cadence®

Virtuoso Layout Suite XL: Connectivity Driven Editing

Product Version IC23.1
November 2023

© 2023 Cadence Design Systems, Inc.

Printed in the United States of America.

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522. All other trademarks are the property of their respective holders.

Restricted Permission: This publication is protected by copyright law and international treaties and contains trade secrets and proprietary information owned by Cadence. Unauthorized reproduction or distribution of this publication, or any portion of it, may result in civil and criminal penalties. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. Unless otherwise agreed to by Cadence in writing, this statement grants Cadence customers permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used only in accordance with a written agreement between Cadence and its customer.
2. The publication may not be modified in any way.
3. Any authorized copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement.
4. The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

Cadence is committed to using respectful language in our code and communications. We are also active in the removal and replacement of inappropriate language from existing content. This product documentation may however contain material that is no longer considered appropriate but still reflects long-standing industry terminology. Such content will be addressed at a time when the related software can be updated without end-user impact.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

Contents

1
Technology File Requirements for Layout XL 37
Prerequisites for Using the Layout XL Connectivity Extractor 40
Prerequisites for Using Virtuoso Photonics Solution (Virtuoso Photonics Option) . . . 41
Connectivity Rules 42
validLayers 44
validLayers – Layer-Purpose Pairs Specification 44
validLayers – Stop Layers Specification 45
validLayers – Enclosed Stop Layers Specification 47
validLayers – Color-Dependent Stopped and Stop Layers Specification 48
validLayers – Bulk Area Layers Specification 49
validLayers – Substrate Layer Specification 50
Derived Layers 51
validPurposes 52
viaDefs 53
Parallel Connectivity Stacks 56
FDSOI Connectivity Stack 62
Connection Requiring an Extra Layer 64

2
Connectivity Source Preparation 65
Schematic Design Elements 65
Design Variables 66
Design Constraints 68
One-to-Many Mapping of Instance or Pin 69
One-to-Many Mapping with Iterated Instances and Bus Pins 69
Properties in Layout XL 70
Pins and Pin Names in Layout XL 71
Extra Pins in the Symbol or Layout Views 71
External Connections 72

© 2023 Cadence Design Systems, Inc.

3
Physical Hierarchy Configuration

Physical Hierarchy Configuration . . . 75
LAM Files and Configuration Views . . . 76
Data Conversion to Use the Layout XL Schema . . . 77
Data Conversion Flow for the Layout XL Schema . . . 79
Schematic Library Conversion . . . 80
Layout Design Conversion . . . 81
Launching Configure Physical Hierarchy . . . 82
Configure Physical Hierarchy Window . . . 84
Configure Physical Hierarchy: License Requirements . . . 88
Hierarchy Configuration Mode in CPH . . . 88
Hierarchy Configuration Instances Table . . . 90
Hierarchy Configuration Cells Table . . . 94
Marking Cells for Virtual Hierarchy Creation . . . 95
Specifying Logical to Physical Mappings for the Components in CPH . . . 96
Mapping Parameter and Terminal Names . . . 98
How to Ignore Components in Physical Hierarchy . . . 100
Ignoring a Cell for Generation or Check Against Source . . . 101
Ignoring an Instance or Occurrence for Generation or Check . . . 103
Splitting Mfactored Devices . . . 104
Splitting Sfactored Devices . . . 105
Splitting Fingered Devices . . . 106
Ignoring Parasitic Devices . . . 108
Setting the Instance Stop Level . . . 110
Specification To Check Parameters . . . 111
Checking a Parameter on a Cellview . . . 114
Checking a Parameter on an Instance or Occurrence . . . 115
Ignore Parameters and Terminals . . . 116
Ignoring a Parameter on a Cellview . . . 116
Ignoring a Parameter on an Instance or Occurrence. . . . 117
Rounding Parameter Values . . . 118
Syntax for Rounding Parameter Values . . . 121
How to Control CPH with Schematic Properties . . . 122
Component Types Mode in CPH . . . 124
Component Types Cells Table . . . 127

© 2023 Cadence Design Systems, Inc.

Defining a Design-Level Component Type . . . 129
Modifying a Component Type . . . 131
Removing a Component Type . . . 132
Soft Block Mode . . . 132
Soft Block Instances Table . . . 134
Soft Block Table Folders . . . 135
Setting the Cell Type . . . 136
Setting the Cell Type using the Shortcut Menu . . . 138
Setting the Cell Type from the CIW . . . 139
Defining Soft Block Parameters . . . 140
Specifying the Boundary . . . 142
Creating a Fixed Rectangular Boundary . . . 143
Creating a Non-Fixed Rectangular Boundary . . . 144
Creating a Polygonal Boundary . . . 146
Adding an Area Estimator Function for Layout Generation . . . 148
Specifying I/O Pins . . . 149
Specifying LayerHalo Obstructions . . . 154
Specifying Cover Obstructions . . . 157
Comparison between LayerHalo and Cover Obstructions . . . 161
Removing Soft Block Parameters . . . 162
Creating a Physical Binding . . . 162
Loading Soft Block Parameters from Another Cellview . . . 163
Saving and Loading Soft Block Parameters to a Floorplan File . . . 164
Generating the Physical Hierarchy . . . 165

4
Device Abutment . . . 167
Prerequisites for Automatic Device Abutment in Layout XL . . . 168
Pcell Set Up for Abutment . . . 169
Automatic Mirroring and Pin Permutation During Abutment . . . 170
Automatic Abutment Flow . . . 172
Sample Parameterized Cells Set Up for Abutment . . . 174
Creating CMOS Pcells to Use with Abutment . . . 177
Multiple Pins Abutment . . . 180
Automatic Device Abutment . . . 180

© 2023 Cadence Design Systems, Inc.

Chaining Transistors Automatically during Layout Generation . . . 181
Chaining Transistors Automatically when Updating Components and Nets . . . 181
Manual Device Abutment . . . 182
Chaining Devices Interactively . . . 183
Chaining Devices Using the Shortcut Menu . . . 186
Removing a Device from a Chain . . . 187
Chaining Devices when Generating Selected From Source . . . 188
Moving Chained Transistors . . . 192
Viewing Extended Chain in Navigator . . . 193
Debugging Abutment . . . 197
Using the Debug Assistant . . . 198
Abutment of Dummy Shapes in Pcells . . . 199
Dummy Shape Abutment Setup . . . 203
Abutment in Virtuoso Photonics Solution (Virtuoso Photonics Option) . . . 204
Abutment Function Flow . . . 205
Abutment Event 5 . . . 207
Abutment Event 2 . . . 209
Abutment Event 1 . . . 212
Abutment Event 6 . . . 214
Abutment Event 3 . . . 217
Abutment Event 4 . . . 218
Unabutment Flow . . . 219
Dummy Abutment Example . . . 220

5
Layout Generation . . . 223
Naming Convention during Layout Generation . . . 224
General Naming Convention . . . 224
Mfactored Devices and Folded Devices . . . 224
Series-Connected Devices . . . 225
lxCombination Devices . . . 225
Nets in Series-Connected and lxCombination Devices . . . 226
Constraint Transfer During Layout Generation . . . 226
Constraint Manager Assistant During Layout Generation . . . 227
Constraint Mapping in One-to-Many Relationships . . . 228

© 2023 Cadence Design Systems, Inc.

Creation and Synchronization of Clusters . . . 228
Generating a PR Boundary . . . 229
Constraint-Driven PR Boundary Generation . . . 230
Moving Generated Components inside the PR Boundary (Placing As In Schematic) . . . 230
Updating the Placement Status Automatically . . . 232
Generating All Components from Source . . . 233
Schematic Parameters when Generating All From Source . . . 235
Inherited Connections when Generating All From Source . . . 236
Viewing Ungenerated Instances In Navigator . . . 237
Specifying the Components to be Generated . . . 241
I/O Pin Generation . . . 243
Specifying Default Values for All Pins During Layout Generation . . . 245
Specifying Values for Selected Pins During Layout Generation . . . 247
Adding a New Pin During Layout Generation . . . 249
Creating Pin Labels During Layout Generation . . . 251
Ignoring Mismatched Pins . . . 253
Generation of Optical Pins and Chains (Virtuoso Photonics Option) . . . 253
Generation of RF TLine Chains (Virtuoso MultiTech Framework) . . . 254
Specifying the Place and Route Boundary During Layout Generation . . . 256
Preserving Floorplanning Objects and Snap Patterns during Layout Generation . . . 258
Layout Generation from Source Using SKILL . . . 260
Placement Modes for Generating Selected Components from Source . . . 260
Generating Components As In Schematic . . . 263
Generation of Virtual Hierarchy using Selected Schematic Components . . . 269
Automatic Snapping of FinFET Devices during Layout Generation . . . 270
Array Generation . . . 271
Array Generation - Instances . . . 273
Array Generation - Modgen . . . 275
Array Generation - Mosaic . . . 277
Array Generation - Group Array . . . 281
Generating an Array . . . 286
Generating Individual Instances . . . 288
Generating Individual Pins . . . 292
Listing Unplaced Components after Layout Generation . . . 294
Clone Generation . . . 297

© 2023 Cadence Design Systems, Inc.

Exact and Non-Exact Matches for Cloning . . . 298
Types of Clones . . . 300
Instance Cloning . . . 301
Clone Source Selection and Target Search and Generation . . . 303
Searching for and Generating the Clone Target . . . 308
Editing a Target Structure Before Generation . . . 312
Instance Cloning Using a Clone Family Constraint . . . 313
Cloning between Multiple Cellviews . . . 319
Mfactored Component Cloning . . . 326
Clone Placement as an Array . . . 335
Creating a Group Array when Generating Clones . . . 345
Generating Synchronous Clones . . . 348
Generating a Family of Synchronous Clones . . . 350
Exceptions to Synchronous Clone Generation . . . 355
Editing a Clone in a Synchronized Family . . . 356
Coloring Synchronization in Synchronous Clones . . . 359
Updating Clone Families . . . 361
Verifying a Clone Family . . . 362
Renaming a Clone Family . . . 363
Removing a Clone from a Family . . . 365
Creating a New Clone Family . . . 367
Adding a Group to an Existing Family . . . 369
Synchronous Copy Generation . . . 371
Generating a Synchronous Copy . . . 372
Managing a Clone Family . . . 374
Synchronous Copy with syncClone Source Group . . . 376
Synchronous Copy with Other Source Groups and Different Source Objects . . . 380
Synchronous Cloning SKILL Functions . . . 382
Dummy Instance Generation . . . 383
Creating Dummy Instances . . . 386
Support for Flexible Connectivity in Dummy Instances . . . 389
Chaining Dummy Instances . . . 390
Dummy Instances Backannotation . . . 395
Net Voltages BackAnnotation . . . 397
Mosaic Instances Generation . . . 398
Creating a Mosaic Instance . . . 400

© 2023 Cadence Design Systems, Inc.

Viewing a Mosaic Instance in the Navigator Assistant . . . 403
Layout XL Commands Supported for Mosaic Instances . . . 405
Layout XL Commands Currently Not Supported . . . 408
Generating Parameterized Layout Cells (VPLGen) . . . 408
Basic VPLGen Flow . . . 411
Creating a VPLGen Core Layout . . . 415
Setting the Default Layout . . . 417
Generating VPLGen Pcell Instances . . . 420
Copying a VPLGen from Layout . . . 420
Updating VPLGen Pcell Instances . . . 424
(Non)parameterized Layout Cells Generation (VLGen) . . . 424
Generating Master Variants . . . 425
Generating Master Variant . . . 430
Transparent Layout Hierarchy Generation . . . 431
Make Cell . . . 432
Make Cell – Transparent . . . 434
Make Cell – Connected . . . 436
Make Cell – Pins Created . . . 437
Creating a Made Cell . . . 438
Make Cell Pin Creation . . . 443
Set Instance as Transparent . . . 447
Creating a Transparent Instance . . . 449
Unset a Transparent Instance . . . 451
Hierarchical Layout Instance Flattening . . . 453
Flattening a Layout Instance . . . 456
The Load Physical View Command . . . 459
Creating, Loading, and Modifying a Source Cellview . . . 459
Loading Physical Information from Another Cellview . . . 462
Specifications for Loading Cellview Information from Source to Target Cellview . . . 463
Load Physical View Summary Report . . . 477
Pseudoparallel Connections . . . 480
Defining a Pseudoparallel Connection . . . 485
Resetting a Pseudoparallel Connection . . . 486
Generating a Layout without Chaining and Folding on Pseudoparallel Nets . . . 487
Generating a Layout with Chaining and Folding on Pseudoparallel Nets . . . 490
Selected Components with Pseudoparallel Nets Generation . . . 492

© 2023 Cadence Design Systems, Inc.

6
Layout Editing 493

Constraint-Aware Editing 495
Constraint-Aware Editing Considerations 497
Enabling and Disabling Constraint-Aware Editing Mode 498
Moving Objects 499
Changing the Appearance of Draglines 499
Swapping Components 500
Locking Devices 501
Unlocking a Device 505
Adding an Instance in Layout 506
Adding a Pin 507
Assigning Instance Pins to a Net 508
Viewing Extended Selection for Instances, Pins, and Nets in Navigator 509
Transistor Folding 513
Interactive Transistor Folding 514
Folding a Transistor or FinFET Device 515
Split-Fingered Device Folding 523
Folding a Multi-fingered Device to Create Split Fingers 524
Ignoring Schematic Mfactor during Folding 525
Unit Identifiers in Layout XL 530
Pin Permutation 530
Permuting Pins Manually 532
Permuting Pins Automatically 533
Disabling Permutation for an Instance 535
Incomplete Nets 537
Incomplete Nets Commands 541
Showing and Hiding Current Incomplete Nets 543
Showing Common Incomplete Nets for Selected Instances 545
Showing Incomplete Nets in Preselection Mode 547
Behavior by Object Type in Preselection Mode of Incomplete Nets 548
Showing Incomplete Nets in Postselection Mode 549
Behavior by Object Type in Postselection Mode of Incomplete Nets 552
Searching for Specific Incomplete Nets 555
Showing Newly Created Incomplete Nets Automatically 558

© 2023 Cadence Design Systems, Inc.

Assigning Colors to Incomplete Nets . . . 561
Zooming In on Incomplete Nets . . . 561
Editing In Place . . . 562
Soft Block Editing . . . 564
Changing the Block Type . . . 564
Changing the Boundary of Soft Blocks . . . 566
Changing I/O Pin Attributes . . . 571
Changing Obstructions . . . 574
Enabling Level-1 Editing Mode . . . 579
Moving an Object in Level-1 Editing Mode . . . 583
Stretching an Object in Level-1 Editing Mode . . . 586
Stretching a Single Edge in Level-1 Editing Mode . . . 587
Maintaining the Area of a Soft Block during Stretch . . . 589
Reshaping an Object in Level-1 Editing Mode . . . 594
Chopping an Object in Level-1 Editing Mode . . . 596
Soft Block Pins . . . 603
Creating a Soft Pin . . . 604
Updating Parameters for a Soft Pin . . . 608
Deleting a Soft Pin . . . 608
Creating a Feedthrough Terminal Pin . . . 609
Updating Parameters for a Feedthrough Terminal Pin . . . 613
Deleting a Feedthrough Terminal Pin . . . 613
Snapping Soft Block Pins . . . 613
Snapping Soft Block Pins to Grid . . . 616
Snapping All Pins to Grid . . . 619
Snapping Selected Pins to Grid . . . 621
Working With Pin Connectivity Models . . . 623
Supported Pin Connectivity Models . . . 623
Setting a Pin Connectivity Model . . . 624
Smart Display . . . 629
Enabling Smart Display . . . 632
Accessing Smart Display Options . . . 633
Display of Mosaics in Abstracted Mode . . . 634
Guard Rings in Abstracted Mode . . . 636
Hierarchical Blocks in Abstracted Mode . . . 638
figGroups in Abstracted Mode . . . 641

© 2023 Cadence Design Systems, Inc.

Limitations of Smart Display . . . 643
Net Name Display . . . 644
Net Name Filtering . . . 649
Setting the Net Name Display Options . . . 650
Composite Waveguides (Virtuoso Photonics Option) . . . 653
Adding a Waveguide Element . . . 654
Deleting a Waveguide Element . . . 655
Changing the Position of a Waveguide Element . . . 655
Adding an Optical Connector . . . 656
Waveguide Element Rotation . . . 657

7
Connectivity Extraction in Layout XL . . . 659

Connectivity Extraction Use Model . . . 660
Connections Recognized by the Extractor . . . 661
Instance Terminal Connectivity Considered During Extraction . . . 662
Support for Diffusion Layers During Extraction . . . 664
Weakly-Connected Instance Pins During Extraction . . . 666
Connectivity from Attached Labels to Consider During Extraction . . . 666
Connectivity from stampLabels During Extraction . . . 668
Extracting a Top-Level Design . . . 670
Extraction Stop Level . . . 671
Extracting Hierarchical Cellviews . . . 672
Extracting Connectivity by Area . . . 677
Extracting Connectivity for Unverified Areas . . . 680
Extracting Connectivity by Nets . . . 681
Enabling and Disabling Incremental Extraction . . . 682
Connectivity Extraction from Substrate and Well Layers . . . 683
Soft Connection Extraction . . . 684
Bulk Area Connectivity Extraction . . . 685
Bulk Area Connectivity Extraction - Identifying the Substrate Area . . . 686
Bulk Area Connectivity Extraction - Isolating the Substrate and Well Layers . . . 690
Bulk Area Connectivity Extraction - Identify the Bulk Area . . . 701
Bulk Area Connectivity Extraction - Identify the Substrate and Well Tap Vias . . . 703
Connectivity Propagation through the Substrate and Well Areas . . . 704

© 2023 Cadence Design Systems, Inc.

Hierarchical Overlaps Processed During Extraction . . . 707
Connectivity Extraction from Mosaics . . . 713
Connectivity Extraction for Stacked Cut Layers . . . 717
Connectivity Extraction for Recognition Layer . . . 720
Support for Shape Trimming during Connectivity . . . 720
Support for Cut Shape Trimming during Extraction . . . 722
Support for Color-Dependent Shape Trimming . . . 724
Defining Color-Dependent Stopped Layers . . . 724
Defining Color-Dependent Stop Layers . . . 725
Enabling Virtual Connections . . . 727
Markers Generated by Connectivity Extractor . . . 734
Grouping Markers in the Annotation Browser . . . 735
Sorting Markers in the Annotation Browser . . . 740
Filtering Markers in the Annotation Browser . . . 743
Assigning Colors to Markers . . . 753
Zooming In on Markers . . . 754
Fixing Markers Interactively in Annotation Browser . . . 757
Hint Support for DRC/DFM Tab of Annotation Browser . . . 763

8
Design Data Checks . . . 769
Selecting and Deselecting Nets in Navigator . . . 770
Selected Net Routing . . . 772
Locking and Unlocking Nets to Set Routing Status . . . 774
Editing Net Attributes and Properties . . . 780
Design Probing . . . 783
General Probing Behavior . . . 783
Object Probing Using the Right Mouse Button . . . 785
Adding a Probe using the Right Mouse Button . . . 786
Removing Probes using the Right Mouse Button . . . 790
Dynamic Probing . . . 793
Probing Objects by Clicking in a Window . . . 794
Probing Objects using the XL Probe Form . . . 796
Probing a Hierarchical Design . . . 797
Tracing Nets . . . 800

© 2023 Cadence Design Systems, Inc.

Net Tracer Toolbar . . . 802
Adding Net Traces . . . 804
Saving All Visible Traces . . . 807
Loading Physical Traces . . . 808
Creating a Step Trace . . . 809
Finding Neighboring Shapes of a Traced Shape . . . 813
Adding a Trace Using the Trace Manager . . . 816
Editing In Place a Trace . . . 817
Tracing Nets using Navigator . . . 819
Checking Shorts, Incomplete Nets, and Illegal Overlaps . . . 820
Locating Shorts Using Short Locator . . . 822
Locating the Shapes Causing the Short . . . 826
Locating Short with Large Number of Common Shapes . . . 833
Design Check Against Source . . . 834
Checking a Layout Against a Schematic . . . 836
Pinning a CAS Glyph Information Balloon . . . 843
CAS TAB Marker Categories . . . 844
Fixing CAS Mismatches . . . 846
Fixing CAS Tab Markers . . . 847
Suppressing the Check Against Source Form . . . 849
Ignoring Components with no Schematic Counterpart . . . 850
Design Check for Layout XL Compliance . . . 852
Viewing XL Status . . . 855
Design Check for Manufacturability . . . 861
Match and Fix Flow . . . 861
Match and Fix Rule Definition . . . 864
Batch Checker . . . 866
Running the Batch Checker . . . 867
Checking Routability Using Batch Checker . . . 869
Checking Process Rules Using Batch Checker . . . 871
Checking Connectivity Using Batch Checker . . . 873
Checking Specialty Routing Using Batch Checker . . . 875
Checking Fabrication Using Batch Checker . . . 877
Checking WSP | SP Active Using Batch Checker . . . 879
Finding Violations Using Annotation Browser . . . 883

© 2023 Cadence Design Systems, Inc.

9
Layout XL Design Data Updates

Layout XL Design Data Updates . . . 885
Components and Nets Updates . . . 886
Updating Components and Nets . . . 887
Updating Selected Layout Components . . . 890
Updating Nets and Instance Name Mismatches Only . . . 893
Specifying the Components to be Updated . . . 894
I/O Pins and PR Boundary Updates . . . 899
Updating Layout Constraints . . . 899
Layout Parameters Updates . . . 901
Updating Layout Parameter Values in Preselection Mode . . . 906
Updating Layout Parameter Values in Postselection Mode . . . 907
Updating Parameter Values for Folded Devices . . . 908
Excluding a Specific Parameter from the Update . . . 908
Schematic Parameter Updates . . . 909
Updating Schematic Parameter Values in Preselection Mode . . . 914
Updating Schematic Parameter Values in Postselection Mode . . . 914
Device Correspondence Updates . . . 916
User-Defined Device Correspondence . . . 918
Dynamic Updates to the Define Device Correspondence . . . 918
Defining a One-to-One Device Correspondence . . . 920
Defining a One-to-Many Device Correspondence . . . 925
Defining a Many-to-One Device Correspondence . . . 926
Defining a Many-to-Many Device Correspondence . . . 927
Assigning Layout Instance Terminals to Nets . . . 928
Deleting a Device Correspondence . . . 929
Binding Using the Bind Command in the Shortcut Menu . . . 933
Define Device Correspondence SKILL Function . . . 934
Updating the Connectivity Reference . . . 935
Updating the Schematic View . . . 935
Updating the Physical Configuration View . . . 935
Binding Updates . . . 936
Update Binding Flow . . . 938
Update Binding Report . . . 945
Updating All Physical Bindings . . . 951

© 2023 Cadence Design Systems, Inc.

10
Troubleshoot Layout XL Issues 953
Troubleshoot Startup Issues 954
Troubleshoot Configure Physical Hierarchy 956
Troubleshoot Constraint Migration 958
Troubleshoot Parameter Evaluation 958
Troubleshoot Layout Generation Problems 960
Troubleshoot Cloning 961
Troubleshoot Chaining 967
Troubleshoot Editing 967
Troubleshoot Connectivity Extraction 969
Troubleshoot Layout XL Compliance 972
Troubleshoot Binding Update Issues 992
Troubleshoot Performance 993

A
Layout XL Environment Variables 995
Setting Environment Variables 995
Setting Environment Variables in a .cdsenv or .cdsinit File 996
Setting Environment Variables in the CIW 996
Setting Environment Variables in the Connectivity Form 996
List of Layout XL Environment Variables 998
abutAddUniqueDummyNet 1019
abutChainEvent 1020
abuttedPcellParamUpdate 1021
abutSetEditedAsMoved 1022
abutWithoutConnectivity 1023
addlxStickyNetPropOnDummy 1024
allowPinResizing 1025
allowPinResizingInEdit 1026
allowSelectedParamUpdate 1027
alwaysGenerateProps 1028
alwaysUseDescriptionForFiltering 1029
autoAbutment 1030

© 2023 Cadence Design Systems, Inc.

autoArrange 1031
autoHideEmptyTabs 1033
automaticHighlight 1034
autoMirror 1035
autoMirrorChains 1037
autoMirrorFilter 1040
autoPermuteNoAbutProp 1041
autoSavePhysConfigUpdate 1042
autoSpace 1043
autoZoomEnabled 1045
autoZoomFactor 1046
autoZoomIsFixed 1047
backAnnotateInstances 1048
backAnnotateIgnoreBoundMasters 1050
backAnnotatePackingMethod 1051
backAnnotateXSpacing 1052
backAnnotateYSpacing 1053
bindAllCellsInDesign 1054
bindClearConn 1055
bindCorrectMaster 1056
bindCreateTransparent 1057
bindCrossRefFile 1058
bindCurrentLevel 1059
bindExtract 1060
bindExtractedNetlistFile 1061
bindFactorWarnLimit 1062
bindFile 1063
bindFlattenLayoutCreate 1064
bindIgnoreBulk 1065
bindIgnoreDummies 1066
bindIgnoreDummiesFullyConn 1067
bindIgnoreRouteCells 1068
bindIncr 1069
bindIncrAddIgnore 1070
bindIncrConn 1071
bindIncrMessage 1072

© 2023 Cadence Design Systems, Inc.

bindIncrPropagateConn 1073
bindInitConn 1074
bindInitInstTerms 1075
bindLayoutStop 1076
bindNameOnly 1077
bindNewSplitParam 1078
bindPermuteRules 1079
bindPreserveExistingBindings 1081
bindPreserveLayoutHierarchy 1082
bindPreserveUserBindings 1083
bindPVSRulesFile 1084
bindPVSScale 1085
bindRemoveIgnores 1086
bindRemoveUnboundTerminalsNoPins 1087
bindReportCellsNoTerms 1088
bindReportDummies 1089
bindReportHierarchy 1090
bindReportReduction 1091
bindReportRouteCells 1092
bindReportTerminalMismatch 1093
bindReportUnconnInstTerms 1094
bindSourceSkip 1095
bindSourceStop 1096
bindStrict 1097
bindUpdateName 1098
bindUpdateTermName 1099
bindUseFingerWidth 1100
bndRemoveDeviceConnectivity 1101
capacitanceParamNames 1103
casAppendToLogFile 1105
casDisplayBusTerminalsMismatch 1106
casDisplayConnMismatch 1107
casDisplayDummyMismatch 1108
casDisplayMastersMismatch 1109
casDisplayNameMismatch 1110
casDisplayParamsMismatch 1111

© 2023 Cadence Design Systems, Inc.

casDisplayUnboundInsts . . . 1112
casDisplayedDiffsLimit . . . 1113
casGlyphs . . . 1114
casInfoBalloonOnHover . . . 1115
casInfoWindow . . . 1116
casLogFileName . . . 1117
casOpenWorkspace . . . 1118
casMarkers . . . 1119
casMarkerTypeLimit . . . 1120
casSelected . . . 1121
casUngenRemoveSplitParamDiff . . . 1122
casVisibleUnboundMarkers . . . 1123
chainAbutStrategy . . . 1124
chainAllowSingleBulk . . . 1125
chainCellOrient . . . 1126
chainDummyFlexBothEndNets . . . 1127
chainDummyAbutment . . . 1128
chainDummyUpdate . . . 1129
chainEffort . . . 1130
chainExtendSelection . . . 1131
chainFolds . . . 1132
chainInPlace . . . 1133
chainInPlaceDirection . . . 1134
chainLeftNet . . . 1135
chainMirror . . . 1137
chainMirrorEquivOrients . . . 1139
chainOptimize . . . 1141
chainPartitionByNumFins . . . 1142
chainPartitionByCompType . . . 1143
chainPartitionSizeMax . . . 1144
chainPermute . . . 1145
chainPermutePins . . . 1146
chainPreserveExistingChains . . . 1148
chainPreserveRows . . . 1149
chainSyncChains . . . 1150
chainSyncChainsSDFirst . . . 1151

© 2023 Cadence Design Systems, Inc.

chainUseAbutSpacing 1152
chainUseDeviceOrder 1153
checkRFUnimplementedTerms 1154
checkBindings 1155
checkCellsAsLeaf 1156
checkCompType 1157
checkDuplicateObjects 1158
checkExcludeCellViews 1159
checkExtractLayout 1160
checkMasterCompType 1161
checkMissingParamsOrProps 1162
checkMissingTermPins 1163
checkParamsOnly 1164
checkPermute 1165
checkRemoveDevice 1166
checkShortsAndOpens 1167
checkStateConfirmModeChange 1168
checkStrictFingers 1169
checkTermLabelAllowBracketDiff 1170
ciwWindow 1171
cloneArrayFlipFirstColumn 1172
cloneArrayFlipFirstRow 1173
cloneArrayFlipNextFromPreviousColumn 1174
cloneArrayFlipNextFromPreviousRow 1175
cloneArrayRowOrColValue 1176
cloneArrayXSpacing 1177
cloneArrayYSpacing 1178
cloneJustify 1179
cloningAutomaticUpdateLayoutParameters 1180
cloningDoExactMatch 1181
cloneNumberPerRowOrCol 1182
cloneSynchronizeUnlockedColor 1183
cloneTransformFrom 1184
cloningSearchIn 1185
cloningSearchInConstraintOnly 1186
cloningSearchInSelSetOnly 1187

© 2023 Cadence Design Systems, Inc.

cloningUpdateConstraint 1188
closeSchWindow 1189
colorDraglines 1190
colorOrNot 1192
constraintAwareEditing 1193
cphAutoUprev 1195
cphCheckEmptyConfig 1197
cphCheckTopCellName 1198
cphHierForceDescendStopLibs 1199
cphIncrementalElab 1200
cphLayoutLibraryInLibList 1201
cphLayoutSync 1202
cphMasterLibraryOnly 1203
cphNonDefaultAutomaticName 1204
cphOpenWin 1205
cphPromptConfigOrSchematic 1206
cphScrollShowCellsMenu 1207
cphShowInstanceColumns 1208
cphStopLevel 1209
cphStopLogicalElabAtPhysLeaf 1210
cphStrictSchematicCheck 1212
cphUseLayoutLibrary 1213
cphUseLayoutViewName 1214
cphVHSizeLimit 1215
createBoundaryLabel 1216
createImplicitBusTerminals 1217
createPinsFromLabels 1219
crossProbeHier 1220
crossSelect 1221
crossSelect 1222
crossSelectInstTerm 1223
crossSelectLayoutShape 1224
deleteConfirmModeChange 1225
deleteUnmatchedInsts 1226
deleteUnmatchedPins 1227
deviceExtractType 1228

© 2023 Cadence Design Systems, Inc.

disableCASOptionsPopUp 1229
disableFolding 1230
disableVXLClonedModgenEditPopUp 1231
displayConfigureTab 1232
drdUseNetName 1233
dummyBackAnnotateLocalNetActive 1235
dummyBackAnnotateRelaxActiveRule 1236
electricalPinSetup 1237
enableCloneColoringSupport 1238
equivalentAbutClasses 1239
excludedViewList 1241
extractShortLocatorAutoZoomToFit 1243
extractShortLocatorCommonShapesColor 1244
extractShortLocatorDisplayMode 1245
extractShortLocatorHaloTransparency 1246
extractShortLocatorHighlightLimit 1247
extractShortLocatorNetColor1 1248
extractShortLocatorNetColor2 1249
extractShortLocatorNumThreads 1250
extractShortLocatorOtherShapesColor 1251
extractShortLocatorShowInstancePins 1252
extractShortLocatorShowPinsAndLabels 1253
extractShortLocatorShowVirtualLabelNetOnShapes 1254
extractShortLocatorTrueColor 1255
extractShortLocatorUseInstancePins 1256
extractShortLocatorVirtualLabelNetOnShapesFontHeight 1257
extractShowMustConnectMarkers 1258
extractShowUnimplementedInstTerms 1259
extractShowWeakMarkers 1260
fingerSplit 1261
fingerSplitSetWidth 1262
flattenHierNameSep 1263
flattenKeepLabels 1264
flattenPreserveConn 1265
flattenPreserveUnthreadedConn 1266
flightLineEnable 1267

© 2023 Cadence Design Systems, Inc.

flightLineForAllMustConnectInstTerms . . . 1269
foldEnforceWidthMatch . . . 1270
foldFingerSplitUseSchWidth . . . 1271
foldIgnoreMfactor . . . 1272
foldInPlace . . . 1273
foldInPlaceDirection . . . 1274
foldKeepFolds . . . 1275
foldsLimit . . . 1277
foldOptimizeSupplyAndGroundNets . . . 1278
foldUseThreshold . . . 1279
generateModgens . . . 1280
generatePinsDisplayLimit . . . 1281
generateXSpacing . . . 1284
generateYSpacing . . . 1285
hardBlockColor . . . 1286
hideDraglinesForGlobalNets . . . 1287
hideMarkersWhenBrowserHidden . . . 1289
hierarchyDepth . . . 1290
hierMfactorNames . . . 1291
hierSummaryColumns . . . 1292
hierSummaryLogFileName . . . 1294
highlightColors . . . 1295
highlightedIsSelectable . . . 1296
highlightObjRMB . . . 1297
highlightPotentialSwap . . . 1298
highlightPotentialSwapColor . . . 1299
highlightShapes . . . 1300
ignoreTermsWithNonRoutingLabels . . . 1301
incrementalCas . . . 1302
incrementalCasLimit . . . 1303
incompleteNetFilter . . . 1304
incompleteNetFilterDefaultList . . . 1305
incompleteNetFilterMode . . . 1306
inductanceParamNames . . . 1307
infoWindow . . . 1309
infoWindowPos . . . 1310

© 2023 Cadence Design Systems, Inc.

initAreaFuncName . . . 1311
initAspectRatio . . . 1312
initAspectRatioOption . . . 1314
initCreateBoundary . . . 1316
initCreateInstances . . . 1317
initCreateMTM . . . 1318
initCreatePadPins . . . 1320
initCreatePins . . . 1321
initCreateSnapBoundary . . . 1322
initDoFolding . . . 1323
initDoStacking . . . 1324
initEstimateArea . . . 1325
initGlobalNetPins . . . 1326
initIOLabelType . . . 1327
initIOPinLayer . . . 1329
initIOPinLayerNames . . . 1331
initIOPinName . . . 1333
initIOPinPurposeNames . . . 1334
initPinHeight . . . 1336
initPinMultiplicity . . . 1338
initPinWidth . . . 1339
initPrBoundaryH . . . 1341
initPrBoundaryW . . . 1342
initUtilization . . . 1343
insertChain . . . 1345
insertChainAbutViolation . . . 1346
lamReadCdsLibs . . . 1347
lamReadFileMessage . . . 1348
lamReadUserLibs . . . 1349
lamReuseCache . . . 1350
layoutWindow . . . 1351
lswWindow . . . 1352
lxAbutPassGroupIdToEvent1 . . . 1353
lxAllowPseudoParallelNets . . . 1354
lxcuAllCellsInDesign . . . 1356
lxcuAllCellsInLibrary . . . 1357

© 2023 Cadence Design Systems, Inc.

lxcuApp . . . 1358
lxcuBinder . . . 1359
lxcuCreateHTMLReport . . . 1360
lxcuDetailedReturn . . . 1361
lxcuExcludeCells . . . 1362
lxcuExtract . . . 1363
lxcuFilterMessage . . . 1364
lxcuIncludeCells . . . 1365
lxcuLayoutComplete . . . 1366
lxcuLeafCells . . . 1367
lxcuLogFile . . . 1368
lxcuLogFileDirectory . . . 1369
lxcuOpenHTMLReport . . . 1370
lxcuOpenWorkspace . . . 1371
lxcuSvdbPath . . . 1372
lxcuSvdbScale . . . 1373
lxcuUpdate . . . 1374
lxcuWithoutSchematic . . . 1375
lxChainAlignNMOS . . . 1376
lxChainAlignPMOS . . . 1378
lxCanSwapSDFunction . . . 1380
lxDeltaWidth . . . 1381
lxDummyBackAnnotateAll . . . 1382
lxDummyBackAnnotateMFactor . . . 1383
lxDummyBackAnnotateMFactorName . . . 1385
lxDummyBackAnnotateMissingTerm . . . 1386
lxEvalCDFCallbacks . . . 1387
lxFingeringNames . . . 1393
lxGenerateInBoundary . . . 1395
lxGenerationOrientation . . . 1396
lxGenerationTopLevelOnly . . . 1398
lxGetSignifDigits . . . 1399
lxGroundNetNames . . . 1401
lxInitResetSource . . . 1402
lxLocalAbutment . . . 1403
lxPositionCompact . . . 1405

© 2023 Cadence Design Systems, Inc.

lxPositionMinSep . . . 1406
lxPositionPinsOnBoundary . . . 1407
lxPositionUseInstPrB . . . 1408
lxRetainFoldOrient . . . 1409
lxSchematicDefaultApp . . . 1410
lxSchematicEditMode . . . 1412
lxSchExtractTopLevelOnly . . . 1414
lxStackMinimalFolding . . . 1416
lxStackPartitionParameters . . . 1417
lxSupplyNetNames . . . 1419
lxSwapSDFunction . . . 1420
lxUpdateFoldedWidth . . . 1422
lxUseLibList . . . 1425
lxValidateXLParameterEvaluation . . . 1427
lxWidthTolerance . . . 1429
makeCellCreatePins . . . 1430
makeCellPinsBelowBoundary . . . 1431
makeCellPRBoundary . . . 1432
makeCellPreserveConn . . . 1433
makeCellTransparent . . . 1434
matchedToIgnore . . . 1435
mfactorNames . . . 1436
mfactorSplit . . . 1438
moveAsGroup . . . 1439
netTracerAutoZoomOnSelected . . . 1440
netTracerColor . . . 1441
netTracerComputeNeighborNet . . . 1442
netTracerComputeNeighborNetMethod . . . 1443
netTracerConnectedLayer . . . 1444
netTracerConstraintGroup . . . 1445
netTracerCrossHighlightNetInSchematic . . . 1446
netTracerDisplayDepth . . . 1447
netTracerDisplayMode . . . 1448
netTracerExcludePurposes . . . 1449
netTracerExcludePurposesName . . . 1450
netTracerFromNet . . . 1451

© 2023 Cadence Design Systems, Inc.

netTracerHaloFadeoutShapes 1452
netTracerHaloForPins 1453
netTracerHaloForShapes 1454
netTracerHaloStyleThick 1455
netTracerHaloTransparency 1456
netTracerMaxCutShapes 1457
netTracerNeighborsOnLayer 1458
netTracerNumShapesLimitForRefresh 1459
netTracerNumThreads 1460
netTracerOn 1461
netTracerPopupSelectionWindow 1462
netTracerRange 1463
netTracerRegion 1465
netTracerSaveMethod 1466
netTracerShowPreview 1467
netTracerStartLevel 1468
netTracerStepMode 1469
netTracerStopLayer 1470
netTracerStopLevel 1471
netTracerUpdateIncrementally 1472
netTracerViaLayer 1473
openConnRef 1474
openConnRefTab 1475
openLocation 1476
openMode 1477
opticalPinSetup (Virtuoso Photonics Option) 1478
paramsToCheck 1479
paramsToIgnore 1481
paramsToIgnoreForCheck 1483
paramTolerance 1484
passFigIdsToEvent3 1486
phoAbutClass (Virtuoso Photonics Option) 1487
phoAbutFunction (Virtuoso Photonics Option) 1488
phoComposeMaster (Virtuoso Photonics Option) 1489
phoComposeMasterPromptOff (Virtuoso Photonics Option) 1490
phoAbutNonPcells (Virtuoso Photonics Option) 1491

© 2023 Cadence Design Systems, Inc.

phoPinInputAngle (Virtuoso Photonics Option) . . . 1492
phoPinLayer (Virtuoso Photonics Option) . . . 1493
phoPinRadius (Virtuoso Photonics Option) . . . 1494
phoPinWidth (Virtuoso Photonics Option) . . . 1495
physOnlyTerminalsRemoveBang . . . 1496
pibCellName . . . 1497
pibCreateBlockageType . . . 1498
pibCreatePinLPPChoice . . . 1499
pibFloatPurposeNames . . . 1500
pibLayerNames . . . 1501
pibLibName . . . 1502
pibPinLPPName . . . 1503
pibPinPurposeName . . . 1504
pibPurposeNames . . . 1505
pibPushAllAsBlockage . . . 1506
pibPushBlockage . . . 1507
pibPushCloseWire . . . 1508
pibPushFloats . . . 1509
pibPushNonPowerGndNet . . . 1510
pibPushPowerGndNet . . . 1511
pibPushRows . . . 1512
pibPushMode . . . 1513
pibPushSelection . . . 1514
pibPushWhole . . . 1515
pibNetType . . . 1516
pibReMasterInst . . . 1517
pibViewName . . . 1518
pibWSPActivePattern . . . 1519
placeCloneAsArray . . . 1520
prBoundarySnap . . . 1521
prefixLayoutInstNamesWithPipe . . . 1522
preserveAreaBoundary . . . 1523
preserveBlockages . . . 1524
preserveClusters . . . 1525
preserveClusterBoundaries . . . 1526
preservePRBoundary . . . 1527

© 2023 Cadence Design Systems, Inc.

preserveRows . . . 1528
preserveSnapBoundary . . . 1529
preserveSnapPatterns . . . 1530
preserveTrackPattern . . . 1531
probeCycleHilite . . . 1532
probeDevice . . . 1533
probeDuringCreate . . . 1534
probeHiliteLayer . . . 1535
probeInfoInCIW . . . 1536
probeNet . . . 1537
probePin . . . 1538
processBatchViolations . . . 1539
propagateAssignNet . . . 1540
propsUsedToIgnoreObjs . . . 1541
propsUsedToIgnoreObjsForCheck . . . 1543
removeDuplicateObjects . . . 1544
renameClonesWhenFamilyIsRenamed . . . 1545
renameConflictingLayoutInstances . . . 1546
resistanceParamNames . . . 1547
saveViewSubType . . . 1549
schematicWindow . . . 1550
schLayLibraryPair . . . 1551
scopeLevel . . . 1552
segmentParamNames . . . 1554
setPPConn . . . 1555
sfactorNames . . . 1556
sfactorSplit . . . 1558
showDraglinesForDistantConns . . . 1559
showIncompleteNets . . . 1561
showPropagateConnectivityReport . . . 1562
showTabs . . . 1563
showUnplacedInstances . . . 1564
smartDeletePathSegBetweenVia . . . 1565
snapCpaToPlacementGrid . . . 1566
softBlockColor . . . 1567
stopList . . . 1568

© 2023 Cadence Design Systems, Inc.

stretchWithPRBoundary 1570
substrateNetParamNames 1571
suppressAbutFailMsgs 1572
suppressExpansion 1573
swapIncludeOrientation 1575
swapOrigin 1576
tabPosition 1577
transistorWidthParamNames 1578
turnInfixOffWhenSmartSnapping 1580
unfoldUseLayoutWidths 1581
updateBindMfactorSplit 1582
updateBindPhysicalBinding 1583
updateBindUsePhysicalBinding 1584
updateLayoutInstNames 1585
updateLayoutNetNames 1586
updateLayoutParameters 1587
updateNetMinMaxVoltage 1588
updateNetSigType 1589
updateNetsOnly 1590
updateOneToOneMappings 1591
updateParamChangeNumFolds 1592
updateParamCopyFolds 1593
updateParamsForCheck 1594
updatePlaceBelowBoundary 1595
updatePlacementStatus 1596
updatePlacementStatusInBoundary 1597
updateReplacesMasters 1598
updateSchParamWithEvalValue 1599
updateSelectedComponents 1602
updateWithMarkers 1603
useConstantAreaStretch 1604
useHierMfactorNames 1605
viewList 1606
vlGenAutoDetectCells 1608
vlGenCreateCells 1609
vlGenCreateCellsInPhysConfigLib 1610

© 2023 Cadence Design Systems, Inc.

vplGenCreateCells . . . 1611
vplGenLibs . . . 1612
vplGenParams . . . 1613
vplGenSetDefaultLayout . . . 1614
xlComplianceCheckDevices . . . 1615
xlComplianceCreateCasMarkers . . . 1616
xlComplianceFilter . . . 1617
xlComplianceHtml . . . 1618
xlComplianceHtmlOpen . . . 1619
xlComplianceLogDir . . . 1620
xlComplianceLogs . . . 1621
xlStatus . . . 1622
xlStatusHierShorts . . . 1623
Batch Checker Environment Variables . . . 1624
WSP | SP Active Checker Environment Variables . . . 1624
activeOn . . . 1625
color . . . 1626
depth . . . 1627
ignoreValidJogs . . . 1628
mergeShapes . . . 1629
selectLayer . . . 1630
selectPurposes . . . 1631
vioLimit . . . 1632
width . . . 1633
wireType . . . 1634
Placement Checker Environment Variables . . . 1635
constraints . . . 1636
rowOrGrid . . . 1637
spacing . . . 1638
tapCell . . . 1639
padding . . . 1640
minPinAccess . . . 1641
colorAware . . . 1642
reportViolations . . . 1643
createMarkers . . . 1644
selOnly . . . 1645

© 2023 Cadence Design Systems, Inc.

GUI Options and Corresponding Environment Variables 1646
Hidden Environment Variables with No Corresponding GUI Option 1653

B
Command Quick Reference 1655

C
Layout XL Properties 1661
Setting Layout XL Properties 1663
abNetsColumnValue Property 1665
abutAccessDir Property 1666
abutClass Property 1667
abutCondInclusion Property 1669
abutFunction Property 1670
abutGateNet Property 1674
abutMosStretchMat Property 1675
abutStretchMat Property 1679
abutOffset Property 1681
caeIgnoreInCluster Property 1683
extractStopLevel Property 1684
extractToLevel Property 1685
ignore Property 1686
lvsIgnore Property 1688
lxAutoAbut Property 1690
lxAutoSpace Property 1691
lxCombination Property 1692
lxLocalAbutment Property 1695
lxMfactorSplit Property 1696
lxNetNamePrefix Property 1697
lxParamsToCheck Property 1700
lxParamsToIgnore Property 1701
lxParamsToIgnoreForCheck Property 1703
lxRemoveDevice Property 1705
lxRounding 1706
lxSeriesTerms Property 1707

© 2023 Cadence Design Systems, Inc.

lxSfactorSplit Property 1709
lxStickyNet Property 1710
lxStopList Property 1712
lxUseCell Property 1713
lxViewList Property 1714
mfactor Property 1715
nonStickyTerminals Property 1718
permuteRule Property 1721
physOnly Property 1727
sfactor Property 1728
vxlInstSpacingDir Property 1731
vxlInstSpacingRule Property 1734

D
Layout XL Forms 1737

Add Area Estimators Form 1741
Add A New Pin Form 1742
Add Soft Pin Form 1743
Application Readiness Checker Form 1745
Assign Layout Instance Terminals Form 1754
Assign Nets Form 1755
Annotation Browser Options Form 1756
Back Annotate Dummies Form 1758
Batch Checker Form 1760
Block Parameters using Physical View Form 1768
Check Against Source Form 1769
Check XL Compliance Form 1772
Clone Array Settings Form 1775
Composite Waveguide Editor Form (Virtuoso Photonics Option) 1776
Connectivity Form 1778
Convert Libraries to Use Physical Configuration Views Form 1814
Create Cluster Boundary Form 1816
Create Group Array Form 1819
Create Feed Through Terminal Block Pins Form 1821
Create Path Form 1823

© 2023 Cadence Design Systems, Inc.

Create Physical Configuration View Form 1826
Custom Show Filter Form 1827
Define Device Correspondence Form 1829
Design Summary Form 1832
Edit Net Attributes Form 1833
Edit Soft Blocks Form 1834
Extract Layout Form 1842
Extractor Diagnostics Options Form 1845
Generate Chained Devices Form 1848
Generate Clones Form 1852
Generate Folded Devices Form 1860
Generate Layout Form 1869
Generate Selected Components Form 1884
Generate Selected Components - Pin Options Form 1889
Incomplete Net Filter Form 1891
Instance/Pin List Form 1893
Load Physical View Form 1894
Load Traced Nets View Form 1902
Master Variant Form 1903
Match And Fix Form 1905
Modify Correspondence Form 1913
Move Cells Form 1914
Net Name Display Options Form 1915
Net Tracer Custom Setup Form 1920
Net Tracer Display Options Form 1924
Net Tracer Neighbors Form 1932
Net Tracer Options Form 1934
Open Physical Configuration View Form 1939
Permutation Information Form 1940
Save Short Locator View Form 1941
Save Traced Nets View Form 1942
Selection Options Form 1944
Set Net Order Form 1952
Set Pin Label Text Style Form 1953
Short Locator Form 1955
Smart Display Options Form 1963

© 2023 Cadence Design Systems, Inc.

Snap Pins Form 1974
Soft Block Global Options Form 1979
Startup Option Form 1982
Swap Components Form 1984
Trace Manager Form 1985
Update Binding Form 1987
Update Clone Families Form 1997
Update Components and Nets Form 2001
Update Connectivity Reference Form 2014
Update Layout Device List Form 2016
View By Form 2017
XL Probe Form 2018

E
Configure Physical Hierarchy Window 2021
CPH Menu Bar 2024
CPH Toolbar 2025
Hierarchy Configuration Mode in the CPH Window 2026
Component Types Mode in the CPH Window 2035
Soft Block Mode in the CPH Window 2041
. 2048

F
Library and Attributes Mapping File Syntax 2049
LAM File Locations 2050
Basic LAM File Structure 2052
LAM File Logical Section 2054
LAM File Support for SDL Pcell 2061
LAM File Physical Section 2063
LAM File Component Type Section 2065
LAM File Mapping Section 2080
Wildcard Support for Cell-Level Mapping 2086
Sample LAM File 2089

© 2023 Cadence Design Systems, Inc.

G

Layout XL Assistants

Layout XL Assistants 2095
Annotation Browser Assistant 2096
Annotation Browser Toolbar 2098
Annotation Browser Tabs 2106
Annotation Browser Pane 2112
Annotation Browser Description Pane 2117
Annotation Browser Shortcut Menus 2119
Navigator 2125
Schematic Assistant 2127

© 2023 Cadence Design Systems, Inc.

1

Technology File Requirements for Layout XL



Virtuoso® Layout Suite XL layout editor (Layout XL) is a connectivity-based editing tool that automates each stage of the design flow, from component generation through automatic and interactive routing.

When used as part of an automated custom physical design methodology, Layout XL lets you generate a custom layout from a schematic and edit existing layouts that have defined connectivity. It continuously monitors connections of components in the layout and compares them with connections in the schematic. You can use Layout XL to view incomplete nets, shorts, invalid connections, and overlaps to help you wire your design.

Prerequisites

This documentation is aimed at developers and designers of integrated circuits and assumes that you are familiar with:

- The Virtuoso Studio design environment and application infrastructure mechanisms supporting consistent operations between all Cadence tools.
- The applications for designing and developing integrated circuits in the Virtuoso Studio design environment, notably the Virtuoso Layout Suite L layout editor.
- Virtuoso technology data.
- Component description format (CDF), which lets you create and describe your own components for use with Layout XL.

© 2023 Cadence Design Systems, Inc.

The Layout XL and other Cadence® layout applications require technology-specific information about your design to be stored in a technology file for the design library.

The Virtuoso technology file comprises the following sections:

Layout Editor Rules	The `leRules` section of the technology file specifies the layers that are listed in the Layer Selection Window (LSW) assistant and the order in which they are listed.
Layer Definitions	The `layerDefinitions` section of the technology file defines the layers used in the current technology. All the layers used in your design must be defined in this section.

Layer Rules	The `layerRules` section carries the `functions` table, which specifies the layers used in the current technology and rules such as the function of each layer and its `maskNumber`.
	Here is an example:
	`functions(`
	` ;( layer function [maskNumber] )`
	` ;( ----- -------- ---------- )`
	`  ( NWELL "nwell" 1 )`
	`  ( PWELL "pwell" 2 )`
	`  ( DIFF "diff" 3 )`
	`  ( NIMP "nplus" 6 )`
	`  ( PIMP "pplus" 7 )`
	`  ( DIFF2 "pdiff" 8 )`
	`  ( DIFF "diff" 9 )`
	`  ( POLY "poly" 10 )`
	`  ( LI "li" 11 )`
	`  ( CONT "cut" 12 )`
	`  ( METAL1 "metal" 13 )`
	`  ( VIA1 "cut" 14 )`
	`  ( METAL2 "metal" 15 )`
	`);functions`

Constraint Groups	A constraint group lets you specify the different sets of design constraints to be applied under different design circumstances or for different design objects. This allows the flexibility to experiment with more or less stringent process rules at different stages of the design process. You can group together any set of constraints into a constraint group, which can then be applied to any design associated with the technology library.

© 2023 Cadence Design Systems, Inc.

Devices

The two main types of technology devices used in Layout XL are vias and multipart paths.

Specifying Vias

Vias are required by the *Create – Via* and *Create – Wire* commands and by the *Routing – Start Router* command. Vias are defined by using `standardViaDefs`, which are equivalent to the `symContactDevices` used in previous releases; and `customViaDefs`, which reference a via cellview.

All the vias used in your design must be defined in the `viaDefs` section of the technology file and listed in the `validVias` constraint in the `virtuosoDefaultExtractorSetup` constraint group. Any via cellviews referenced by `customViaDefs` must also exist.



Tip

For information about how via definitions are used to derive the connectivity rules for Layout XL, see Connectivity Rules.

Specifying Multipart Paths

Multipart path templates are defined in the `multipartPathTemplates` section in the technology file.

Physical Rules

Physical rules are specified in the `foundry` constraint group.

The minimum requirement for Layout XL is the `minWidth` constraint which is used by the *Create – Shape – Path* and *Create – Wire* commands to determine the minimum width of objects and the default width of a wire.

© 2023 Cadence Design Systems, Inc.

	For example,
	`constraintGroups(`
	` ; ( group [override] )`
	` ; ( ----- -------- )`
	`  ( "foundry"`
	`  ; physical constraints`
	`  spacings(`
	`  ;( constraint layer1 layer2 value )`
	`  ;( ---------- ------ ------ ----- )`
	`  ( minWidth METAL1 0.02`
	`  );spacings`
	`  );foundry`
	`);constraintGroups`

Connectivity Rules	The connectivity stack is a representation of the extractable layers and the associated electrical connections used by the Layout XL connectivity extractor.

Related Topics

Constraint Groups and Constraints

Virtuoso Technology Data ASCII Files Reference

Virtuoso Technology Data Constraint Reference

Virtuoso Technology Data SKILL Reference

Virtuoso Layout Suite SKILL Reference

Prerequisites for Using the Layout XL Connectivity Extractor

Prerequisites for Using the Layout XL Connectivity Extractor

If you are using the Layout XL Connectivity Extractor, you must define a `virtuosoDefaultExtractorSetup` constraint group. For example,

```
constraintGroups(
 ; ( group [override] )
 ; ( ----- -------- )
  ( "virtuosoDefaultExtractorSetup" nil
  ;  layer constraints
     interconnect(
     ( validLayers    ( Nwell Oxide Nimp Pimp Poly Cont Metal1 Via1 Metal12
     Via2 Metal3 )   )
     );interconnect
    );virtuosoDefaultExtractorSetup
);constraintGroups
```


© 2023 Cadence Design Systems, Inc.

Layout XL issues message LX-2063 if the constraint group is not defined appropriately.

Related Topics

Connectivity Rules

Prerequisites for Using Virtuoso Photonics Solution (Virtuoso Photonics Option)

If you are using the Virtuoso Photonics Solution, you can update the technology file to define an additional constraint group:

If specified, the `virtuosoDefaultOpticalPinSetup` constraint group defines the optical layers to use for generating and updating optical pins.

The `opticalPinSetup` environment variable controls the name of the constraint group to use for determining the optical layers available for pin generation.

You can also define the `virtuosoDefaultElectricalPinSetup` constraint group to specify the electrical layers to use for generating and updating electrical pins. To control the name of the constraint group to use for determining the electrical layers available for pin generation, use the `electricalPinSetup` environment variable.

If the `virtuosoDefaultOpticalPinSetup` or the `virtuosoDefaultElectricalPinSetup` constraint group is not defined, the `validLayers` defined in the default Layout XL constraint group, `virtuosoDefaultSetup`, are used for pin generation.

Related Topics

opticalPinSetup (Virtuoso Photonics Option)

electricalPinSetup

© 2023 Cadence Design Systems, Inc.

Connectivity Rules

The connectivity stack is derived from the following sections of the technology file.

Technology section	For information, see...
`functions`	Specifies the layer functions and assigns mask numbers to the layers. The functions table is defined in the `layerRules` section of the technology file.
`validLayers`	Specifies the layers that you want to extract the connectivity through. If the `excludeLPPs` constraint is used to specify the layer-purpose pairs to exclude from consideration, then a layer-purpose pair is excluded even if it is listed in `validLayers`.
`validPurposes`	Specify the valid layer purpose pairs.
`viaDefs`	Defines both standard and custom vias in the `ViaDefs` section of the technology file.
`equivalentLayers`	The `equivalentLayers` rule is used by the connectivity extractor mainly for legacy purpose. It allows two layers to be declared as strictly identical in terms of connectivity. For example, two layers that correspond to two different voltages of a metal can be declared as equivalent using the definition: `equivalentLayers( ((M1a drawing) (M1b drawing) M1c) )` This indicates that `(M1a drawing)`, (`M1b drawing)`, and all the purposes of `M1c` are in fact the same metal. The `equivalentLayers` rule does not support derived layers. If a derived layer is included in the rule definition, a warning message is issued.
`validVias`	Cadence recommends not to define a `validVias` constraint in `virtuosoDefaultExtractorConstraintGroup` as it restricts the via definitions, which are parsed by the extractor to build the connectivity stack. However, in some technologies, such as those containing legacy via definitions, defining the `validVias` constraint can be useful.

© 2023 Cadence Design Systems, Inc.

To view information about the connectivity stack of a design, see the Diagnostics Report, which is available through the *Connectivity* tab of the Connectivity form.

The default constraint group that is read by the extractor for information about the extractable layers and connections defined using the connectivity stack is `virtuosoDefaultExtractorSetup`.

If required, you can change the default constraint group by setting the environment variable, "`layout`" "`setupConstraintGroup`".

Related Topics

Connectivity Form

© 2023 Cadence Design Systems, Inc.

validLayers

Specifies the layers that you want to extract the connectivity through. The layers can be *physical*, such as metal, poly, and so on, or *derived*, such as those used to define stop layers and bulk layers.

Each physical layer must be defined in the `layerRules` section of the technology file, whereas each derived layer must be defined in the `techDerivedLayers` section.

Let us consider an example of a `validLayers` constraint definition that indicates that all the purposes of layers `M1, V1, M2, V2`, and `M3` are extractable.

```
(validLayers (M1 V1 M2 V2 M3))
```

Cadence recommends that you always provide an exhaustive list of extractable layers—metal, poly, diffusion, implant, well, cut, and recognition layers—in the `validLayers` constraint. If there are no extractable layers defined in the constraint group, Layout XL is launched with the connectivity extractor disabled.

Related Topics

validLayers – Layer-Purpose Pairs Specification

validLayers – Stop Layers Specification

validLayers – Enclosed Stop Layers Specification

validLayers – Color-Dependent Stopped and Stop Layers Specification

validLayers – Bulk Area Layers Specification

validLayers – Substrate Layer Specification

validLayers – Layer-Purpose Pairs Specification

validLayers – Layer-Purpose Pairs Specification

By default, all purposes of a layer specified in the `validLayers` constraint are extractable. However, a constraint group can have multiple purposes defined for a layer, forming multiple layer- purpose pairs. You can directly add layer-purpose pairs in the `validLayers` constraint.

© 2023 Cadence Design Systems, Inc.

For example, the constraint definition `validLayers((METAL1 drawing) METAL2)` indicates that the only extractable purpose of `METAL1` is "`drawing`" whereas for `METAL2`, all the purposes are extractable.

If the `validLayers` constraint definition has both (`METAL1 drawing`) and `METAL1` specified, the specification limiting `METAL1` to `drawing` purpose is ignored and all the purposes of `METAL1` are considered extractable.

Depending on your requirements, you can set the `validLayers` constraint on a net or on a design. If you add the constraint in the schematic constraint editor, you can define different `validLayers` for a specified net.

If you set the constraint at the design level, you can restrict the pin layers further than the extract layer level set for Layout XL. This can restrict the pin layers available for creating pins but will continue to extract the design up to the specified extraction level.

Related Topics

validLayers

validLayers – Enclosed Stop Layers Specification

validLayers – Color-Dependent Stopped and Stop Layers Specification

validLayers – Bulk Area Layers Specification

validLayers – Substrate Layer Specification

validLayers – Stop Layers Specification

You can also use the `validLayers` constraint to specify the layers that stop another layer, such as when a transistor is implemented as a single piece of diffusion cut into two parts by a poly shape. The two parts in this case, form the drain and the source of a MOS device and the poly shape forms the transistor gate.

A stop layer is defined using a derived layer with a `NOT` operation between two layers.

For accurate connectivity extraction, if the `validLayers` constraint contains a poly and a diffusion layer, Cadence recommends that you check that the poly is intended to stop the diffusion and the derived layers are appropriately defined in the `validLayers` constraint. If the constraint is not appropriately defined in the technology file, a warning message is issued and the extractor may report false shorts.

© 2023 Cadence Design Systems, Inc.

Consider the following derived layer definitions in the `validLayers` constraint:

```
(StoppedLayer 10001 (Oxide 'not Poly))
(validLayers (StoppedLayer))
```

The constraint definition indicates that the:

- `Poly` and `Oxide` are extractable layers.
- `Oxide` is being stopped by `Poly`.

Consider another example where a layer-purpose pair “stops” another purpose pair on the same layer:

```
(D1 10001 (Metal 'select slot))
(StoppedLayer 10002 (Metal 'not D1))
(validLayers (StoppedLayer))
```

The constraint definition indicates that the:

- All purposes of `Metal1` are extractable.
- `(Metal1 slot)` “stops” all other purposes of `Metal1`

In addition to enabling poly or metal shapes be stopped by shapes on another valid layer, Layout XL supports that standalone cut shapes or cut shapes inside vias be stopped by another shape. For this, the cut layer and the stop layer should be defined in the `validLayers` constraint.

Consider the following derived layer definitions in the `validLayers` constraint:

```
(StoppedVia1 10040 ( Via1 'not StopVia1 ))
(validLayers (StoppedVia1))
```

The constraint definition indicates that `Via1` is being stopped by `StopVia1`.



Tip

For information on how stop layers can be set at the global level so that all the layout users are benefited by the global settings, see Issues Faced when Setting up the PDK to Improve Layout XL Compliance.

Related Topics

Support for Cut Shape Trimming during Extraction

validLayers

© 2023 Cadence Design Systems, Inc.

validLayers – Layer-Purpose Pairs Specification

validLayers – Enclosed Stop Layers Specification

validLayers – Color-Dependent Stopped and Stop Layers Specification

validLayers – Bulk Area Layers Specification

validLayers – Substrate Layer Specification

validLayers – Enclosed Stop Layers Specification

You can also use the `validLayers` constraint to specify that the areas defined by a `NOT` operation should be enclosed by a third layer. To do this, you must add a constraint to the stop layer using the `INSIDE` operation.

Consider the following example:

```
(Stopped 10001 (substrate 'not Nwell))
(StoppedWithInsideConstraint 10002 (Stopped 'inside Nburied))
(validLayers (StoppedWithInsideConstraint))
```

The constraint definition indicates that the:

- `Substrate`, `Nwell`, and `Nburied` are extractable layers
- Substrate is being stopped by the `Nwell` layer
- Stopped regions of the substrate must be enclosed within the specified `Nburied` layer to be electrically insulated from the rest of the substrate

There can be several `INSIDE` layers for a specified pair of stop layer and stopped layer. For example:

```
(Stopped 10001 (substrate 'not Nwell))
(StoppedWithInsideConstraint1 10002 (Stopped 'inside Nburied1))
(StoppedWithInsideConstraint2 10002 (Stopped 'inside Nburied2))
(validLayers (StoppedWithInsideConstraint1 StoppedWithInsideConstraint2))
```

This indicates that the stopped regions of the substrate must be enclosed within one of the specified `Nburied` layers to be electrically insulated from the rest of the substrate.

Related Topics

validLayers

© 2023 Cadence Design Systems, Inc.

validLayers – Layer-Purpose Pairs Specification

validLayers – Stop Layers Specification

validLayers – Color-Dependent Stopped and Stop Layers Specification

validLayers – Bulk Area Layers Specification

validLayers – Substrate Layer Specification

validLayers – Color-Dependent Stopped and Stop Layers Specification

To enable defining color-dependent stopped and stop layers, the technology file supports a `color` operator.

Consider the following derived layer definitions:

```
techDerivedLayers(
  ( D1                1000(M1 'color mask1Color))
  ( M1InterConnect    1001(D1 'not CM1))
)
validLayers( (M1InterConnect) )
```

In the technology file definition of a color-dependent derived layer:

- The first operand is a layer, such as `M1`, in the example above.
- The second operand is one of the following:
  - `mask1Color`
  - `mask2Color`
  - `grayColor` (no color specified)
  - `anyColor` (color independent)

  In the example above, the second operand is `mask1Color`.
- The third operand is optional and it applies only to *locked* or *unlocked* shapes.

In the example above, adding the derived layer `M1InterConnect` in the `validLayers` constraint indicates that the `M1` shapes on `mask1` are stopped by `CM1` shapes. Because the third operand is not added, there is no restriction about the lock status of the shapes.

Let us consider another derived layer definition:

```
techDerivedLayers(
```


© 2023 Cadence Design Systems, Inc.

```
    ( D1                1000(M1 'color mask2Color) 'locked)
    ( M1InterConnect    1001(D1 'not CM1))
  )
  validLayers( (M1InterConnect) )
```

Here, the derived layer indicates that only *locked* `M1` shapes on `mask2` are stopped by the `CM1` shapes.

Now, let us consider another example:

```
D1 1000 (M1 'color mask1Color) 'locked)
D2 1001 (CM1 'color mask1Color) 'locked)
D3 1002 (D1 'not D2)
```

Here, the derived layer indicates that locked `M1` shapes on `mask1` are stopped by the locked `CM1` shapes on `mask1`.

Related Topics

validLayers

validLayers – Layer-Purpose Pairs Specification

validLayers – Stop Layers Specification

validLayers – Enclosed Stop Layers Specification

validLayers – Bulk Area Layers Specification

validLayers – Substrate Layer Specification

validLayers – Bulk Area Layers Specification

You can use the `validLayers` constraint to define the area of a shapeless bulk terminal. The area of a shapeless bulk terminal is determined using an `AND` operation between the shapes on one layer with the shapes on another layer.

Consider the following `validLayers` constraint definition:

```
(BulkLayer 10001 (Oxyde 'and Poly))
(validLayers (BulkLayer))
```

The constraint definition indicates that:

- All purposes of `Poly` and `Oxide` are extractable

© 2023 Cadence Design Systems, Inc.

- Intersections of `Poly` and `Oxide` define the area of the shapeless bulk terminals.

Related Topics

validLayers

validLayers – Layer-Purpose Pairs Specification

validLayers – Stop Layers Specification

validLayers – Enclosed Stop Layers Specification

validLayers – Color-Dependent Stopped and Stop Layers Specification

validLayers – Substrate Layer Specification

validLayers – Substrate Layer Specification

You can use the `validLayers` constraint to define the substrate as extractable when the *Verify connections to substrate and set well layers as soft* option in the Connectivity form – Connectivity – Verify tab is enabled.

Consider the following derived layer definitions in the `validLayers` constraint:

```
(StoppedSubstrate 10000 (substrate 'not Nwell))
(validLayers (StoppedSubstrate))
```

The constraint definition indicates that the:

- `Nwell` and `substrate` are extractable layers
- `substrate` is being stopped by `Nwell`
- The polarity of the `substrate` is p-type since it is stopped by an n-type well layer

When the *Verify connections to substrate and set well layers as soft* option is enabled:

- `substrate` should be stopped by a well layer. If that is not the case, a warning message is issued and substrate connectivity extraction is disabled.
- `substrate` should be stopped by a single well layer. If the substrate is stopped by several well layers of different polarities, a warning message is issued and substrate connectivity extraction is disabled.
- To derive the polarity of the substrate, the extractor uses the `validLayers` constraint definition, which indicates that the substrate should be considered p-type if the stopping

© 2023 Cadence Design Systems, Inc.

well layer is n-type and vice versa. If the substrate is not defined as stopped by a well layer, the extractor considers the standard via definitions in `standardViaDefs` where the `well` or `substrate` is set as `substrate`. For more information, see Substrate Layer Connectivity.

Related Topics

validLayers

validLayers – Layer-Purpose Pairs Specification

validLayers – Stop Layers Specification

validLayers – Enclosed Stop Layers Specification

validLayers – Color-Dependent Stopped and Stop Layers Specification

validLayers – Bulk Area Layers Specification

Derived Layers

A derived layer is composed as the result of a binary operation between two other layers that may be derived or physical. The layers used as operands for the binary operation must be of a lower layer number compared to the layer to be composed. The SELECT operation is the only operation supported by the extractor where the second operand is a “purpose” instead of a “layer”. If a derived layer with an unsupported operation is processed by the extractor, a warning message is issued.

Consider the following definition in the `techDerivedLayers` section of a technology file.

```
(D1 10001 (M1 'select drawing))
(D2 10002 (M2 'select pin))
(D3 10003 (M2 'select drawing))
(D4 10004 (D2 'or D3))
(D5 10005 (D4 'or D1))
(validLayers (D5))
```

Here:

- `D5` represents the list of physical layers: `(M1 drawing)`, `(M2 pin)`, and `(M2 drawing)`. These physical layers are extractable because `D5` is defined in the `validLayers` constraint.
- `D4`, `D3`, `D2`, and `D1` are “intermediate derived layers”

© 2023 Cadence Design Systems, Inc.

- Top and intermediate derived layers only support the `OR` or `SELECT` operations.
- `AND`, `NOT`, and `INSIDE` are supported as top derived layers.
- `GROW`, `AND`, and `COLOR` are supported in intermediate derived layers for the operands of `NOT`.
- The first operand of `INSIDE` can only be a `NOT` derived layer. The second operand of `INSIDE` can only be a physical layer or a `SELECT` derived layer.
- The first operand of `COLOR` can only be a physical layer.

Related Topics

dbSetSoftConnectTermPinlessLayer

validPurposes

If you are using the Layout XL Connectivity Extractor, you can define the valid layer purpose pairs using the `validPurposes` constraint that the connectivity extractor should consider. For example:

```
constraintGroups(
 ; ( group                              [override] )
 ; ( -----                               --------  )
   ( "virtuosoDefaultExtractorSetup"     nil
   ;  layer constraints
      interconnect(
      ( validLayers("Metal12" "Poly")
      ( validPurposes 'include ("drawing" "pin" "net") )
      );interconnect
     );virtuosoDefaultExtractorSetup
);constraintGroups
```

Here, only the following layer-purpose pairs (LPP) are considered extractable:

- `Metal12 drawing`
- `Metal12 pin`
- `Metal12 net`
- `Poly drawing`
- `Poly pin`
- `Poly net`

© 2023 Cadence Design Systems, Inc.

You can also use the `'exclude` option to exclude an LPP from the valid layer purpose list, as illustrated in the example below:

```
constraintGroups(
 ; ( group                              [override] )
 ; ( -----                               --------  )
   ( "virtuosoDefaultExtractorSetup"     nil
   ;  layer constraints
      interconnect(
      ( validLayers("Metal12" "Poly")
      ( validPurposes 'exclude ("blockage" "boundary") )
      );interconnect
     );virtuosoDefaultExtractorSetup
);constraintGroups
```

For layers "`Metal12`" and "`Poly`", all purposes other than `"blockage"` and `"boundary"` are extractable .

Instead of using the `'exclude` option, you can also use the `excludeLPPs` constraint to specify the layer purpose pairs to be ignored for extraction.

Related Topics

`validPurposes`

`excludeLPPs`

viaDefs

Vias—both standard and custom—are defined using the `ViaDefs` section of the technology file.

The connectivity extractor derives the following information from via definitions.

- Via Layer Connectivity
- Substrate Layer Connectivity
- Implant Layer Connectivity

Via Layer Connectivity

Via layer connectivity is defined in terms of:

- Cut Layer Connectivity
- Local Interconnect Connectivity

© 2023 Cadence Design Systems, Inc.

Cut Layer Connectivity

Cut layers always connect to the *associated pair of layers*. For example, for the following via definition:

```
(M2_M1 M2 M1 ("V1" 0.1 0.1) (1 1 (0.1 0.1))
(0.04 0.0) (0.02 0.02) (0.0 0.0) (0.0 0.0) (0.0 0.0))
```

Layout XL derives the following connectivity stack:

```
M2 -> V1 -> M1
```

Local Interconnect Connectivity

Local interconnect layers, which are contact-less metal layers, connect directly to the *associated layer*. For example, for the following via definition:

```
(LI_POLY LI POLY ("LI" 0.1 0.1) (1 1 (0.1 0.1))
(0.04 0.0) (0.02 0.02) (0.0 0.0) (0.0 0.0) (0.0 0.0))
```

Layout XL derives the following connectivity stack:

```
LI -> POLY
```

In this example, the via definition, `LI_POLY` is included in the technology file to properly define the connectivity stack.

Substrate Layer Connectivity

A substrate via can be defined using the `well/substrate` construct in the `standardViaDefinition`, such as in the example below:

```
standardViaDefs(
;(viaDefName layer1 layer2 (cutLayer cutWidth cutHeight
;[resistancePerCut])
;(cutRows cutCol (cutSpace))
;(layer1Enc) (layer2Enc);(layer1Offset) (layer2Offset) (origOffset)
;[implant1 (implant1Enc) [implant2 (implant2Enc)]])
;[well/substrate]]])
);standardViaDefs
```

The `well/substrate` construct can be defined using the keyword `substrate` or a physical well layer name to allow the connectivity extractor to connect to these using the diffusion defined in the `viaDef`. When the function of the diffusion is defined as `diff` instead of `ndiff` or `pdiff`, the connection is made through the `implant` defined in the `viaDef`.

© 2023 Cadence Design Systems, Inc.

Implant Layer Connectivity

Implant layers are currently derived only from `standardViaDefs`; for example,

```
standardViaDefs(
 (M1_NWELL Oxide Metal1 ("Cont" 0.12 0.12)
 (1 1 (0.16 0.16))
 (0.07 0.07) (0.06 0.06) (0.0 0.0) (0.0 0.0) (0.0 0.0)
 Nimp (0.15 0.15) Nwell (0.24 0.24)S)
);standardViaDefs
```

The implant layers are `Nimp` and `Nwell`. Often the second implant is used to target a well layer, not as a real implant definition. You can also use a `well/substrate` construct in such situations to avoid using the second implant. For more information on defining a `well/substrate` construct, see Substrate Layer Connectivity.

In the example shown above,

- The implant `Nimp` gives the `Oxide` diffusion layer its type (n). Layout XL derives the following layer interconnection stack.

  ```
  Metal1 -> Cont -> ( Oxide & Nimp ) -> Nwell (or N-substrate)
  ```

  The `&` relationship shown above means that `Oxide` connects to the well only if there is some implant around that particular diffusion layer. A via of this type allows Layout XL to propagate the correct connectivity to the substrate of a MOS device from a top-level power or ground `Metal1`.

- Similarly, a via definition with an implant of p-type material might lead to the following layer interconnection stack.

  ```
  Metal1 -> Cont -> ( Oxide & Pimp ) -> Pwell (or P-substrate)
  ```

  Here the oxide shape is implanted with a p-type implant shape to connect to a p-type well shape.

In both the cases, a cut shape will connect to an oxide shape only if there is an implant shape of n-type or a p-type available.

Note: Some manufacturing processes define diffusion layers that are implicitly implanted by an n-type or p-type material. This is the case when the layer function of the diffusion layer is set to `ndiff` or `pdiff` in the `functions` table in the technology file. In this case, no implant shapes are required. The foundry uses the diffusion layer to create the masks both for diffusion and for n- or p-type implants. The layer interconnection stack is typically defined as:

```
Metal1 -> Cont -> Ndiff -> Nwell or N-substrate
Metal1 -> Cont -> Pdiff -> Pwell or P-substrate
```

Note: Some manufacturing processes define only one diffusion layer and one implant layer, for example a, n-type diffusion and a p-type implant. In that case, the implant layer has precedence over the implicitly implanted diffusion:

© 2023 Cadence Design Systems, Inc.

```
Metal1 -> Cont -> Ndiff -> Nwell or N-substrate
Metal1 -> Cont -> ( Ndiff & Pimp ) -> Pwell or P-substrate
```

Related Topics

`extractConsiderAllCutLayersInVias`

Parallel Connectivity Stacks

A parallel connectivity stack is formed when the same cut layer connects a top layer—a layer with a higher mask number—to various bottom layers, which are layers with lower mask numbers.

In the technology file, a parallel connectivity stack is defined using several `viaDefs` that have the same cut layer but different bottom layers.

A parallel connectivity stack can be represented as:

```
via2: Metal2 < MCBOT < MCTOP
```

To view information about the parallel connectivity stacks in a design, you can access the *Diagnostics Report*, which is available through the *Connectivity* tab of the Connectivity form.

Note: A `viaDef` contains information about the following:

- Top connecting layer
- Cut layer
- Bottom connecting layer

Depending on the material to which a cut layer connects, a parallel connectivity stack can be formed by:

- **Metal layers**
  - Metal-Insulator-Metal Capacitor (MIMCAP) Stack
  - Two-Metal Stack
- **Poly layers**
  - Poly-Insulator-Poly Capacitor (PIPCAP) Stack

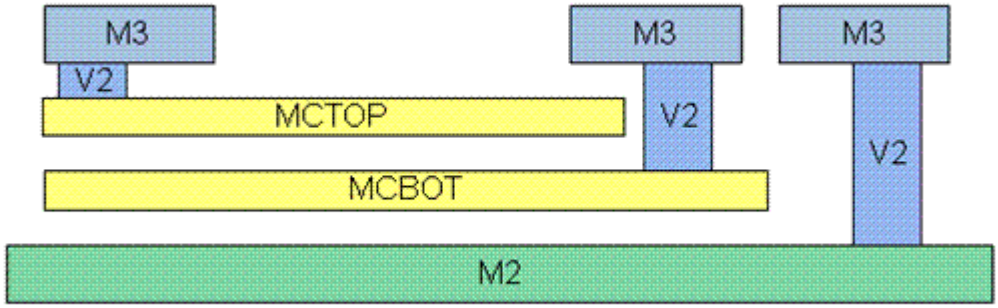
© 2023 Cadence Design Systems, Inc.

Metal-Insulator-Metal Capacitor (MIMCAP) Stack

A Metal-Insulator-Metal Capacitor (MIMCAP) stack uses a cut layer that connects a metal layer of a higher mask number, such as M(n+1), to another metal layer of a lower mask number, such as M(n), and to two intermediate layers.

The intermediate layers in such a connectivity stack are used to define capacitors and these layers are defined in the technology file with the `mimcap` function.

The figure below represents a MIMCAP connectivity stack. Here, the cut layer, `V2`, connects the metal layer, `M3`, to another metal layer, `M2`, which has a lower mask number. In addition, the cut layer connects the top metal layer to intermediate metal layers, `MCTOP` and `MCBOT`.



MIMCAP Connectivity Stack

In the technology file, the connectivity stack represented above is defined with the `viaDefs`:

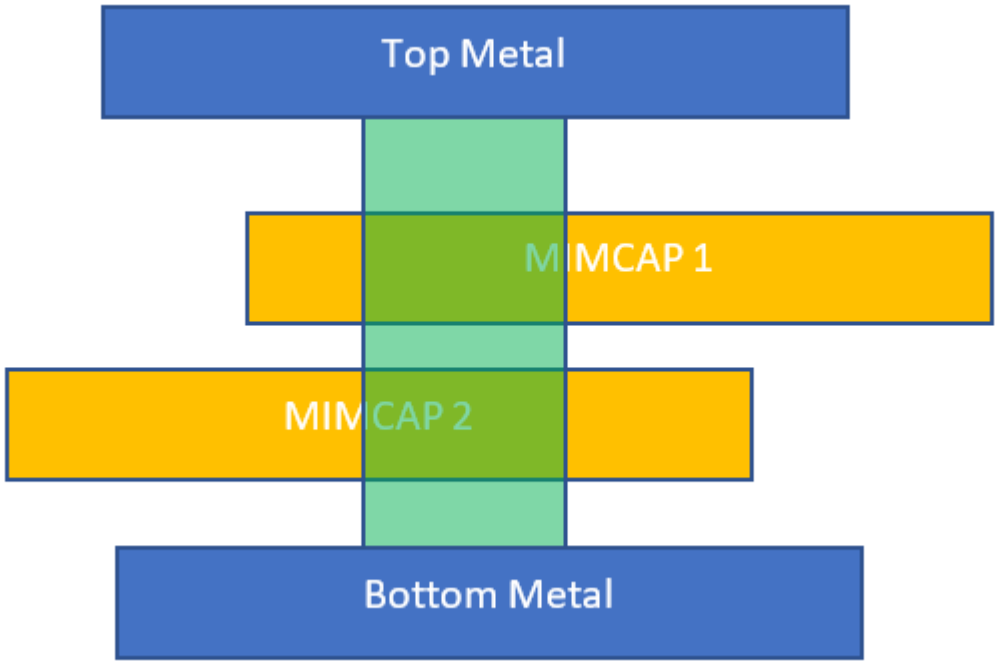
```
"M3_M2":    M3 -> V2 - > M2
"M3_MCTOP": M3 - > V2 - > MCTOP
"M3_MCBOT": M3 - > V2 - > MCBOT
```

The corresponding layer function table can be represented as:

layer	function	mask
M2	metal	9
MCBOT	mimcap	10
MCTOP	mimcap	11
V2	cut	12
M3	metal	13

© 2023 Cadence Design Systems, Inc.

Alternatively, the cut in a MIMCAP stack may not stop at another MIMCAP layer but can go all the way down to the bottom metal layer. This means that all the layers between the top and bottom metal layers are connected by the cut shape, as shown in the figure.



For the connections in such a stacked MIMCAP to be correct, all the `viaDefs` must be present in the `validVias` constraint of the extractor constraint group. For example, for the connections in the given stacked MIMCAP figure to be correct, the following `viaDefs` should be present in the `validVias` example: `Top/MIMCAP1`, `Top/MIMCAP2`, `Top/Bottom`, `MIMCAP1/MIMCAP2`, `MIMCAP1/Bottom`, `MMCAP2/Bottom`.

In the technology file, the `viaDefs` definition of the stacked MIMCAP is:

```
"Top_Mimcap1": Top -> V -> Mimcap1
"Top_Mimcap2": Top -> V -> Mimcap2
"Top_Bottom": Top -> V -> Bottom
"Mimcap1_Mimcap2": Mimcap1 -> V -> Mimcap2
"Mimcap1_Bottom": Mimcap1 -> V -> Bottom
"Mimcap2_Bottom": Mimcap2 -> V -> Bottom
```


© 2023 Cadence Design Systems, Inc.

The corresponding layer function table can be represented as:

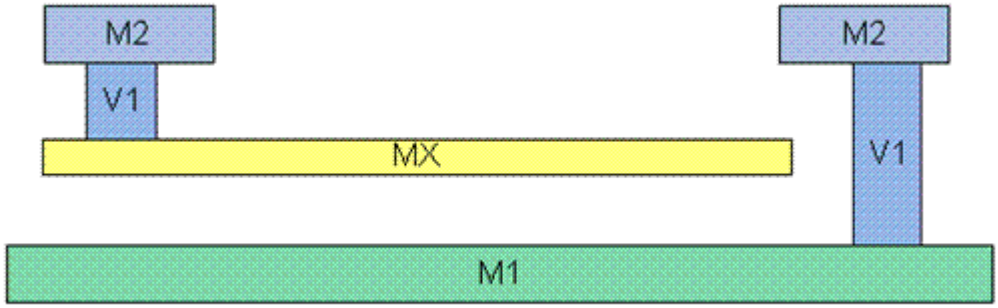
```
layer           function          mask

Bottom          metal             9
Mimcap2         stackedMimcap     10
Mimcap1         stackedMimcap     11
 V              cut               12
Top             metal             13
```

Two-Metal Stack

A “two-metal” stack uses a cut layer that connects a metal layer of a higher mask number, such as M(n+1), to another metal layer of lower mask number, such as M(n), and to a single intermediate metal layer.

The figure below represents a “two-metal” connectivity stack. Here, the cut layer, `V1`, connects the metal layer with a higher mask number, `M2`, to another metal layer, `M1`, which has a lower mask number. In addition, the cut layer connects the top metal layer to an intermediate metal layer, `MX`.



“Two-metal” Stack

In the technology file, the connectivity stack represented above is defined with the `viaDefs`:

```
“M2_M1”:     M2 - > V1 - > M1
“M2_MX”:     M2 - > V1 - > MX
```


© 2023 Cadence Design Systems, Inc.

The corresponding layer function table can be represented as:

```
layer       function        mask

M1          metal           9

MX          metal           10

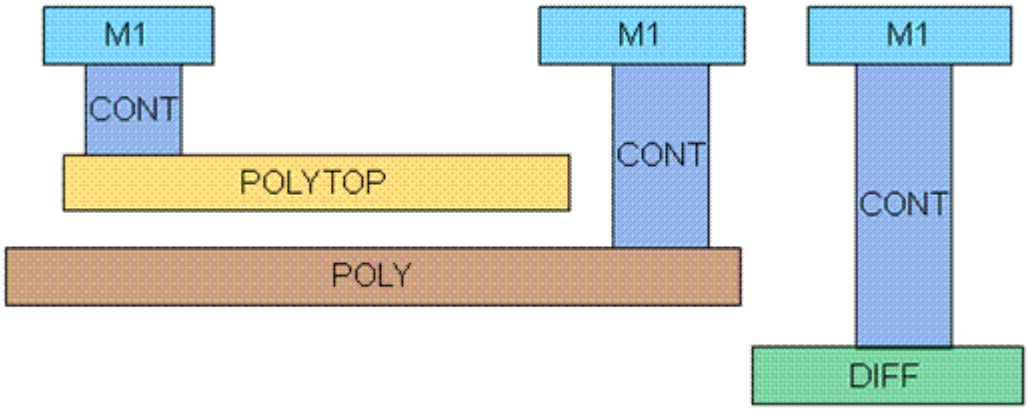
V1          cut             11

M2          metal           12
```

Poly-Insulator-Poly Capacitor (PIPCAP) Stack

A Poly-Insulator-Poly Capacitor (PIPCAP) stack uses a cut layer that connects a metal layer to two poly layers. The poly layer at the bottom is routable, whereas the poly at the top is nonroutable and is used to define capacitors. The technology file defines the top poly layer by using the function, `pipcap`.

The figure below represents a PIPCAP connectivity stack. Here, the cut layer, `CONT`, connects the metal layer, `M1`, to two poly layers, `POLYTOP` and `POLY`.



PIPCAP Connectivity Stack

As displayed in the figure, the cut layer can also connect the metal layer, `M1`, to the diffusion layer, `DIFF`. To ensure that the top poly layer does not overlap diffusion, the top poly layer must be enclosed within the bottom poly.

In the technology file, the connectivity stack represented above is defined with the `viaDefs`:

```
"M1_DIFF"    :    M1 - > CONT - > DIFF
```


© 2023 Cadence Design Systems, Inc.

```
"M1_POLYTOP":   M1 - > CONT - > POLYTOP
"M1_POLY"   :   M1 - > CONT - > POLY
```

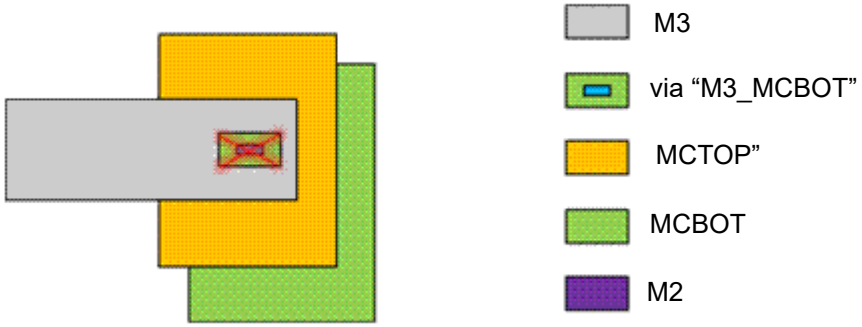
The corresponding layer function table can be represented as:

layer	*function*	*mask*
DIFF	ndiff	2
POLY	poly	3
POLYTOP	pipcap	4
CONT	cut	5
M1	metal	6

Illegal Via Connections

The extractor checks that the top and bottom layers of a via are effectively connected. If a violation is identified, the extractor creates an "Illegal via connection" marker on the via.

Let us consider a MIMCAP parallel connectivity stack as displayed in the figure below. Here, the metal layer `M3` is connected to metal layers, `MCTOP`, `MCBOT`, and `M2` through the same cut layer, `V2`.



The `viaDefs` for the parallel connectivity stack are defined in the technology file as:

```
"M3_M2" : M3 -> V2 - > M2
"M3_MCTOP": M3 - > V2 - > MCTOP
"M3_MCBOT": M3 - > V2 - > MCBOT
```

In the figure, the via is expected to connect the shape on `M3` to a shape on `MCBOT`. But, a shape on `MCTOP` overlaps the via, preventing the connection. However, due to the overlap with the shape on `MCTOP`, the via effectively connects the shape on `M3` to the shape on `MCTOP`. As this was not the expected connection, the extractor creates an “illegal via connection” marker on the via.

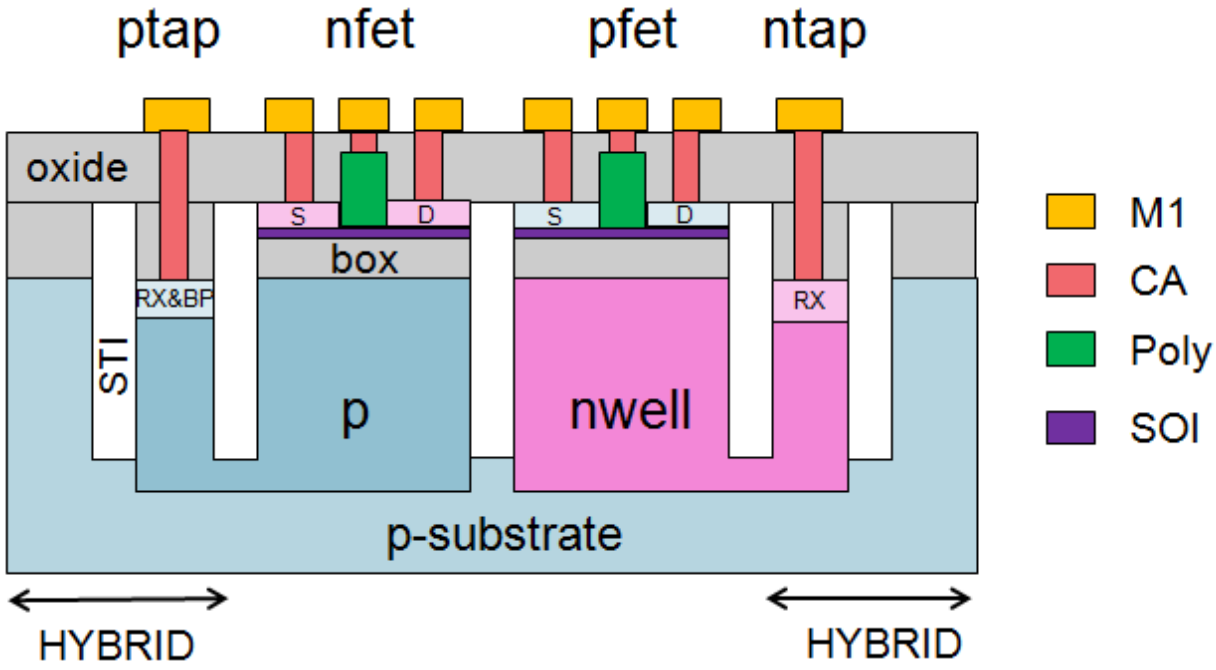
© 2023 Cadence Design Systems, Inc.

Related Topics

Connectivity Form

FDSOI Connectivity Stack

An FDSOI (Fully Depleted Silicon-On-Insulator) connectivity stack is formed when a thin, fully depleted Si channel (`SOI`) and the diffusion (RX) are insulated from the substrate and wells by a buried oxide (`box`), as shown in the figure below.



The `HYBRID` layer defines the area where `SOI` and `box` are removed such that the diffusion directly connects to the substrate or wells, as shown in the figure above. The wells are laterally insulated by shallow trench isolation (STI), as shown in the figure above.

Therefore, the `HYBRID` layer is defined as a `cut` layer. `viaDefs`, as displayed below, are added to the technology file to define that `RX` can be connected to the well and substrate only through the `HYBRID` layer. In the example below, RX is an n-implanted diffusion (ndiff) and BP is a p-implant (pplus).

```
        layer1     layer2      cut      implant1  implant2  sub/well
M1_RX:   M1          RX         CA
RX_sub:  RX      substrate    HYBRID      BP
RX_NW:   RX        NWELL      HYBRID
```


© 2023 Cadence Design Systems, Inc.

Therefore, the FDSOI connectivity stack can be represented as:

```
M1 -> CA -> RX
(RX & BP) -> HYBRID -> substrate
RX -> HYBRID -> NWELL
```

The corresponding layer function table can be represented as:

Layer	Function	Mask
(substrate)*	p-type substrate	--
(NW)	n-type well	1
(HYBRID)	cut	2
(RX)	n-type diffusion	3
(PC)	poly	4
(BP)	p-type implant	5
(CA)	cut	6
(M1)	metal	7

For routing purposes, the physical via definitions, as defined below, are `M1_sub` and `M1_NW`,.

```
          layer1  layer2   cut   implant1   implant2    sub/well
M1_PC:    M1      PC       CA
M1_sub:   M1      RX       CA    BP         HYBRID      substrate
M1_NW:    M1      RX       CA    NWELL      HYBRID
```

You must set up the `validVias` constraint in the extractor constraint group, as displayed below, such that the extractor excludes the physical viaDefs `M1_sub` and `M1_NW`.

```
("virtuosoDefaultExtractorSetup"nil
  interconnect(
    (validLayers (... M1 CA PC RX BP HYBRID NWELL) )
    ( validVias   (... M1_PC M1_RX RX_sub RX_NW) )
  )
)
```

Similarly, you must set the `validVias` constraint in the wire editing constraint group, as displayed below, to exclude the viaDefs: `M1_RX`, `RX_sub`, and `RX_NW`.

`M1_RX`, `RX_sub`, and `RX_NW` are not physical viaDefs. These are added in the technology file only to properly define the connectivity stack for the connectivity extractor.

```
("virtuosoDefaultSetup"nil
  interconnect(
    (validLayers (... M1 PC) )
    ( validVias   (... M1_PC M1_sub M1_NW) )
  )
```


© 2023 Cadence Design Systems, Inc.

```
)
```

Related Topics

Connectivity Form

Connection Requiring an Extra Layer

A `stdViaDef` is used to define a connectivity model in which an extra layer is required to connect to the diffusion layer.




```
("MD"    "OD"      ("MD")     "NP"      "RECOGNITION"   "NW"       )

(layer1 layer2     cut      implant1   implant2        well/substrate)
```

The extra layer is considered only if it is a recognition layer for a connection between diffusion and well or substrate. In the syntax, the slots `implant1` and `implant2` are interchangeable.

© 2023 Cadence Design Systems, Inc.

2

Connectivity Source Preparation

You need to prepare a schematic connectivity source for your design, which you can then use to generate, place, and route a layout with the Virtuoso® Layout Suite XL layout editor (Layout XL).

- Schematic Design Elements
- Design Variables
- One-to-Many Mapping of Instance or Pin
- Properties in Layout XL
- Pins and Pin Names in Layout XL

Schematic Design Elements

You can use any schematic to generate a layout with Layout XL. However, the process is easier if you:

- Place symbols of devices and pins in the schematic approximately where you want them to be in the layout.

  The *Place As In Schematic* command lets you place generated layout devices and pins in positions that correspond to the positions of their schematic symbols. If your design carries any unbound instances and pins, these are positioned below the PR boundary.

- Use consistent parameter defaults for comparable symbols.

  Layout XL applies the parameters of the symbols to the layout devices.

Important

Layout XL does not pass to the layout instance CDF parameters that are not evaluated (i.e., that are not AEL expressions) and have `cdfParamRec~>storeDefault=nil`. These are master parameters and are

© 2023 Cadence Design Systems, Inc.

expected to match for both schematic and layout master. If you require these parameters to be passed, set `cdfParamRec->storeDefault` to `t` for the schematic master parameters.

When transferring information from the schematic to the layout, Layout XL flattens the schematic (i.e., expands schematic symbols into corresponding devices) if you provide an `lxViewList` or an `lxStopList` to tell Layout XL which view of the lower-level instances to use.

Layout XL also supports:

- Design variables.
- One-to-many mapping.
- Many-to-many and many-to-one binding.

If you are preparing a hierarchical design, you also need to make sure the symbol view of each top-level design element is mapped to the correct layout view of the corresponding layout element for generating the layout.

Related Topics

lxViewList Property

lxStopList Property

Design Variables

One-to-Many Mapping of Instance or Pin

Design Variables

You can use the following design variables with Layout XL.

- Netlist Processor Expressions
- Analog Expression Language Expressions
- Simulation Design Variables.

© 2023 Cadence Design Systems, Inc.

Netlist Processor Expressions

Netlist Processor (NLP) expressions are properties that specify parameter values. These expressions are used by the Open Simulation System (OSS) in netlisting.

Layout XL evaluates CDF parameters that begin with the string '`[@`' as netlist processor (NLP) expressions regardless of the value of the `parseAsCEL` environment variable.

Analog Expression Language Expressions

Analog Expression Language (AEL) expressions, such as `iPar` and `pPar`, define the value of a parameter as a function of other instance parameters or parameters passed from other levels of hierarchy. If you specify the value of a parameter using an AEL expression, the parameter

- Must be defined in the component description format (CDF) for the cell of which the symbol is a view.
- Must be a string for which `parseAsNumber` and `parseAsCEL` properties are set to `t`.
- Must not have a CDF callback (because the evaluation of the expression does not trigger the execution of the callback).

If Layout XL detects a parameter value defined with `iPar`, `pPar`, or other AEL expressions not defined in the CDF, you see a warning in a message box.

Netlisting Mode

To ensure that Layout XL always evaluates CDF parameters correctly, make sure that

- The CDF parameters in question have the *parseAsCEL* option set to *yes*.
- The `CDS_Netlisting_Mode` shell environment variable is set to `Analog` before you launch Layout XL.

You can also set `CDS_Netlisting_Mode` for the current session only. To do this,

➤ Type the following commands in the CIW.

```
setShellEnvVar("CDS_Netlisting_Mode=Analog")

cdsSetNetlistMode()
```

To check which mode is currently set,

➤ Type the following command in the CIW.

© 2023 Cadence Design Systems, Inc.

```
cdsGetNetlistMode()
```

Note: If you have the `CDS_Netlisting_Mode` shell environment variable set to `Digital`, Layout XL will not evaluate the CDF parameters by default. But, you can set the `dbSearchCDF` SKILL function to `t` to cause CDF to be searched for parameter values.

For more information on AEL expressions, see *Scope of Parameters* in the *Virtuoso ADE Explorer User Guide*.

Simulation Design Variables

When you use simulation design variables to specify the value of a parameter in the circuit, Layout XL uses the value last saved during the simulation of the circuit as the value for the layout implementation.

For more information about simulation design variables, see *Design Variables and Simulation* in the *Virtuoso ADE Explorer User Guide*.

Related Topics

Open Simulation System Reference

dbSearchCDF

Customizing the Simulation Environment

Design Constraints

On startup Layout XL transfers all the constraints defined in the schematic view to the layout view, correctly mapping the constraints and their members between the two views.

Constraints in the schematic are transferred to the top-level layout view. Constraints that have been created or changed in the schematic but not yet saved are also transferred.

Device correspondence information is maintained during the transfer. Logical and physical name bindings are tracked and constraints are updated appropriately to take account of folded instances in the layout.

Related Topics

Constraint Transfer During Layout Generation

© 2023 Cadence Design Systems, Inc.

One-to-Many Mapping of Instance or Pin

One-to-many mapping lets you map a single instance or pin in the schematic to multiple instances or pins in the layout. You can implement one-to-many mapping in Layout XL designs by:

- One-to-Many Mapping with Iterated Instances and Bus Pins
- Using the multiplication factor (mfactor Property)
- Using the series-connected factor (sfactor Property)
- Defining a One-to-Many Device Correspondence

One-to-Many Mapping with Iterated Instances and Bus Pins

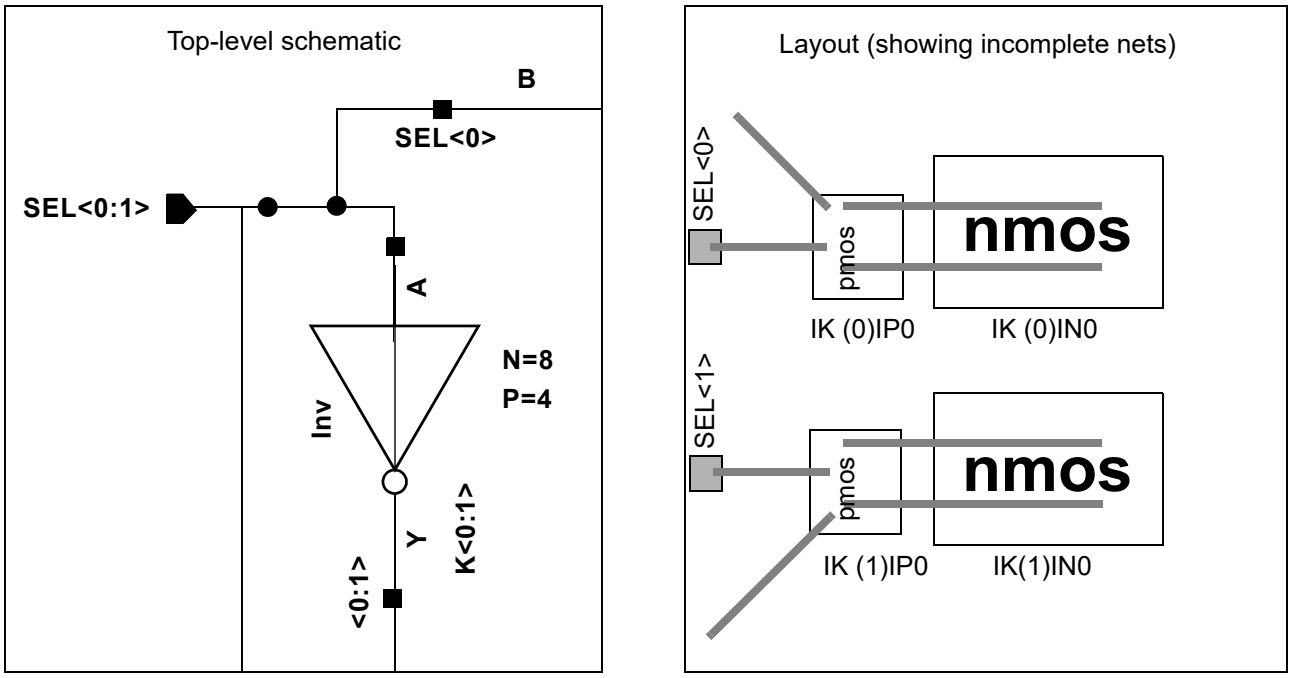
You can use iterated instances and bus pins to enable one-to-many mapping between pins and devices in the schematic and multiple instances of the pins and devices in the layout.

For example, the figure below shows

- Bus pin `SEL<0:1>` in the schematic mapped to bus pins `SEL<0>` and `SEL<1>` in the layout.

© 2023 Cadence Design Systems, Inc.

- Iterated instance `K<0:1>` mapped to instances `IK(0)|P0`, `IK(0)|N0`, `IK(1)|P0`, and `IK(1)|N0` in the layout.



Related Topics

Adding Instances Using an Iterative Expression

Properties in Layout XL

Before the Layout XL layout editor can create a layout from a schematic, you must create a layout device for every symbol in the schematic. The layout master of a device or contact can be a fixed cell, a parameterized cell (Pcell), or a device or contact defined in the technology file.

Note: The default values for any given property must be the same in the schematic and the layout. If the default value of a property differs between the two cellviews, and the property has a value of `storeDefault=nil`, then when you start the *Generate All From Source* command, the layout instance is not updated (i.e., the CDF default of the layout is used). This can lead to a parameter mismatch between the layout and the schematic.

© 2023 Cadence Design Systems, Inc.

Pcells are often the most effective because you can assign the dimensions of the device at the time you generate the layout and vary the sizes of a contact each time you place the cell. For more information about Pcells, see the *Virtuoso Parameterized Cell Reference*.

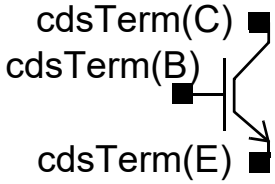
Related Topics

Layout XL Properties

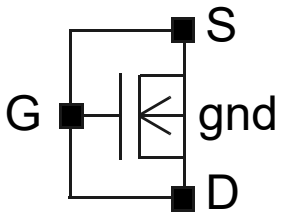
Pins and Pin Names in Layout XL

For connectivity assignment tracing and cross-probing to work correctly in Layout XL, the pins and pin names in the layout cellview of a device must match those in the corresponding schematic symbol.

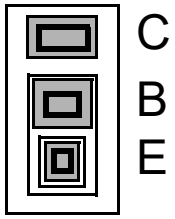
analoglib library style pin symbol



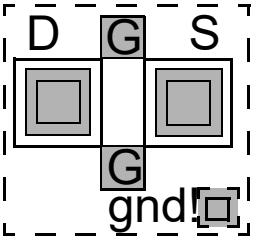
basic library style pin symbol



analoglib library style pin layout



basic library style pin layout



Extra Pins in the Symbol or Layout Views

Layout XL cannot maintain connectivity for any extra pins in the symbol view because there is no corresponding pin for the device in the layout view.

Conversely, Layout XL maintains connectivity for extra pins whose names are global nets (for example, `vdd!`) in the layout view. It also maintains connectivity for any extra pins in the layout that have their connectivity defined by inherited connections. The inherited connection can be defined relative to the layout instance itself or relative to the schematic hierarchy that ends with the schematic instance corresponding to that layout instance.

© 2023 Cadence Design Systems, Inc.

The look-up order used by Layout XL to determine the net to connect to an extra pin is:

1. Inherited net expression on the layout pin.
2. Global net — global attribute or `!` at the end of the net name.
3. Property named `sub`, `sub_inh`, or `bn` on the layout master.
   - Layout XL uses the value of the property as the substrate net name and connects the extra terminal to that net.

Important

Cadence recommends not to place pins where you do not want to make a connection. For example, on a polysilicon layer that covers the gate area of a FET.

Related Topics

Adding a Pin

External Connections

You can also define pins to be connected externally to the design.

The commands in the *Connectivity – Pins* submenu let you connect pins in four different ways.

- *Must Connect* connects selected pins in a net externally at a higher level of the hierarchy.
- *Strongly Connected* connects selected pins within the device. By default, pins are connected internally (strongly).
- *Weakly Connected* connects selected pins in a limited external connection to avoid specific internal connections (typically ones with high-resistance paths).
- *Pseudo Parallel Connect* connects selected instance terminals on the same net within an instance as though they were connected externally; that is, they are defined as a connection but need never be physically connected.

Related Topics

Using Connectivity

© 2023 Cadence Design Systems, Inc.

permuteRule Property

Device Abutment

© 2023 Cadence Design Systems, Inc.

© 2023 Cadence Design Systems, Inc.

3

Physical Hierarchy Configuration

is a physical hierarchy configuration utility that lets you specify how a layout implementation is generated from a specific schematic design.

Note: Alternative to using the CPH window for defining the layout implementation, you can directly import the information from the library and attributes mapping (LAM) file, if defined.

The CPH window has three modes of operation.

Operation Mode	Description
Hierarchy Configuration Mode in CPH	Controls how the physical hierarchy is generated from your logical design, including which logical components are to be ignored in the physical implementation and which physical views are used to implement the logical components.
Component Types Mode in CPH	Lets you view the library-level component types. In addition, this mode lets you create, edit, and remove the design-level component types, which identify the *NMOS* and *PMOS* transistor cells in the design and set the parameters for device chaining and folding. ■ The *Component Types Mode* also lets you identify *NFIN* and *PFIN* devices in the design and set the parameters for device chaining and folding. ■ If the Configure Physical Hierarchy window is not enabled, the component type information is directly read from the `cph.lam` file.

© 2023 Cadence Design Systems, Inc.

Operation Mode	Description
Soft Block Mode	Lets you configure and specify bindings for the soft blocks that will be created by the *Floorplan – Generate Physical Hierarchy* command. You select the instances to be included and define the boundary, pin, and blockage parameters for the block you want to create. You can also remove the parameters from an existing soft block to prevent it from being created when the physical hierarchy is generated. To edit an existing soft block that has already been generated in the layout view, use the *Edit Soft Blocks* command.

The introduction of this functionality means that the layout designer no longer needs to have `write` access to the schematic design to add or modify the properties that drive layout generation. As these properties are now controlled in the Configure Physical Hierarchy Window, a layout designer can define different physical configurations and view different physical implementation possibilities without changing the schematic view.

Related Topics

Library and Attributes Mapping File Syntax

LAM Files and Configuration Views

Data Conversion to Use the Layout XL Schema

Launching Configure Physical Hierarchy

Configure Physical Hierarchy Window

Soft Block Editing

LAM Files and Configuration Views

Underlying the Configure Physical Hierarchy functionality is a Layout XL Schema. It comprises two new data formats.

- For schematic and layout library data, there is a **library and attributes mapping (LAM) file** to store the symbol properties that drive layout generation, component type information for a given library, and library and cell-level mapping information. This file is called `cph.lam` and is stored in the following location:

  library`/.cadence/dfII/cph/lam/cph.lam`

© 2023 Cadence Design Systems, Inc.

There is no graphical user interface to help you create or edit a LAM file.

- For layout design data, there is a **physical configuration (physConfig) view**; which comprises a number of files that store design-specific overrides, including the schematic expansion rules that drive layout generation; design component types; and cell and instance mapping rules. Every Layout XL session operates in the context of a physical configuration view, whether generated automatically or user-specified.

  When you launch Layout XL from a schematic window, the schematic view is re-opened in the context of the physical configuration view being used for the session, which might be different from the configuration the schematic was using previously. The schematic window banner updates to indicate the name of the physical configuration view currently being used.

  One effect of this is that you might need to re-extract the schematic view in order to take into account the settings in the physical configuration view. The system will inform you if this extraction is required.

  If you close the Layout XL session by closing the layout window only, the schematic view is re-opened with no physical configuration context.

Related Topics

Library and Attributes Mapping File Syntax

Data Conversion to Use the Layout XL Schema

This topic explains how you decide whether you need to convert your existing data to use the new Layout XL schema; what happens to the data during the conversion; and how to run the conversion commands.

You need to convert data to the Layout XL schema, if your schematic library or layout design data uses any of the following features:

- Library-level component type information stored in a `data.dm` (or `prop.xx` for earlier releases) file in the library directory, or design-level component type overrides stored as an `lxComponentType` property on the layout view.
- The following properties (or their CDF equivalents) stored on a symbol or schematic view.

  lxMfactorSplit Property
  lxParamsToIgnore Property
  lxParamsToIgnoreForCheck Property

© 2023 Cadence Design Systems, Inc.

lxRemoveDevice Property
lxRounding
lxStopList Property
lxUseCell Property
lxViewList Property

After your data has been converted, if you change the name of an instance, or change it from vectored to non-vectored, then Configure Physical Hierarchy will not be able to find the converted properties for the new instance name.

These properties will not be converted if the variable `CPH_USE_SCHEM_PROP` is enabled.

- A Layout XL map file referenced from a layout view.

Automatic Mode

If you use none of the features described above, then you do not need to convert your data to use physical configuration views.

When you start Layout XL, set the *Configuration* option to *Automatic* in the Startup Option form. Layout XL creates a temporary physical configuration view, handles all operations related to the physical configuration view automatically, and removes the temporary view again when it is no longer required; that is., when you convert your data to use physical configuration views or edit the automatic configuration and save it.

Related Topics

Data Conversion Flow for the Layout XL Schema

Schematic Library Conversion

Layout Design Conversion

How to Control CPH with Schematic Properties

Startup Option Form

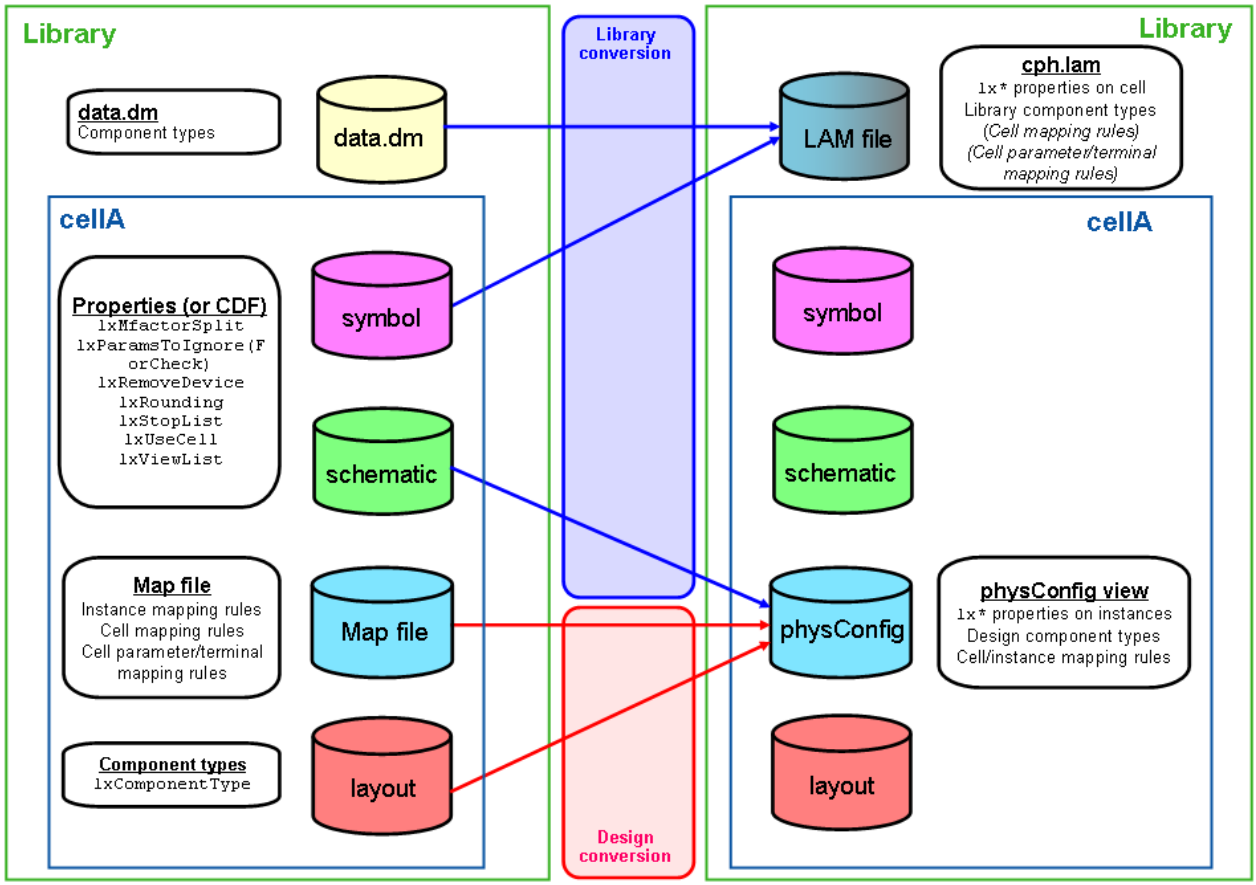
Starting Layout XL from a Physical Configuration View

Troubleshoot Configure Physical Hierarchy

© 2023 Cadence Design Systems, Inc.

Data Conversion Flow for the Layout XL Schema

The figure below shows how legacy data is converted to use the Layout XL schema.



Data is converted in two separate operations.

- A schematic library conversion (indicated by the top three arrows in the picture above).

  You can run it manually using the utility described in Converting Schematic Libraries and Designs.

- A layout design conversion (indicated by the bottom two arrows in the picture above).

  This runs automatically when you load your design into Layout XL, or you can run it manually using the `cphUprevDesign` SKILL function.

Related Topics

Layout Design Conversion

© 2023 Cadence Design Systems, Inc.

cphUprevDesign

Schematic Library Conversion

The schematic library conversion is typically performed by the library designer, who provides a complete library and attributes mapping (LAM) file for the libraries you are using.

The schematic library conversion converts:

- Library-level component types stored in CDF files or (if none exist in CDF) in the `data.dm` file in the library directory.

  This data is converted into a LAM file called *`library`*`/.cadence/dfII/cph/lam/cph.lam`.

  If the LAM file already exists, any new information is appended to the existing file. Where there is a conflict, existing information is overwritten with new information. If there is no data to be converted, then no LAM file is created or updated.

  If the same component type is defined with identical attributes in two different libraries, the definitions are merged into a single definition in the LAM file. All required cells are then assigned to that single component type.

- The properties (or their CDF equivalents) set on a symbol view for a cell.

  This data is also converted into the LAM file under the same conditions as described above. After the conversion, the properties remain on the symbol view but are considered obsolete.

- The properties (or their CDF equivalents) set on a schematic view for a cell.

  The library conversion determines whether a view is a schematic view based on its type, not its name.

  This data is converted into the physical configuration (`physConfig`) view for the cell in question. If the view already exists, any new information is appended to the existing view. Where there is a conflict, existing information is overwritten with new information. If there is no data to be converted, no physical configuration view created or updated.

  After the conversion, the properties remain on the schematic view but are considered obsolete.

You can convert schematic libraries manually using the `cphUprevLibrary` SKILL function.

© 2023 Cadence Design Systems, Inc.

Related Topics

Library and Attributes Mapping File Syntax

Data Conversion Flow for the Layout XL Schema

Layout Design Conversion

cphUprevLibrary

Layout Design Conversion

Layout designs are converted automatically the first time you load the design into Layout XL. The layout design conversion converts:

- Design-level component types stored directly on the layout view using the `lxComponentType` property.

  This data is converted into a physical configuration (`physConfig`) view for the cell in question.

  If the physical configuration view already exists, any new information is appended to the existing view. Where there is a conflict, existing information is overwritten with new information. If there is no data to be converted, no physical configuration view is created or updated.

- The contents of any map file referenced from the layout view.

  The Layout XL map file was used in previous releases to define cell and instance property mapping rules. It also mapped terminal and parameter names in the schematic to their equivalents in the layout. This data is also converted into the physical configuration view under the same conditions as those described for the previous item.

  Note: You can also define these mapping rules at the library level by adding them to the LAM file if required.

  The design conversion does not support the following map file features.

  - `paramSet`, which lets you set defaults for parameter values.
  - Parameters that apply to all defined cells; for example,

    ```
    (paramNameMap ("w2" "w") ("l2" "l"))
    ```

  Warnings are issued if these are found in a map file.

- The design conversion also handles the stopList and viewList environment variables, both of which are considered obsolete after the conversion.

© 2023 Cadence Design Systems, Inc.

The following environment variables remain valid after conversion:

mfactorSplit
paramsToIgnore
paramsToIgnoreForCheck
propsUsedToIgnoreObjs
propsUsedToIgnoreObjsForCheck

When the conversion is complete, a property is stored in the physical configuration view to indicate that the layout design has been converted. This prevents it from being converted needlessly the next time it is opened in Layout XL.

You can check the physical configuration view that is created using the *Launch – Configure Physical Hierarchy* command.

Manual Layout Design Conversion

You can convert a layout design manually using the `cphUprevDesign` SKILL command.

Related Topics

cphUprevDesign

Troubleshoot Configure Physical Hierarchy

Launching Configure Physical Hierarchy

To open the Configure Physical Hierarchy window, do one of the following:

- From the Layout XL (or GXL) window, choose *Launch – Configure Physical Hierarchy*.
- To launch Layout XL (or GXL) from a schematic, check the *Open CPH* option in either the Create Physical Configuration View Form or Open Physical Configuration View Form.
- From the CIW, choose *File – New* or *File – Open* and create or open a cellview of type `physConfig`.

The system checks out a Layout XL license if there is not one currently checked out.

© 2023 Cadence Design Systems, Inc.

If you have a large hierarchical design that takes a long time or fails to open in Layout XL, it might be that the logical elaboration performed by Configure Physical Hierarchy is causing the problem.

You can use the physical stop view list to limit the logical elaboration by switching on the `cphStopLogicalElabAtPhysLeaf` environment variable. This stops the elaboration of the logical design hierarchy when a node is reached which maps to one of the view names specified in the *Physical stop view list* field.

When `cphStopLogicalElabAtPhysLeaf` is set to `t`, global nets below the leaf instance in the schematic are not considered during the elaboration. If your design relies on such nets, Cadence recommends that you do not switch on this environment variable.

Related Topics

Starting the Design Environment for Layout XL

cphStopLogicalElabAtPhysLeaf

© 2023 Cadence Design Systems, Inc.

Configure Physical Hierarchy Window

The Configure Physical Hierarchy window contains a number of different elements.



- **Top Cell:** The Top Cell pane displays the top-level schematic cellview and the corresponding layout cellview to be generated in the context of the current physical configuration. It is a dockable window, which you can dock on either side of the main Configure Physical Hierarchy window, or leave undocked as a floating window.

  You can use the *Open* buttons to open (or raise, if already open) the specified views in the context of the current physical configuration view (which might be different from the configuration the schematic was using previously). If either cellview is not fully specified, this is indicated by the text `<unknown>` in the relevant field and the *Open* button is grayed out.

© 2023 Cadence Design Systems, Inc.

If there are multiple open layout views, all using the same physical configuration view, the Top Cell pane shows the layout view in the last layout window you clicked.

- If there is a layout cellview specified, any changes you make in the Configure Physical Hierarchy window affect the currently active Layout XL session between the layout and schematic views listed.
- If you open the Configure Physical Hierarchy window using the CIW's *File – Open* command or using *File – Open* or *File – New* from another Configure Physical Hierarchy window, the *Physical* cellview is listed as `<unknown>`, and the changes you make do not affect any running Layout XL session.

If the content of the top schematic cellview changes on disk, this typically invalidates the rest of the information shown in the Configure Physical Hierarchy window. To refresh the session, use the dbRefreshCellView SKILL command in the CIW.

- **Global Bindings**: The settings in the Global Bindings pane control how the logical design is traversed and how the logical to physical correspondence for leaf cells (stop points) is made. It is a dockable window, which you can dock on either side of the main Configure Physical Hierarchy window, or leave undocked as a floating window.
  - *Physical library list* is the list of libraries that are searched to find the corresponding physical cell for a given logical cell. By default, the library containing the logical cell is always searched first.
  - *Logical switch view list* specifies the view names that are used to descend into a hierarchical design to find schematic views.
  - *Physical stop view list* specifies the view names that are used to determine the corresponding physical view for a given logical view. When traversing a hierarchy, Configure Physical Hierarchy uses the first view it encounters with one of the specified names.
  - The connectivity extractor does not stop at the stop view but descends into the hierarchy until it reaches the leaf symbol. This ensures that Layout XL finds all the required pins, including global nets, which are sometimes defined on the schematic view rather than on the symbol view.
  - *Physical stop library list* specifies the library names that contain a physical view that can be used to stop CPH from elaborating further during a hierarchical force descend.
  - *Constraint view list* lists the names of the views containing constraint data. This list is passed to the Virtuoso Schematic Editor when it is opened in the context of a physical configuration view, allowing it to determine whether the contents of the Constraint Manager must be updated.

© 2023 Cadence Design Systems, Inc.

- **Table View**: The table view lists the devices in the design in a format appropriate to the mode in which you are operating. It is not dockable.
  - In Hierarchy Configuration mode you can choose between an *Instances* view and a *Cells* view. The former shows a hierarchical representation of the logical design, that is, the schematic top-level cell – and how each instance in the logical design is mapped to a layout view. The latter shows a table view of the cells in the logical design.
  - In Component Types mode, the table view lists the physical cells grouped by component type. If a cell is not assigned to a particular component type, it is put in the *No component type* directory.
  - In Soft Block mode, the table view lists the instances in the design organized by cell type into *Core*, *Custom*, *Hard Blocks*, *I/Os*, and *Soft Blocks*. Instances with no physical view are also grouped together.

Important

When you select an entry in the table view, the corresponding object is cross-selected in the schematic Navigator assistant and canvas and, in turn, in the layout canvas and Navigator assistant. However, if you close the schematic view, there is no cross-selection between the Configure Physical Hierarchy Window and the layout window, because there is no active Layout XL session.

- **Attributes**: Lists the attributes of the component or components currently selected in the table view. The attributes and controls are different depending on the mode in which you are operating.

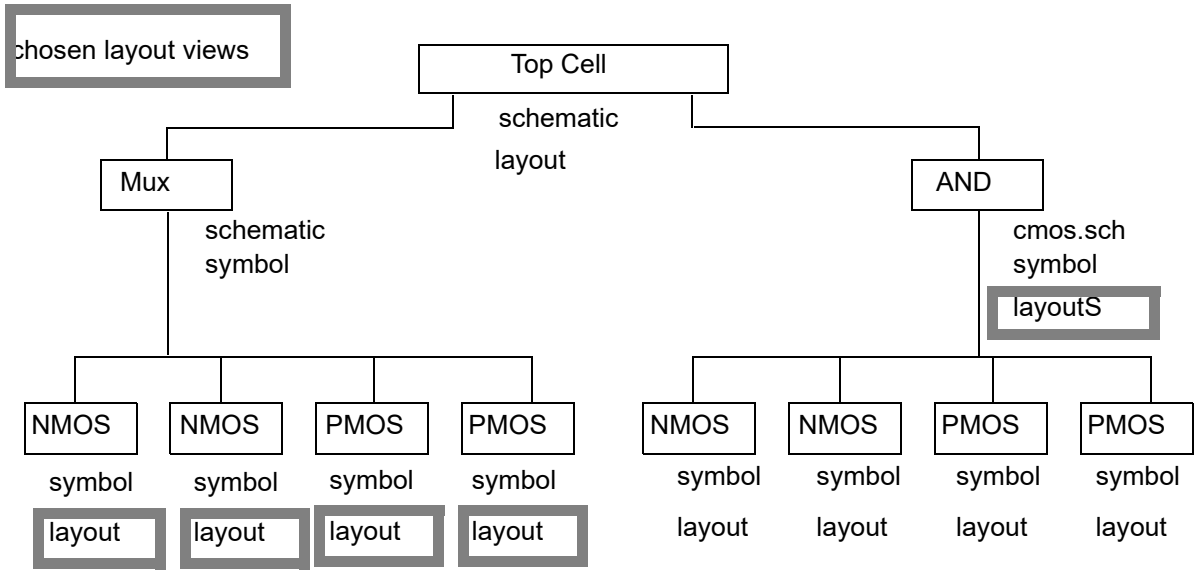
Hierarchy Traversal

The following figure represents the schematic hierarchy and shows which layout views Configure Physical Hierarchy finds under the specified conditions:

- *Physical stop view list* is set to `layout layoutS`
- *Logical switch view list* is set to `schematic cmos.sch`

© 2023 Cadence Design Systems, Inc.

The chosen layout views are shown with thick-line boxes.



1. As Configure Physical Hierarchy builds the layout, it looks at each symbol in `Top Cell`.
2. The `Multiplier (Mux)` cell has no view with a name listed in the *Physical stop view list*, so Configure Physical Hierarchy goes to the *Logical switch view list* and switches into the `Mux` schematic view.
3. All the devices in the `Mux` schematic have a view (`layout`) with a name that is in the *Physical stop view list*, so Configure Physical Hierarchy uses that view for each device.
4. The `AND` cell has a view (`layoutS`) with a name that is in the *Physical stop view list*, so Configure Physical Hierarchy uses that view for the `AND` device and never switches into the `cmos.sch` view.

Related Topics

Configure Physical Hierarchy Window

Hierarchy Configuration Mode in the CPH Window

Component Types Mode in the CPH Window

Soft Block Mode in the CPH Window

© 2023 Cadence Design Systems, Inc.

Configure Physical Hierarchy: License Requirements

The Configure Physical Hierarchy graphical user interface requires a Layout XL license, regardless of whether:

- You are using Hierarchy Configuration, Component Types or Soft Block mode.

  When using multiple SKILL APIs for working with component types, it is recommended that you checkout a Layout XL license beforehand to enhance the system performance. But if you already have a CPH window or an XL session open, the XL license will already be checked out.

- The physical configuration view is open in read or edit mode.

  In read mode, all menu items and graphical controls that let you change the physical configuration are disabled. You can still expand and collapse the nodes in the *Instances* tree, use the Options form to change the way information is displayed, and save the physical configuration under a new name.

  Note: If a layout is open in read-only mode, the physical configuration view also opens in read-only mode. However, you can make a read-only physical configuration view editable by selecting the *Make Editable* command from the *Configure Physical Hierarchy* menu.

If you launch Configure Physical Hierarchy from a Layout XL window, the system uses the Layout XL license that is already checked out.

If you open a physical configuration view directly using the CIW's *File – Open* command, the system checks out a Layout XL license, if there is not one checked out currently. When you close the Configure Physical Hierarchy window, the Layout XL license is released (provided there is no other instance of Configure Physical Hierarchy, or an associated Layout XL session running).

Related Topics

Configure Physical Hierarchy Window

Hierarchy Configuration Mode in CPH

Use Configure Physical Hierarchy in the Hierarchy Configuration mode:

- To make any required changes before you update the layout from the schematic.

© 2023 Cadence Design Systems, Inc.

- To produce alternative physical implementations of an existing design, by using a different logical to physical configuration as a “what-if” experiment.

In each case, the main role of the physical hierarchy configuration is to specify the mappings between logical and physical views. When you change a setting, you can see the effects of that change immediately in the layout view by choosing either the *Generate All From Source* or *Update Components And Nets* command. The layout view is regenerated (or updated) based on the latest settings in the Configure Physical Hierarchy window.

Related Topics

Configure Physical Hierarchy Window

Hierarchy Configuration Mode in the CPH Window

Specifying Logical to Physical Mappings for the Components in CPH

Mapping Parameter and Terminal Names

How to Ignore Components in Physical Hierarchy

Splitting Mfactored Devices

Splitting Fingered Devices

Ignoring Parasitic Devices

Specification To Check Parameters

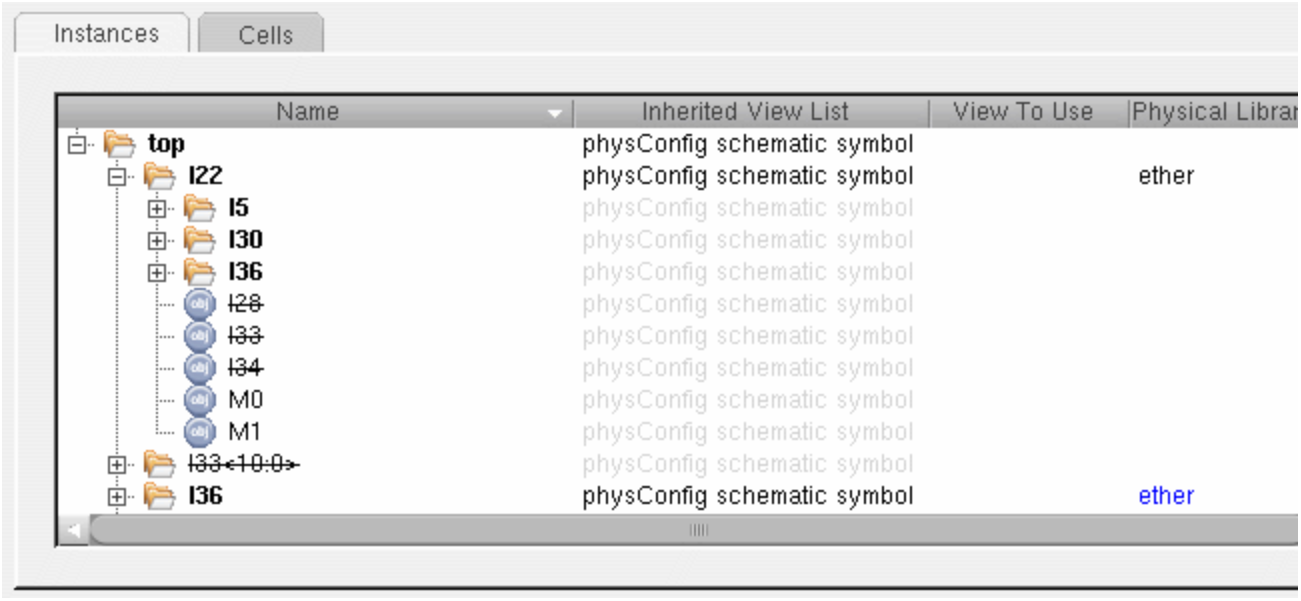
Ignore Parameters and Terminals

Rounding Parameter Values

© 2023 Cadence Design Systems, Inc.

Hierarchy Configuration Instances Table

The *Instances* tree shows a hierarchical representation of the design – that is, the schematic top-level cell – and how each instance in the logical design is mapped to a layout view.



For each instance in the design, the table shows the view list used to reach the schematic instance and the corresponding layout cellview found. The table also supports display of vector instances.

There are a number of operations you can perform directly in the table view.

- Sort the tree alphabetically by column within each hierarchical node, click the header of the column you want to sort. Alternate clicks sort in ascending and descending order.
- Edit an entry in a table field directly, double-click the text you want to change and choose a value from the drop-down list or type in the new setting manually.
- Edit multiple entries for a particular column,

  a. Select the rows you want to edit.

  b. Right-click the column you want to change and choose *Edit multiple <fieldName>*.

  c. Choose a value from the drop-down list or type in the new setting manually and press `Return`.

  The value is updated for all the selected rows.

© 2023 Cadence Design Systems, Inc.

- To make an instance a stop point in the hierarchy traversal, click the right mouse button and choose *Set/clear stop point*. A 'STOP' icon against the schematic instance indicates that it is to be regarded as a leaf instance. To clear the stop point, select the instance and choose *Set/clear stop point* again.
- To force hierarchy traversal to proceed beyond a leaf instance, click the right mouse button and choose *Set/remove force descend*. A 'go' icon indicates that the selected instance will now be traversed even if it would normally be a leaf instance. To clear the force descend, select the instance and choose *Set/remove force descend* again.
  - You can also set the force descend individually on vector bit instances by selecting the instance and choosing *Set/remove force descend* from the shortcut menu.
  - You can set the force descend only on instances that have a physical binding.
- To force hierarchy traversal to proceed down until the physical view from the physical stop library list is reached, right-click and choose *Set/remove hierarchical force descend*. A 'go' icon with inverted arrow indicates that the selected instance will be traversed until the physical view from the specified physical stop library list is reached. To clear the force descend, select the instance and choose *Set/remove hierarchical force descend* again.

  Note: To set the hierarchical force descend on an instance, there must be at least one library specified in the *Physical stop library list*. See **Global Bindings: The settings in the Global Bindings pane control how the logical design is traversed and how the logical to physical correspondence for leaf cells (stop points) is made. It is a dockable window, which you can dock on either side of the main Configure Physical Hierarchy window, or leave undocked as a floating window.**.

Instance Mode and Occurrence Mode

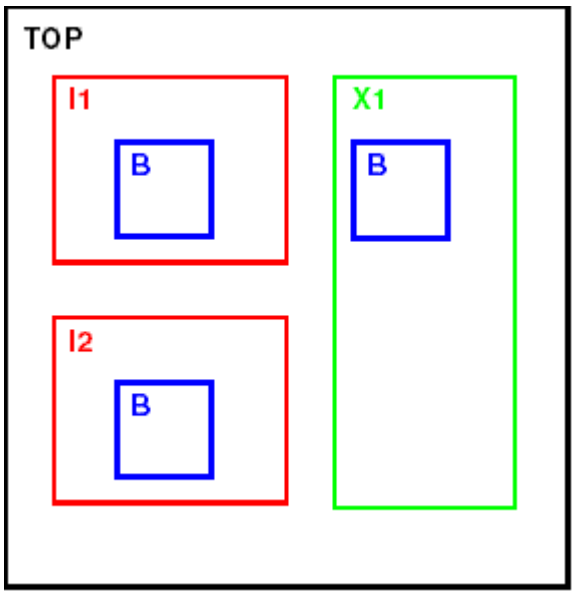
The *Instances* table has two modes: Instance mode and Occurrence mode.

- Use Instance mode to change a particular setting (for example, the physical binding) or attribute for all identical instances by changing the setting on just one of the instances.
- Use Occurrence mode to change a setting or attribute for only a single occurrence of a particular instance; for example, to use a specific variant of a cellview in a particular area of your design.

You switch between them using the toggle button  . When the button is pressed, you are working in Occurrence mode. When it is not, you are working in Instance mode. You can check the current mode by hovering over the button to display a tooltip.

© 2023 Cadence Design Systems, Inc.

Consider the following simple example.



- In Instance mode, any changes you make to the table values or attributes for a particular instance are replicated in all other identical instances in the design.

  For example, if you specify a physical binding for instance **I1:B**, the same physical binding is applied to instance **I2:B** because the two instances of **B** are identical and are contained in identical instances one level up.

  The physical binding is not applied to instance **X1:B** because **B** is contained inside an instance of a different cellview and is therefore considered different to the other two.

  To make the same change to all three instances of **B**, edit the setting for the master cellview in the *Cells* table.

- In Occurrence mode, any changes you make to the table value or attributes are applied only to the specific occurrence of the instance that you edited.

  In the example above, if you specify a physical binding for instance **I1:B**, that physical binding is applied only to **I1:B**.

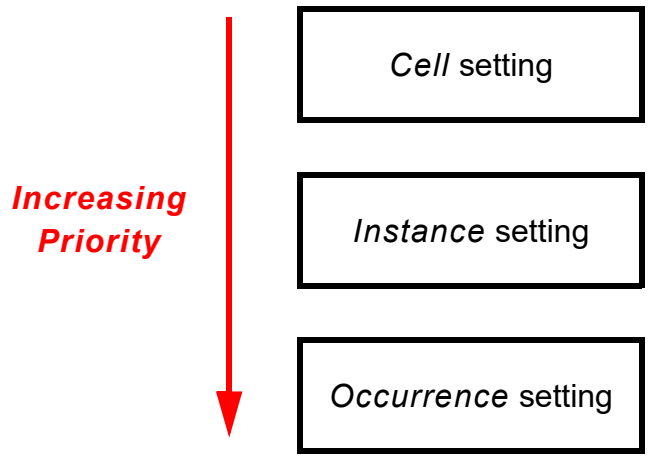
  The physical binding is not applied to either **I2:B** or to **X1:B** because these are different occurrences of instance **B**.

Inheritance of Settings

All settings are inherited from the higher level unless overridden at the lower level. Settings made in Occurrence mode override prior or subsequent settings made in either Instance

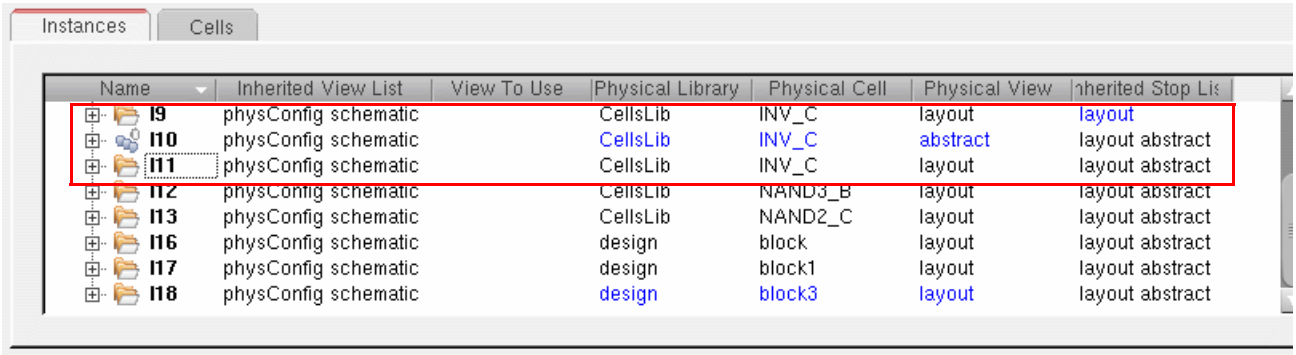
© 2023 Cadence Design Systems, Inc.

mode or in the *Cells* table. Settings made in Instance mode override prior or subsequent settings made in the *Cells* table.



If you delete a higher priority setting, the item in question inherits the setting from the next level up the chain.

Instances with occurrence-based settings are indicated using the Occurrence mode icon in the table. For example, in the picture below, instance **I10** has had its *Physical View* set in Occurrence mode. The occurrence icon replaces the normal directory icon for that entry, and the text is colored blue to indicate that it was set manually by the user.



Related Topics

Hierarchy Configuration Mode in the CPH Window

Color Coding in the Hierarchy Configuration Instances and Cells Tables

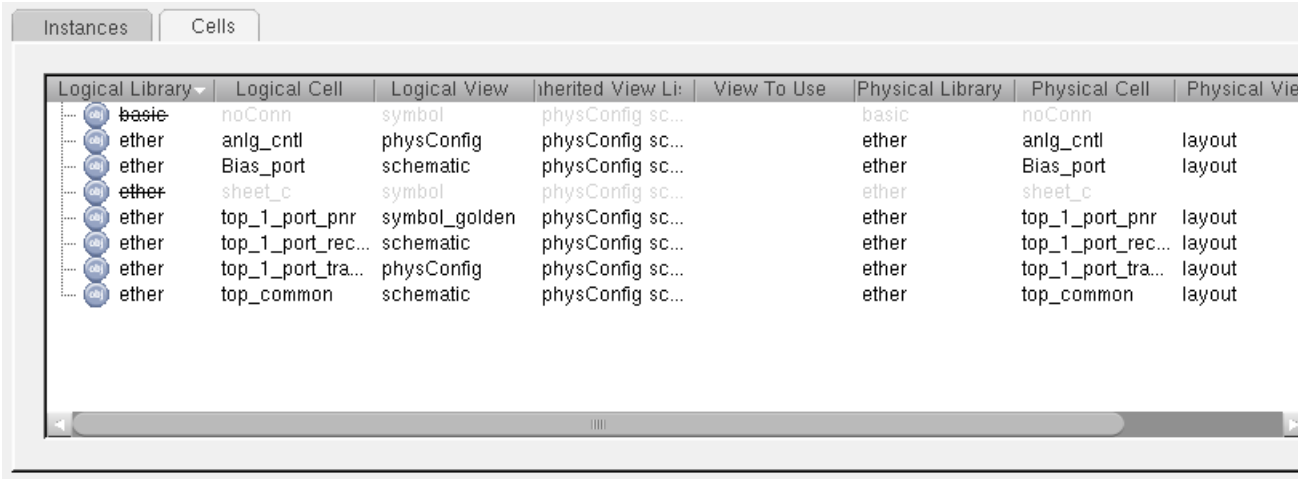
Shortcut menu for the Hierarchy Configuration Instances and Cells Tables

© 2023 Cadence Design Systems, Inc.

Hierarchy Configuration Cells Table

Hierarchy Configuration Cells Table

The *Cells* table view shows a flat view of the cellviews used in the schematic design and lists the corresponding layout cellview.



The table lists each cellview explicitly, so if your design contains two instances of the same cell and you specify different views to be used for each instance, then the *Cells* table lists two entries for that cell, one for each view to use.

There are a number of operations you can perform directly in the table view.

- To sort the tree alphabetically by column within each hierarchical node, click the header of the column you want to sort. Alternate clicks sort in ascending and descending order.
- To edit an entry in a table field directly, double-click the text you want to change and choose a value from the drop-down list or type in the new setting manually.
- To edit multiple entries for a particular column:

  a. Select the rows you want to edit.

  b. Click the right mouse button on the column you want to change and choose *Edit multiple <fieldName>*.

  c. Choose a value from the drop-down list or type in the new setting manually and press `Return`.

  The value is updated for all the selected rows.

© 2023 Cadence Design Systems, Inc.

Related Topics

Hierarchy Configuration Mode in the CPH Window

Color Coding in the Hierarchy Configuration Instances and Cells Tables

Shortcut menu for the Hierarchy Configuration Instances and Cells Tables

Marking Cells for Virtual Hierarchy Creation

You can use the Configure Physical Hierarchy window to perform selective generation of virtual hierarchies for a layout. Selective virtual hierarchy generation mode can be enabled using the `vhSelectiveMode` environment variable or by using the *Generate – Virtual hierarchy* option on the Virtual Hierarchy Options form.

In this mode, the layout is generated completely flat even when the *Virtual Hierarchy* option on the Generate Layout form is selected. But, you can use the CPH window to selectively choose cells for virtual hierarchy generation.

To choose cells for selective virtual hierarchy generation using the CPH window:

1. In the table view, select the Cells tab.
2. In the Cells table, right-click the cell for which you want to create the virtual hierarchy and choose *Create/remove Virtual Hierarchy*.

The icon corresponding to the cell changes to  shape to indicate that the cell is marked for generation during the next *Generate All From Source* run. You can also select multiple cells for marking in one go.

Note:

- To unmark a cell that was previously selected for virtual hierarchy generation, right-click the cell and choose *Create/remove Virtual Hierarchy.* The icon changes to indicate that the cell will be generated as flat during the next the *Generate All From Source* run.

Related Topics

vhSelectiveMode

Virtual Hierarchy Options

© 2023 Cadence Design Systems, Inc.

Specifying Logical to Physical Mappings for the Components in CPH

You can use the Configure Physical Hierarchy window to specify the logical to physical mappings for the components in your design. You can specify the mapping at the cell-level, in which case all instances of that cell inherit the mapping you specify, or you can override the cell-level setting for one or more instances or occurrences in your design.

Specifying the Mapping for a Cell

To specify the logical to physical mapping for a cell:

1. With the design loaded, choose *Launch – Configure Physical Hierarchy* from the Layout XL window.

   The Configure Physical Hierarchy window is displayed.

2. In the Configure Physical Hierarchy window, click the *Cells* tab.

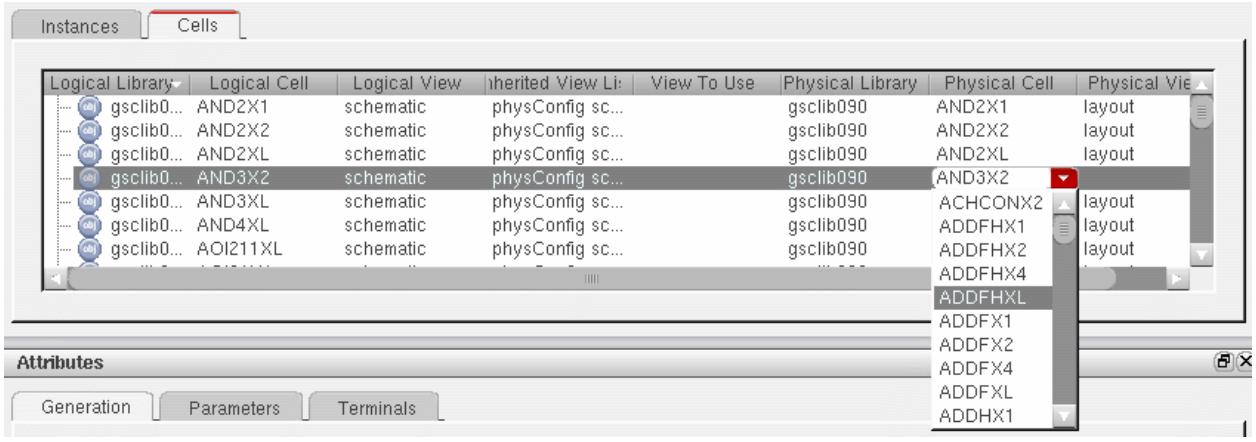
3. Click the row containing the cell for which you want to change the mapping.

   The row is highlighted.

4. Double-click the text in the *Physical Cell* column.

   The *Physical Cell* entry becomes a drop-down list containing the names of all the available views for the cell in question.

5. Choose the cell name you want from the list and press `Return` to confirm the setting.



6. In the same way, select the *Physical View* to use and press `Return` to confirm the setting.

   The logical to physical mapping for the cell in question is changed.

© 2023 Cadence Design Systems, Inc.

7. From the menu bar, choose *File – Save* to save the physical configuration view.

   The next time you generate or update the layout view, the system uses the new view for that cell instead of the one previously specified.

Specifying the Mapping for an Instance or Occurrence

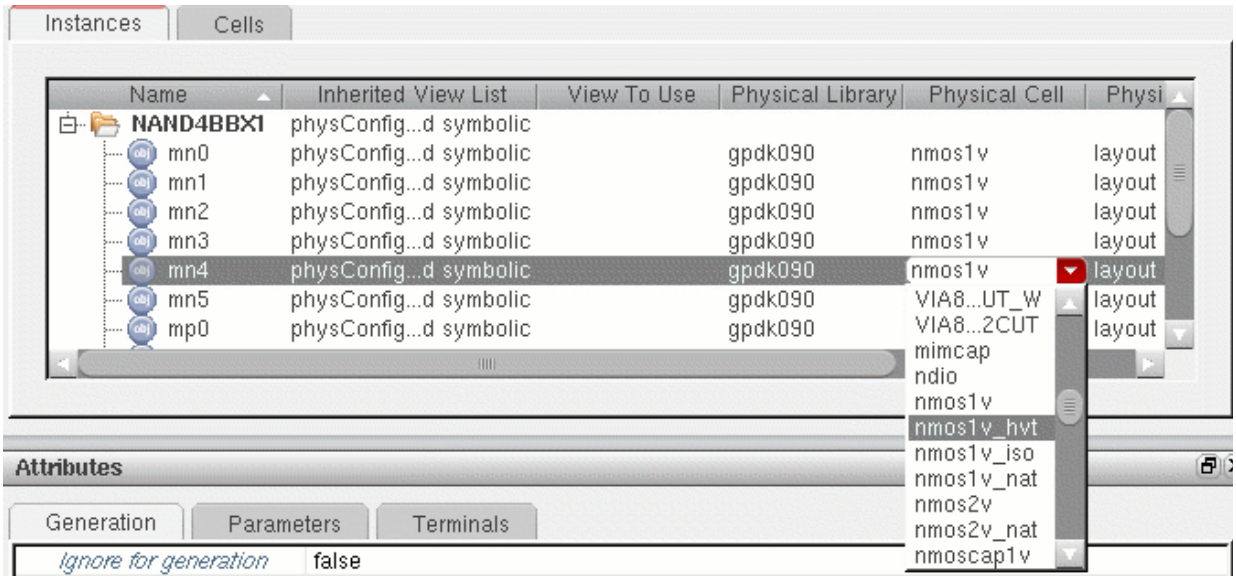
To change the logical to physical mapping for a single instance or occurrence:

1. In the Configure Physical Hierarchy window, click the *Instances* tab.
2. Click the row containing the instance for which you want to change the mapping.

   The row is highlighted.
3. Do one of the following:

   - To specify the mapping for all the occurrences of the selected instance, ensure that Occurrence mode is turned off in the toolbar.
   - To specify the mapping for an individual occurrence, make sure Occurrence mode is turned on.
4. Double-click the text in the *Physical Cell* column.

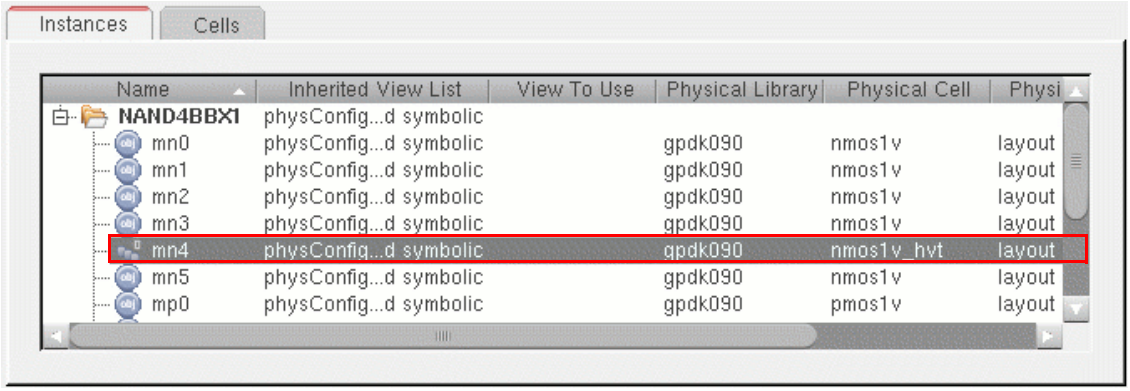
   The *Physical Cell* entry becomes a drop-down list containing the names of all the available views for the cell in question.
5. Choose the cell name you want from the list and press `Return` to confirm the setting.



© 2023 Cadence Design Systems, Inc.

6. In the same way, select the *Physical View* to use and press `Return` to confirm the setting.

   The logical to physical mapping for the instance in question is changed. If you specified an occurrence mapping, the symbol is also changed to indicate this.



7. From the menu bar, choose *File – Save* to save the physical configuration view.

   The next time you generate or update the layout view, the system uses the new view for the instance or occurrence instead of the one previously specified.

Related Topics

Launching Configure Physical Hierarchy

Instance Mode and Occurrence Mode

Mapping Parameter and Terminal Names

The *Name mapping* options on the *Parameters* and *Terminals* tabs of the Attributes pane let you define mapping between parameter and terminal names in schematic and layout cells.

You cannot override this setting for an individual instance or occurrence of a cell. The field is not available among the instance attributes.

This topic uses the *Parameters* tab as an example. The instructions apply equally to the *Terminals* tab.

Mapping a Parameter Name in CPH

To map a parameter name between a schematic cell and its layout equivalent:

© 2023 Cadence Design Systems, Inc.

1. In the Configure Physical Hierarchy window, click the *Cells* tab.
2. Click the row containing the cell with the parameter names you want to map.

   The row is highlighted and the options in the Attributes pane are enabled.
3. In the *Parameters* tab, type the mapping you want into the *Name mapping* field.

   The syntax is illustrated in the example below, which maps schematic parameters `l` and `w` to layout parameters `L` and `W` respectively.

   ```
   l L ; w W ;
   ```
4. From the menu bar, choose *File – Save* to save the physical configuration view.

   The settings are saved for all instances of the cell in question. The specified parameter names are mapped accordingly the next time you update the layout view.
5. To restore the value to its default, select the value and click the *Revert to default* button on the right.

Mapping a Terminal Name in CPH

To map a terminal name between a schematic cell and its layout equivalent:

1. In the Configure Physical Hierarchy window, click the *Cells* tab.
2. Click the row containing the cell with the terminal names you want to map.

   The row is highlighted and the options in the Attributes pane are enabled.
3. In the *Terminal*s tab, type the mapping you want into the *Name mapping* field.
4. From the menu bar, choose *File – Save* to save the physical configuration view.

   The settings are saved for all instances of the cell in question. The specified terminal names are mapped accordingly the next time you update the layout view.
5. To restore the value to its default, select the value and click the *Revert to default* button on the right.

Related Topics

Mapping a Parameter Name in CPH

Mapping Parameter and Terminal Names

© 2023 Cadence Design Systems, Inc.

How to Ignore Components in Physical Hierarchy

You can use Configure Physical Hierarchy to specify ignore properties at the cell-level, in which case all the instances of that cell are ignored, or you can override the cell-level setting for one or more instances or individual occurrences of instances in your design.

The value of the `propsUsedToIgnoreObjs` property set on a device in the schematic design is used as the default value for the *Ignore for generation* and *Ignore for check* attributes in the Configure Physical Hierarchy window.

If you do not change the value of these properties in the Configure Physical Hierarchy window, the *Generate All From Source* command honors the value set for the `propsUsedToIgnoreObjs` in the schematic design and any changes made to the value of that property in the schematic design.

If you change a value in the Configure Physical Hierarchy window to specify that certain components are to be ignored either during layout generation and/or check, *Generate All From Source* respects the changed value and disregards any changes made to the `propsUsedToIgnoreObjs` property in the schematic design.

To restore the default, select the value and click the *Revert to default* button on the right.



Generate All From Source again honors the value on the `propsUsedToIgnoreObjs` property in the schematic design and any changes made to the value of that property in the schematic design.

© 2023 Cadence Design Systems, Inc.

The table below shows the inheritance rules for ignore values on cells and instances.

Inheritance Rules for Ignore Properties

Cell	Instance	Instance is ignored?
No ignore property	`ignore` = true	Yes.
No ignore property	`ignore` = true `lvsIgnore` = false	Yes, because `lvsIgnore` OR `ignore` => true
`ignore` = true	No ignore property	Yes, because it inherits the `ignore` property setting from the cell
`ignore` = true	`lvsIgnore` = false	Yes, because the `ignore` property is inherited from the cell meaning that `lvsIgnore` OR `ignore` => true
`ignore` = true	`ignore` = false	No, because the `ignore` property exists on the instance and its setting overrides the setting on the cell.

Related Topics

Ignoring an Instance or Occurrence for Generation or Check

Ignoring a Cell for Generation or Check Against Source

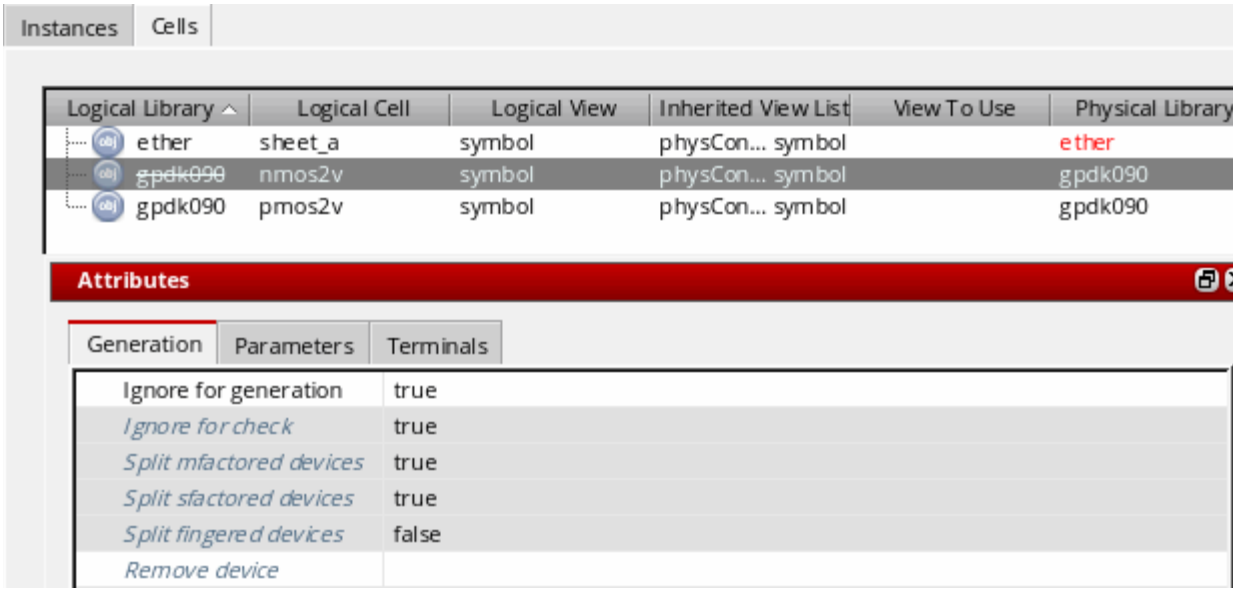
To specify that all instances of a particular cell are to be ignored when the layout view is generated or when the *Check Against Source* command is run:

1. In the Configure Physical Hierarchy window, click the *Cells* tab.
2. Click the row containing the cell you want to ignore.

   The row is highlighted and the options in the Attributes pane are enabled.
3. In the *Generation* tab, set *Ignore for generation* to true and press `Return`.

© 2023 Cadence Design Systems, Inc.

The cell name is now displayed with a strikethrough.



Setting this option automatically sets the *Ignore for check* option as well because it is irrational to check something that you have purposely excluded from the layout. You can switch on the *Ignore for check* option independently by first disabling the *Ignore for generation* option.

4. From the menu bar, choose *File – Save* to save the physical configuration view.

   The ignore properties for the cell in question are saved.

   The next time you generate or update the layout view, no layout representation is generated for the instance in question.

Tip

To generate an instance that was earlier set as *ignore for generation*, make sure the *Remove device* option is also set to `false`. Else, the instance will have its nets merged and will continue to be ignored for generation, even though *ignore for generation* is set to `false`.

Related Topics

Design Data Checks

Inheritance Rules for Ignore Properties

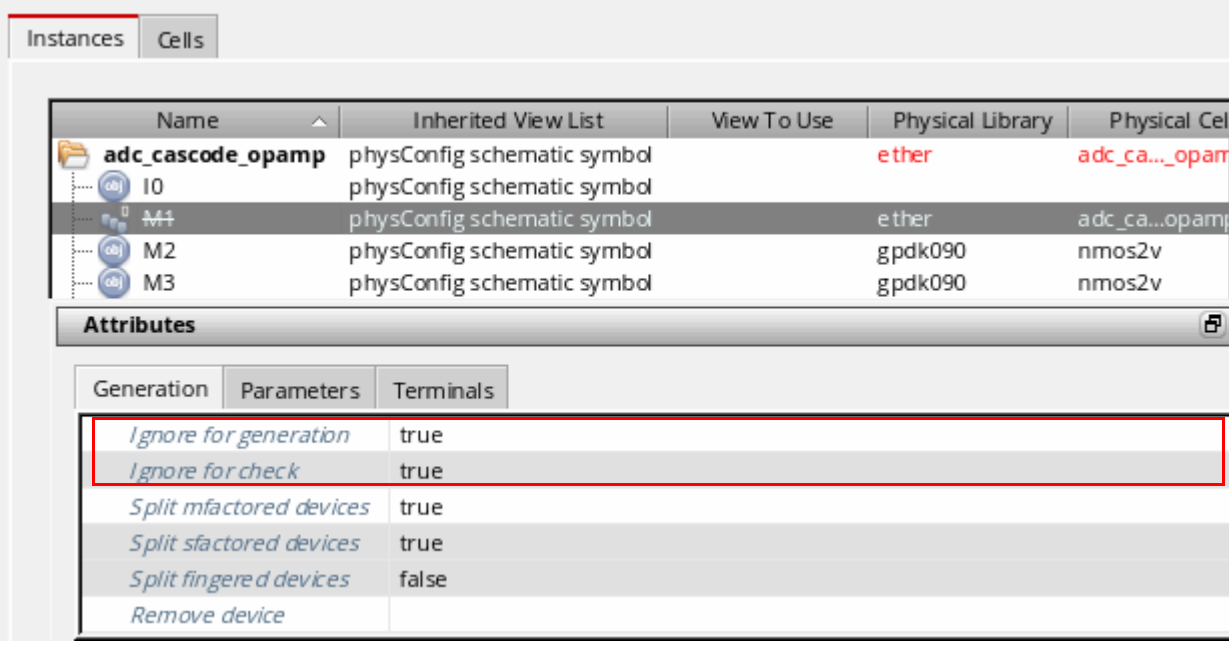
© 2023 Cadence Design Systems, Inc.

Ignoring an Instance or Occurrence for Generation or Check

To specify that an instance is not to be generated in the layout view or checked by the *Check Against Source* command:

1. In the Configure Physical Hierarchy window, click the *Instances* tab.
2. Click the row containing the instance you want to ignore.

   The row is highlighted and the options in the Attributes pane are enabled.



3. Do one of the following:

   - To specify the ignore property for all the occurrences of the selected instance, ensure that Occurrence mode is turned off in the toolbar,  .
   - To specify the ignore property for an individual occurrence, make sure the Occurrence mode is turned on.

4. In the *Generation* tab, set *Ignore for generation* to true and press `Return`.

   The instance or occurrence name is now displayed with a strikethrough. If you specified an occurrence mapping, the symbol is also changed to indicate this.

   Setting *Ignore for generation* automatically sets *Ignore for check* as well because it is irrational to check something that you have purposely excluded from the layout. You can switch on the *Ignore for check* option independently by first disabling the *Ignore for generation* option.

© 2023 Cadence Design Systems, Inc.

5. From the menu bar, choose *File – Save* to save the physical configuration view.

   The next time you generate or update the layout view, no layout representation is generated for the instance in question.

Related Topics

Instance Mode and Occurrence Mode

Inheritance Rules for Ignore Properties

Splitting Mfactored Devices

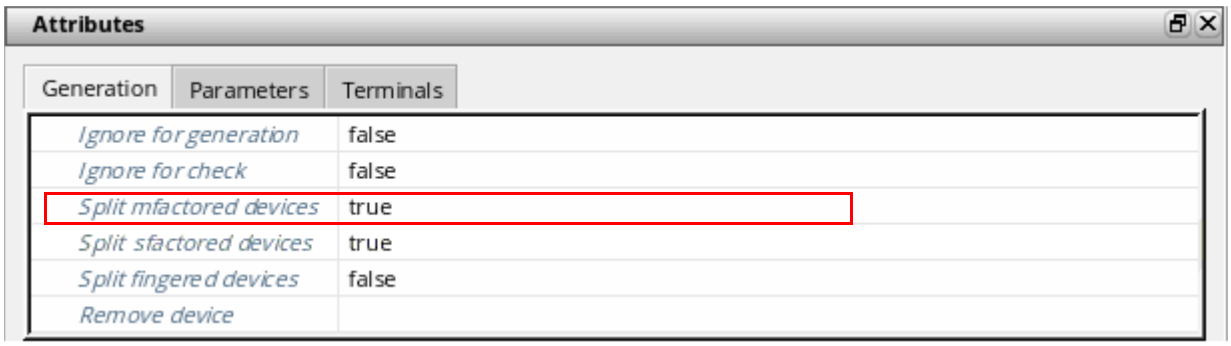
The *Split mfactored devices* option controls whether Layout XL places schematic devices with the `mfactor` property as multiple devices in the layout or as single devices. It overrides the `mfactorSplit` environment variable, which can also be set on the *Generation* of the Connectivity form.

Note: *Split mfactored devices* controls *only* whether Layout XL generates one device or several devices in the layout view. It does not change the widths of the generated devices.

To set or unset this option,

1. In the Configure Physical Hierarchy window, click the *Instances* tab or the *Cells* tab, and select the device in question.

   The *Split mfactored devices* option is checked or unchecked depending on the default value set on the *Generation* of the Connectivity form.



   Split mfactored devices is grayed out if the selected instances do not have the `mfactor` property.

2. Set the option as required and choose *File – Save* from the menu bar to save the physical configuration view.

© 2023 Cadence Design Systems, Inc.

The new setting takes effect the next time you generate or update the layout view.

Related Topics

mfactor Property

mfactorSplit

Generation

Connectivity Form

Splitting Sfactored Devices

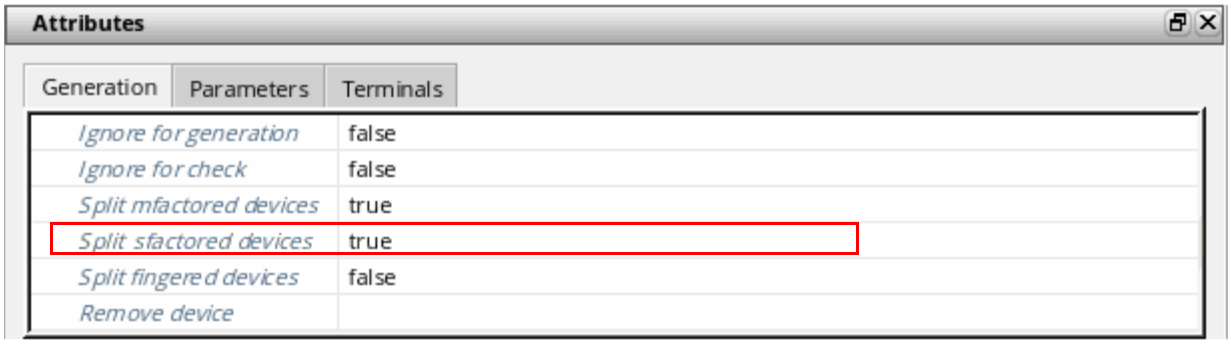
The *Split sfactored devices* option controls whether Layout XL places schematic devices with the `sfactor` property as series-connected devices in the layout or as single devices. It overrides the `sfactorSplit` environment variable, which can also be set on the *Generation* of the Connectivity form.

Note: *Split sfactored devices* controls *only* whether Layout XL generates one device or several devices in the layout view. It does not change the widths of the generated devices.

To set or unset this option:

1. In the Configure Physical Hierarchy window, click the *Instances* tab or the *Cells* tab, and select the device in question.

   The *Split sfactored devices* option is checked or unchecked depending on the default value set on the *Generation* of the Connectivity form.



   Split sfactored devices is grayed out if the selected instances do not have the `sfactor` property.

© 2023 Cadence Design Systems, Inc.

2. Set the option as required and choose *File – Save* from the menu bar to save the physical configuration view.

   The new setting takes effect the next time you generate or update the layout view.

Related Topics

sfactor Property

sfactorSplit

Generation

Connectivity Form

Splitting Fingered Devices

The *Split fingered devices* option controls whether each finger of a schematic device will be placed in the layout as a separate device.

For example, if an instance has three fingers in the schematic and finger splitting is selected, three devices, each with one finger, is generated in the layout. If the schematic device also has an `mfactor` value set, the number of devices generated in the layout is a multiple of the `mfactor` value and the number of fingers.

Therefore, for a schematic device that has `3` fingers and an `mfactor` value of `2`, the number of devices generated in the layout is:

```
(number of fingers X mfactor value)

(3 X 2) = 6
```

The *Split fingered devices* option also controls the width of each finger generated in the layout when the finger width is not specified as a CDF parameter but the *total width* and the *number of fingers* are specified. In this case, the resulting finger width may not match the schematic finger width because the split finger width is calculated as:

```
(total width X number of fingers)
```

To control the generated finger width, you can set the `fingerSplitSetWidth` environment variable to `t`.

Alternatively, finger splitting can be set by:

- Setting the `fingerSplit` environment variable to `t`.

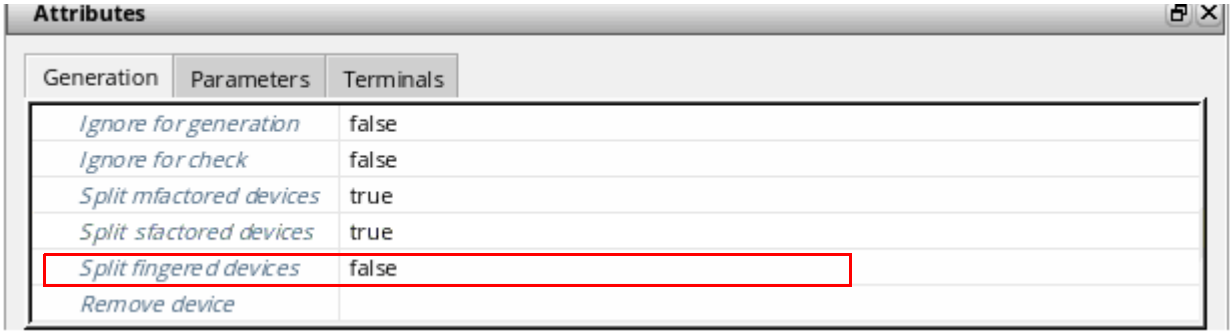
© 2023 Cadence Design Systems, Inc.

- Selecting the *Split fingered devices* option in the *Generation* of the Connectivity Form form.

To set or unset finger splitting using the *Configure Physical Hierarchy* window:

1. Click the *Instances* tab or the *Cells* tab, and select the device in question.

   The *Split fingered devices* option appears selected (`true`) or deselected (`false`), depending on the default value set on the *Generation* tab of the Connectivity form.



2. Set the option as required and choose *File – Save* from the menu bar to save the physical configuration view.

The new setting takes effect the next time you generate or update the layout view.

Note: If a multi-fingered device requires to be split after the device has been generated in layout, you can fold the multi-fingered device in order to create a separate device for each finger.

Related Topics

fingerSplitSetWidth

fingerSplit

Generation

Connectivity Form

Folding a Multi-fingered Device to Create Split Fingers

© 2023 Cadence Design Systems, Inc.

Ignoring Parasitic Devices

Use the *Remove device* option on the *Generation* tab to cause parasitic devices to be ignored by merging nets connected to the terminals of a single instance at a time.

You can also use this option to merge nets connected to the terminals of an iterated instance. To do so, specify the option for the instance being iterated, not for the hierarchical block containing the iterated instances.

Note:

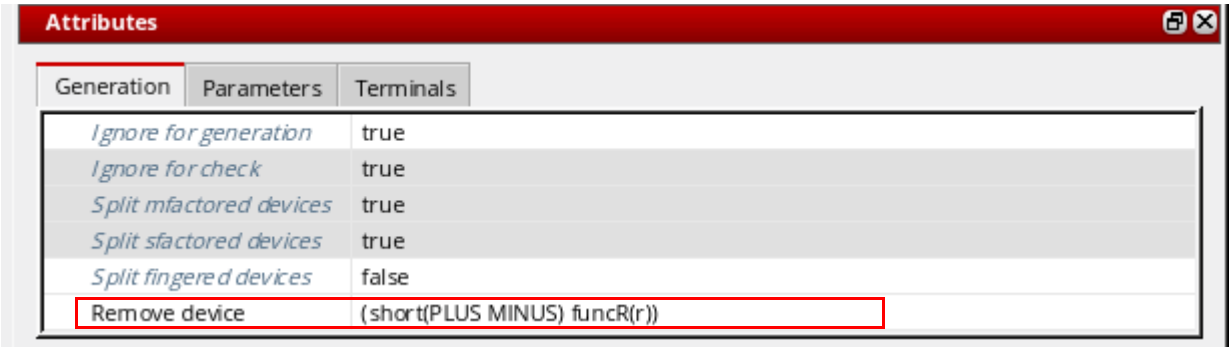
- The use of this option is not restricted to devices with only two terminals; it also works for devices with more than two terminals.

To generate a device that was earlier set as *Ignore for generation,* you must also disable the *Remove device* option to allow the device to be regenerated. Unless the Remove device option is also set to `false`, the nets connected to the instances are merged and the device is ignored for generation.

To ignore parasitic devices:

1. In the Configure Physical Hierarchy window, click the *Instances* tab.
2. Click the row containing the instance you want to ignore.

   The row is highlighted and the options in the Attributes pane are enabled.
3. In the *Generation* tab, specify the *Remove device* argument in the text field provided.



For example, for a resistor with terminal names PLUS and MINUS,

- `(short(PLUS MINUS))` would short the terminals.
- `(short(PLUS MINUS) funcR(r))` would short the terminals only if the user-defined SKILL function `funcR` returns non-nil.

  A sample `funcR` for the above case would be as follows.

© 2023 Cadence Design Systems, Inc.

```
procedure( funcR(r)
    if(r<100 then t
    else nil)
)
```

The `funcR` SKILL function must already be defined in your in your `.cdsinit` or `libInit.il` file. The first parameter of the SKILL function is `r` and must be an instance or cell parameter of the instance that you are testing for removal (defining, for example, the resistance value for a resistor).

The system searches the instances in the design for the parameter `r` and evaluates the value for that parameter in each of the instances in which it exists. For the purposes of this example, parameter `r` evaluates to 30.

The system then calls the `funcR` function, passing the value 30 as the first argument. The `funcR` function compares this value to a predefined value (in this case 100).

- If it is less than 100, the function returns `t`, implying that the resistor instance is to be removed from the design (i.e., shorted).
- If it is greater than or equal to 100, the function returns `nil`, implying that the instance should not be removed.

4. From the menu bar, choose *File – Save* to save the physical configuration view.

   The setting takes effect the next time you generate or update the layout view.

Rules to Handle Device Removal

1. A single function applies for all the `short()` lists.
2. If the function is missing, the terminals are shorted as specified in the short rule.
3. If the device is not removed, the terminals are not shorted.
4. A warning is issued when the device is not ignored in the netlisting.

Rules for Net Merging during Remove Device

The system uses the following rules to decide which net survives after the device is shorted.

1. Global nets, such as gnd!, or nets connected to I/O pins always survive.
2. If there are no global or external nets, the net at a higher level of the hierarchy survives. For example, `net1` survives over `/I1/net2`.

© 2023 Cadence Design Systems, Inc.

3. If both the nets are at the same hierarchy level, the labeled net survives. For example, the labeled net, `net10,` survives over the non-labeled net, `net1.`
4. If both the nets have a label, the net that has the shortest name and is the first one alphabetically, survives. For example, the labeled `netAA` survives over the labeled `netZZ` and over the non-labeled `netA`.
5. If the nets are not labeled and their names are of the same length, the net that survives is the first one alphabetically. For example, `netM` survives over `netN`.

When merging the nets of a shorted device, if you want a particular net to survive, you must add a label to the net. Labeled nets survive over non-labeled nets even if the name of the non-labeled net is shorter.

Related Topics

Configure Physical Hierarchy Window

Setting the Instance Stop Level

The *Stop level* attribute controls the hierarchy level to which the selected instance will elaborate relative to the top level.

If an instance at level 1 and another one at level 2 are both set to a *Stop level* value of 5, both instances will elaborate to level 5 from the top.

The new stop level value takes effect the next time you generate or update the layout view.

To set the stop level for an instance using the Configure Physical Hierarchy window:

1. In the Configure Physical Hierarchy window, click the *Instances* tab.
2. Select the row containing the instance for which you want to set the stop level.

   The row is highlighted and the options in the *Attributes* pane are enabled.

© 2023 Cadence Design Systems, Inc.

3. In the *Generation* tab, type the required *Stop level* value.



4. Choose *File – Save* to save the update.

The new *Stop level* value for the updated instance takes effect the next time you generate or update the layout view.

Related Topics

Configure Physical Hierarchy Window

Specification To Check Parameters

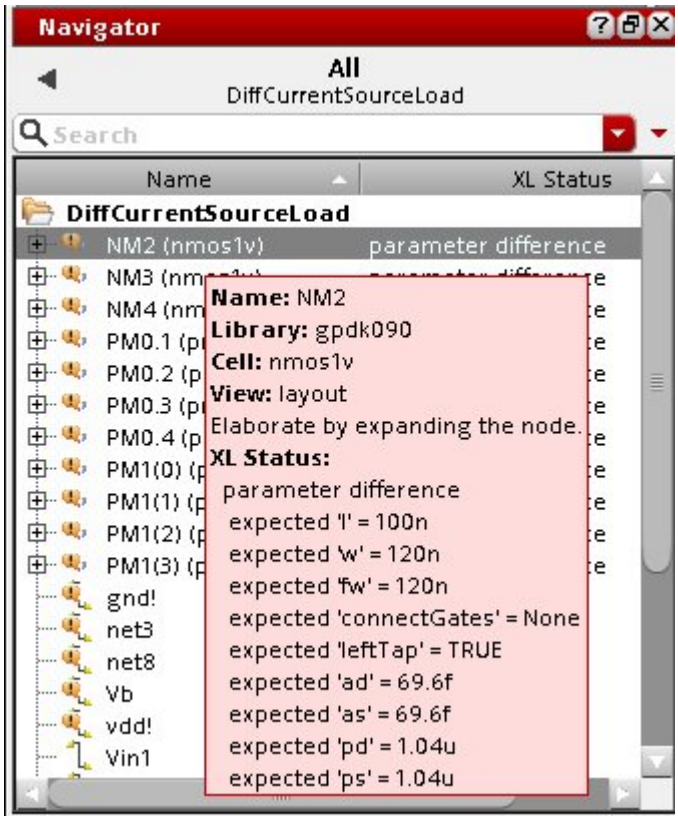
The *To Check* field on the *Parameters* tab of the *Attributes* pane lets you specify the names of the schematic parameters to be checked by the *Check Against Source* command.

For designs that have a long list of layout-only parameters that need to be ignored during a *Check Against Source* run, it is more efficient to specify the parameters that need to be checked instead of specifying the ones that need to be ignored. This not only reduces the effort of setting the parameters, but also makes the *check* and *update* procedures more effective as there is a lesser number of parameters to process.

You can use the *To Check* field to specify the list of parameters to check. When the *parameters to check* list is specified, Layout XL uses that list instead of using the *parameters to ignore during check* list.

© 2023 Cadence Design Systems, Inc.

For example, in the figure displayed below, no parameters were set for *ignore*. Therefore, the instance info balloon displays only those parameters that were set for check.



In Layout XL, the parameters to check can be specified using one of the following:

- LAM file
  - Cadence recommends that the parameters to check be defined in the LAM file. This ensures that the parameters to check are set at the library level and can benefit all the layout engineers using the particular cell in their design. However, any updates to the LAM file will need to be made by the CAD team engineers.
- *To Check* field in the *Parameters* tab in the CPH attributes pane
  - The *To Check* field displays the parameters set for checking, if defined in the LAM file. You can use the *To Check* field to modify the parameters to check for the current design. Any change in the CPH parameters to check list does not change the settings in the LAM file. The library-level settings remain the same, unless modified in LAM.
- *Parameters to check* option on the *Parameters* tab in the Connectivity form.

© 2023 Cadence Design Systems, Inc.

- If the parameters to check are defined in the Connectivity form, the setting is applicable for all the Pcells. This is unlike setting the parameters to check in the LAM file, which allows the parameters to be set for a particular Pcell.

- `paramsToCheck` environment variable
  - By default, the environment variable is set to "", which means the parameters to check are not specified. In this case, the `paramsToIgnore` list is used.

    If the `paramsToCheck` environment variable is set, it takes precedence over the `paramsToIgnoreForCheck` environment variable.

When the *parameters to check* list is set up, the parameters in the list are the only ones that are updated by the Layout Parameters Updates or Schematic Parameter Updates commands.

Example

Let us now consider an example that illustrates a typical design scenario that uses the *parameters to check* list also for running the *Update* command.

Let us assume that you want to check the total width, `w`.

In this case, it is likely that you set for check only the *total width* (`w`) parameter and ignore the *finger width* (`fw`) parameter.

But:

```
w = fingers * fw
```

If you only check and update the width (`w`) parameter, the Pcell does not change its finger width (`fw`), unless the CDF callbacks are fired. Therefore, if you only want to check the width parameter (`w`), but want to update both the width (`w`) and finger width (`fw`) parameters, you must set the `lxEvalCDFCallbacks` environment variable to `t`.



Tip

In the event that `lxEvalCDFCallbacks` cannot be set to `true`, ensure that the *parameters to check* list matches the *parameters to update* list.

For the current example, if CDF callback evaluation is not set, you must also include finger width (`fw`) as a parameter to check. This ensures that the lists of parameters to be checked and updated are in sync.

Alternatively, you can use the following options on the *Parameters* tab in the Connectivity form to set the parameters to check and update:

© 2023 Cadence Design Systems, Inc.

- *Parameters to ignore during check*
- *Parameters to ignore during generation and update*

Related Topic

LAM File Mapping Section

Checking a Parameter on a Cellview

Connectivity Form

paramsToCheck

paramsToIgnore

paramsToIgnoreForCheck

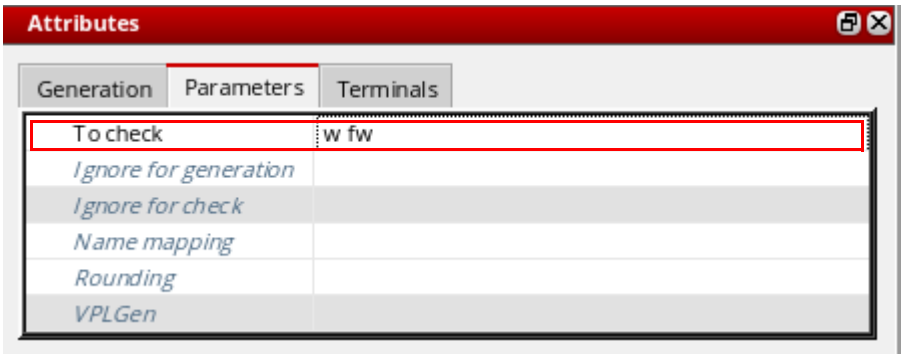
lxEvalCDFCallbacks

Checking a Parameter on a Cellview

To check a specified cellview parameter:

1. In the Configure Physical Hierarchy window, click the *Cells* tab.
2. Click the row containing the cell with the parameters you want to check.

   The row is highlighted and the options in the Attributes pane are enabled.
3. In the *Parameters* tab – *To check* field, type the name of any parameter to be checked by the *Check Against Source* command.



4. From the menu bar, choose *File – Save* to save the physical configuration view.

© 2023 Cadence Design Systems, Inc.

The settings are saved for all the instances of the cell in question. The specified parameters are checked the next time you run the *Check Against Source* command or update the layout view.

Related Topics

Checking a Parameter on an Instance or Occurrence

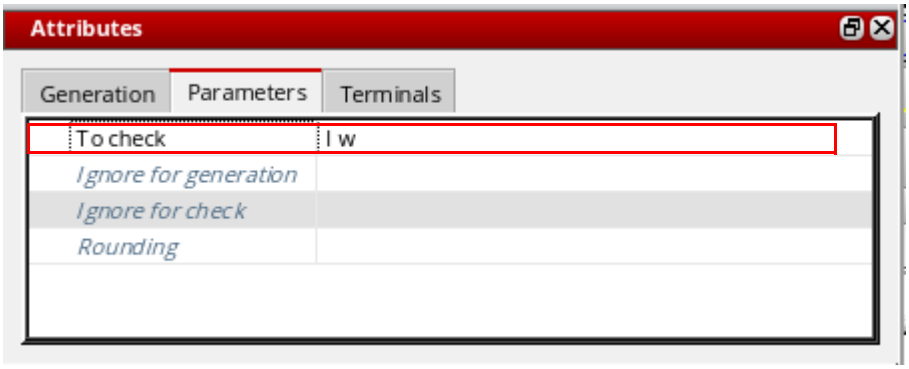
Checking a Parameter on an Instance or Occurrence

To check a specified parameter on a single instance or occurrence:

1. In the Configure Physical Hierarchy window, click the *Instances* tab.
2. Click the row containing the instance with the parameters you want to check.

   The row is highlighted and the options in the Attributes pane are enabled.
3. Do one of the following.
   - To specify the parameter to check for all the occurrences of the selected instance, ensure Occurrence mode is turned off in the toolbar.

     
   - To specify the parameter to check for an individual occurrence, ensure Occurrence mode is turned on.
4. In the *Parameters* tab – *To Check* field, type the name of the parameter to be checked by *Check Against Source* and press `Return`.



The settings are made for the selected instance or occurrence. If you specified an occurrence mapping, the symbol is also changed to indicate this.

© 2023 Cadence Design Systems, Inc.

5. From the menu bar, choose *File – Save* to save the physical configuration view.

The specified parameters are checked the next time you run the *Check Against Source* command.

Instance Mode and Occurrence Mode

Related Topics

Ignore Parameters and Terminals

The *Ignore for generation* and *Ignore for check* controls on the *Parameters* and *Terminals* tabs of the Attributes pane let you specify the names of the schematic parameters or terminals that are not to be transferred to the layout view or checked by the *Check Against Source* command.

Note: To ignore a layout-only terminal, an `ignore` property must be added on the terminal in the layout instance master.

The sections below use the *Parameters* tab as an example. The instructions apply equally to the *Terminals* tab.

- Ignoring a Parameter on a Cellview
- Ignoring a Parameter on an Instance or Occurrence

Ignoring a Parameter on a Cellview

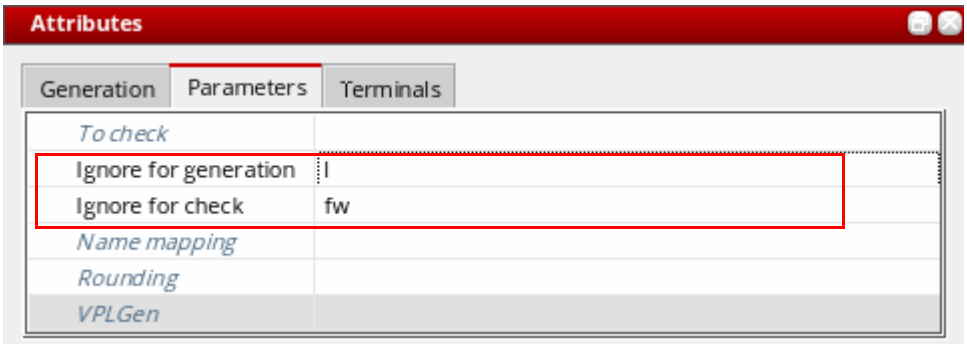
To ignore a specified cellview parameter:

1. In the Configure Physical Hierarchy window, click the *Cells* tab.
2. Click the row containing the cell with the parameters you want to ignore.

   The row is highlighted and the options in the Attributes pane are enabled.

© 2023 Cadence Design Systems, Inc.

3. In the *Parameters* tab, type the name of any parameter to be ignored by layout generation in the *Ignore for generation* field.



4. Specify the name of any parameter to be ignored by *Check Against Source* in the *Ignore for check* field.
5. From the menu bar, choose *File – Save* to save the physical configuration view.

   The settings are saved for all instances of the cell in question. The specified parameters are ignored the next time you update the layout view or the next time you run the *Check Against Source* command.

Related Topics

Ignoring a Parameter on an Instance or Occurrence.

Ignoring a Parameter on an Instance or Occurrence

To ignore a specified parameter on a single instance or occurrence:

1. In the Configure Physical Hierarchy window, click the *Instances* tab.
2. Click the row containing the instance with the parameters you want to ignore.

   The row is highlighted and the options in the Attributes pane are enabled.
3. Do one of the following.

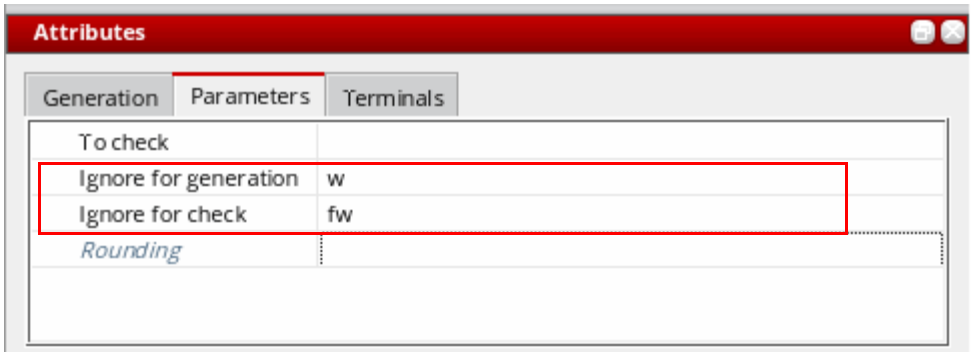
   - To specify the parameter to ignore for all the occurrences of the selected instance, ensure Occurrence mode is turned off in the toolbar.



© 2023 Cadence Design Systems, Inc.

- To specify the parameter to ignore for an individual occurrence, ensure Occurrence mode is turned on.

4. In the *Parameters* tab, type the name of the parameter to be ignored by layout generation in the *Ignore for generation* field and press `Return`.



5. Type the name of the parameter to be ignored by *Check Against Source* in the *Ignore for check* field and press `Return`.

   The settings are made for the selected instance or occurrence. If you specified an occurrence mapping, the symbol is also changed to indicate this.

6. From the menu bar, choose *File – Save* to save the physical configuration view.

   The specified parameters are ignored the next time you update the layout view or the next time you run the *Check Against Source* command.

Related Topics

Instance Mode and Occurrence Mode

Rounding Parameter Values

The *Rounding* option on the *Parameters* tab lets you define how the value of a specified parameter is rounded when it is evaluated by Layout XL. It is typically used to prevent folded devices from becoming off-grid.

- Rounding Cellview Parameter Values
- Rounding Instance or Occurrence Parameter Values

In previous releases, you could use the equivalent `lxRounding` property to set a tolerance when Layout XL compared parameter values during the Generating All Components from Source, Design Check Against Source, Layout Parameters Updates, and Schematic

© 2023 Cadence Design Systems, Inc.

Parameter Updates commands. To do this in the current release, use the paramTolerance environment variable instead.

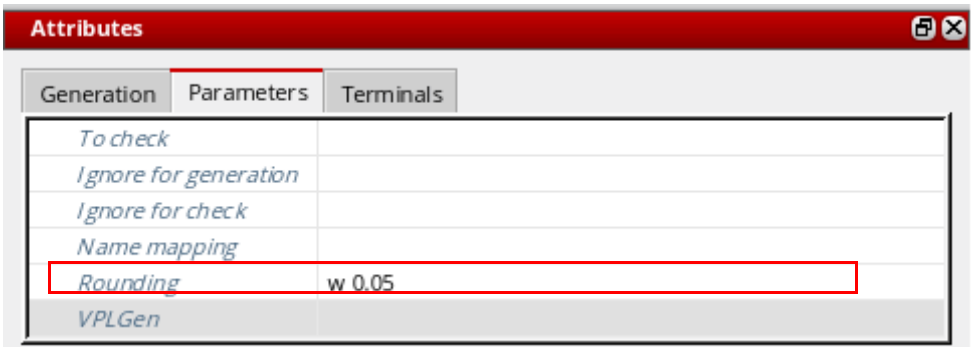
Rounding Cellview Parameter Values

To specify how a parameter value is rounded for a cell:

1. In the Configure Physical Hierarchy window, click the *Cells* tab.
2. Click the row containing the cell with the parameter value to be rounded.

   The row is highlighted and the options in the Attributes pane are enabled.
3. In the *Parameters* tab, type the *Rounding* argument you need in the field provided.

   For example, `(w 0.05 round)` rounds the value of a parameter called `w` to the closest multiple of 0.05 microns.



4. From the menu bar, choose *File – Save* to save the physical configuration view.

   The setting is saved for all instances of the cell in question.

Rounding Instance or Occurrence Parameter Values

To specify how an instance or occurrence parameter value is rounded:

1. In the Configure Physical Hierarchy window, click the *Instances* tab.
2. Click the row containing the instance with the parameter values to be rounded.

   The row is highlighted and the options in the Attributes pane are enabled.
3. Do one of the following.

© 2023 Cadence Design Systems, Inc.

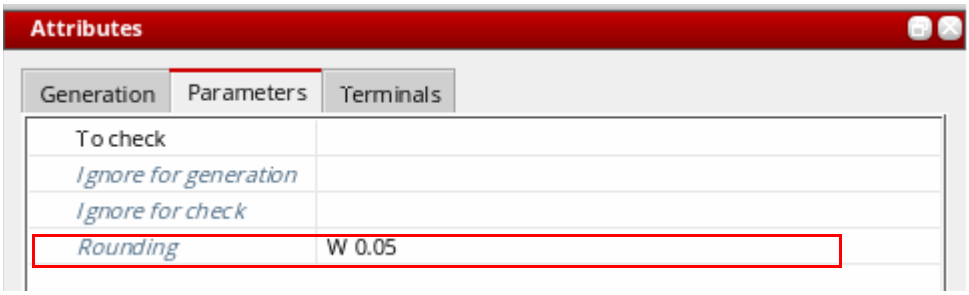
- To specify the rounding option for all the occurrences of the selected instance, ensure Occurrence mode is turned off in the toolbar.



- To specify the rounding option for an individual occurrence, ensure Occurrence mode is turned on.

4. In the *Parameters* tab, type the *Rounding* argument you need in the field provided.

   For example, `(w 0.05 round)` rounds the value of `w` to the closest multiple of 0.05 nanometers.



5. From the menu bar, choose *File – Save* to save the physical configuration view.

   The setting is saved for the specified instance or occurrence. If you specified an occurrence mapping the, the symbol is also changed to indicate this.

Related Topics

Syntax for Rounding Parameter Values

Rounding Instance or Occurrence Parameter Values

Instance Mode and Occurrence Mode

Syntax for Rounding Parameter Values

© 2023 Cadence Design Systems, Inc.

Syntax for Rounding Parameter Values

The *Rounding* definition comprises three parts: (*paramName value keyword*).

paramName

The name of the parameter whose value is to be rounded. Valid parameter names are

- Those specified by the transistorWidthParamNames environment variable.
- The width parameter name specified in the component type to which the cellview is assigned.

value

A floating point number. If the specified parameter is to be rounded, the value returned is a nonzero integer multiple of the *value*.

The default unit for this value is microns. To specify a different unit, use one of the identifiers listed in the table below.

Note: Layout XL never rounds to zero. If the closest rounded value is zero, no rounding takes place.

keyword

Specifies the rounding method used. Allowed values are

- `round` rounds to the nonzero integer multiple at or closest to the current value. If the rounded result is zero, Layout XL issues a warning and the original value is returned unchanged.
- `floor` rounds to the nonzero integer multiple at or below the current value. If the rounded result is zero, Layout XL issues a warning and the original value is returned unchanged.
- `ceil` rounds to the nonzero integer multiple at or above the current value. If the rounded result is large, Layout XL issues a warning inviting you to check the rounding value.

Note: Using `ceil` or `floor` with a float value that is not an integer can produce unexpected results.

© 2023 Cadence Design Systems, Inc.

Use the following identifiers to specify the rounding unit.

Identifier	Denotes	Factor	Example
`f`	femto	10^{-15}	`0.05f`
`p`	pico	10^{-12}	`0.05p`
`n`	nano	10^{-9}	`0.05n`
`u` (the default)	micro	10^{-6}	`0.05u` (or `0.05`)
`m`	milli	10^{-3}	`0.05m`
`c`	centi	10^{-2}	`0.05c`
`k`	kilo	10^{3}	`0.05k`
`M`	mega	10^{6}	`0.05M`
`G`	giga	10^{9}	`0.05G`

Related Topics

Rounding Instance or Occurrence Parameter Values

Instance Mode and Occurrence Mode

Syntax for Rounding Parameter Values

How to Control CPH with Schematic Properties

CPH now allows you to configure layout generation using the following types of properties:

- AEL expression
- NLP expression
- SKILL expression (ilExpr)

The other schematic properties previously converted to LAM or physConfig view will no longer be converted (up-revved) and CPH will honor all changes made to these properties.

To enable this feature, the shell environment variable "`CPH_USE_SCHEM_PROP=1`" should be turned on, which is by default off.

Note:

© 2023 Cadence Design Systems, Inc.

- The default value of "`CPH_USE_SCHEM_PROP`" is subject to change in future releases.
- If the evaluation of the property fails, the unevaluated property will be used.

Impact in handling incremental changes in schematic properties

- Changes to the lxProperties made through the property editor, the db SKILL interface or CDF callbacks impact the active CPH/Layout XL session.
- For the CDF Editor, only the base CDF Parameter modification impact the active CPH when they are saved.

Impact on subConfiguration

The subConfiguration is considered as a Hard IP. Information coming from higher hierarchies does not impact it. Therefore, the context needs to start from the last subConfiguration Root node on top of the node to evaluate.

Impact of changes on dependent properties

- ILExpr: no changes till next full re-elaboration. This can happen when CPH is launched or switch view list is changed.
- AEL/NLP: no changes till the node using this property is re-elaborated. This could possibly lead to inconsistency between occurrences of the same instance.

Impact on lxParamsToIgnore and lxParamsToIgnoreForCheck look-up

In accordance with 5.1.41 use model, the lxParamsToIgnore Property and lxParamsToIgnoreForCheck Property properties at the cell and instance levels are merged instead of overridden.

Note: Due to an implementation limitation, the ‘Schematic Instance’ property does not override the result of the look-up from the Instance Master to the PhysConfig Cell but is merged with it.

Related Topics

Library and Attributes Mapping File Syntax

© 2023 Cadence Design Systems, Inc.

Component Types Mode in CPH

A component type is the name given to that classification of layout cells which share common characteristics.

- *Component class* (one of *PMOS*, *NMOS*, *RESISTOR*, *CAPACITOR*, *INDUCTOR*, *STDCELL*, *STDSUBCONT*, *BOUNDARYCELL*, *FILLER, ROUTING, GUARDRING, NGAA, and PGAA*). This class is required to identify *NMOS* and *PMOS* transistor cells for folding, chaining, and assisted MOS and standard cell operations in Layout XL and the Custom Digital Placer.
- *Width parameter* name, *Length parameter* name, *Folding threshold*, *Active layer* and *View* names, and other associated component attributes

  Note: For FinFET devices—devices of component class *PFIN* or *NFIN*—the common characteristics useful for device chaining and folding are *Number of fins*, *Width per fin*, and *Folding threshold fins.*
- Names defined for *Source*, *Gate*, *Drain*, and *Bulk* terminals.
- (Virtuoso Advanced Node Option for GAA) In Layout EXL or higher tiers, *NGAA and PGAA* are used with Gate All Around transistor devices.

© 2023 Cadence Design Systems, Inc.

The following Virtuoso Layout Suite operations require cells to be assigned to a component type.

Application	Operation	Comments
Layout XL	Device folding	*Generate All From Source* needs the folding threshold to be set for instances. **Note:** ■ If the disableFolding environment variable is set and the folding threshold for the component type is set to `0`, folding is disabled. ■ For folding FinFET devices, the *Folding threshold fins* value must be set. *Generate All From Source*, *Generate Selected From Source*, and interactive folding require that the PMOS and NMOS transistors be defined. **Note:** If the design carries *PFIN* and *NFIN* FinFET devices, the components should be defined.
	Generate layout	*Generate All From Source*, *Generate Selected From Source,* and interactive chaining require that the PMOS and NMOS transistors be defined. **Note:** If the design carries *PFIN* and *NFIN* FinFET devices, the components should be defined.
	Determination of pseudoparallel nets	Uses the component type to access the width parameter, which, in turn, is used to confirm that the instances are identical.

© 2023 Cadence Design Systems, Inc.

Application	Operation	Comments
Custom Digital Placer	Placement of components in rows	Uses the *Component class* to correctly identify standard cells (*STDCELL*), substrate contacts (*STDSUBCONT*), boundary cells (*BOUNDARYCELL*), and filler cells (*FILLER),* and place them accordingly in rows.

Library Component Types and Design Component Types

There are two categories of component types.

- A *library-level* component type is defined in the library and attributes mapping (LAM) file for the library in question.

  Library-level component types are read-only; you can use the controls in the Configure Physical Hierarchy window to override a library-level component type, but you cannot change it.

  When you change the attributes of a library-level component type in the Configure Physical Hierarchy window, the system copies the library-level component type to a new design-level component type that you can edit. When you remove the design-level component type that you have edited, the overrides are removed, but the library-level component type remains.

  To define a new library-level component type, or modify or remove an existing one, you need to manually edit the LAM file for the design.

- A *design-level* component type applies only to the cells in your design. Where there is a conflict, a design-level component type overrides the library-level component type. However, you can add, modify, and remove design-level component types using the *Edit Component Types* command. This brings up the Configure Physical Hierarchy window in the Component Types Mode in the CPH Window mode.

Related Topics

Component Types Mode in the CPH Window

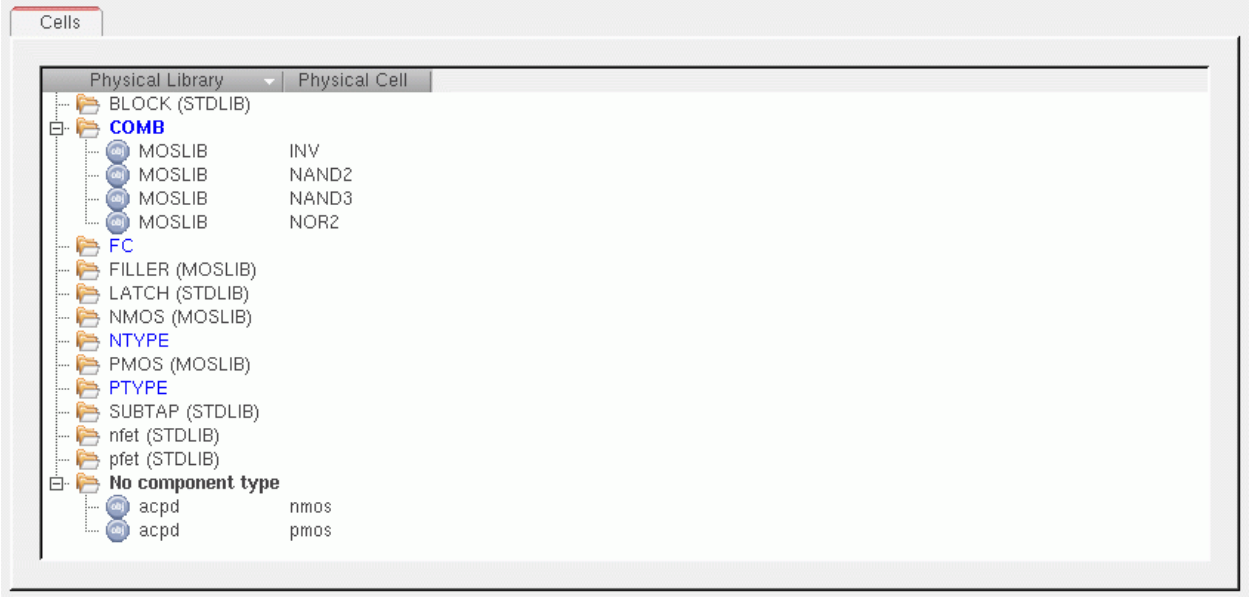
Soft Block Mode in the CPH Window

LAM Files and Configuration Views

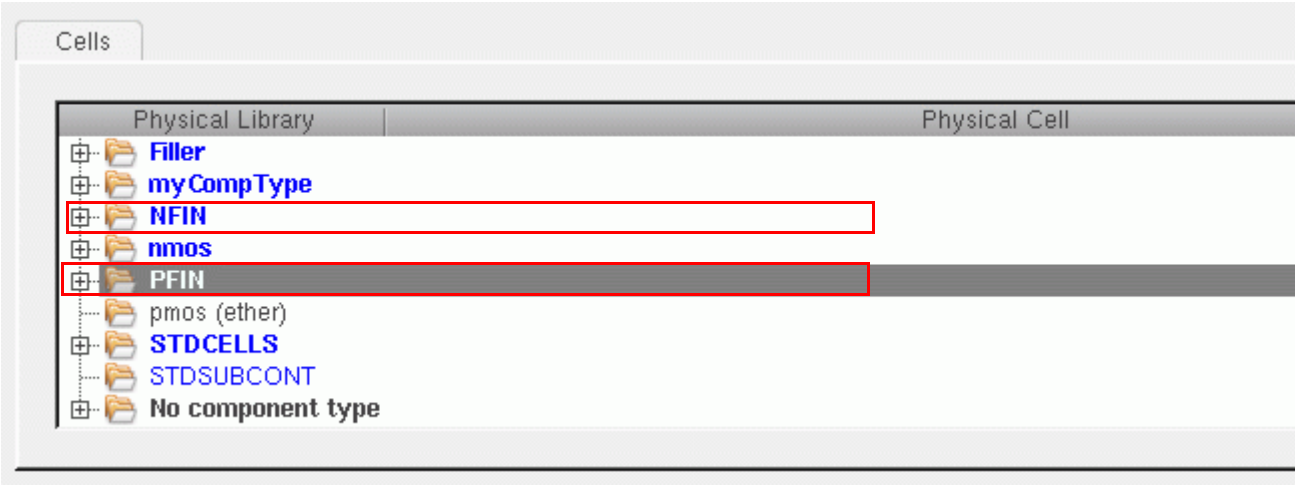
© 2023 Cadence Design Systems, Inc.

Component Types Cells Table

The *Cells* table in Component Types mode shows the cells in the design organized by component type.



To represent FinFET devices, the *Cells* table includes cells of the component type, *NFIN* and *PFIN*, as displayed in the figure below.



Each component type in the cells table is represented by a directory containing the cells that are assigned to it. This includes cells such as filler cells that have only a physical representation but no logical representation. If a cell is not assigned to a particular component type, it is put in the *No component type* directory.

© 2023 Cadence Design Systems, Inc.

The names of design-level component types are in blue; library-level ones in black. If a component type has cells assigned to it, its name is in bold type of the appropriate color.

By default, only the component types for the cells in the current design are shown. You can filter the cells that are shown on a per-library basis using the *Show cells* command on the shortcut menu. The *Physical library list* is prepended to the list of libraries from your library definitions file in order to build the correct library order.

You can perform the following operations directly in the table view.

- To add a new component type, right-click and choose *Add component type*. Use the Create Component Type dialog to specify a name for the new component type.
- To remove a component type, select the component type you want to remove, right-click and choose *Remove component type*. You can remove only design-level component types. Evicted cellviews are put into the *No component type* directory.
- To move cells between component types, select the cells to move, right-click and choose *Move cells*. Use the Move Cells dialog box to select a new component type for the selected cells.

Resolution of Conflicts in Component Type Names

When the same component type name exists in multiple LAM files, the first definition found in the (*Physical library list*, `cds.lib`) list is the one that is used, as illustrated in the examples below. Component type definitions shown in bold are used; those in normal type are disregarded.

```
lib list: lib1, lib2, lib3

lib3:    CT1(lib3/C1)   CT2(lib3/C2)           CT3(lib3/C3 )
lib2:                   CT2(lib2/C1 lib2/C2)
lib1:                                          CT3(lib1/C1 lib1/C2)

effective LAM: CT1(lib3/C1) CT2(lib2/C1 lib2/C2) CT3(lib1/C1 lib1/C2)
```

Conflicting component type settings are resolved, as indicated below.

```
lib list: lib1, lib2, lib3

lib3:    CT1(lib1/C1)   CT2(lib1/C2)   CT1(lib1/C3)
lib2:                   CT2(lib1/C2)
lib1:                                  CT3(lib1/C1)

effective LAM:  CT1:lib1/C3 CT2:lib1/C2 CT3:lib1/C1
```


© 2023 Cadence Design Systems, Inc.

To edit the attributes for a particular component type, you must select that component type in the table view. The settings you specify in the Attributes pane are applied to all the cells assigned to that component type.

Related Topics

Defining a Design-Level Component Type

Removing a Component Type

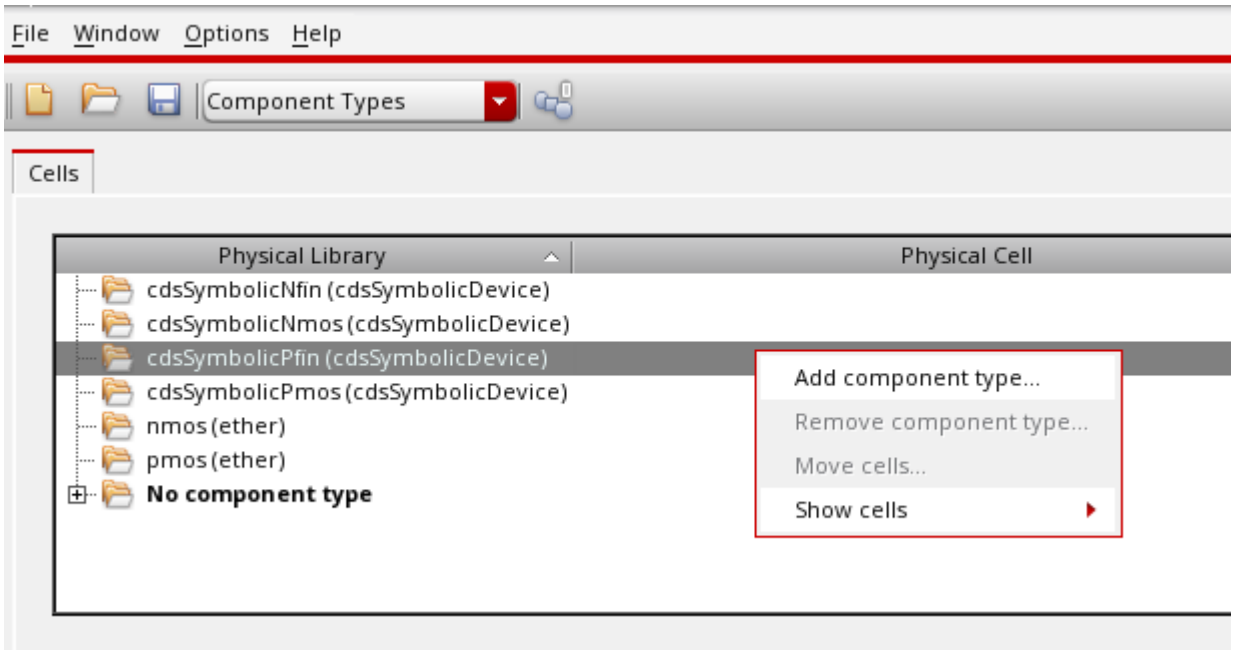
Defining a Design-Level Component Type

Defining a Design-Level Component Type

To define a design-level component type for the current design and assign cells to it,

1. From the layout window menu bar, choose *Edit – Component Types*.

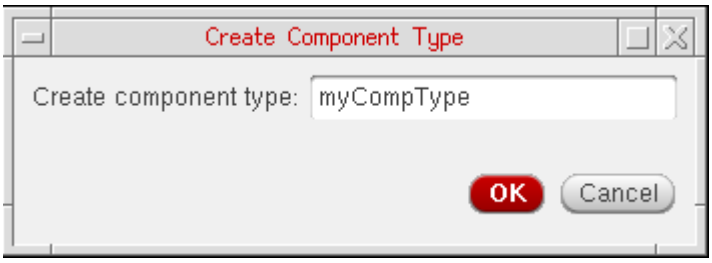
   The Configure Physical Hierarchy window is opened in Component Types mode.



2. In the *Cells* table, right-click and choose *Add component type*.

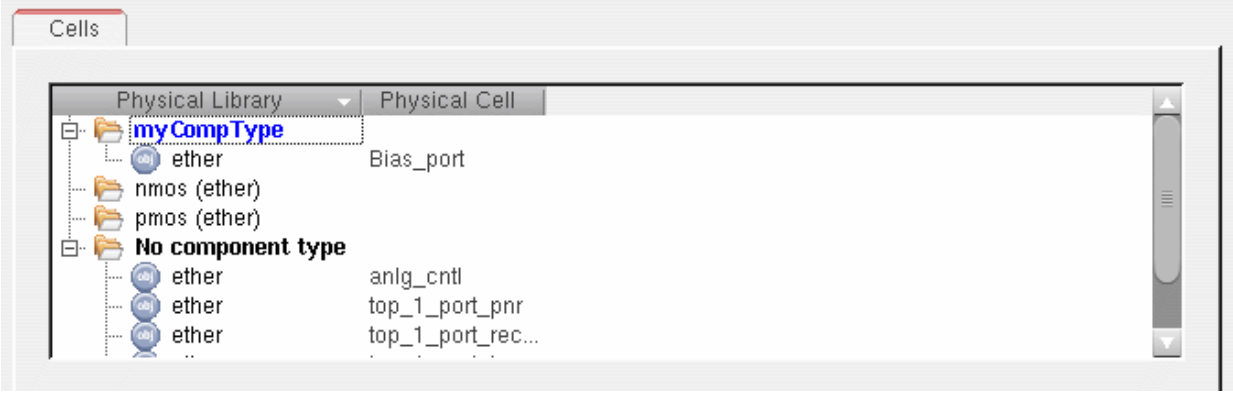
© 2023 Cadence Design Systems, Inc.

The Create Component Type dialog is displayed.



3. Type the name of the new component type in the text field and click *OK*.

   The system creates a directory for the new component type in the *Cells* pane. Component type folders with cells assigned to them are picked out in bold text. User-defined component types are labeled in blue.



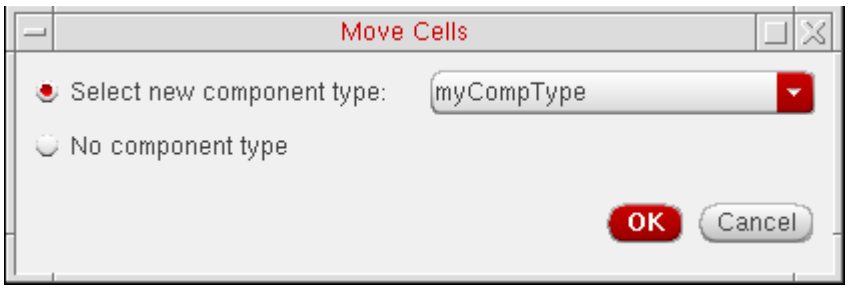
4. Select the new component type in the table.

   The options in the Attributes pane are enabled.

5. Set the attributes you require for the component type.

6. In the *Cells* pane, select the cells you want to assign to this component type, click the right mouse button, and choose *Move cells*.

   The Move Cells dialog is displayed.



© 2023 Cadence Design Systems, Inc.

7. Select the component type you want to move the cells to from the drop-down list and click *OK*.

   The selected cells are moved to the specified component type.

8. Choose *File – Save* to save the component type definition.

Related Topics

Component Types Mode in the CPH Window

Configure Physical Hierarchy Window

Modifying a Component Type



Important

You cannot use the *Edit Component Types* command to change a library-level component type, but you can override the settings it contains.

To modify a component type,

1. From the layout window menu bar, choose *Edit – Component Types*.

   The Configure Physical Hierarchy window is opened in Component Types mode.

2. Select the component type you want to modify.

   The options in the Attributes pane are enabled.

3. Set the attributes you require for the component type.

   If you change the *Folding threshold*, you must run *Generate All From Source* for the changes to take effect.

4. Choose *File – Save* to save the changes.

Related Topics

Component Types Mode in the CPH Window

Configure Physical Hierarchy Window

© 2023 Cadence Design Systems, Inc.

Removing a Component Type

To remove a design-level component type from your design,

1. From the layout window menu bar, choose *Edit – Component Types*.

   The Configure Physical Hierarchy window is opened in Component Types mode.

2. Select the component type you want to remove, click the right mouse button, and choose *Remove component type*.

   - Cells that were originally assigned to a different design-level component type are reassigned to the *No component type* directory.
   - Cells that were originally assigned to a library-level component type (i.e., one that is defined in the LAM file) are reassigned to the component type defined for them in the LAM file and are no longer displayed in the *Cells* pane.

3. Choose *File – Save* from the Configure Physical Hierarchy window menu bar to save the changes.

Related Topics

Component Types Mode in the CPH Window

Configure Physical Hierarchy Window

Soft Block Mode

Soft Block mode lets you define the soft blocks that will be created by the *Floorplan – Generate Physical Hierarchy* or the *Generate Abstract From Symbol* commands. You select the instances to be included and define boundary, pin, and blockage parameters for the block you want to create. You can also remove the parameters from an existing soft block to prevent it being created when the physical hierarchy is generated.

Generate Physical Hierarchy (GPH) is the equivalent of a hierarchical *Generate All From Source* command. It generates instances from source at different levels of hierarchy, including soft blocks that are defined with type *LAYOUT*.

In order for the bus terminal and bus bit order (ascending or descending) information to be transferred from the schematic to the layout, you need to set the environment variable, `createImplicitBusTerminals`, before running GPH. This creates implicit bus terminals and bus order information in the layout view, derived from the bus terminals and bit order

© 2023 Cadence Design Systems, Inc.

information in the schematic cellview. In addition, this avoids the need to run verilogAnnotate before referencing the layout cellview when:

- Running verilog2oa
- When importing another verilog design into Innovus

See createImplicitBusTerminals.

Further, for the implicit bus information to be carried to the abstract, the abstract must be generated with the bus annotation option enabled. This can be achieved by setting the `AnnotateBusInAbstract` option to “True” in the *Abstract Options* file.

See Bus Annotation in Abstract Generator.

Unlike the GPH command, *Generate Abstract From Symbol* command generates soft blocks that are defined with type *ABSTRACT*.

Use *Configure Physical Hierarchy* in Soft Block mode to define soft blocks before initial generation or if you need to make more radical changes (by, for example, generating new soft blocks or flattening others that you no longer require). If you want to make changes to existing soft blocks that do not require the physical hierarchy to be regenerated, use the *Edit Soft Blocks* command instead. See Soft Block Editing.

You define a soft block by selecting a component in the soft block *Instances* table and specifying the following in the Attributes pane.

- The type of soft block that will be created.
- The height, width, area, or shape of the place and route boundary.
- The length, width, layer purpose, and signal type of soft pins.
- Routing and placement obstructions inside the block.

Alternatively, you can load soft block parameters directly from another layout cellview using the *Initialize Soft Block Parameters Using Physical View* command.

You also need to define parameters for the top-level instance so that the *Generate Physical Hierarchy* command has the information it requires to generate the boundary and pins. The top level instance has type *LAYOUT*, which cannot be changed.

Related Topics

Setting the Cell Type

© 2023 Cadence Design Systems, Inc.

Defining Soft Block Parameters

Specifying the Boundary

Specifying I/O Pins

Specifying LayerHalo Obstructions

Specifying Cover Obstructions

Loading Soft Block Parameters from Another Cellview

Saving and Loading Soft Block Parameters to a Floorplan File

Generating the Physical Hierarchy

Creating a Physical Binding

Soft Block Instances Table

The *Instances* table in the Soft Block mode lets you see how your logical design is currently partitioned and helps you decide whether you can gain further efficiencies by generating more or fewer soft blocks, or by changing the shape or internal configurations of existing soft blocks.

For each instance in the design, the table shows the view list used to reach the schematic instance and the corresponding layout cellview found. It also shows the type of block and view that will be created when you use the *Floorplan – Generate Physical Hierarchy* command.

If a schematic instance is set to be ignored, it is not displayed in the table. Instances under a valid soft block instance are in gray text, indicating that although the binding has been set up, it is never reached because of the physical stop point at a higher level. Red text in a row means that there is no physical representation for the cell in question.

There are a number of operations you can perform directly in the table view.

- To sort the tree alphabetically by a particular column within each node, click the header of the column you want to sort. Alternate clicks sort in ascending and descending order.
- To define soft block attributes, click the right mouse button on the soft block in question and choose *Define Soft Block Parameters*. This enables the controls in the Attributes pane.

© 2023 Cadence Design Systems, Inc.

- To reset soft block attributes, click the right mouse button on the soft block in question and choose *Remove Soft Block Parameters*. This enables the controls in the *Attributes* pane.
- To edit the entries in any of the fields, double-click the text you want to change and either type into the field directly or choose a value from the drop-down list.

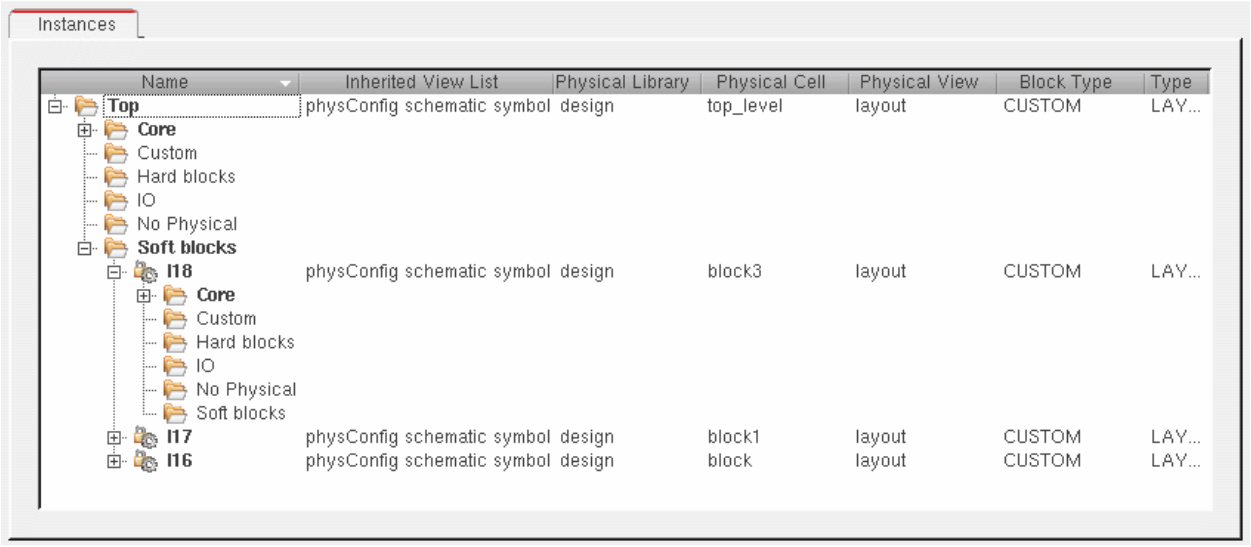
Note: You can also select a soft block instance from the table and view its attributes in the Attributes pane. To edit the shape or size of the soft block boundary or to modify the pin configuration or the internal routing obstructions of the soft block, right-click the soft block and select *Edit Soft Block Parameters*. This makes the corresponding soft block parameters in the Attributes pane available for editing.

Related Topics

Soft Block Mode in the CPH Window

Soft Block Table Folders

When you move into the Soft Block mode, the instances in the design are automatically moved into one of the *Core*, *Custom*, *Hard block*, *IO*, and *Soft block* folders based on the existence of one of the physical views specified in the *Physical stop list*.



Note: If a soft block has already been generated, it will be listed under *Soft block*. If it does not exist yet, it is displayed under *No Physical*.

© 2023 Cadence Design Systems, Inc.

By default, instances with no physical representation are put into the *No Physical* directory. If you change the *Physical Library*, *Physical Cell*, and *Physical Cell* to point to a cellview that does exist, the cellview is put into the appropriate directory based on its cell type.

In the *No Physical* directory:

- The *Physical Library*, *Physical Cell*, and *Physical Cell* fields indicate only the potential binding for the instance in question.

  The *Generate Physical Hierarchy* command flattens the instance because there is no binding defined.

- If you change the *Physical Library*, *Physical Cell*, and *Physical Cell* to point to a cellview that does not exist, the instance remains in the *No Physical* directory.

  The *Generate Physical Hierarchy* command attempts to find the specified view during layout generation; however, the view does not exist and the logical node is flattened instead.

  To clear the physical binding, click the right mouse button and choose *Clear Physical Binding*.

- If you define soft block parameters for any instance in the *No Physical* directory, the *Generate Physical Hierarchy* command generates the appropriate view and uses it during layout generation.

  To clear the physical binding and remove the soft block parameters, click the right mouse button and choose *Clear Physical Binding*.

  To remove any soft block parameters but preserve the physical binding, click the right mouse button and choose *Remove Soft Block Parameters*.

Related Topics

Soft Block Mode in the CPH Window

Setting the Cell Type

Cell types are used to categorize the blocks in your design.Valid settings for soft blocks are.

Soft Block	*blockBlackBox*	type ABSTRACT
	softMacro	type LAYOUT

© 2023 Cadence Design Systems, Inc.

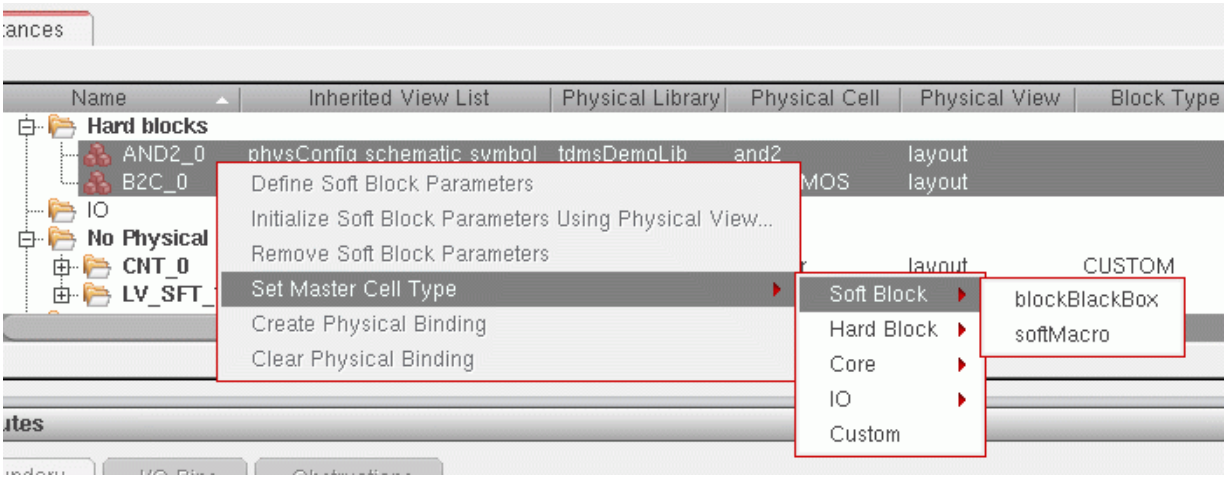
The cell type is automatically set to the appropriate value for any existing soft blocks in the design. To set the cell type for a soft block that has not yet been generated:

1. In the *Instances* table, select the instance for which you want to set the cell type.

   Note: To select multiple instances, keep the `Ctrl` key pressed while you select the instances.

2. Click the right mouse button and choose *Set Master Cell Type*.

   A submenu opens showing the cell types you can set.



3. Set the cell type of your soft block to either *blockBlackBox* or *softMacro* as required.

   The instance is moved into the *Soft blocks* bin, unless it was stored there already, and the following information message, CPH-1014, is displayed in the CIW:

   CellType for the Lib/Cell/View, '%s/%s/%s', has been changed to '%s'

Irrespective of the cell type you choose to apply for the selected instance, the information message is displayed when the master cell type has been set.

Related Topics

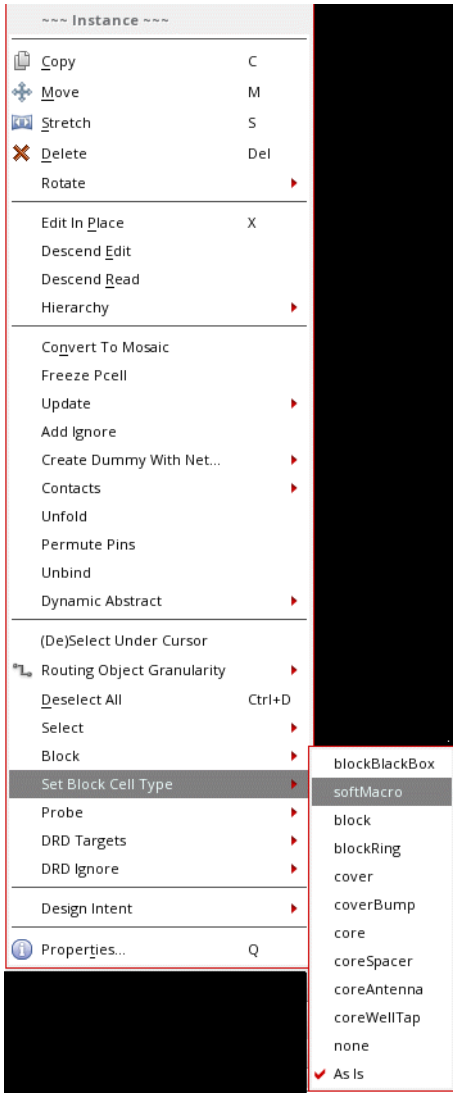
Soft Block Mode in the CPH Window

© 2023 Cadence Design Systems, Inc.

Setting the Cell Type using the Shortcut Menu

You can use the shortcut menu to set the cell type for blocks. To do this:

1. Select the required blocks in the design canvas.
2. Right-click to display the shortcut menu.



3. Choose *Set Block Cell* Type. A sub-menu that lists all available block types is displayed.
4. Select the required block type.

The cell types for the selected blocks are updated.

© 2023 Cadence Design Systems, Inc.

With multiple blocks selected, if all the blocks have the same cell type, a check mark is displayed beside the cell type in the shortcut menu. If the selected blocks are of different cell types, a check mark is displayed beside *As is*.

Related Topics

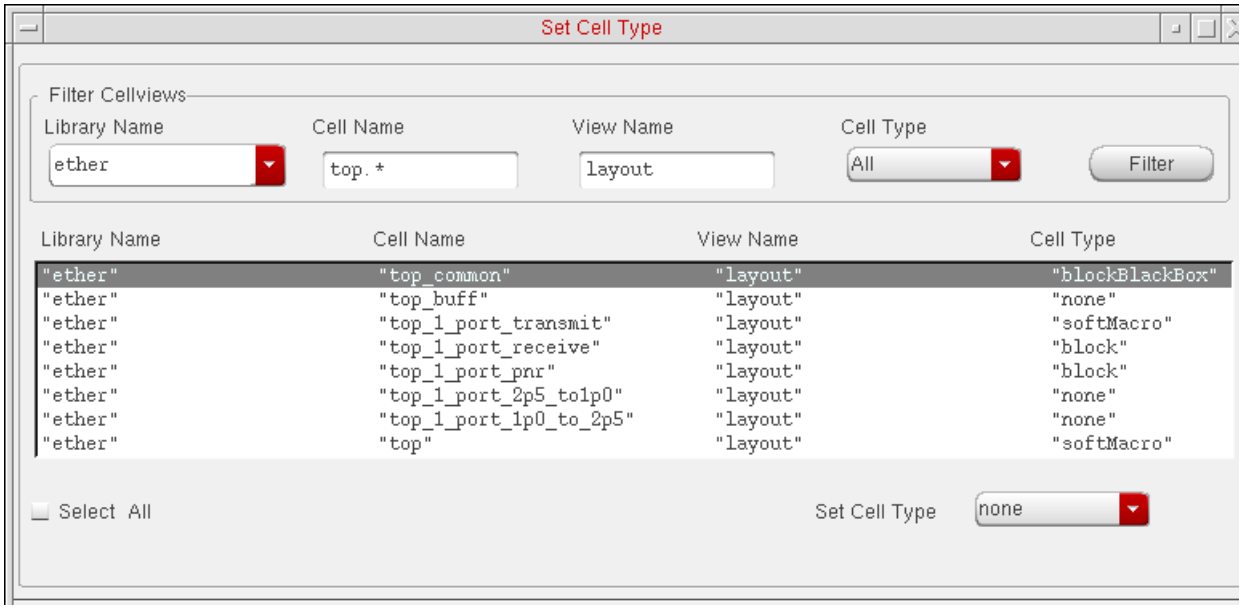
Soft Block Mode in the CPH Window

Setting the Cell Type from the CIW

You can set the cell type for multiple instances using the *Tools – Set Cell Type* command from the CIW menu bar.

1. From the CIW, choose *Tools – Set Cell Type*.

   The Set Cell Type form is displayed.



2. To find the cells you want to set the cell type for:

   a. Select a *Library Name* from the cyclic list.

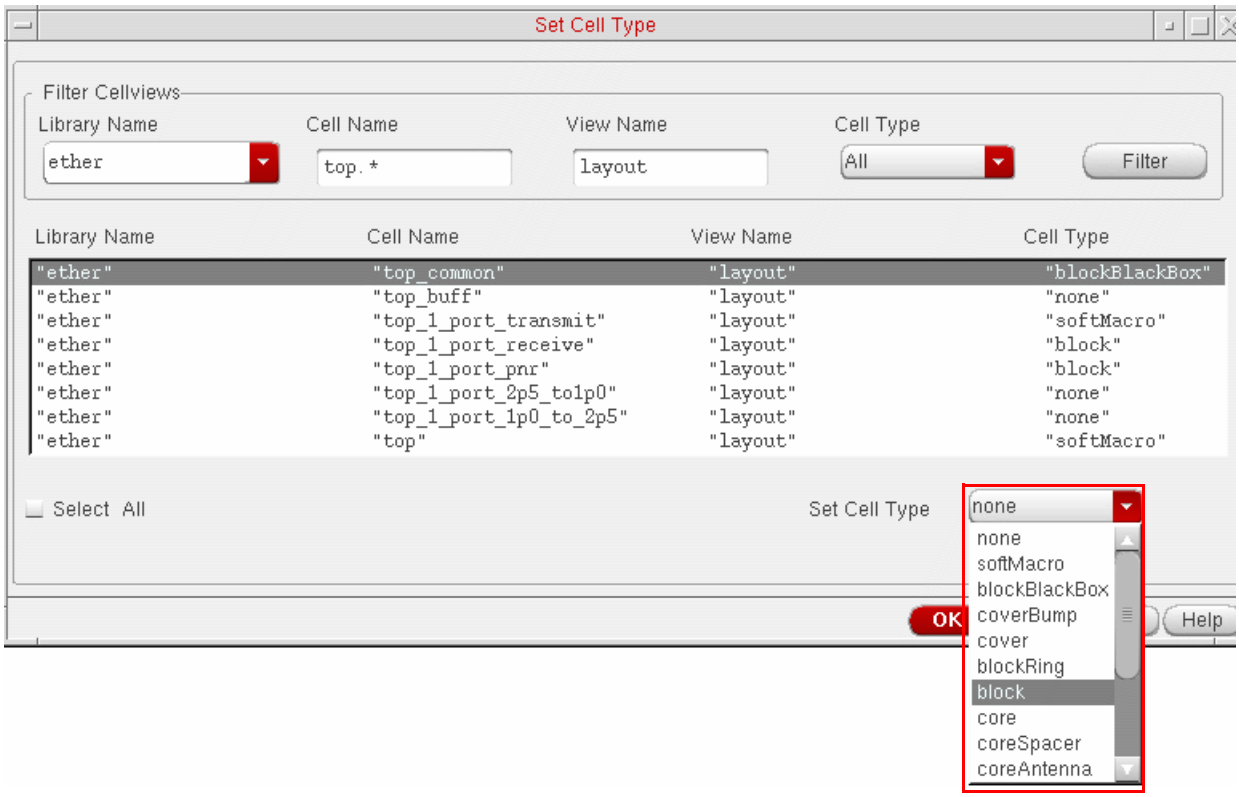
   b. Type the strings you want to filter by in the *Cell Name* and *View Name* fields and select a *Cell Type* from the drop-down list.

   Note: For performance reasons, the *View Name* field does not accept regular expressions.

© 2023 Cadence Design Systems, Inc.

c. Click *Filter* to filter the list to contain only those cells that match the options settings.

3. Select the cells you want from the list and choose the cell type you want to set from the *Set Cell Type* drop-down list.



4. Click *Apply* or *OK*.

The specified cell type is set for the selected cells.

Related Topics

Soft Block Mode in the CPH Window

Defining Soft Block Parameters

Use the *Define Soft Block Parameters* command to define the boundary, pins, and blockage parameters for a selected instance in the schematic design. These parameters are used by the *Generate Physical Hierarchy* command to create soft blocks for these instances in the physical implementation.

© 2023 Cadence Design Systems, Inc.

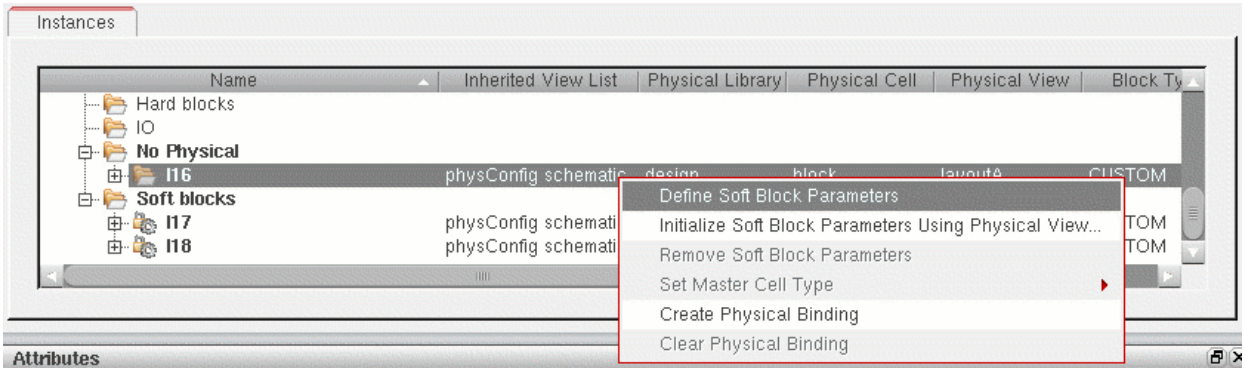
When the *Define Soft Block Parameters* command is running, only the controls in the Attributes pane are enabled, ensuring you do not change the instance selection while you are editing the parameters. All other controls in the Configure Physical Hierarchy window remain disabled until you either save or discard the changes you have made to the soft block parameters. Similarly, cross-selection from other Virtuoso windows or assistants is delayed until you save or discard your changes.

To make changes to existing soft blocks that do not require the physical hierarchy to be regenerated, use the *Edit Soft Blocks* command instead. See Soft Block Editing.

To define soft block parameters,

1. Select a component in the *Instances* table.
2. Right-click the instance and choose *Define Soft Block Parameters*.

   If the selected component already has soft block parameters defined, the command is labeled *Edit Soft Block Parameters*.

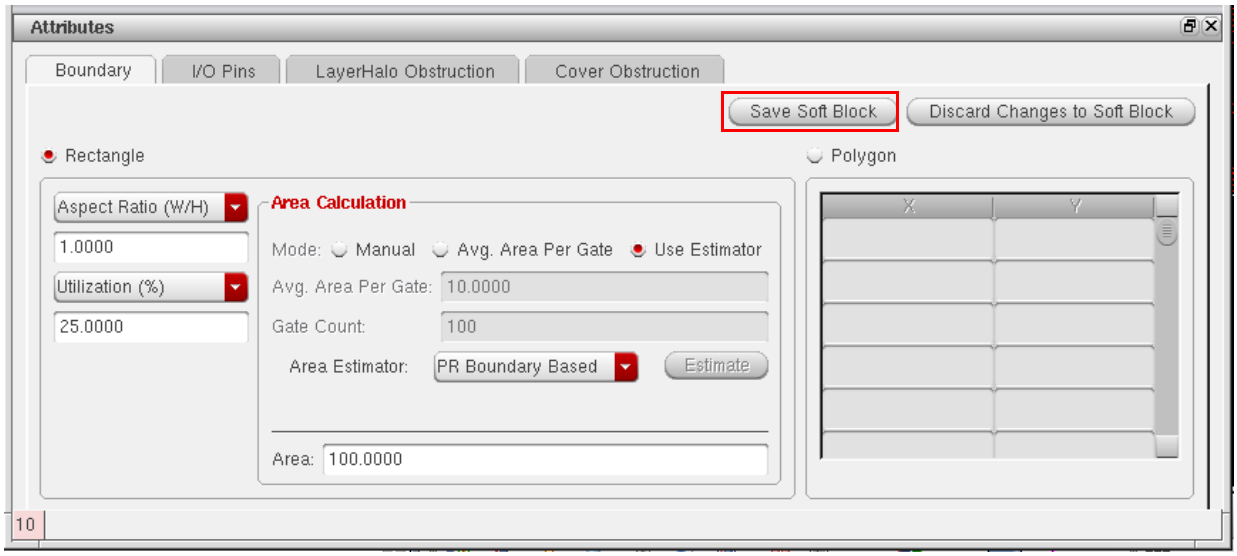


   The *Instances* table is disabled and the Attributes pane enabled,

3. Edit the parameters you require for your soft block. See
   - Specifying the Boundary
   - Specifying I/O Pins
   - Specifying LayerHalo Obstructions
   - Specifying Cover Obstructions

© 2023 Cadence Design Systems, Inc.

4. To save the parameters for the selected component, click *Save Soft Block* in the Attributes pane.



   The Attributes pane is disabled and the *Instances* table re-enabled.

   To discard any changes you made since the Attributes pane was enabled, click *Discard Changes to Soft Block*.

5. Repeat steps **1**-**4** for all the soft blocks you want to define.

6. From the Configure Physical Hierarchy menu bar, choose *File – Save Floorplan File* to save all your settings to a floorplan file for later re-use.

   See Saving and Loading Soft Block Parameters to a Floorplan File.

Related Topics

Soft Block Mode in the CPH Window

Specifying the Boundary

The *Boundary* tab lets you specify the shape and size of the boundary that is created when the soft block is generated. This topic describes how to create the most common types of boundary.

- Creating a Fixed Rectangular Boundary
- Creating a Non-Fixed Rectangular Boundary

© 2023 Cadence Design Systems, Inc.

- Creating a Polygonal Boundary

It also describes the area estimation functions you can use to determine the size of the boundary, and explains how you can define and register your own area estimation functions.

- Creating and Registering a User-Defined Area Estimation Function

Related Topics

Component Types Mode in the CPH Window

Creating a Fixed Rectangular Boundary

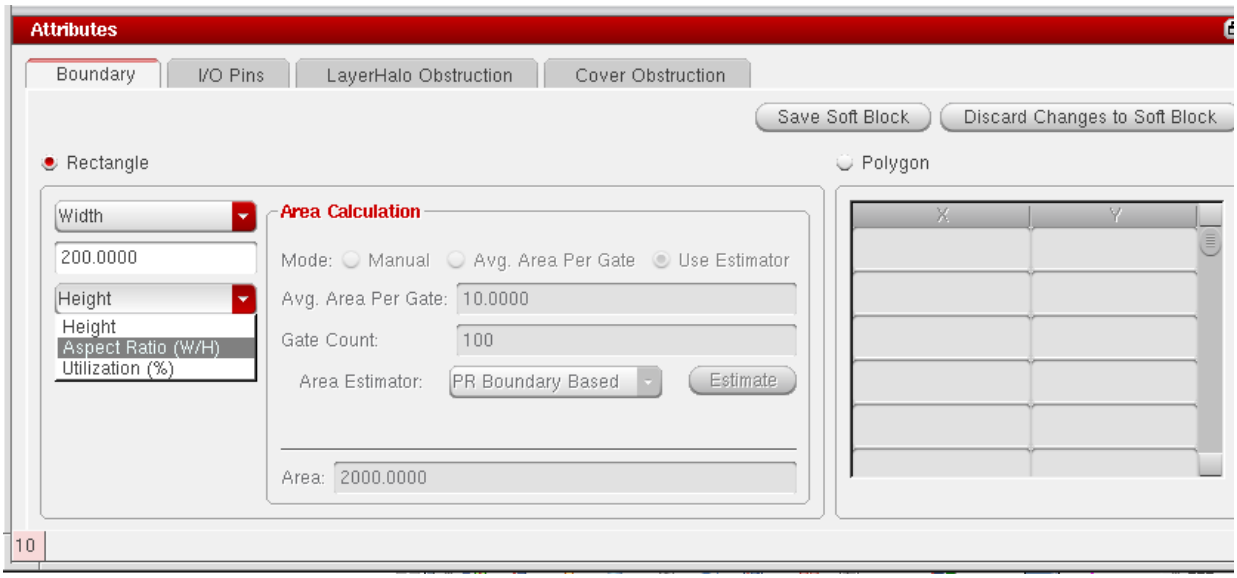
Fixed refers to a boundary defined by specifying a combination of width, height, and aspect ratio. The boundary is considered fixed because it is defined by two fixed variables, and is not derived by estimating the area of the block.

To create a fixed rectangular boundary:

1. Select the *Rectangle* radio button in the *Boundary* tab.
2. Choose one of the following.
   - *Aspect Ratio (W/H)* in the first cyclic field and *Height* in the second. In this case, the boundary width is calculated from the boundary height and aspect ratio.
   - *Aspect Ratio (W/H)* in the first cyclic field and *Width* in the second. In this case, the boundary height is calculated from the boundary width and the aspect ratio.
   - *Width* in the first cyclic field and *Height* in the second. In this case, the values you type in are used to create the boundary.

© 2023 Cadence Design Systems, Inc.

The *Area Calculation* group box is disabled, indicating that area estimation is not required.



3. Type the value you require into each of the fields provided.
4. Type a value in the *Rail Height* field, if required.

   Note: The aspect ratio, width, and height values determine the core area, whereas the rail height value is used to extend the total area to accommodate the rails at the top and bottom of the core area. By specifying a rail height in this form, you do not need to calculate the rail area explicitly before populating the GUI.
5. Click *Save Soft Block*.

   The boundary parameters are set. The Attributes pane is disabled and the Instances table re-enabled.

Related Topics

Soft Block Mode in the CPH Window

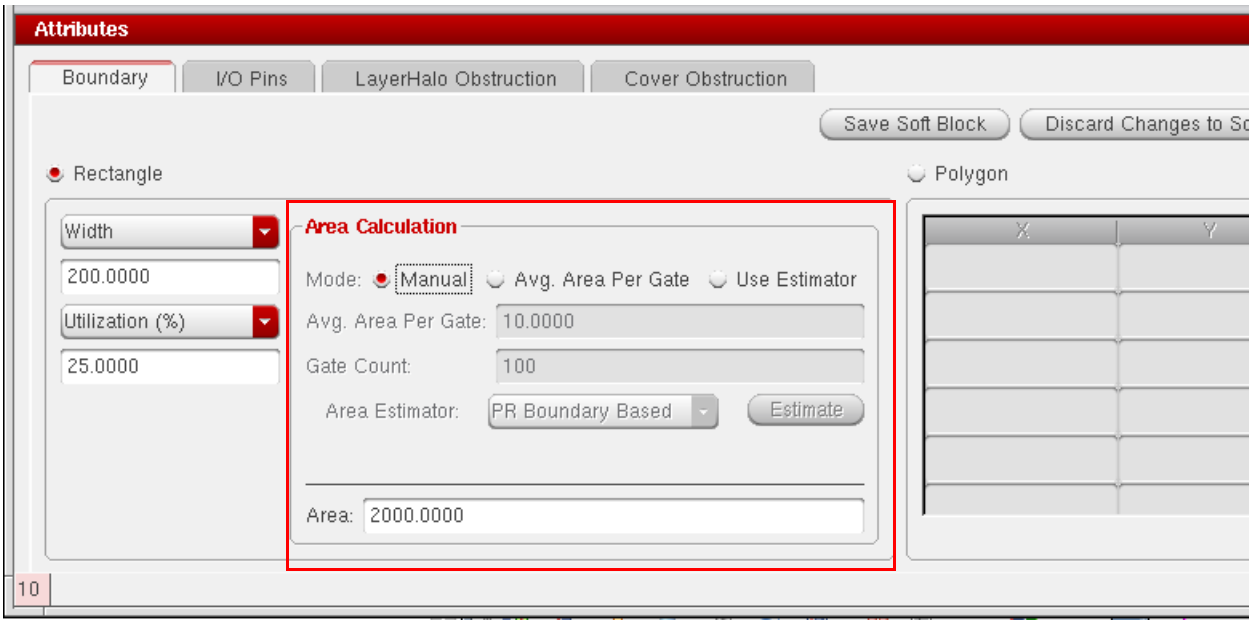
Creating a Non-Fixed Rectangular Boundary

Non-fixed refers to a boundary defined by estimating the area required by the block in question. The boundary is non-fixed because it is derived from only one fixed parameter (such as the height or width), an area utilization factor, and an area estimation function.

© 2023 Cadence Design Systems, Inc.

To create a non-fixed rectangular boundary,

1. Select the *Rectangle* radio button in the *Boundary* tab.
2. Choose either *Width* or *Aspect Ratio (W/H)* from the first cyclic field and *Utilization* from the second cyclic field.



The *Area Calculation* group box is enabled, indicating that area estimation is required.

3. Choose the area estimation method you prefer:
   - *Manual* lets you directly type the value you want into the *Core Area* field.
   - *Avg. Area Per Gate* lets you specify the gate count and the average area per gate. The system calculates the area by multiplying the two values.
   - *Use Estimator* lets you choose between *PR Boundary Based* and *BBox Based* area estimator. The PR Boundary Based estimation sums up the polygonal area of the individual instances to derive the overall area estimation. The BBox Based estimation, on the other hand, sums up the minimum bounding box area of the individual instances to derive the overall area estimation.

     You can also register your own area estimation functions and use them, as appropriate. For more information, see Creating and Registering a User-Defined Area Estimation Function.
4. (Optional) To modify any user-defined parameters for the selected area estimation function, click *Estimate*.

© 2023 Cadence Design Systems, Inc.

5. Click *Save Soft Block*.

   The boundary parameters are set. The Attributes pane is disabled and the *Instances* table re-enabled.

Related Topics

Soft Block Mode in the CPH Window

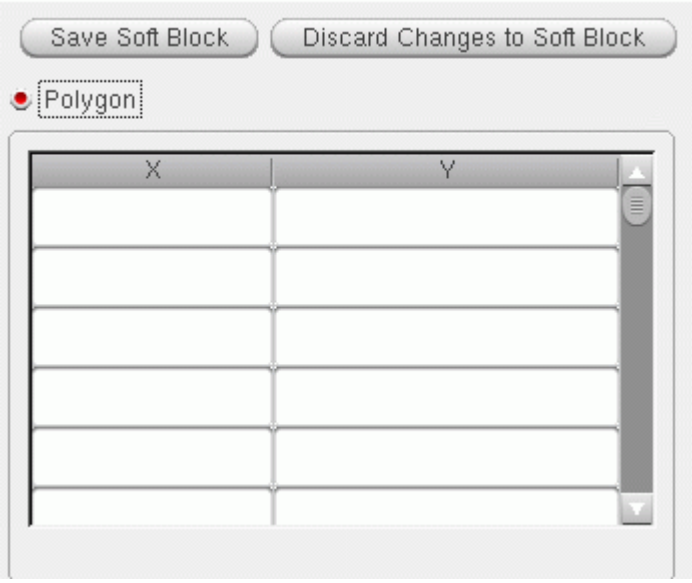
Creating a Polygonal Boundary

To define the boundary parameters required to create a soft layout or soft abstract with a polygonal boundary,

1. Select the *Polygon* radio button in the *Boundary* tab.

   The table on the right is enabled. All other controls on the tab are disabled because no area estimation is required.

2. Type in the coordinates of the polygonal boundary – the x-coordinates into the left-hand column, the y-coordinates into the right-hand column.



3. Click *Save Soft Block*.

   The boundary parameters are set. The Attributes pane is disabled and the *Instances* pane re-enabled.

© 2023 Cadence Design Systems, Inc.

Creating and Registering a User-Defined Area Estimation Function

You might write a simple area estimator as follows.

```
procedure(myFunction(maskLayoutId @optional (contactArea  1.0)
  prog((area)
    // Function body to calculate area involving layout instances
    foreach(ins maskLayoutId~>instances
      area = area + ins~>master~>prBoundary~>area * contactArea
    )
    return(area)
  )
)
```

You register the function as shown below.

```
leRegAreaEstimator(
    'myFunction
        "simpleAreaEst"
        '('cellView)
        '((contactArea 1.0))
        "deferred"
        )
```

Where `deferred` means that the function is applied only when all the required information is available. In this example, the layout instances will be available only when the *Generate All From Source* command is run, and only then will the area estimator be executed to calculate the area.

Alternatively, you can register the function as shown below.

```
leRegAreaEstimator(
    'myFunction
        "simpleAreaEst"
        '('cellView)
        '((contactArea 1.0))
        "direct"
        )
```

Where `direct` means the function will be applied as soon as you click *Estimate* in the *Boundary* tab. The result is shown in the *Area* field.

When registered, the area estimator function appears in the *Area Estimator* list in the *Boundary* tab of the Configure Physical Hierarchy window in Soft Block mode.

You can also add an area estimation function to the *PR Boundary* tab of the Generate Layout form. See Adding an Area Estimator Function for Layout Generation.

For more information on registering an area estimation function, see leRegAreaEstimator in the *Virtuoso Layout Suite SKILL Functions Reference*.

© 2023 Cadence Design Systems, Inc.

Related Topics

Soft Block Mode in the CPH Window

Adding an Area Estimator Function for Layout Generation

To add an area estimator function to the Generate Layout form:

1. In the PR Boundary tab, click the *Register* button in the Area Estimation group box.

   The Add Area Estimators form opens.

2. In the CIW, type the command given below to load the SKILL file that defines the area estimator function:

   ```
   load("area.il")
   ; where area is the name of the SKILL file.
   ```

3. In the *Function name* field, type the name of the area estimator function defined in the SKILL file.

4. In the *Nickname* field, specify the name to use for registering the area estimator function. The name specified in this field will be displayed in the area estimator drop-down list after the function is registered.

5. Select the appropriate value from the *Type* drop-down list. Choose *PR boundary* to register the function for PR boundary area estimation. Choose *Area boundary* to register the function for the estimation of virtual hierarchy area boundary.

6. In the *Parameters* field, specify any parameters to be considered when calculating the area estimates.

   The area estimator function registered for PR boundary appears in the *Estimator* drop-down list with the same name as specified in the *Nickname* field.

   The area estimator function registered for virtual hierarchy area boundary appears in the drop-down list associated with the *Soft Block – Area* field. The registered function is listed using the same name as specified in the *Nickname* field. For more information, see Adjust Area Boundary form in *Virtuoso Design Planner User Guide*.

Related Topics

Add Area Estimators Form

Adjust Area Boundary

Soft Block Mode in the CPH Window

© 2023 Cadence Design Systems, Inc.

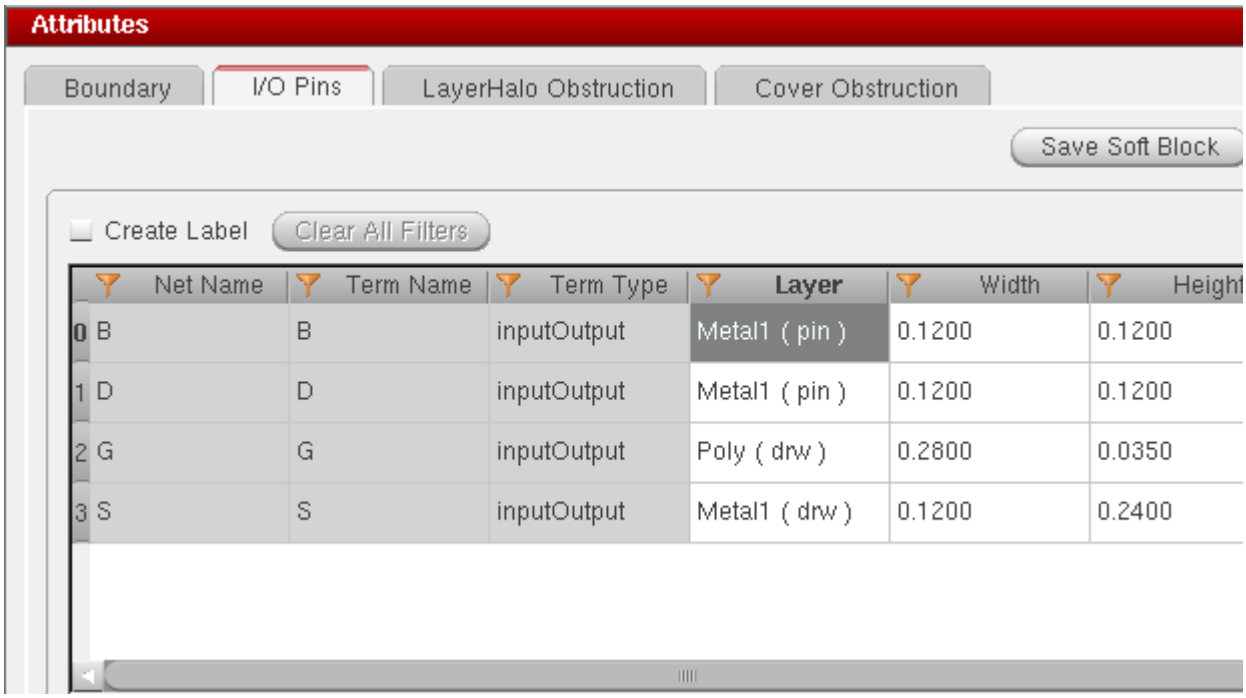
Specifying I/O Pins

In addition to defining layer-based rules, you can define layer purpose based rules while defining the soft block parameters in the *I/O Pins* tab.

To specify I/O pins for the selected soft block:

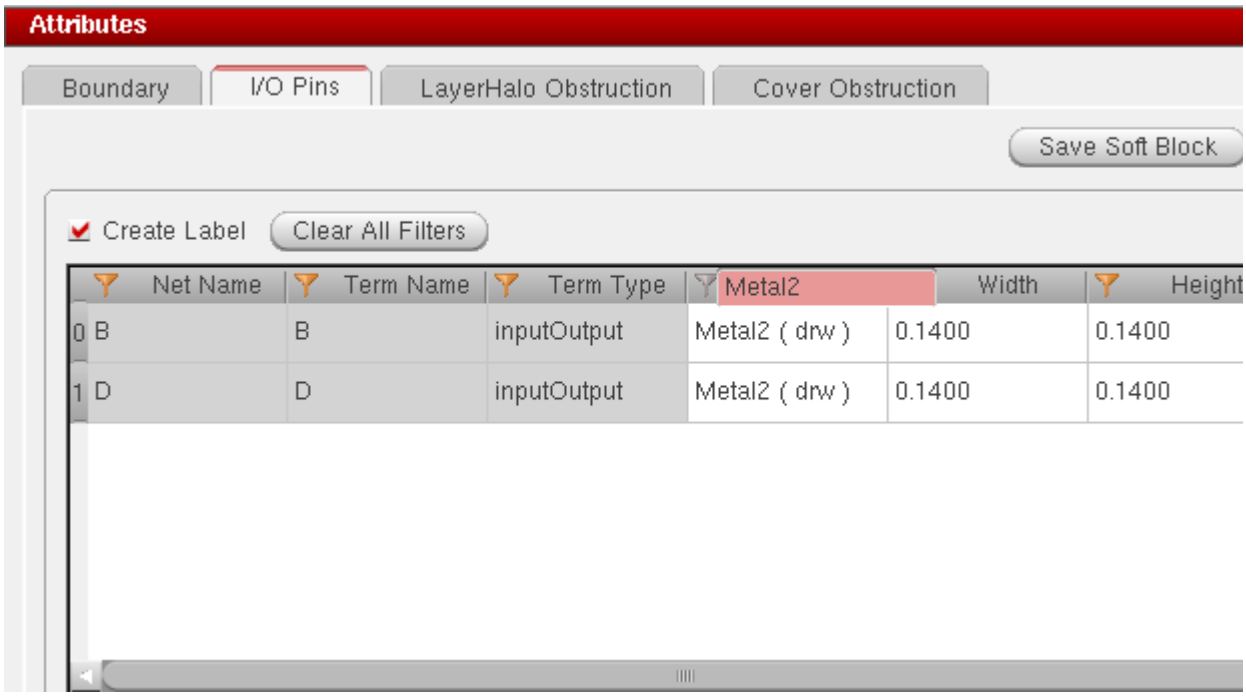
1. Click the *I/O Pins* tab to see the pins in the design.

   CPH honors existing net process rule override (net PRO) definitions in the schematic and displays the values for corresponding options, such as the pin *Width* and pin *Layer*. The net PRO values that are read from the schematic are used for initial population of the respective fields. However, you can override these values manually. You must ensure that the net PRO constraint values are not violated. See Support for Net PRO Definitions.



© 2023 Cadence Design Systems, Inc.

All the fields with a white background are editable.



2. To change the value for an individual cell:

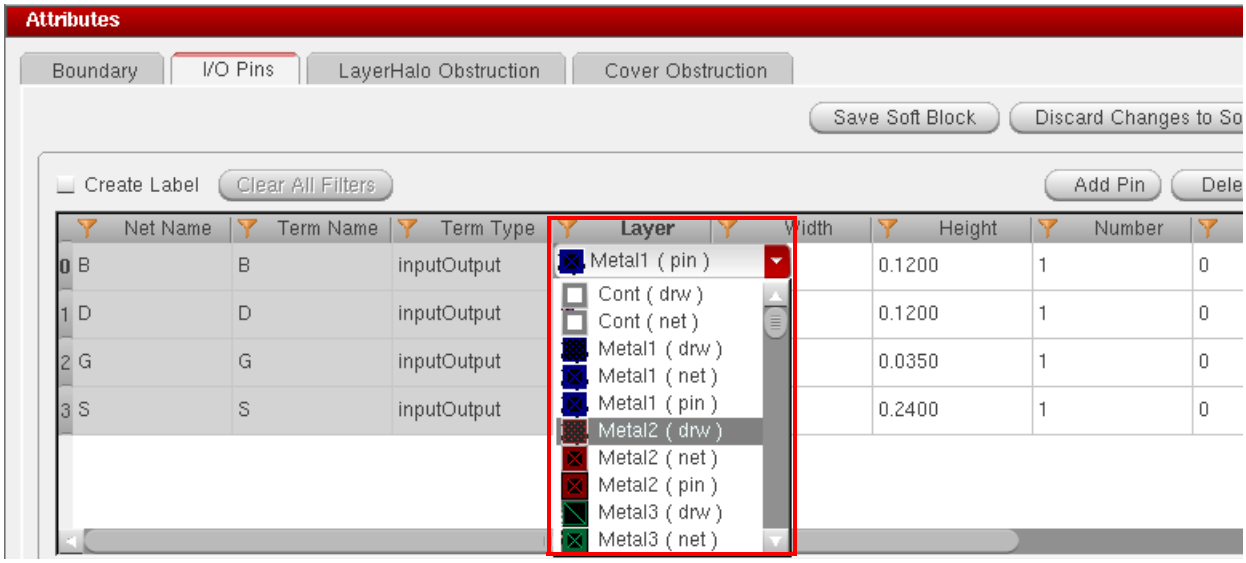
   a. Double-click the cell.If a drop-down arrow appears, select a new value from the drop-down list. If the value in the cells gets highlighted, type in a new value directly.

      Note: Alternatively, you can select a cell and press the space bar to display the value list box for the column. If the selected cell is a text field, pressing the space bar makes the text editable. You can then type in the new value directly.

   b. Click any where within the editable area of the tab page.

© 2023 Cadence Design Systems, Inc.

The value of the selected cell changes to the new value you selected.



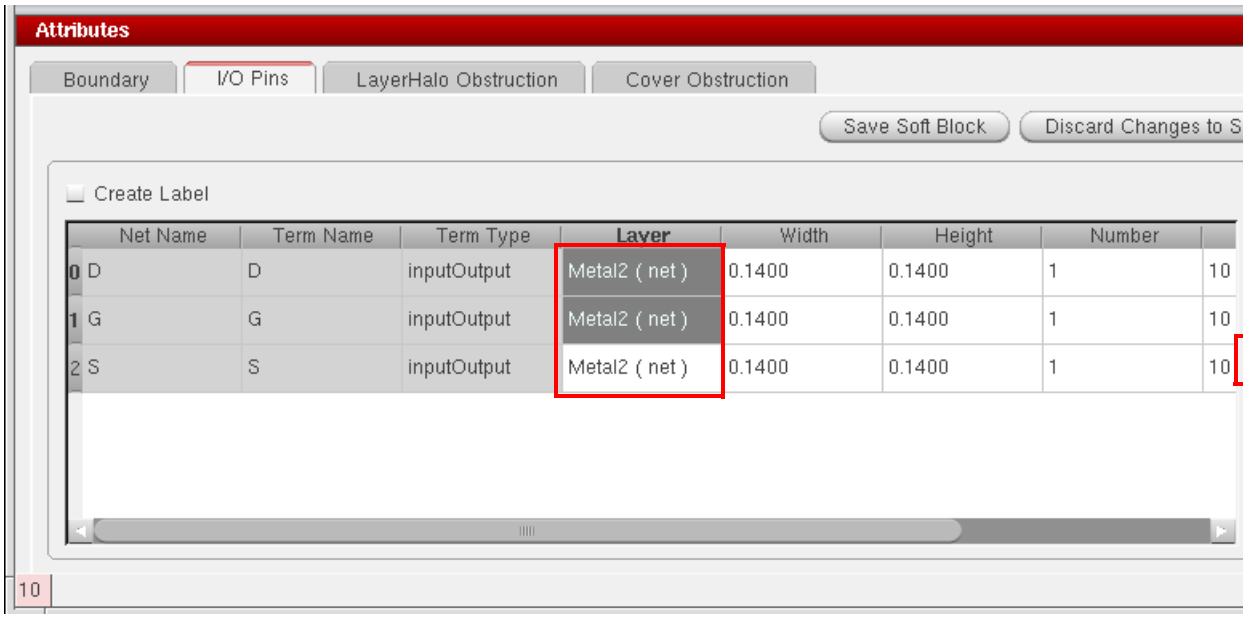
3. To change the value for multiple cells within a column:

   a. Hold down the `Ctrl` key and select the cells in a column that you want to update.

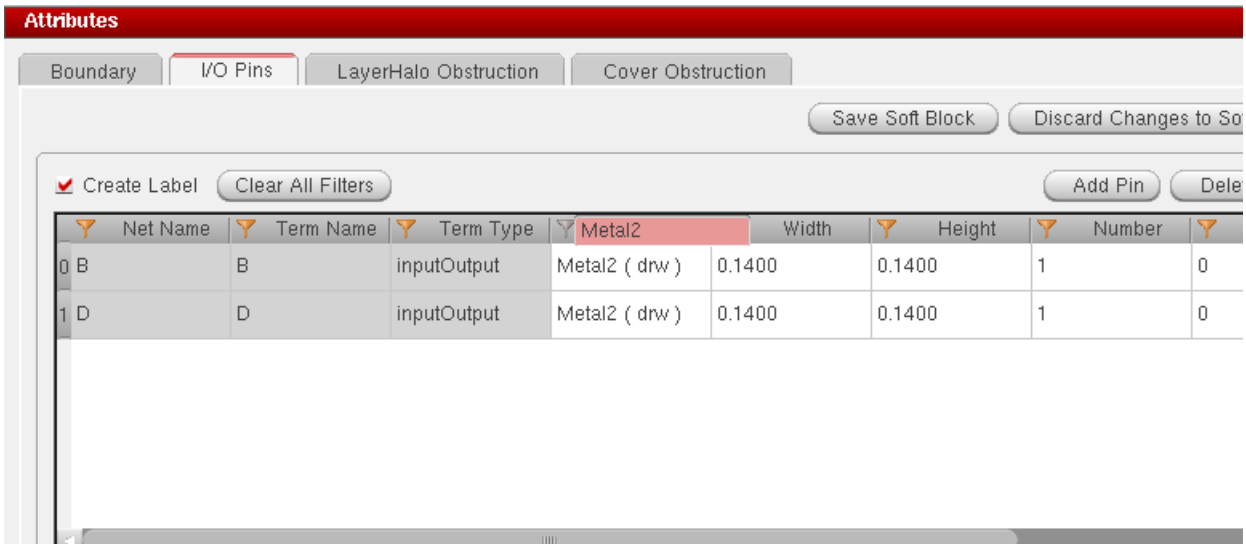
      - If you want to select all the cells in the column, select the column header.With the `Ctrl` key pressed, double-click one of the selected cells and choose the new value from the value list box that appears.
      - Alternatively, release the `Ctrl` key and press the space bar to display the value list box for the column. Then, choose an appropriate value to apply. If the selected cell is a text field, pressing the space bar makes the text editable. You can then type in the new value directly.

© 2023 Cadence Design Systems, Inc.

b. Click any where within the editable area of the tab page. The value of the selected cells changes to the new value you selected.



4. To filter rows based on a keyword, double-click the column name and specify the keyword. Only those rows with matching values are filtered out and listed.



5. To add a new pin:

c. Click *Add*.

A new row is added to the table with a default *Net Name* and *Term Name*. You can change these, if required.

© 2023 Cadence Design Systems, Inc.

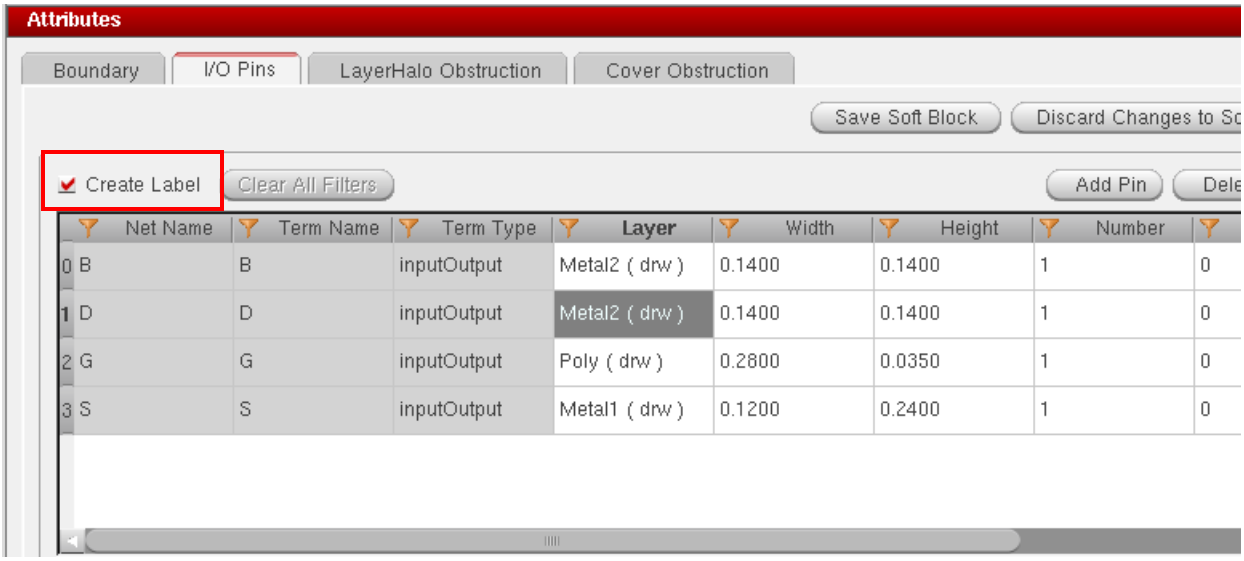
d. The other fields display their default values. Edit the values in the other fields as required.

- To know more about the default values in the *Width* and *Height* fields, see Initial Pin Width Setting and Initial Pin Height Setting.
- To specify the layers that need to be listed in the *Layer* column, use the initIOPinLayerNames or initIOPinPurposeNames environment variable, as appropriate.

6. To delete a pin, click *Delete*.

The selected pin is removed from the list.

7. To generate labels for all the pins in the selected soft block, select the *Create Label* option.



To generate labels for all the pins of *all* the soft blocks in your design:

a. Choose *Options – Soft Block Options* from the Configure Physical hierarchy menu bar.

b. Check the *Create Label* check box in the Soft Block Global Options form.

c. Select appropriate label creation options under the *Pin Label Text Style* group box.

8. Click *Save Soft Block*.

The I/O pin parameters are set. The Attributes pane is disabled and the *Instances* pane re-enabled.

© 2023 Cadence Design Systems, Inc.

Initial Pin Width Setting

The initial pin width value displayed in the *Width* field is determined in the following order:

- If there are any Process Rule Overrides (PROs) on the current net in the schematic view, then the `minWidth` is determined based on the net PROs values.
- If there is no PRO, but there is an existing layout, then the `minWidth` is derived from the layout, assuming that the pin widths in the layout are updated.
- When neither of the above exist, the value of initPinWidth environment variable is used the set the `minWidth` value.
- In the absence of a PRO, layout, and initPinWidth environment variable, the default technology file value is considered.

Initial Pin Height Setting

The initial pin height value displayed in the *Height* field is determined in the following order:

- If there are any Process Rule Overrides (PROs) on net in the schematic view, then the `minHeight` is determined based on the net PROs values.
- If there is no PRO, but there is an existing layout, then the `minHeight` is derived from the layout, assuming that the pin heights in the layout are updated.
- When neither of the above exist, the value of initPinHeight environment variable is used the set the `minHeight` value.

  In the absence of a PRO, layout, and initPinHeight environment variable, the default technology file value is considered.

Related Topics

Soft Block Global Options Form

Soft Block Mode in the CPH Window

Specifying LayerHalo Obstructions

LayerHalo obstructions are defined at the child-level, and are effective at the parent-level. Alternatively, you can define cover obstructions to block routing on the specified layer at the top level. The selected layer and all lower layers are automatically obstructed.

© 2023 Cadence Design Systems, Inc.

A soft block obstruction is implemented using a halo object instead of a blockage object, because there is no need to manually remove the halo prior to routing.

- To define a *routing*, *fill*, *slot*, *pin*, *feedthru*, or *screen* obstruction, you need to set the *Blockage Type*, and specify the *Layer* and *Offsets*. There is also a *Material* filter which you can use to select layers based on the material defined for that layer in the technology file.
  - The list of extractable layers is retrieved from the constraint group set for the current layout cellview. If there is no constraint group set for the current layout cellview, the software retrieves the layers from the constraint group specified by the `setupConstraintGroup` environment variable. The CPH soft block reads the environment variable and looks for the constraint group carrying the required information in the associated technology file. The technology file may be an Incremental Technology Database file (ITDB) or a design techfile.
  - If there are no layers defined in either location, it issues a message and disables the soft block definition.
- To define a *placement* obstruction, you need only set the *Offsets*. You can define only one placement obstruction per block.

To specify parameters for a LayerHalo obstruction,

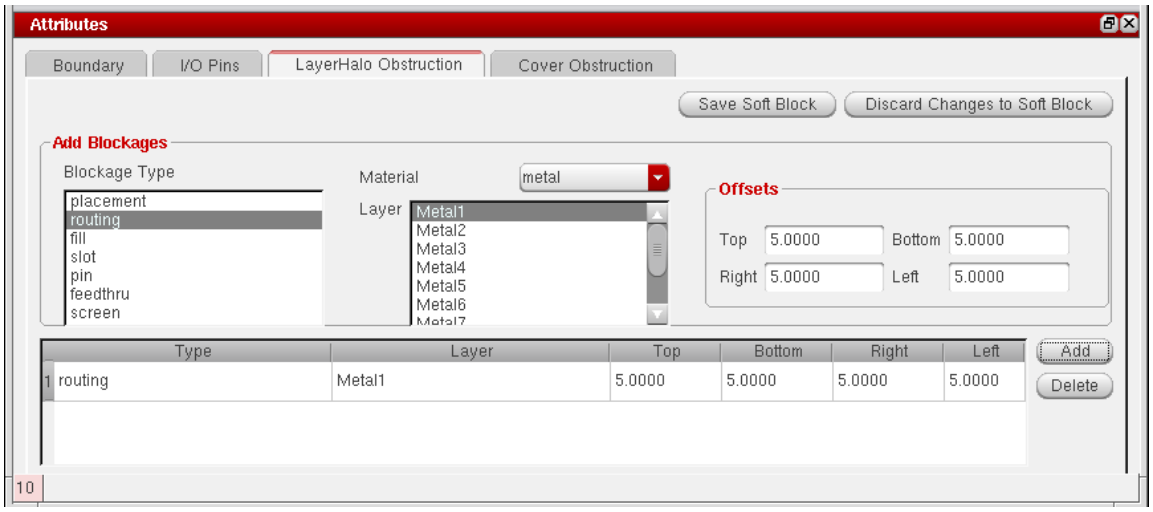
1. Click the *LayerHalo Obstruction* tab.
2. To define a routing obstruction:
   a. Select *routing* in the *Blockage Type* pane.
   b. Choose *metal* from the *Material* drop-down list.

      The *Layer* list is reduced to show only the extractable metal layers defined in the active constraint group.

   c. Set the *Offsets* for the four sides of the soft block.

© 2023 Cadence Design Systems, Inc.

d. Click *Add* to add the obstruction to the list.



3. To define a placement obstruction:

a. Select *placement* in the *Blockage Type* pane.

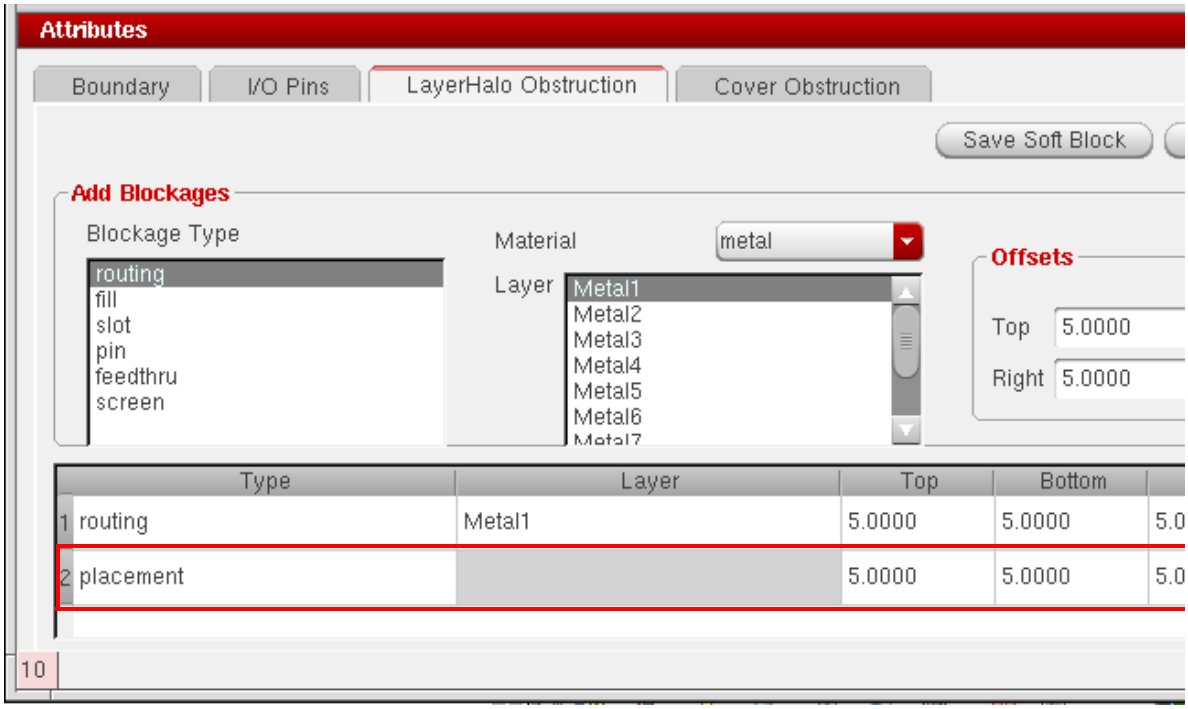
placement is removed from the list of blockage types because you can create only one placement obstruction per soft block.

The *Material* and *Layers* controls are grayed out because they are not required to define a placement obstruction.

b. Set the *Offsets* for the four sides of the soft block.

© 2023 Cadence Design Systems, Inc.

c. Click *Add* to add the obstruction to the list.



4. To update an obstruction, edit the fields you require directly in the table and press `Return` to accept the new value.
5. To delete an obstruction, select it in the table and click *Delete*.

   The obstruction is deleted and removed from the table.

Related Topics

Comparison between LayerHalo and Cover Obstructions

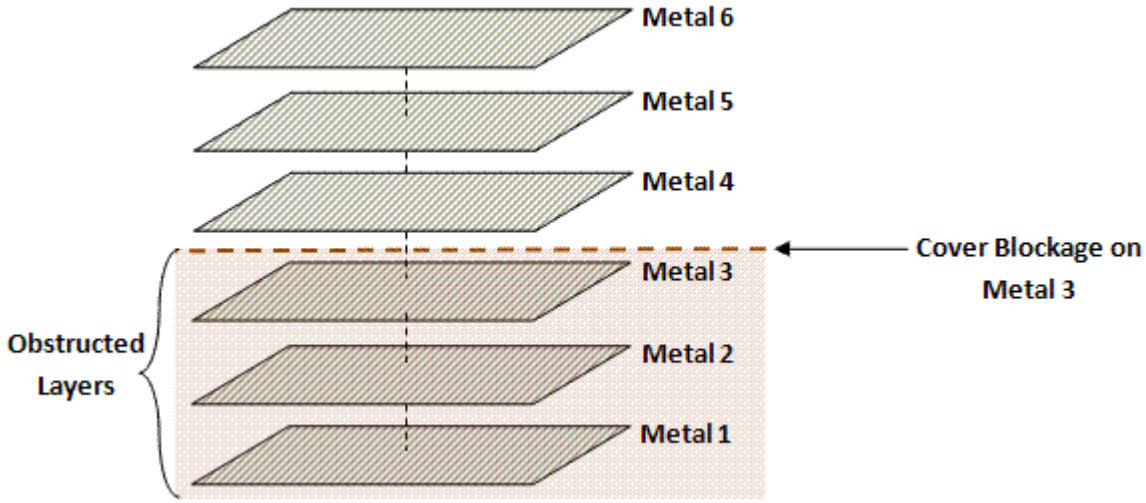
Soft Block Mode in the CPH Window

Specifying Cover Obstructions

You use the *Cover Obstruction* tab to create cover obstructions in your soft block. Cover obstructions are defined at the layer level. When you create a cover obstruction, the selected layer and all lower layers are automatically obstructed. For example, in the following diagram, `Metal 1` is the lowest layer and `Metal 6` is the top most layer. Adding a cover obstruction

© 2023 Cadence Design Systems, Inc.

on `Metal 3` will generate obstructions on layers `Metal 3`, `Metal 2`, and `Metal 1`. As a result, routing cannot be done on these layers.



This type of obstruction is typically generated for layers where you want to prevent over-the-cell routing to improve performance or to avoid electrical effects between tracks.

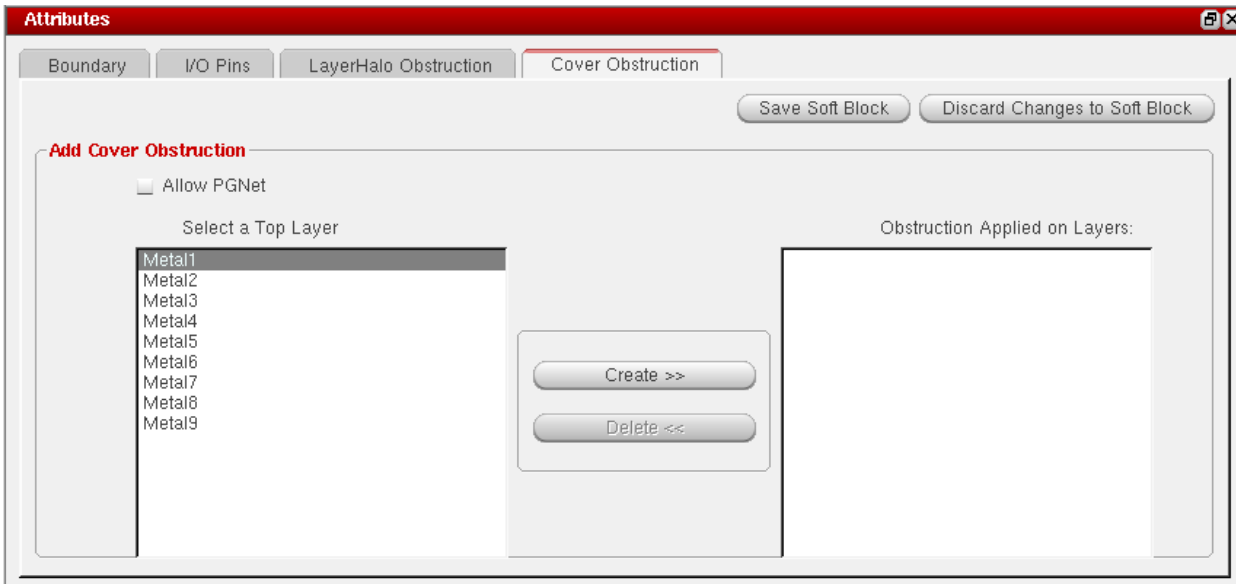
A soft block can be associated with only one cover obstruction. So, any new cover obstruction that is created on a soft block replaces the existing cover obstruction (if any) on it.

Use LayerHalo obstructions to block routing at the child-level. These obstructions are effective at the parent-level.

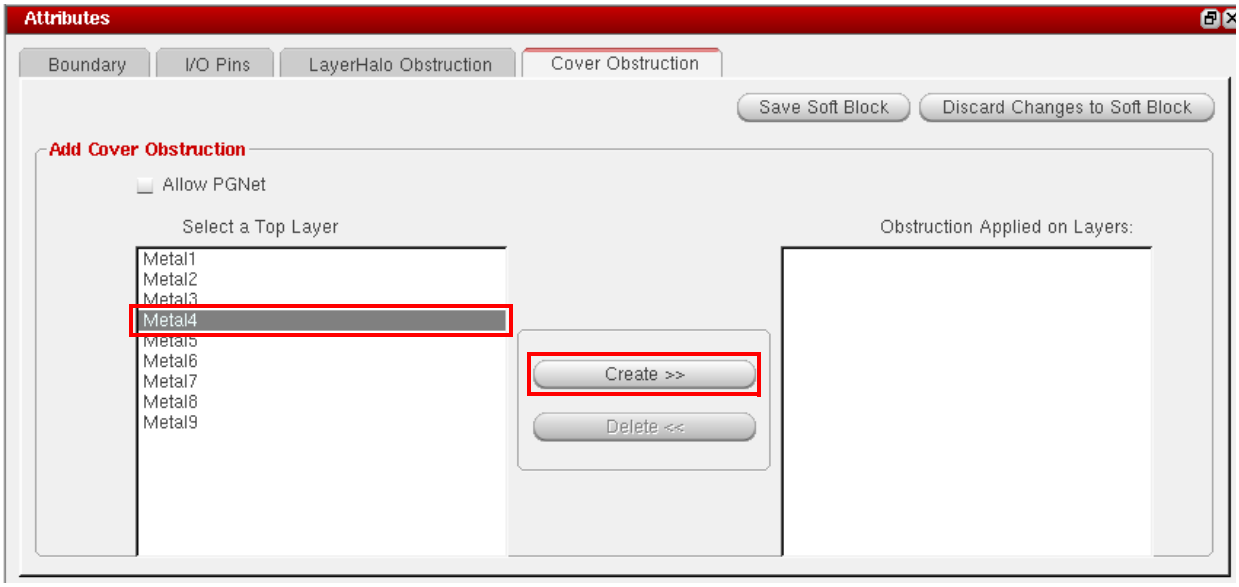
© 2023 Cadence Design Systems, Inc.

Following are the steps to create a Cover Obstruction:

1. Click the *Cover Obstruction* tab in the *Attributes* pane.

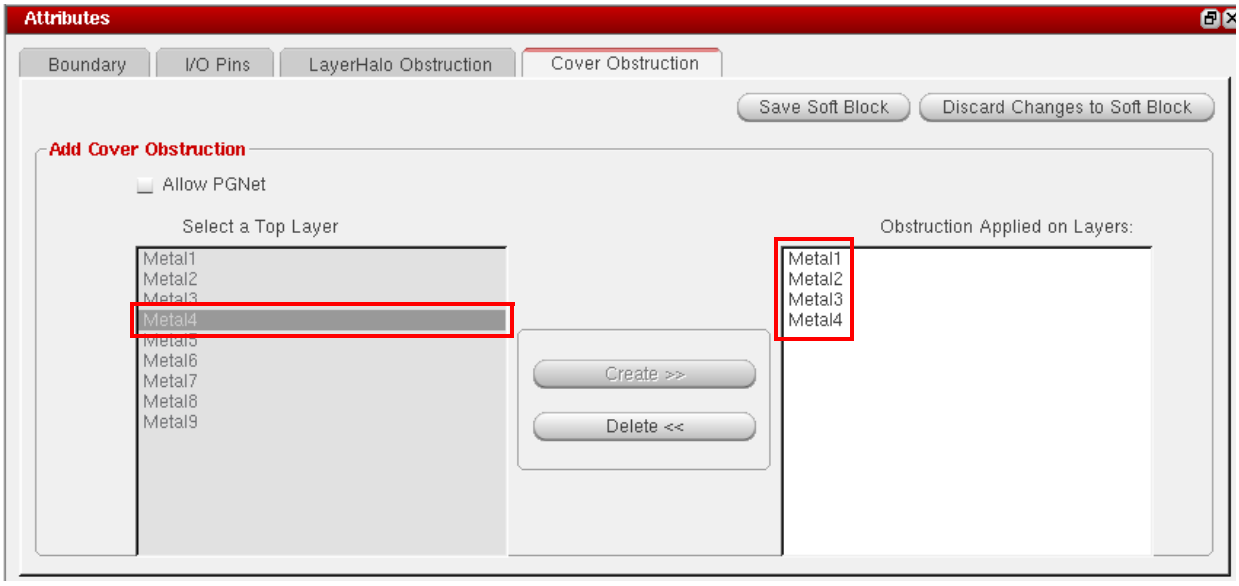


2. Select the layer on which you want to create an obstruction in the *Select a Top Layer* pane and click the *Create* button.



© 2023 Cadence Design Systems, Inc.

The selected layer and lower layers are automatically populated in the *Obstruction Applied on Layers* pane.



3. To allow power and ground nets, check the *Allow PGNet* box.
4. Click *Save Soft Block*.

SKILL Functions

Alternatively you can use the following SKILL functions:

- cphSbDefineCovObstruction: To define cover obstructions
- cphSbDelCovObstruction: To delete cover obstructions
- cphSbDisplayCovObstructionInfo: To display information about cover obstructions
- cphSbHasCovObstruction: To check for the presence of cover obstructions in a soft block

Related Topics

Comparison between LayerHalo and Cover Obstructions

Soft Block Mode in the CPH Window

© 2023 Cadence Design Systems, Inc.

Comparison between LayerHalo and Cover Obstructions

The following table summarizes the similarities and differences between layerHalo and cover obstructions. Depending on your requirement, you can choose the type of obstruction to be defined.

LayerHalo Obstructions	Cover Obstructions
Are defined at the child-level, and are effective at the parent-level.	Are defined for soft blocks. Cover obstructions block routing on the specified layer at the top level.
Prohibits routing parallel to the edge of the selected instance, but allow routing perpendicular to the instance to access pins.	Prohibits routing parallel to the edge of the instance. Also prohibits routing over the cell. Only pins that are placed on the PRBoundary edge can be accessed by the router.
Allows power and ground routes in the entire halo area, around the instance, and over the cell. There is no option to enable/disable these routes.	Provides an option to enable/disable power and ground routing in the entire obstruction area, around the instance, and over the cell.
Can be specified for non-contiguous layers such as M1, M5, and M7.	Can be specified only for contiguous layers, for example for all layers below the top layer. If top layer = M3, then M1 and M2 are also obstructed.
Supports different offsets beyond the instance boundaries.	Does not support different offsets beyond the instance boundaries.
Is not supported by VSR.	Is supported by VSR.

Related Topics

Specifying LayerHalo Obstructions

Specifying Cover Obstructions

Soft Block Mode in the CPH Window

© 2023 Cadence Design Systems, Inc.

Removing Soft Block Parameters

To remove soft block parameters:

1. In the table view, select the instance for which you want to remove parameters.
2. Click the right mouse button and choose *Remove Soft Block Parameters*.

   The soft block parameters are removed and the selected instance will not be generated as a soft block by the *Generate Physical Hierarchy* command.

To redefine soft block parameters, use either *Edit Soft Block Parameters* or *File – Load Floorplan File*.

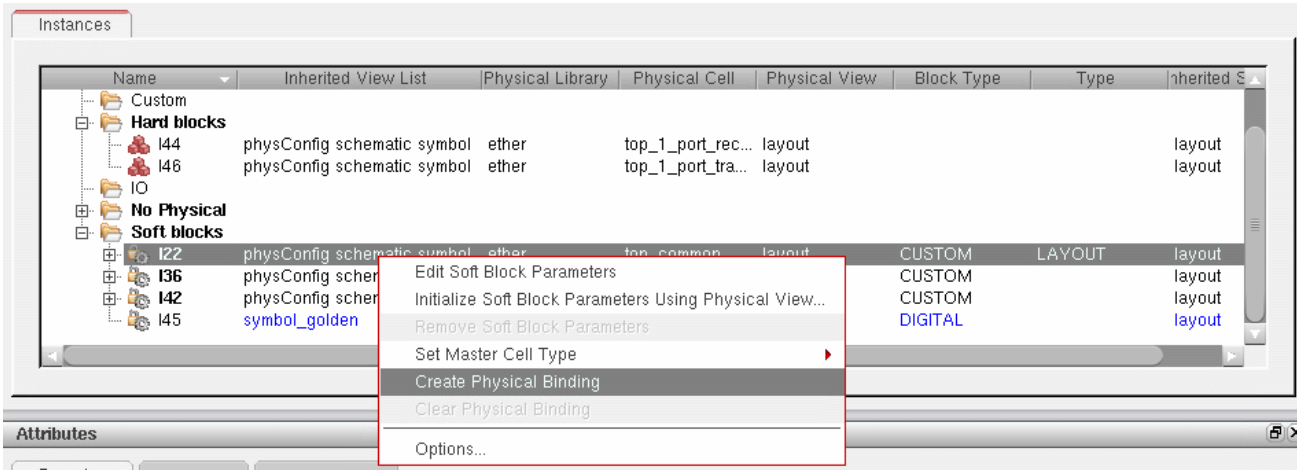
Related Topics

Soft Block Mode in the CPH Window

Creating a Physical Binding

To create a physical binding:

1. In the table view, select the instance for which you want to set a physical binding.
2. Click the right mouse button and choose *Create Physical Binding*.



The physical binding is created.

© 2023 Cadence Design Systems, Inc.

Removing a Physical Binding

To remove a physical binding:

1. In the table view, select the instance for which you want to remove the physical binding.
2. Click the right mouse button and choose *Clear Physical Binding*.

   The physical binding is removed, along with any soft block parameters defined for the instance in question.

To remove soft block parameters but preserve the physical binding, click the right mouse button and choose *Remove Soft Block Parameters*.

Related Topics

Soft Block Mode in the CPH Window

Loading Soft Block Parameters from Another Cellview

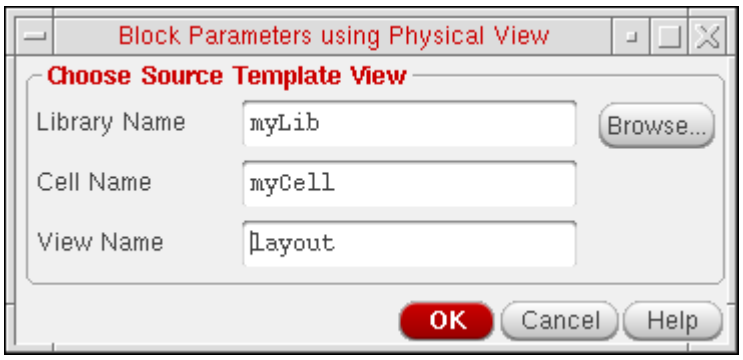
The *Initialize Soft Block Parameters Using Physical View* command imports soft block parameters from an existing layout cellview.

This command uses the same functionality as the *Load Physical View* command.

To load predefined soft block parameters from another cellview:

1. Select a component in the *Instances* table.
2. Right-click the instance and choose *Initialize Soft Block Parameters Using Physical View*.

   The Block Parameters using Physical View dialog is displayed.



© 2023 Cadence Design Systems, Inc.

3. Type in the name of the layout cellview that contains the information you want to import and click *OK*.

   The Attributes pane is updated with the settings from the specified cellview.

4. Edit the parameters as required. For more information, see

   - Specifying the Boundary
   - Specifying I/O Pins
   - Specifying LayerHalo Obstructions
   - Specifying Cover Obstructions

5. Click *Save Soft Block* in the Attributes pane to save the soft block parameters you specified.

   The Attributes pane is disabled and the *Instances* table re-enabled.

   Note: To discard any changes you made since the Attributes pane was enabled, click *Discard Changes to Soft Block*.

6. Repeat steps **1**-**5** for all the soft blocks you want to define in this way.

7. From the Configure Physical Hierarchy menu bar, choose *File – Save Floorplan File* to save all your settings to a floorplan file for later re-use.

Related Topics

Flattening a Layout Instance

Saving and Loading Soft Block Parameters to a Floorplan File

Soft Block Mode in the CPH Window

Saving and Loading Soft Block Parameters to a Floorplan File

You can save the current block parameters to a floorplan file, which you can then re-use in a different Virtuoso session.

To do this:

1. Define the soft block parameters you require.
2. From the Configure Physical Hierarchy menu bar, choose *File – Save Floorplan File*.

© 2023 Cadence Design Systems, Inc.

3. Specify a name for the file and click *OK*.

   The floorplanning parameters are saved to the specified file.

To load parameters from a floorplan file:

1. From the Layout XL window, choose *Launch – Configure Physical Hierarchy*.

   The Configure Physical Hierarchy window appears.
2. Change to Soft Block mode using the drop down list in the toolbar.
3. From the Configure Physical Hierarchy menu bar, choose *File – Load Floorplan File*.

   The Load Floorplan File dialog appears.
4. Specify the file you want to load and click *OK*.

   The floorplanning parameters defined in the file are loaded into the current configuration view.

Related Topics

Defining Soft Block Parameters

Soft Block Mode in the CPH Window

Configure Physical Hierarchy Window

Soft Block Mode in the CPH Window

Generating the Physical Hierarchy

Use the *Generate Physical Hierarchy* command to generate the components in your design while maintaining the hierarchy levels you have defined in Soft Block mode

To generate the physical hierarchy:

1. In the Configure Physical Hierarchy window, define or load the soft block parameters you require. From the layout window menu bar, choose *Launch – Layout XL.* In addition, you must draw 4 Layout GXL tokens.
2. From the layout window menu bar, choose *Floorplan – Generate Physical Hierarchy*.
3. The physical hierarchy is generated in the layout window, including the soft blocks defined in the Configure Physical Hierarchy window.

© 2023 Cadence Design Systems, Inc.

Related Topics

Defining Soft Block Parameters

Saving and Loading Soft Block Parameters to a Floorplan File

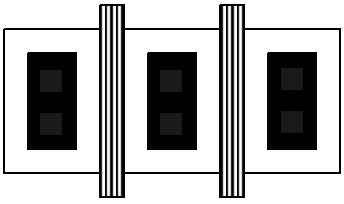
© 2023 Cadence Design Systems, Inc.

4

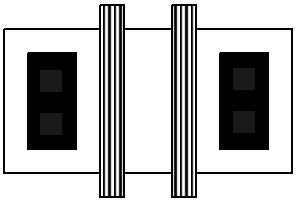
Device Abutment

This topic describes the abutment capabilities of the Virtuoso® Layout Suite XL editor (Layout XL). Abutment allows cells to be automatically overlapped, aligned, and electrically connected without introducing a design rule violation or connectivity error. Abutment reduces both the area occupied by a circuit and the length of the interconnect wiring. You can use abutment during interactive layout generation.

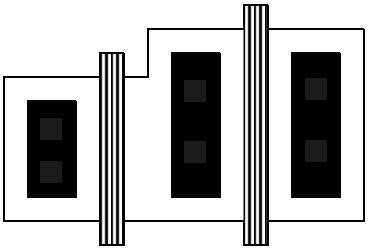
MOS transistors sharing a diffusion



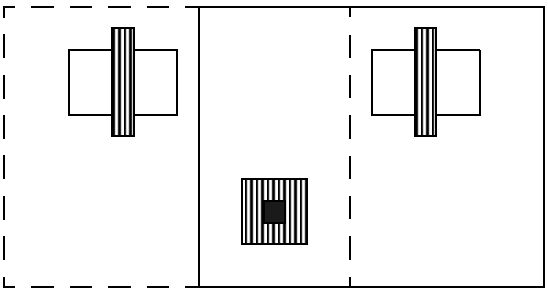
A. Same size, terminals with external connection on same net



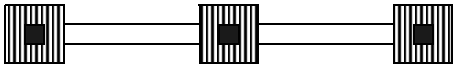
B. Same size, no other connections on the same net



C. Different size, terminals with external connection on the same net



Wells sharing part of their surface and bias contacts



Resistors sharing terminals

Related Topics

Pcell Set Up for Abutment

Automatic Abutment Flow

© 2023 Cadence Design Systems, Inc.

Automatic Device Abutment

Manual Device Abutment

Debugging Abutment

Abutment of Dummy Shapes in Pcells

Abutment Function Flow

Prerequisites for Automatic Device Abutment in Layout XL

Abutment in Layout XL is triggered when two pins of different instances overlap or when named shapes within two Pcell instance submasters overlap. If they meet the following criteria, the abutment engine executes.

- The abutment engine is enabled.
- Both instances must be Pcells set up for abutment or cells with the appropriate abutment properties set.
- Both instances must share the same master or must have the `abutClass` property set.

  If two instances have the same master, then they can be abutted, even if the `abutClass` is set to different values on each instances.

  To abut two cells with different masters, you must add the `abutClass` property to the pin of each cell and enter the same abutment class name as the value for each property.
- The instances must overlap.
- Both instance pins must be connected to the same net.

  To abut overlapping instances with no nets, set the `abutWithoutConnectivity` environment variable.
- Both instance pins must be defined on shapes of the same layer or on layers that are defined as equivalent layers in the technology file.
- Both instance pins can have the same abutment direction if the rotation of one (but not both) of the pins is either `R180` or `MY`, or any other rotation that transforms the left-hand side to the right-hand side.
- During placement, the *Connectivity extractor* and *Abut transistors* options in the Connectivity form must be turned on.

© 2023 Cadence Design Systems, Inc.

When the *Abut transistors* option is turned on, devices that have not been abutted but are overlapping are abutted. If the devices are already abutted, `autoAbutment` does not re-abut the devices.

- If one of the instances is in any type of group (including a synchronous clone), the other instance must be in the same group.

Abutted devices can share diffusion, contacts, metal tabs, or any shape combined in an instance pin.

While creating specific parameters in Pcells and named diffusion shape pins, you also have to add the following properties on the shapes the pins are created on. The auto-abutment properties specify whether or not the Pcells can abut.

- abutClass Property – Pins with the same `abutClass` are able to abut.
- abutAccessDir Property – Defines the directions in which the pins are able to abut.
- abutGateNet Property – Defines the name of the net the gate shapes are on.
- abutCondInclusion Property – Defines the name of the boolean property that adds or removes contact shapes over the diffusion pin.

Related Topics

Pcell Set Up for Abutment

About Pcell Super and Submaster Cells

abutWithoutConnectivity

autoAbutment

Pcell Set Up for Abutment

To set up a Pcell so that it can be abutted, add the following properties on the pins of the cellview:

- `abutFunction`
- `abutOffset`
- `abutAccessDir`
- `abutClass`

© 2023 Cadence Design Systems, Inc.

You can also add these properties to a cellview in order to set up abutment for all the pins in the cellview. If any pin in the cellview has properties that differ from the properties set for the cellview, the properties on the pin override the properties on the cellview.

For examples of how to set abutment properties, see the sample Pcell libraries at:

- *your_install_dir*/tools/dfII/samples/ROD/rodPcells/components/mos/mos.il
- your_install_dir/tools/dfII/samples/ROD/rodPcells/components/simple_mos/simple_mos.il

Related Topics

abutFunction Property

abutOffset Property

abutAccessDir

abutClass

Automatic Mirroring and Pin Permutation During Abutment

If an attempt to abut two devices causes two pins on the same net to touch, a short violation is created. Layout XL tries to resolve this violation automatically by first mirroring the device or chain in question. If that fails, it then attempts to resolve it by permuting pins.

This behavior is controlled by two environment variables, both of which are switched on by default.

- `autoMirror` which specifies that if a short violation is created during abutment, then Layout XL first mirrors the device in question in order to resolve the short. Only if this is unsuccessful does Layout XL attempt to resolve the short by permuting the pins.

  The GUI equivalent is the *Mirror transistors* option on the Layout XL Generation Options form.

  Automatic mirroring is temporarily disabled when constraint-aware editing is on and constraints exist between the instances and chains under consideration.

© 2023 Cadence Design Systems, Inc.

- `autoMirrorChains` which specifies that device chains are mirrored subject to certain conditions being satisfied. This environment variable is honored only if `auto Mirror` is set to `t`. It has no GUI equivalent.

  If one of the devices is in any type of group (including a synchronous clone), the other device must be in the same group for automatic mirroring to occur. Mirroring is done on end-pin violations only if they are valid left/right or top/bottom pairs.

To disable the mirroring of devices and chains:

1. From the layout window menu bar, choose *Options – Connectivity* and click the *Generation* tab.
2. Switch off the *Mirror transistors* option and click *OK*.

   This switches off both device mirroring and chain mirroring during abutment.

Resolving shorts by mirroring rather than permuting pins is desirable because pin permutation

- Assumes that the device is symmetric and therefore the source and drain parameters are not swapped
- Changes the connectivity of the design.

Resolving the short violation through mirroring removes these drawbacks. No pin permutation is performed on chained devices.

Related Topics

autoMirror

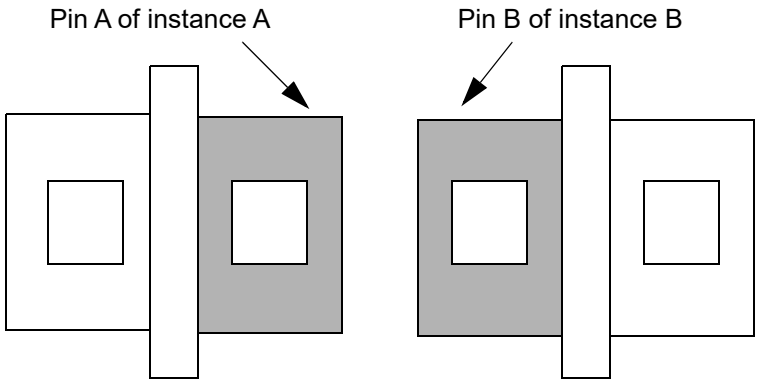
autoMirrorChains

© 2023 Cadence Design Systems, Inc.

Automatic Abutment Flow

1. Layout XL recognizes that two Pcell instances can be abutted together, triggered either by overlapping two instance pins or when you create parameterized cell instances during chaining and folding.

The software determines if pin A of instance A should abut with pin B of instance B

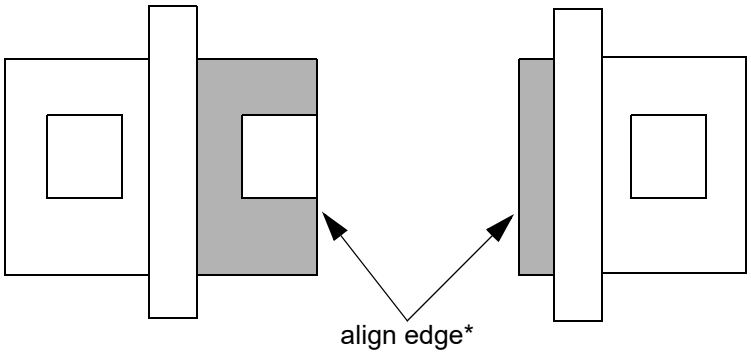


2. Automatic abutment identifies cells for abutment by master name or class.
3. Auto-abutment calls `abutFunction`, a user-defined SKILL function that performs the necessary changes to the parameterized cell instances themselves.

   You must provide the SKILL function because Layout XL has no knowledge of the user-defined parameterized cell.

   `abutFunction` adjusts the parameters of Pcells and calculates reference edge offsets of conventional cells. It stores the old values, which can be called again if you choose to unabut the instances.

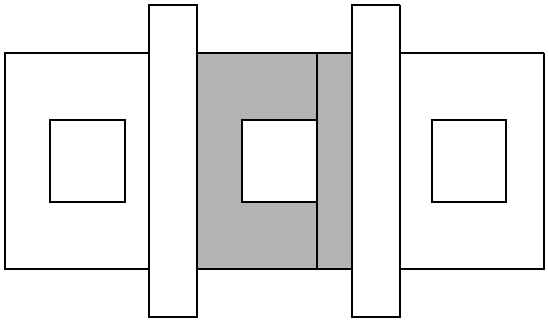
If yes, the software calls the user-defined SKILL function to change the topology of instance A and instance B and return the offset to align pin A of instance A and pin B of instance B.



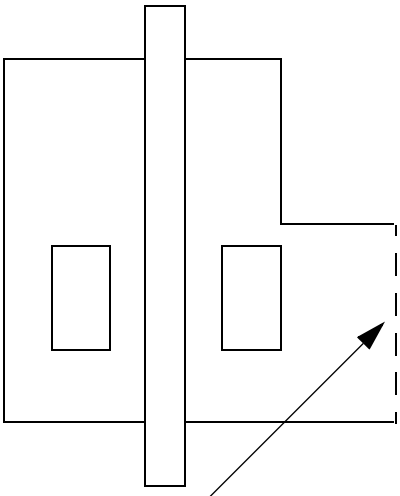
© 2023 Cadence Design Systems, Inc.

4. If the cells can be abutted (the abutment connection condition is 1 or 2), the cells are abutted to the reference edges and the pins are aligned perpendicular to the direction of abutment.

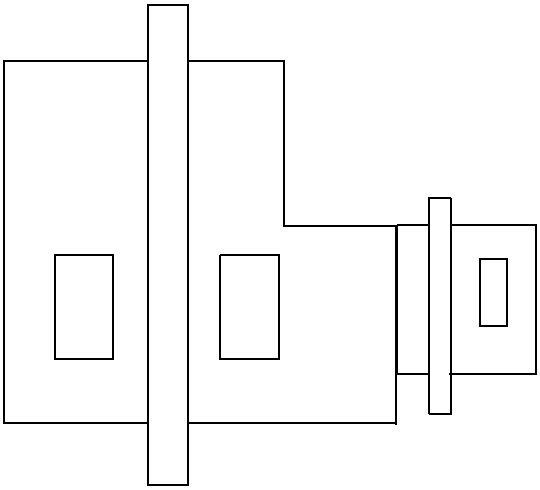
The software correctly aligns pin A of instance A and pin B of instance B



The align edge is always the real outermost edge used, as shown in the figure below. This lets you maintain an L-shaped pin when applying abutment.



Real edge as align edge, not bbox edge



Following abutment, there is no notch DRC error

If the cells cannot be abutted, the abutment connection condition is 3 and the cells remain in their original configuration.

© 2023 Cadence Design Systems, Inc.

Sample Parameterized Cells Set Up for Abutment

Example 1

This example shows what each of the different parts does:

```
------------------------------------------------------------------------
; Function: abutFunction(iA iB pA pB pASide connection event
; @optional (group nil))
;
; Inputs: group - the abutment group that these two cells belong to.
; ____________________________________________________________________
;
;      iA         = Instance Id of cell that will move during abutment
; ____________________________________________________________________
;
;      iB         = Instance Id of cell being abutted to
; ____________________________________________________________________
;
;      pA         = Overlapping Pin Fig of iA
; ____________________________________________________________________
;
;      pB         = Overlapping Pin Fig of iB
; ____________________________________________________________________
;
;      pASide     = Abutting pin access direction
; ____________________________________________________________________
;
       connection = an integer value of 1 or 2  that indicates:
;
;      1. pins are connected to the same net and do not
;         connect to any other pin.
;
;      2. pins are connected to the same net and the net
;         connects to other pins
; ____________________________________________________________________
;
;      event      = integer that represents abutment event:
;
;      1. compute abutment offset
;      2. Pcell parameter adjustment for abutment
;      3. Pcell parameter adjustment for unabutment
; ____________________________________________________________________
;
;      group      = abutment group pointer available to specific events:
;      abutment group pointer available to Events 1, 2, and 3
; ____________________________________________________________________
;
; Outputs: depends
;
; Side effects:
;
procedure( abutFunction(iA iB pA pB pASide connection event @optional (group
nil))
```


© 2023 Cadence Design Systems, Inc.

```
   prog((result)

   case(event
      (1 ; Compute offset
         result = getAbutmentOffset(iA iB pA pB pASide connection)
      )
      (2 ; Adjust Pcell parameters
         result = setAbutmentParams(iA iB pA pB pASide connection group)
      )
      (3 ; Adjust Pcell parameters back to default resetAbutmentParams(group iA iB)
         result = t
      )
      (t ; Anything else return a nil
         result = nil

      )
   )
   return(result)
   )
)
```

Example 2

This example shows how to add abutment properties to pins in an inverter Pcell.

```
; *** the following 4 pins are on metal1 and are on ***
; *** the power and ground rails. This will allow   ***
; *** abutment to other standard cells              ***

    obj = leftVddPin~>dbId
    dbReplaceProp(obj "abutAccessDir" "list" list("left"))
    dbReplaceProp(obj "abutClass" "string" "stdcell")
    dbReplaceProp(obj "abutFunction" "string" "stdCellFunc")

; *** the following property is a technique for passing ***
; *** information to the user defined function. In this ***
; *** case the minimum cell height is dependent on many ***
; *** factors including design rules and is originally  ***
; *** calculated in the Pcell code. This avoids         ***
; *** duplicating calculations in the user-defined      ***
; *** function                                          ***

    dbReplaceProp(obj "minCellHeight" "float" minH)

    obj = rightVddPin~>dbId
    dbReplaceProp(obj "abutAccessDir" "list" list("right"))
    dbReplaceProp(obj "abutClass" "string" "stdcell")
    dbReplaceProp(obj "abutFunction" "string" "stdCellFunc")
    dbReplaceProp(obj "minCellHeight" "float" minH)

    obj = leftGndPin~>dbId
    dbReplaceProp(obj "abutAccessDir" "list" list("left"))
    dbReplaceProp(obj "abutClass" "string" "stdcell")
    dbReplaceProp(obj "abutFunction" "string" "stdCellFunc")
    dbReplaceProp(obj "minCellHeight" "float" minH)

    obj = rightGndPin~>dbId
    dbReplaceProp(obj "abutAccessDir" "list" list("right"))
    dbReplaceProp(obj "abutClass" "string" "stdcell")
```


© 2023 Cadence Design Systems, Inc.

```
    dbReplaceProp(obj "abutFunction" "string" "stdCellFunc")
    dbReplaceProp(obj "minCellHeight" "float" minH)


; *** The following 4 pins are the diffusion pins   ***
; *** of the individual MOS devices in the inverter ***
; *** this will allow abutment of other individual  ***
; *** MOS devices to the inverter                   ***

    obj = POutPin~>dbId
    dbReplaceProp(obj "abutAccessDir" "list" list("right"))
    dbReplaceProp(obj "abutClass" "string" "ptran")
    dbReplaceProp(obj "abutFunction" "string" "mosAbutFunc")
    dbReplaceProp(obj "contactParam" "string" "POutCnts")
    dbReplaceProp(obj "w" "float" pw)
    ; *** the automatic spacing properties are used if abutment fails ***
    dbReplaceProp(obj "vxlInstSpacingDir" "list" list("right"))
    dbReplaceProp(obj "vxlInstSpacingRule" "float" .35 )

    obj = NOutPin~>dbId
    dbReplaceProp(obj "abutAccessDir" "list" list("right"))
    dbReplaceProp(obj "abutClass" "string" "ntran")
    dbReplaceProp(obj "abutFunction" "string" "mosAbutFunc")
    dbReplaceProp(obj "contactParam" "string" "NOutCnts")
    dbReplaceProp(obj "w" "float" nw)
    dbReplaceProp(obj "vxlInstSpacingDir" "list" list("right"))
    dbReplaceProp(obj "vxlInstSpacingRule" "float" .35 )

    obj = PPwrPin~>dbId
    dbReplaceProp(obj "abutAccessDir" "list" list("left"))
    dbReplaceProp(obj "abutClass" "string" "ptran")
    dbReplaceProp(obj "abutFunction" "string" "mosAbutFunc")
    dbReplaceProp(obj "contactParam" "string" "PPwrCnts")
    dbReplaceProp(obj "w" "float" pw)
    dbReplaceProp(obj "vxlInstSpacingDir" "list" list("left"))
    dbReplaceProp(obj "vxlInstSpacingRule" "float" .35 )

    obj = NPwrPin~>dbId
    dbReplaceProp(obj "abutAccessDir" "list" list("left"))
    dbReplaceProp(obj "abutClass" "string" "ntran")
    dbReplaceProp(obj "abutFunction" "string" "mosAbutFunc")
    dbReplaceProp(obj "contactParam" "string" "NPwrCnts")
    dbReplaceProp(obj "w" "float" nw)
    dbReplaceProp(obj "vxlInstSpacingDir" "list" list("left"))
    dbReplaceProp(obj "vxlInstSpacingRule" "float" .35 )
```

- **Calls to dbReplaceProp for Abutment Groups**: When using the function `dbReplaceProp`, you must call `dbReplaceProp` for abutment groups before calls to `dbReplaceProp` for Pcell properties because the IDs of the objects before and after abutment occurs cannot be guaranteed to be the same. This is because the instance master is purged as soon as the instance reference count associated with the instance master becomes zero. Therefore you cannot rely on the same data or IDs after Pcell re-evaluation.
- **Pin Figure Names must be Unique within a Terminal**: When developing transistor Pcells, wherever there is more than one figure attached to any `oaPin` (as is the case

© 2023 Cadence Design Systems, Inc.

when two figures are strongly connected), each figure attached to the pin must have a name that is unique within the same terminal and its corresponding must-connect terminals. If this is not the case, use the `dbSetPinFigName` SKILL function to assign a unique name to each figure attached to the pin.

- **Abutting Pcell Supermasters from Legacy Designs:** If a Pcell supermaster that is instantiated in a legacy design has changed, possibly due to a PDK update since the design was last opened in a newer version of Virtuoso, the Pcell submaster might appear to have incorrect contacts. To resolve the contact mismatch and to allow abutment, open the design in Layout XL, discard the edits, and re-read the design from disk.

Related Topics

abutFunction Property

Creating CMOS Pcells to Use with Abutment

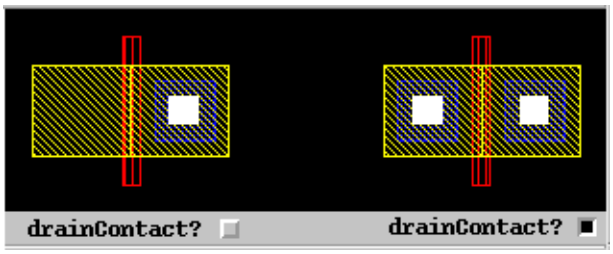
To create Pcells that have a built-in abutment capability:

1. Create a Pcell specifically for abutment.

   Editing an existing Pcell for abutment is much more difficult than creating a new one for the purpose. For information on creating Pcells, see the *Virtuoso Parameterized Cell Reference*.

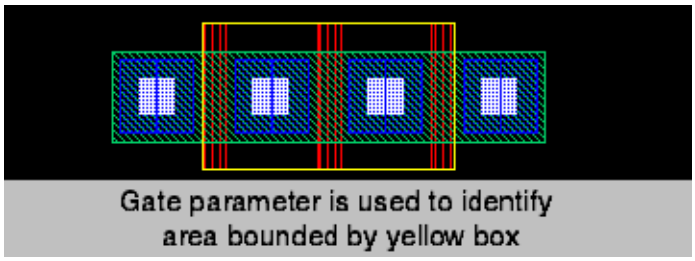
2. Add the following abutment parameters to the Pcell.

   a. Conditional inclusion or exclusion of contacts. For more information, see abutCondInclusion Property.

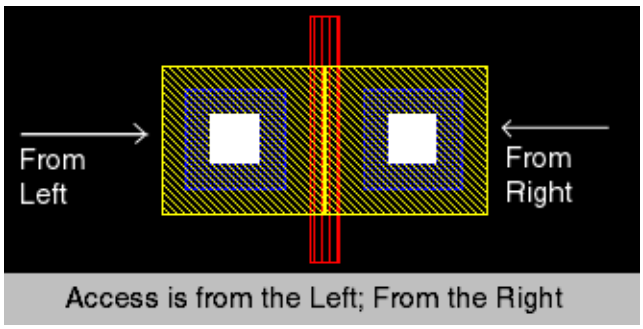


© 2023 Cadence Design Systems, Inc.

b. Gate pin identification. For more information, see abutGateNet Property.

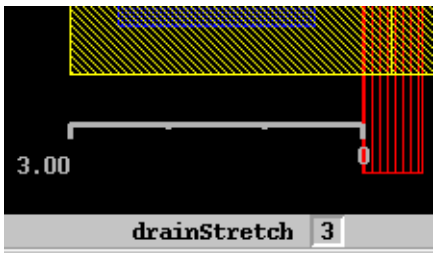


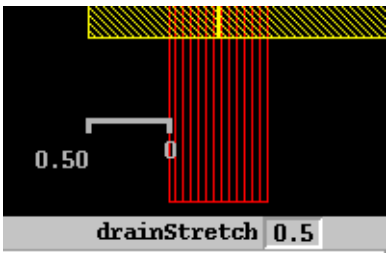
c. Access direction identification. For information, see abutAccessDir Property.



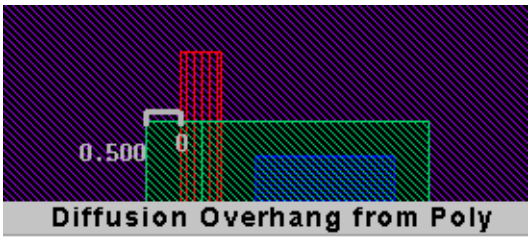
d. Specify stretchable material. For information, see abutMosStretchMat Property.

The first element in the `abutStretchMat` list is `drainStretch`.



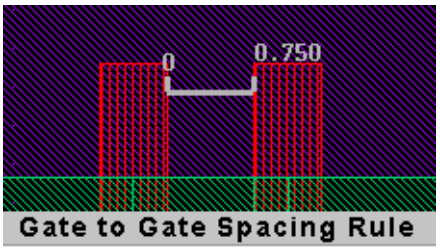


The second element in the `abutStretchMat` list is `abutMinExt`.

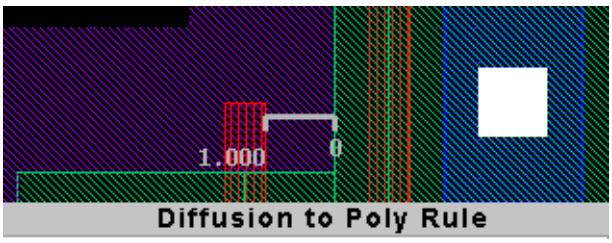


© 2023 Cadence Design Systems, Inc.

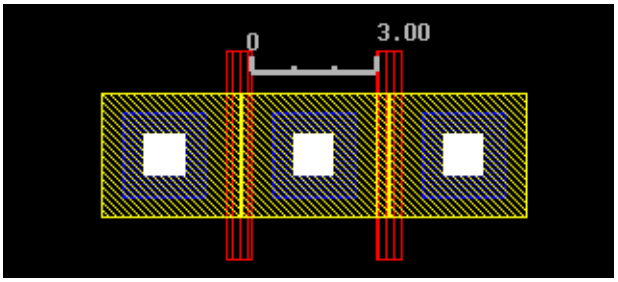
The third element in the `abutStretchMat` list is `abutRule1Ext`.



The fourth element in the `abutStretchMat` list is `abutRule2Ext`.



The fifth element in the `abutStretchMat` list is `abutContactExt`.



MOS Device Abutment without a User-Defined Function

Another way to process abutment and unabutment without creating an `abutFunction` callback, is to use the `abutMosStretchMat` property. The `abutMosStretchMat` property is used with the automatic abutment properties to define abutment; for more information, see abutMosStretchMat Property.

This section describes the automatic abutment properties that you will have to add to Pcells to use auto-abutment on standard MOS Pcells without having to define your own `abutFunction`.

© 2023 Cadence Design Systems, Inc.

Multiple Pins Abutment

Layout XL supports the abutment of more than one pin to a wider single pin.

Multiple pins on the edge of a cell can also be abutted. Any pair of pins with the correct properties can trigger automatic abutment, but once automatic abutment has been triggered, other pins on that edge that touch pins on the cell it is abutted to will not trigger automatic abutment again. If those pins do not connect, auto-permute tries to resolve the conflict. If the conflict cannot be resolved, a connection violation is flagged.

Make sure that you define abutment in such a way that no Design Rule Checker (DRC) or connectivity violations are introduced by abutment between any selected pair of pins.

Snap Instances in the Direction Perpendicular to the Abutment

The layout environment variable `abutPerpSnapOn` controls whether Layout XL snaps an instance in the direction perpendicular to the abutment. When `abutPerpSnapOn` is set to `t`, Layout XL automatically snaps the moving instance vertically such that the lower-left y coordinates of the two abutting pins are the same.

Additional Pins on Ignored Instances

The abutment engine ignores instance terminals belonging to instances with the schematic properties lvsIgnore or `ignore`. This means that additional pins on ignored instances are also ignored, allowing abutment to take place where it otherwise would not and enabling a tighter placement of abutted components.

For example, a schematic has two MOS devices connected in series (one's source to the other's drain). The schematic also places a parasitic capacitor between these MOS devices. The capacitor has the `ignore` property attached to it. When generating the layout, Layout XL correctly ignores the capacitor and its pin, and abuts the two MOS devices.

Automatic Device Abutment

Devices are abutted automatically when the Chain option is selected:

- During layout generation.
- When running the *Update Components And Nets* command.

© 2023 Cadence Design Systems, Inc.

Related Topics

Chaining Transistors Automatically during Layout Generation

Chaining Transistors Automatically when Updating Components and Nets

Chaining Transistors Automatically during Layout Generation

To chain transistors automatically during layout generation,

1. From the layout window menu bar, choose *Connectivity – Generate – All From Source*.

   The Generate Layout form is displayed.

2. On the *Generate* tab, select either of the Chaining options:

   - To automatically chain the transistors, select the *Chain* check box and click *OK*.

     The software chains the transistors automatically during layout generation.

   - To automatically chain the individual folds of a transistor, select the *Chain Folds* check box and click OK.

     The software automatically chains the individual folds of a transistor. The *Chain Folds* check box is active only when the *Chain* option is deselected and the *Fold* option is selected.

   The results of the Chaining operation are affected by the state of the `lxStackPartitionParameters` environment variable.

Chaining Transistors Automatically when Updating Components and Nets

To chain transistors automatically when updating components and nets,

1. From the layout window menu bar, choose *Connectivity – Update – Components And Nets*.

   If the schematic has any devices that are missing from the layout, the Generate Layout form is displayed.

2. On the *Generate* tab, select either of the chaining options:

   - To automatically chain the transistors, select the *Chain* check box and click *OK*.

     The software chains the transistors automatically during layout generation.

© 2023 Cadence Design Systems, Inc.

- To automatically chain the individual folds of a transistor, select the *Chain Folds* check box and click OK.

  The software automatically chains the individual folds of a transistor. The *Chain Folds* check box is active only when the *Chain* option is deselected and the *Fold option* is selected.

Update Components And Nets chains only new devices created by the command itself. The new chains are separate and do not attach to old chains. Devices that had been chained prior to running *Update Components And Nets* are not moved from their existing positions.

Related Topics

Generate Layout Form

lxStackPartitionParameters

Generating All Components from Source

Components and Nets Updates

Manual Device Abutment

Device abutment is the process by which a number of MOS transistors (or the individual folds of folded transistors) are chained in a specific order. Chaining devices helps reduce layout area and capacitance.

Prerequisites Manual Device Abutment

The transistors to be chained:

- Need not be bound to a schematic instance.
- Must be properly set up for abutment.
- Must have the `lxComponentType` property set on the cell or library to which they belong. You set this property using the *Edit Component Types* command.

All parameter values are retained after chaining. The chaining engine always creates pairs of P and N chains with the P chain on top (because the power rail is typically at the top of a cell) and the N chain at the bottom (because the ground rail is typically at the bottom).

© 2023 Cadence Design Systems, Inc.

You can use the following commands to chain devices manually.

- *Generate Chained Devices* lets you chain devices interactively while you are editing a layout.
- The *Chain* command in the instance shortcut menu lets you chain devices selectively while you are editing a layout.
- *Generate Selected From Source* can automatically chain devices while generating a layout.
- *Move* lets you control chaining while moving devices.

The results generated by the chaining engine are affected by the setting for the `lxStackPartitionParameters` environment variable. The *Generate Chained Devices* command considers the first parameter of this environment variable only if there are 100 instances or more in the selected set. *Generate Selected From Source* always considers both parameters.

Related Topics

Pcell Set Up for Abutment

Component Types Mode in the CPH Window

Chaining Devices Interactively

Chaining Devices Using the Shortcut Menu

Chaining Devices when Generating Selected From Source

Moving Chained Transistors

lxStackPartitionParameters

Chaining Devices Interactively

To chain transistors interactively while you are editing a layout:

1. From the layout window menu bar, choose *Connectivity – Generate – Chained Devices*.

© 2023 Cadence Design Systems, Inc.

The Generate Chained Devices form appears.



2. Set the options you require.

   - To prevent existing chains from being broken during interactive chaining, select *Preserve Existing Chains*.
   - To maintain the relative starting positions of the selected devices in the generated chain, select *Use Device Order*.
     - To enable the dummy instances to change their nets on both sides during interactive chaining to support abutment on both sides, set the `chainDummyFlexBothEndNets` environment variable to `true`.

     When *Use Device Order* is deselected, dummy and unconnected devices are not chained. These devices are placed down separately in the order: chains, dummies, and then unconnected devices. Part-connected devices are chained, where possible, with nets assigned to enable device abutment.

     If `useDeviceOrder` is set to `t`, advanced node devices are chained even if their S/D pins are on different nets. In this case, the devices are abutted on dummy pins.
   - To select entire chains with a single click, choose *Extend Selection*. Otherwise, each click selects an individual device, mfactored device, or folded leg.
   - To identify nodes that qualify as pseudoparallel connections so that no current passes through the connecting nets, choose *Interdigitate Chains*.
   - To mirror the instances in a chain, choose *Mirror*.

© 2023 Cadence Design Systems, Inc.

- To permute the instances in a chain, choose *Permute*.
- To create the device chains on layout canvas at the same location as the left-most instance selected for chaining, choose *Create In Place*. If at least one of the instances selected for chaining are on a row region, the chain is created at the same location as the left-most instance in the row region.
- To define the alignment of p- or n-type devices, select an appropriate alignment using the *Align PMOS* or *Align NMOS* drop-down list. By default, the NMOS and PMOS alignment is controlled by the current value of the environment variables, lxChainAlignNMOS and lxChainAlignPMOS, respectively.
- If only one of the devices to be abutted has a bulk, ensure that the chainUseDeviceOrder and chainAllowSingleBulk environment variables are set to `t`.
- If you are using a FinFET design, the alignment options will be displayed as *Align PFIN* and *Align NFIN.*
- In the layout canvas, select the devices you want to chain.

Alternatively, click the *Show Device List* button to see the list of devices available for chaining. The list box appears in the Generate Chained Devices form.

You can also select devices in the layout canvas before you start the command. In this case, the selected devices are highlighted in the form when it opens.

The selected devices are highlighted in the canvas.

3. Click *Apply* and move the cursor into the layout canvas.

   The selected transistors appear as an image of a transistor chain that follows the cursor.

   If the majority of the selected devices are oriented vertically, the generated chain is also oriented vertically.

   a. Click the middle mouse button to rotate a chain by R90.

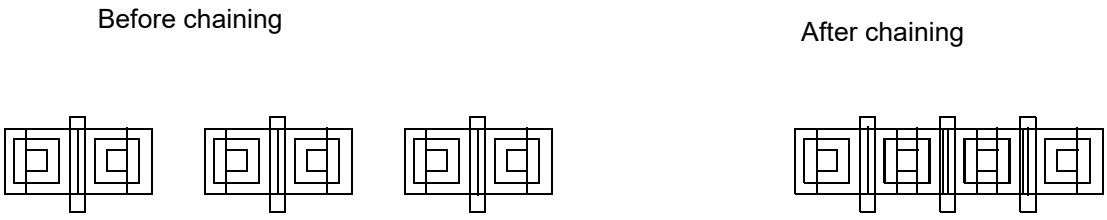
   b. `Shift`-click the middle mouse button to flip a chain alternately about its X and Y axes.

   You must have `lxBindKeys.il` loaded in order to use this functionality.

4. Click the layout canvas where you want to place the chained transistors.

© 2023 Cadence Design Systems, Inc.

The transistors are placed where you click.



You can also chain dummy instances interactively by using the procedure described above. If you select the *Use Device Order* check box, dummy devices are treated as regular devices during the chaining operation and can support abutment on one side or on both sides depending on the state of the `chainDummyFlexBothEndNets` environment variable. Dummy devices with floating S/D nets are also supported for interactive chaining. However, if *Use Device Order* is not selected, dummies are ignored during chaining. As a result, irrespective of their order, dummies are placed at the end of the chain and their original connectivity is retained.

Chains that comprise only NMOS or only PMOS transistors are aligned with the bottom edge. If you use the *Generate All From Source* command to create a cluster comprising both an NMOS and a PMOS transistor, the PMOS is placed on top aligned to the lower edge, and the NMOS is placed below it aligned to the upper edge.

Folded transistors are chained with the number of folds specified in the Generate Folded Devices form.

Related Topics

Generate Chained Devices Form

chainDummyAbutment

Generate Folded Devices Form

Folding a Transistor or FinFET Device

Chaining Devices Using the Shortcut Menu

To manually chain transistors using the instance shortcut menu:

➡ In the Navigator assistant, select the instances to be chained and then right-click to choose *Chain – <Appropriate Align option>* from the shortcut menu.

© 2023 Cadence Design Systems, Inc.

The *Chain* command in the shortcut menu chains the selected set with *Use Device Order* on and aligns the devices to default, top, center, or bottom depending on the alignment option selected from the shortcut menu.

Chaining Devices with or without Contacts

Layout XL enables you to choose whether or not to keep contacts for each chain in the selected set. You can set the option to keep or drop contacts.

To control contacts in a selected chain of chains:

➡ Right-click to select the chain and choose either from the shortcut menu:

- *Contacts – On* to keep contacts.
- *Contacts – Off* to delete contacts.

The *Contacts* option is disabled, if the selected set does not contain a chain.

The selected chains have their contacts kept or deleted in the layout canvas, as appropriate.

Related Topics

Generate Chained Devices Form

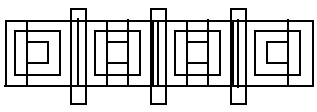
Removing a Device from a Chain

To remove a device from a chain:

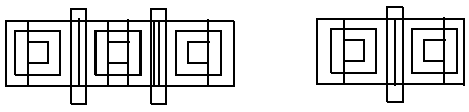
1. In the layout canvas, select the device you want to remove from a chain.

   The `chainExtendSelection` environment variable must be set to `nil` to let you select an individual device in a chain.
2. Choose *Edit – Move*.
3. Click the layout canvas where you want to place the selected device.

Before removing device from chain



After removing device from chain



© 2023 Cadence Design Systems, Inc.

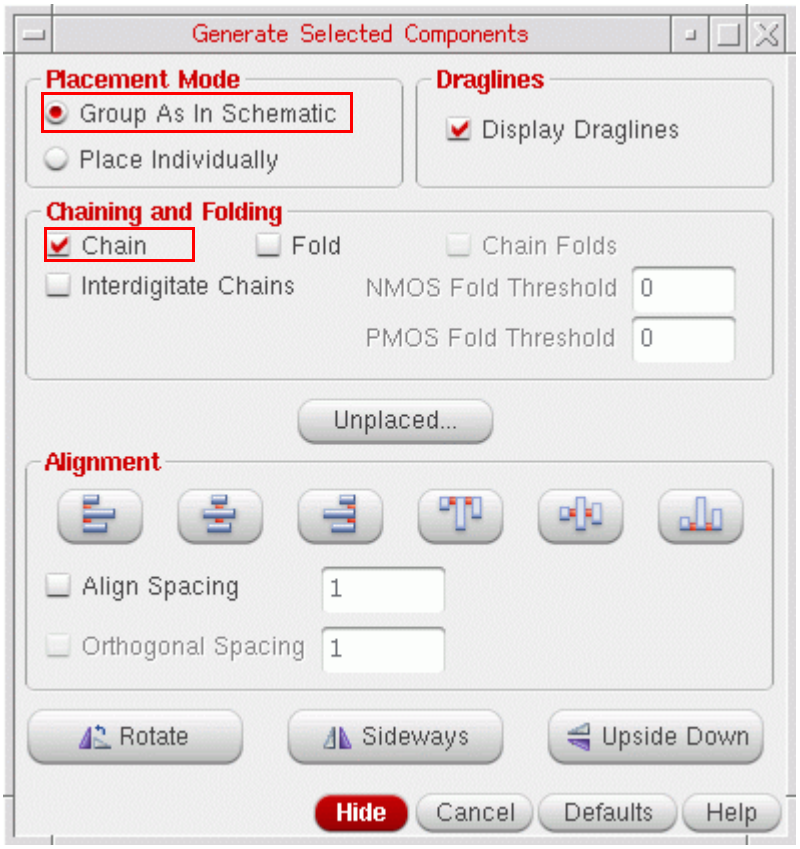
Related Topics

chainExtendSelection

Chaining Devices when Generating Selected From Source

To chain devices automatically during the *Generate Selected From Source* command:

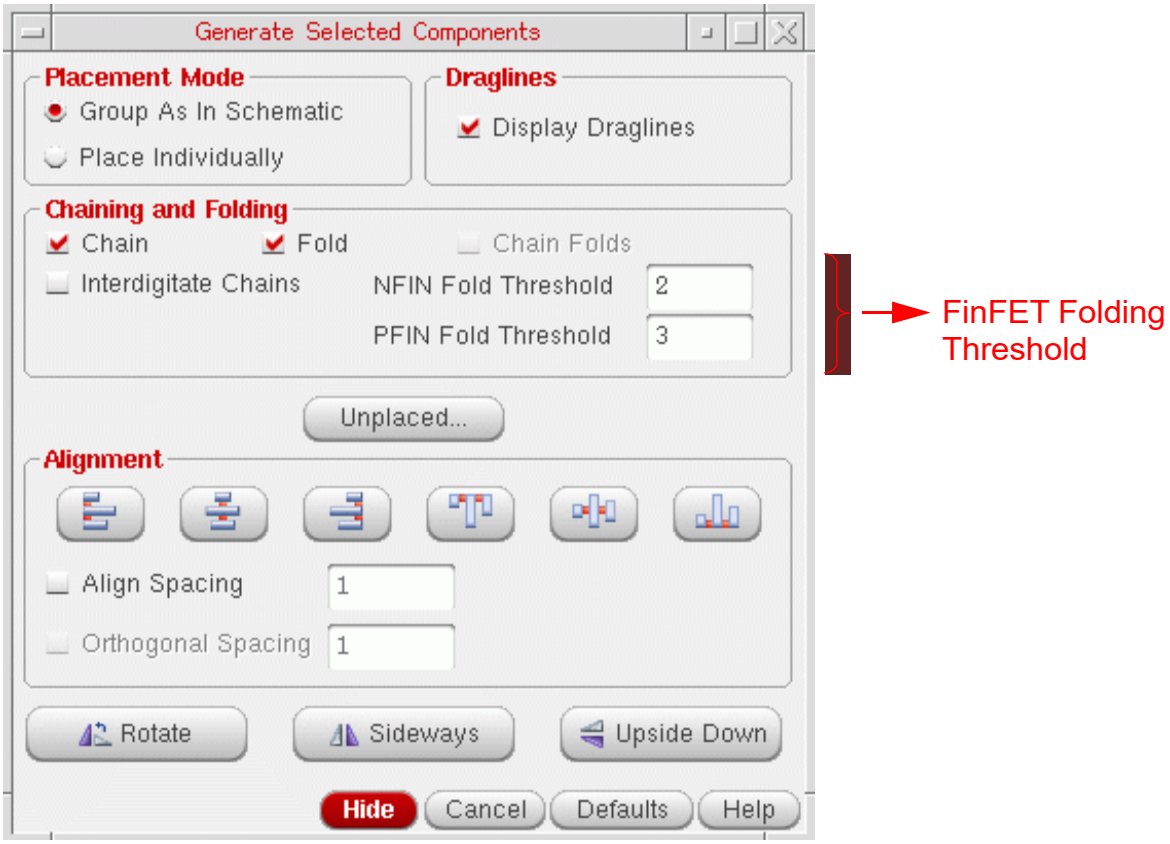
➤ In the Generate Selected Components form, select the *Group As In Schematic* and *Chain* options.



The results generated by the chaining engine during this command are affected by the setting for the `lxStackPartitionParameters` environment variable.

© 2023 Cadence Design Systems, Inc.

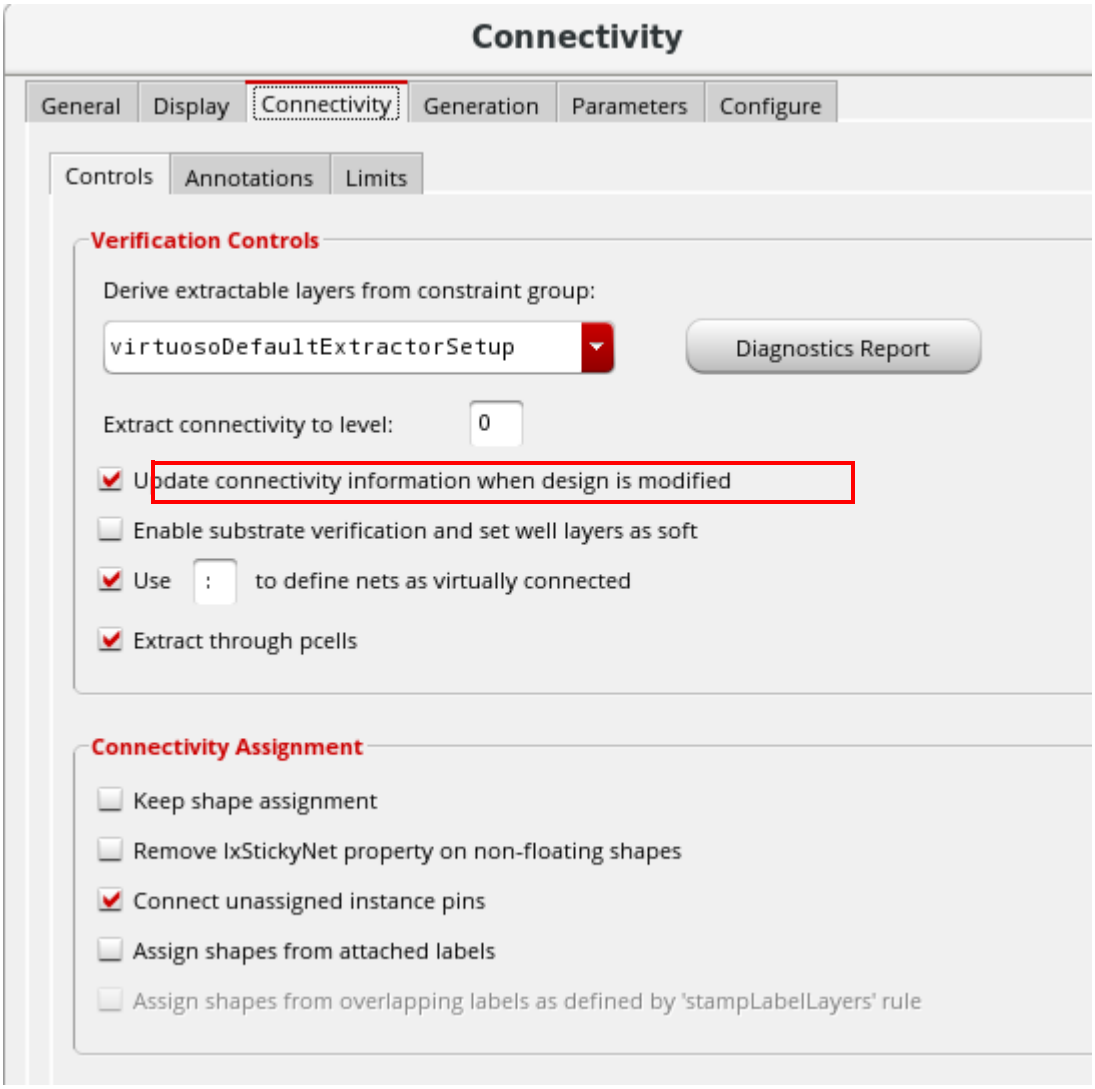
If you are using a FinFET design, the chaining and folding threshold fields are displayed as *NFIN Fold Threshold* and *PFIN Fold Threshold*, respectively, as shown in the figure below.



© 2023 Cadence Design Systems, Inc.

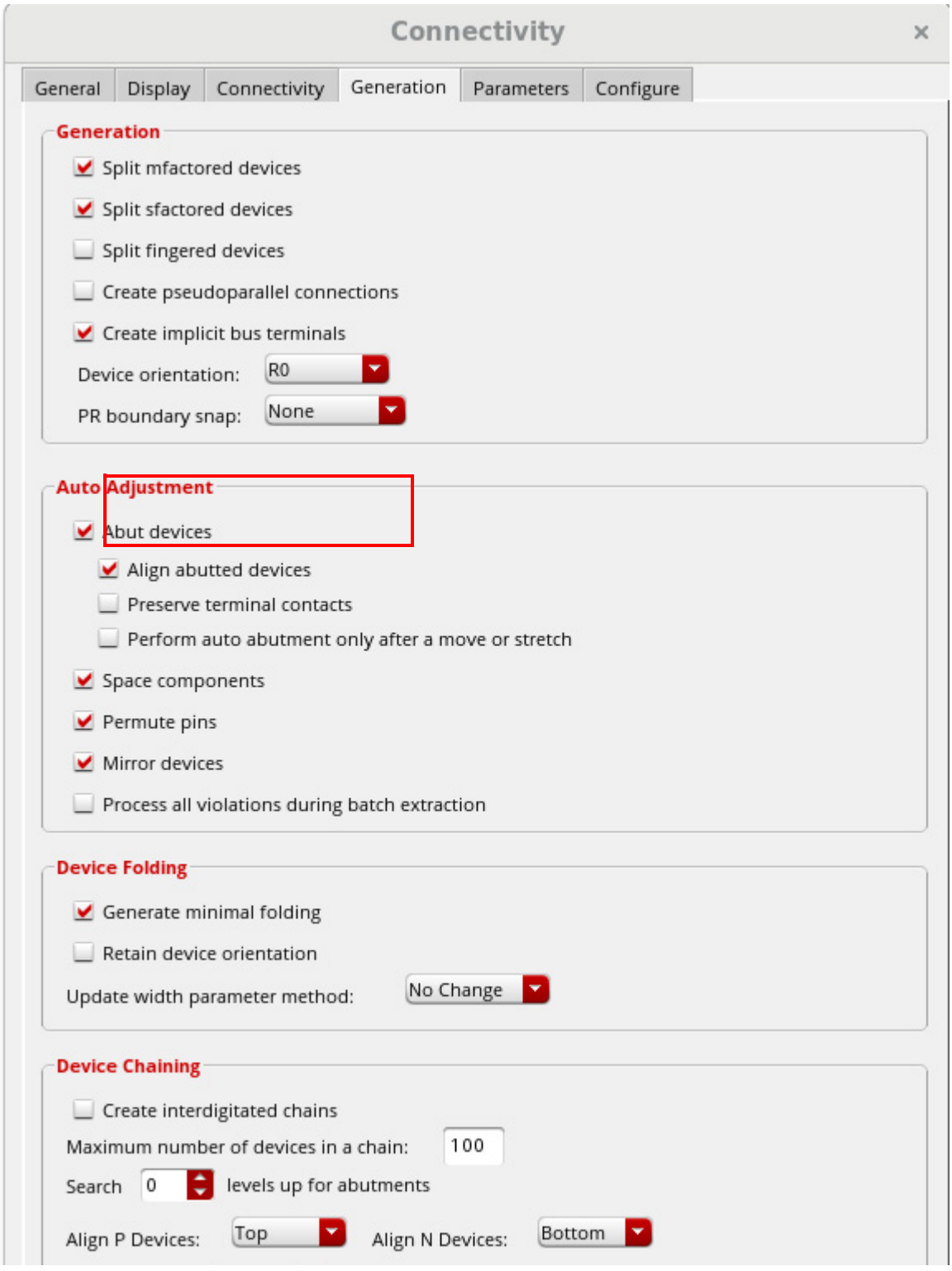
Layout XL cannot automatically chain devices in Place Individually mode. To do this, on the Connectivity form:

1. Open the *Connectivity* tab and choose *Update connectivity information when design is modified*.



© 2023 Cadence Design Systems, Inc.

2. Open the *Generation* tab and select *Abut devices*.



© 2023 Cadence Design Systems, Inc.

3. From the layout window menu bar, choose *Generate Selected From Source*.

   The Generate Selected Components form appears.

4. In the schematic, select the instances you want to place, then click in the layout where you want to place the first and second instances.

   The software places the instances where you click in the same order that you selected them.

5. Where the flight lines indicate that abutment is possible, place the devices so that the pins overlap.

   The devices are chained automatically.

Related Topics

Generate Selected Components Form

Connectivity Form

lxStackPartitionParameters

Moving Chained Transistors

To move chained devices while maintaining the chains:

1. From the layout window menu bar, choose *Edit – Move*.

   The Move form is displayed.

2. Select a chained device and choose the *Chain Mode* as *All*.

3. Click the reference point for the move.

   All the devices in the chain are highlighted.

4. Click in the layout where you want to place the devices.

5. Change the *Chain Mode* option to *Selected* and click the selected device.

   The selected device is moved and the device is no longer a part of the chain.

   To move the device back into the chain, choose *Edit – Undo*.

6. Select a device in the center of the chain and change the *Chain Mode* to *Selected Plus Right*.

© 2023 Cadence Design Systems, Inc.

7. Click the reference point for the move.

   You see that all the devices to the right of the selected device remained chained as you move the devices.

   Selected Plus Left option selects all chained devices to the left of the selected device.

8. Click in the layout where you want to place the devices.

Related Topics

Move Form

Viewing Extended Chain in Navigator

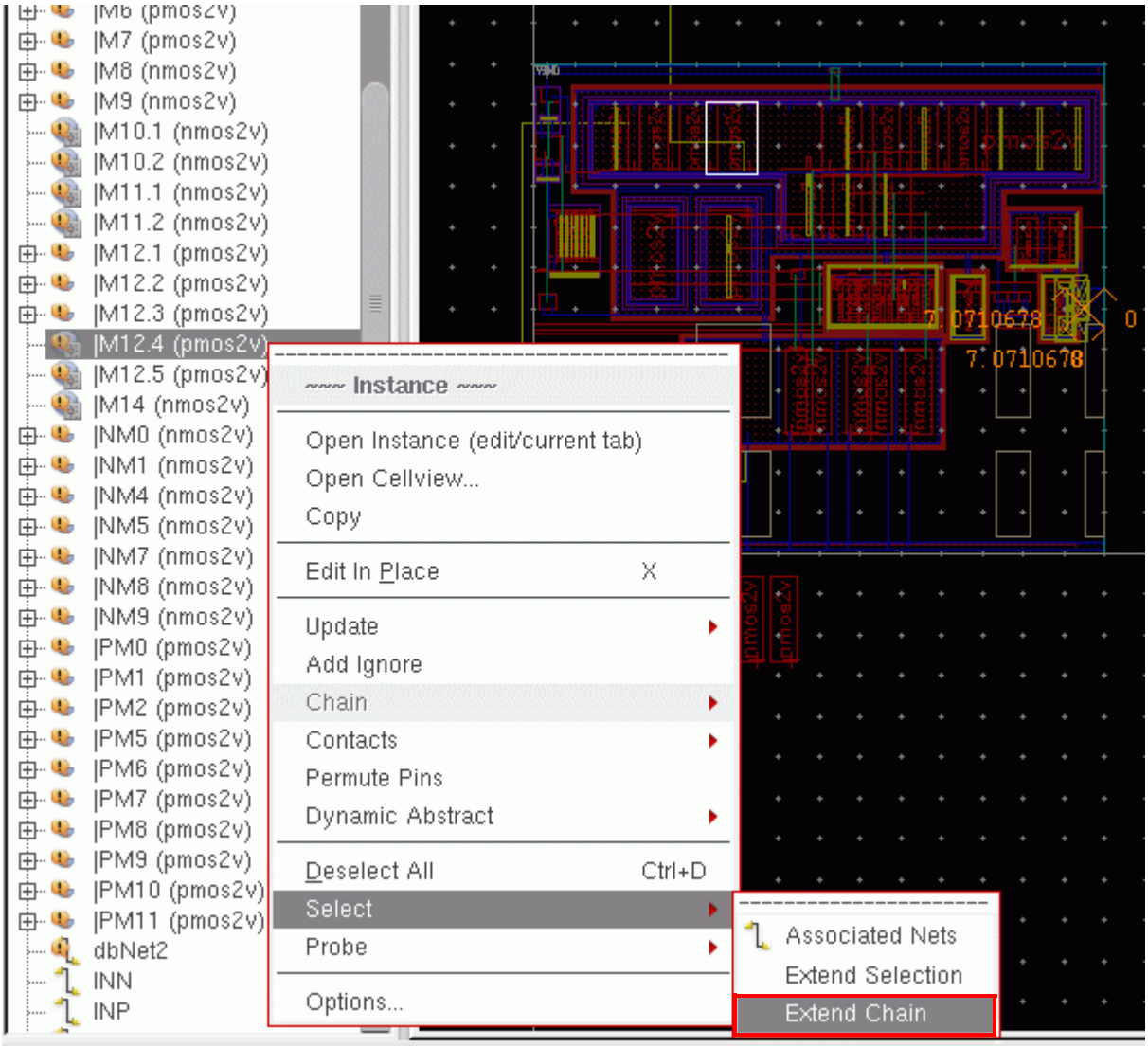
For an abutted instance that you select in the Navigator assistant, you can choose to extend the selection to view the entire chain highlighted in the Navigator assistant and cross-selected in the layout canvas.

To extend the selection to display the entire chain:

➡ Right-click an instance in the Navigator assistant (or in the layout canvas) and choose *Select – Extend Chain*, as shown in the figure below.

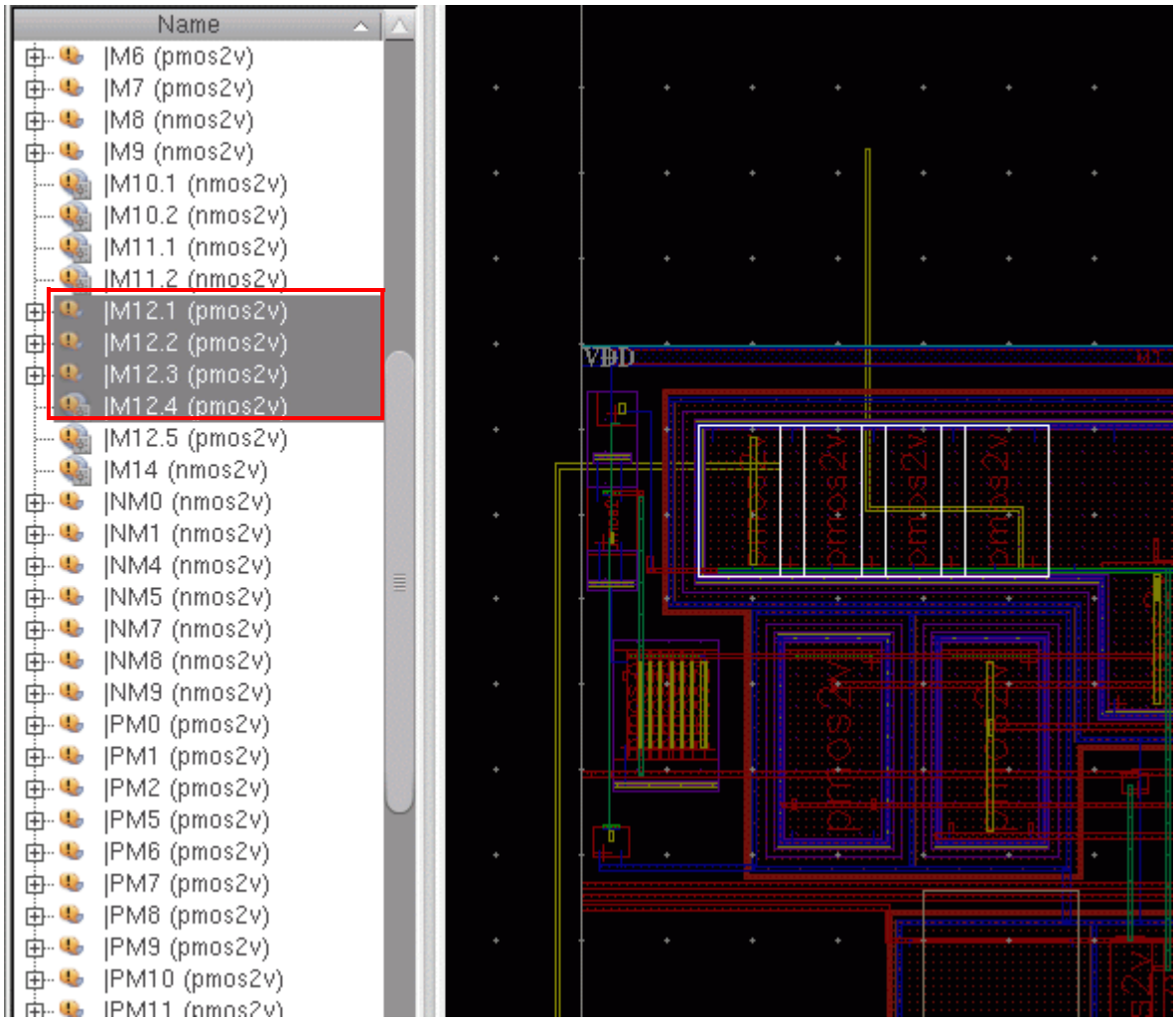
© 2023 Cadence Design Systems, Inc.

Alternatively, select the abutted instance in the Navigator assistant (or in the layout canvas) and call the lxSelectedExtendChain SKILL function.



© 2023 Cadence Design Systems, Inc.

All the instances that are abutted to the selected instance are selected in the Navigator assistant as well as in the layout canvas, as shown below.



If the selected instance is not abutted to any other instance, the *Select* menu does not display the *Extend Chain* command.

Now, consider a layout design that has two chains and an instance that is unabutted, such that:

```
Chain 1 = I1, I2, I3

Chain 2 = I4, I5, I6

Unabutted Instance = I7
```


© 2023 Cadence Design Systems, Inc.

Let us now consider some scenarios to see how the *Extend Chain* command behaves when the selected instance belongs to either or both the chains, or is an unabutted instance from the example above.

Scenario 1: *Select `I1` and run Extend Chain*

Because the instance *`I1`* is abutted to instances *`I2`* and *`I3`*, the selection is extended to the entire chain. Therefore, the selection set now includes: *`I1, I2, I3`*.

Scenario 2: *Select `I1` and `I4` and run Extend Chain*

Here, because each of the selected instances, *`I1`* ***and*** `I4`, is abutted to a set of instances, the selection is extended to include the abutted instances from both the chains. Therefore, the selection set now includes: *`I1, I2, I3, I4, I5, I6`*.

If any of the selected instances are not part of chain, they are still retained in the selection set after the command is run.

Scenario 3: *Select `I1` and `I7` and run Extend Chain*

Here, because the instance *`I1`* is abutted to instances *`I2`* and *`I3`*, the selection is extended to the entire chain. Also, instance `I7` is retained as selected although it is not part of a chain. Therefore, the selection set includes: *`I1, I2, I3, I7`*.

Scenario 4: *Select `I7` and run Extend Chain*

Because the selected instance is not abutted, there are no instances available to extend the selection to. This is the reason why the *Extend Chain* command is not available for an unabutted instance, making this a *hypothetical scenario*. Also, if you select such an instance and call the `lxSelectedExtendChain` SKILL function, the function returns a `nil`, indicating the extend operation failed as no new instances could be added to the selected set.

Related Topics

Navigator

lxSelectedExtendChain

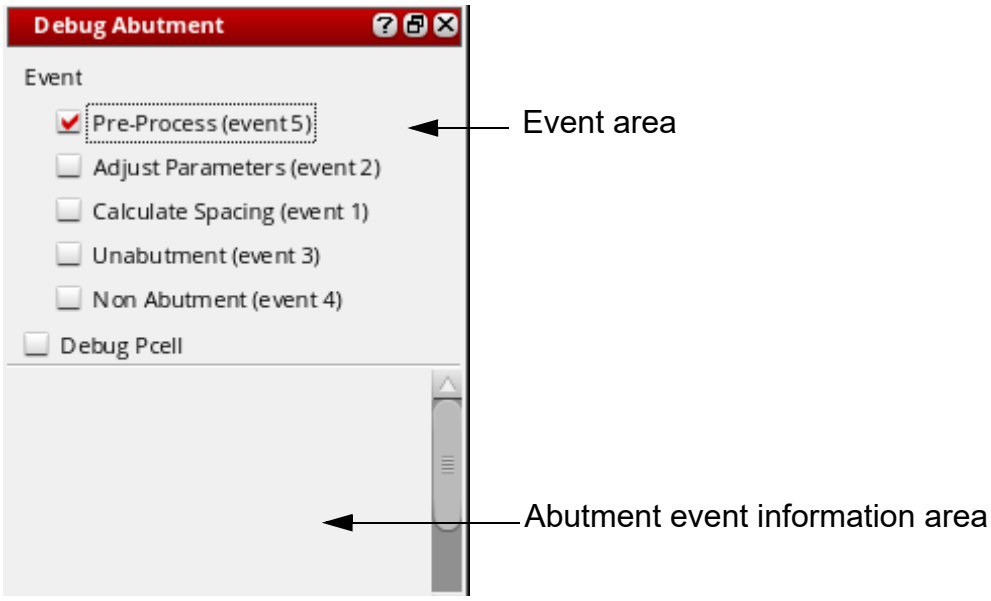
© 2023 Cadence Design Systems, Inc.

Debugging Abutment

The *Debug Abutment* plugin is available for cellviews with `maskLayout` view type and can be used in Layout XL to debug all abutment events.

To launch the plugin, choose *Launch – Plugins – Debug Abutment* from the layout window menu bar.

The Debug Abutment assistant is displayed.



You use this assistant to specify a set of events as break-points in a user-defined abutment function. The **Event** area lets you choose which event to debug.

- *Pre-Process (Event 5)*
- *Adjust Parameters (Event 2)*
- *Calculate Spacing (Event 1)*
- *Unabutment (Event 3)*
- *Non Abutment (Event 4)*

When a command triggers device abutment, the file containing the user abutment function is displayed in SKILL IDE. The device abutment flow pauses in the user-defined abutment function if the current event matches the events selected in the assistant and information on the pair of abutting devices is displayed in the **Abutment event information** area. You can then step through the abutment function using the SKILL IDE commands.

© 2023 Cadence Design Systems, Inc.

Related Topics

Using the Debug Assistant

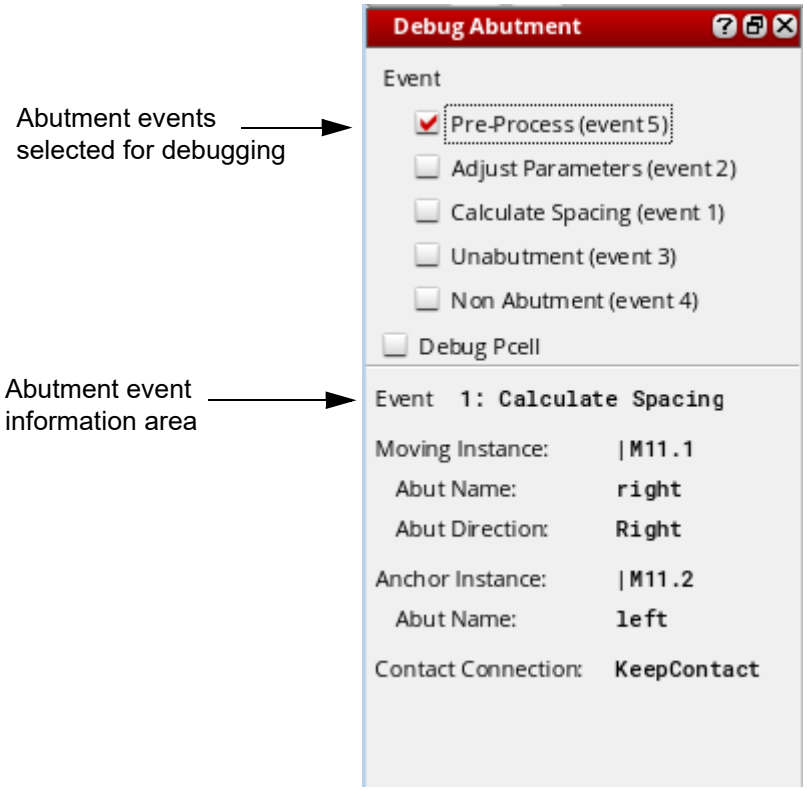
Using the Debug Assistant

This topic describes how to use the Debug Abutment assistant to debug a user-defined abutment function.

1. Choose *Launch – Plugins – Debug Abutment* from the layout window menu bar.

   The Debug Abutment assistant is displayed.

2. Choose the abutment events you want to debug.



When you abut two devices (or two chains), the device abutment information is displayed in the assistant and when one of the selected events is executed:

   a. SKILL IDE is opened automatically and displays the user-defined abutment function.

© 2023 Cadence Design Systems, Inc.

b. The device abutment flow pauses at the user-defined abutment function, giving you the opportunity to step through the abutment function to debug the specified event using SKILL IDE.

c. When current event is finished, control is returned to the abutment engine.

If another selected event is executed, the debug abutment flow returns to step 2a.

3. Select the *Debug Pcell* check box to debug the Pcell evaluation that is triggered by the code block that modifies the Pcell parameter inside the Pcell abutment function.
4. Repeat these steps until device abutment is finished.

The assistant does not show any abutment event results. To see these results, go to the Debug Abutment simulator.

Related Topics

Debugging Abutment

Abutment of Dummy Shapes in Pcells

At advanced nodes, cells are placed close together and such proximity can lead to various interconnect and layout-dependent effects. So, abutment at advanced nodes needs to be driven by specific design rules that will ensure optimal device placement, yield, and performance. For information about the design rules that are applicable at advanced nodes, see Supported Constraints and Parameters: Advanced Nodes Only.

Important

Abutment at advanced nodes requires a *Virtuoso_Adv_Node_Opt_Layout* license.

At advanced nodes, abutment of shapes in Pcell submasters, which are not attached to pins is supported. This is enabled by using SKILL functions for Abutment to provide such shapes an abutment name in the Pcell SKILL code, in addition to setting the usual abutment properties on these shapes. In the context of advanced node abutment, dummy shapes such as dummy poly gates or dummy diffusion areas can be made to abut by giving them an abutment name.

Video

For a video demonstration of dummy shape abutment (non-connectivity based), see Interactive Abutment.

© 2023 Cadence Design Systems, Inc.

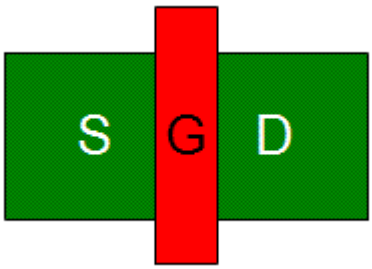
During incremental or full extraction, the connectivity extractor creates abutment violations for shapes inside Pcell submasters that have an abutment name, and are not pin figures. Because these shapes have no assigned connectivity at the top level, the extractor performs a shape chase to identify any such shapes at level 1 even when the default extraction depth is 0 and determines what connectivity should be assigned to each shape.

If the two overlapping shapes identified by the extractor are within Pcell submasters, which contain at least one shape each with an abutment name, then during abutment, the user-defined or PDK SKILL abutment function is called with a new event—Abutment Event 5. This new abutment event allows the abutment function to adjust the Pcell parameters and return a SKILL list containing a pair of shapes to be abutted. In the context of advanced node abutment, this new abutment event allows the abutment function to add or remove dummy shapes during abutment so that the shapes which initially overlapped to trigger the abutment can be totally different shapes (with different names) from the shapes that are finally abutted together.

Dummy shapes are defined by using the `customFill` layer purpose. For more information about the use of this layer purpose for creating these shapes, see Layer Purpose for Defining Dummy Shapes (customFill).

For more information about the Pcell abutment capability supported by Virtuoso Layout Suite XL, see Abutment Event 5. For information about the generic device abutment in Virtuoso Layout XL, see Debugging Abutment.

Let us now look at some examples that depict Pcells with and without dummy shapes. To begin with, below is an example of a Pcell without a dummy shape inserted.



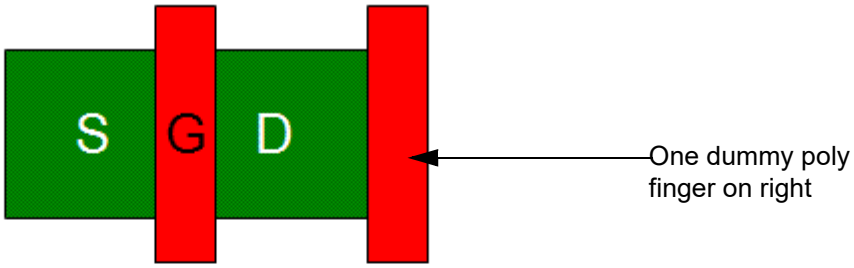
Pcell with No dummy shape

Before we see some examples of Pcells with dummy shapes added, it is useful to know that the dummy shapes in a Pcell can be controlled by using the Pcell parameters.

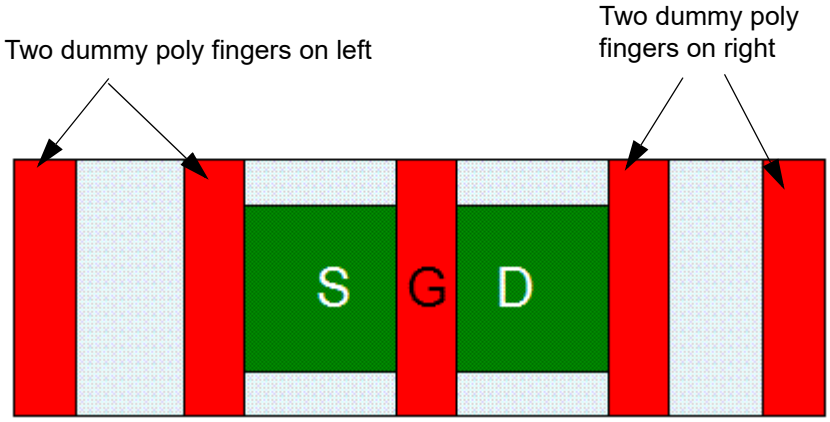
Let us now look at some examples of Pcells that have one or more *dummy "*poly*"* shapes added. Notice that in the figure below, the Pcell at the top has a single dummy poly finger

© 2023 Cadence Design Systems, Inc.

added to the right and the Pcell at the bottom has two dummy poly fingers added on either side.



Pcell with one dummy poly shape

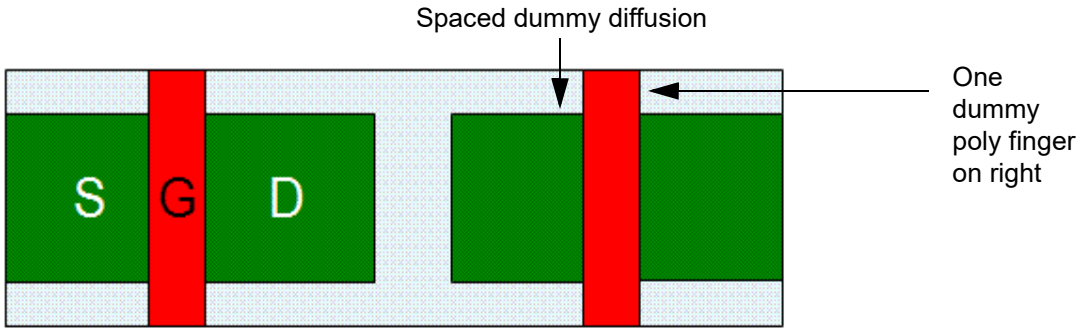


Pcell with two dummy poly shapes (fingers) on right and two dummy poly fingers on left

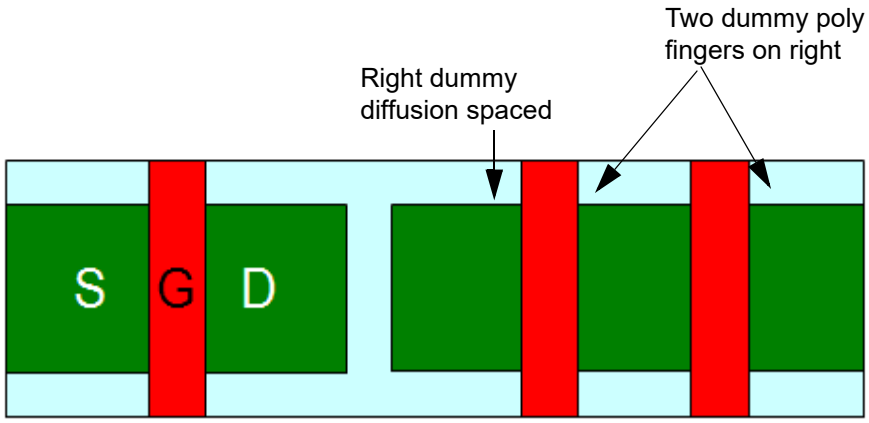
Now, let us look at some Pcells that have one or more *dummy "*poly*"* shapes and one or more "spaced" *dummy "*diffusion*"* shapes added.

© 2023 Cadence Design Systems, Inc.

In the figure below, the Pcell at the top has a single dummy poly shape and a dummy diffusion shape added. However, that the dummy diffusion shape is added at a distance from the transistor diffusion.



Pcell with one dummy poly and spaced dummy diffusion shape



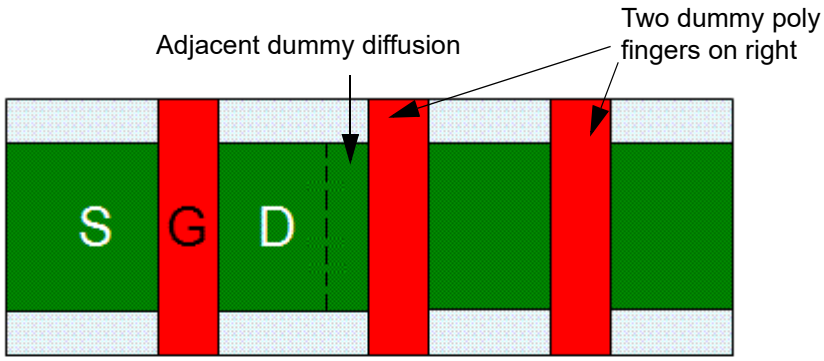
Pcell with two dummy poly shapes and spaced dummy diffusion shape

Now, let us look at some Pcell examples that have one or more *dummy poly* shapes and an adjacent *dummy diffusion* shape added. The dummy diffusion shape is placed immediately adjacent to the transistor poly or diffusion shape.

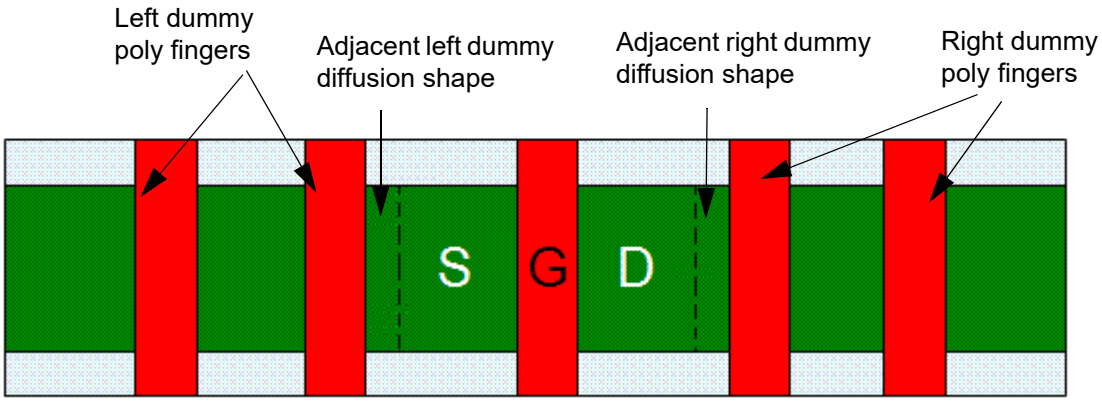
In the figure below, notice that the Pcell at the top has two *dummy* “poly” fingers and an adjacent *dummy* “diffusion” shape added. On the other hand, the Pcell at the bottom has two

© 2023 Cadence Design Systems, Inc.

dummy “poly” shapes on either side and the *dummy* “diffusion” shape is adjacent to the transistor diffusion.



Pcell with two dummy poly shapes and an adjacent dummy diffusion shape



Pcell with two dummy poly shapes and adjacent diffusion shapes on each side

Dummy Shape Abutment Setup

Abutment of shapes in Pcell submasters can trigger due to overlap of two “named” abutment shapes that are not attached to pins, or due to the overlap of shapes that are attached to pins.

For either of the overlap scenarios to result in abutment, the abutment shapes should have the usual abutment properties, such as `abutAccessDir`, `abutName`, and `abutFunction`, defined.

© 2023 Cadence Design Systems, Inc.

The names of the “named” abutment shapes are accessible only through SKILL.

To set up Pcell dummy shapes for abutment at advanced nodes, you must:

1. Use the `dbSetFigAbutName` SKILL function to set the `abutName` of each dummy shape.
2. Set up the same abutment properties as required for any other Pcell:
   - abutFunction Property
   - abutOffset Property
   - abutAccessDir Property
   - abutClass Property

Related Topics

Abutment of Dummy Shapes in Pcells

SKILL Functions for Abutment

dbSetFigAbutName

Abutment in Virtuoso Photonics Solution (Virtuoso Photonics Option)

When using the Virtuoso Photonics Solution with the `Virtuoso_Photonics_Option` license checked out, Layout XL enables abutment of Pcells. For photonic designs, this implies that Layout XL supports the abutment of optical waveguides.

If you set the `phoAbutNonPcells` environment variable to `t`, Layout XL also supports the abutment of non-Pcell instances with Pcell instances and other non-Pcell instances. During non-Pcell abutment, top-level pins can abut with instance pins. By default, instance pins on the same pin layer are abutted and aligned at the center.

For photonic designs, waveguide abutment is also supported.

Related Topics

phoAbutNonPcells (Virtuoso Photonics Option)

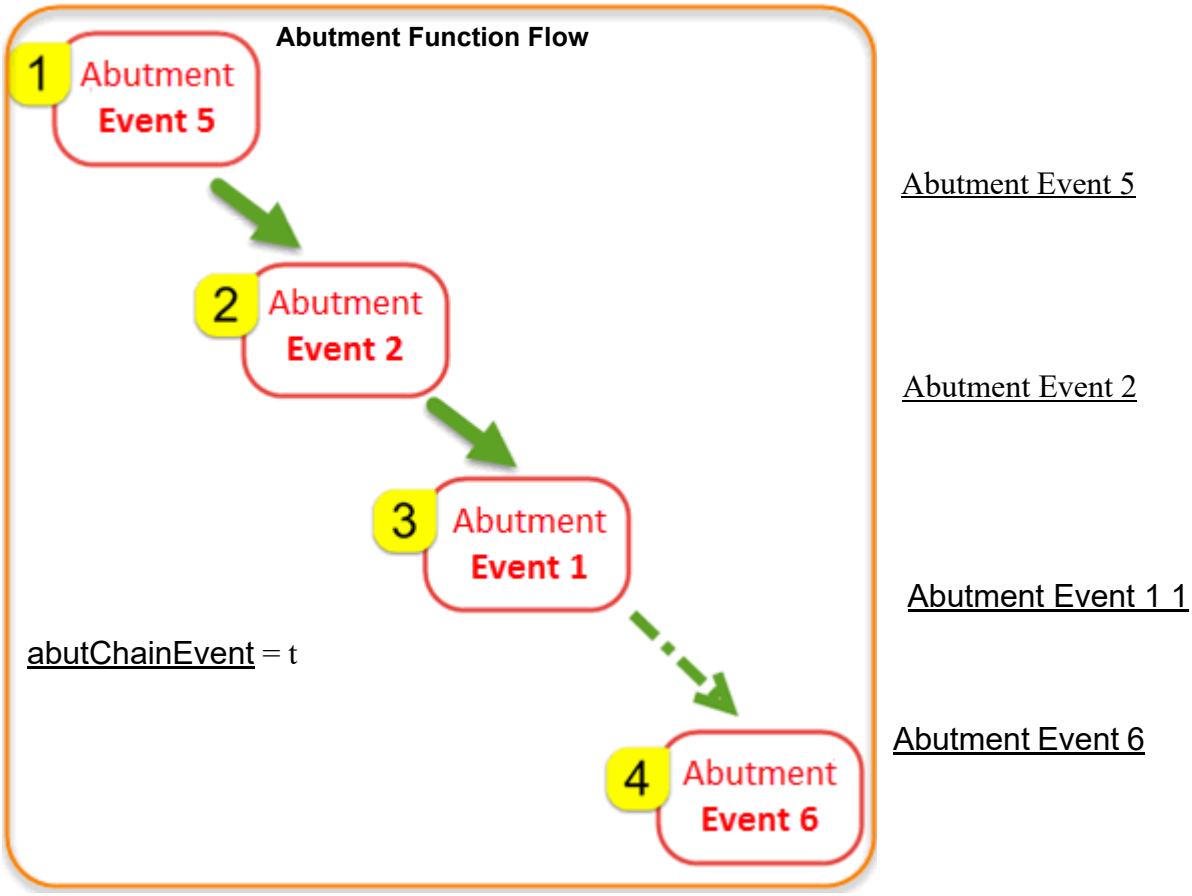
© 2023 Cadence Design Systems, Inc.

Abutment Function Flow

To enable the abutment of Pcell dummy shapes, Event 5 of the `abutFunction` flow gets precedence over all the other abutment events—Events 1 to 4 and Event 6. If both the overlapping Pcell pin figures or shapes support Event 5, the sequence in which the `abutFunction` calls these events is illustrated in the figure below.

Important

Abutment Event 6 is triggered only when the `abutChainEvent` environment variable is set to `true`.



- If either of the overlapping Pcell instances does not support Event 5, the `abutFunction` starts with Event 2.
- In the event that Pcell abutment fails or the dummy shapes are not considered abuttable, the `abutFunction` calls the following events, though the order in which these events are called is not fixed. Event 4 is triggered to allow auto-spacing, when required. Event 6 is triggered only when the `abutChainEvent` environment variable is set to `true`.

© 2023 Cadence Design Systems, Inc.

- Abutment Event 3
- Abutment Event 4
- Abutment Event 6

Related Topics

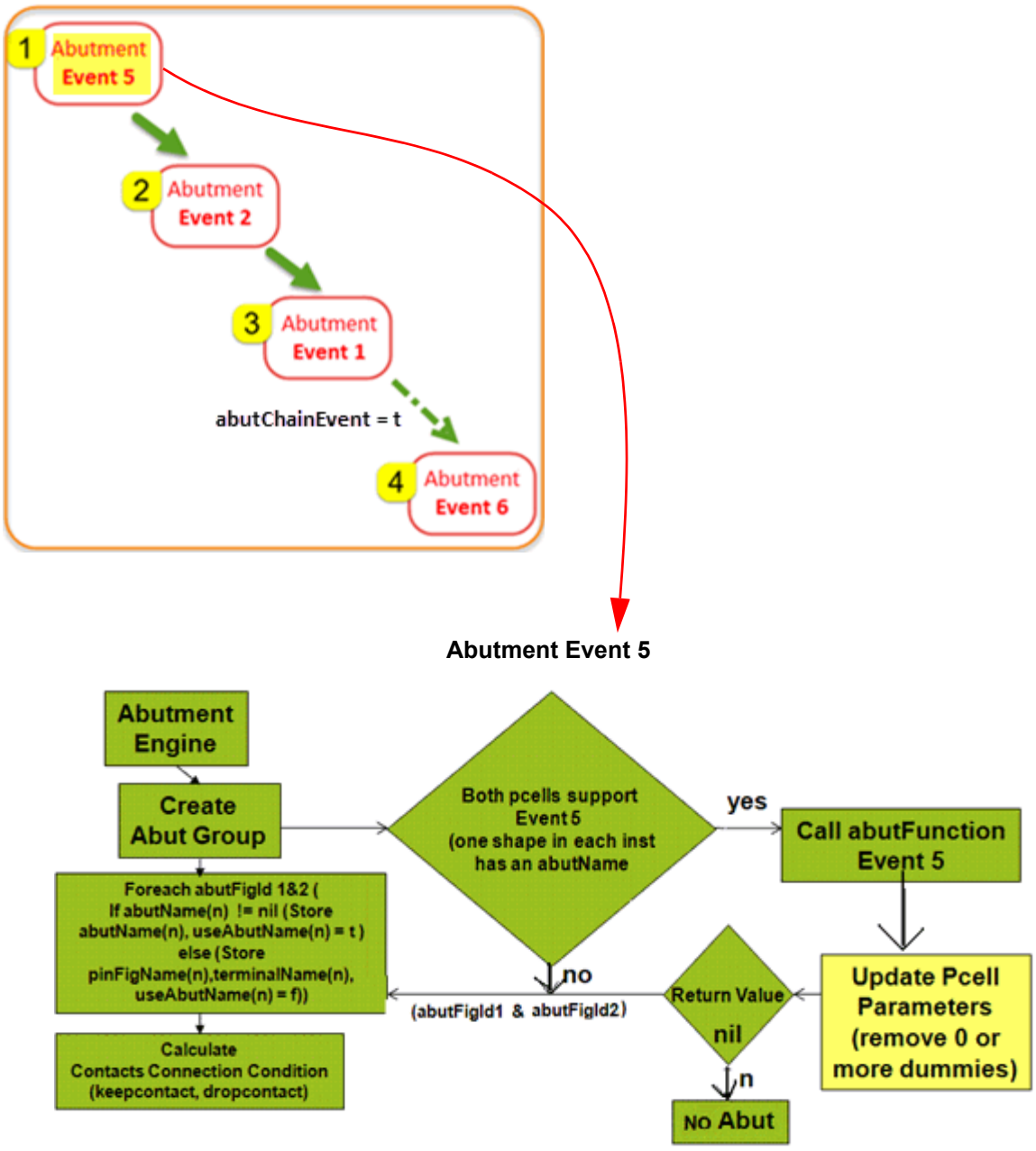
Unabutment Flow

abutChainEvent

© 2023 Cadence Design Systems, Inc.

Abutment Event 5

Abutment Function Flow



In abutment Event 5, after the abutment engine creates the abutment group and adds the instances that have at least one named abutment shape, the parameters of each Pcell instance get updated, if required. The parameter update allows the dummy shapes that

© 2023 Cadence Design Systems, Inc.

actually abut together to be different from the ones that initially overlapped to trigger the abutment. During the update, any undesired dummy shapes—poly or diffusion shapes—are removed.

- Abutment groups are created as dbGroups by the Virtuoso abutment engine. Pcell abutment functions can use these abutment groups to store and retrieve state information through properties, provided the Pcell abutment functions do not add or remove objects to or from the abutment groups, or modify their existing properties or attributes. For more information about an abutment group and the attributes it supports, see the V*irtuoso Studio Virtuoso Studio design environmentDesign Environment SKILL Reference* user guide.
- The abutment engine stores the names of the abuttable pin figures that are passed for Abutment Event 5 (or for Abutment Event 2, as required). The stored figIDs are later used to trigger the re-abutment, if unabutment was triggered due to a parameter update of the Pcell instances. For more information, see Abutment Event 3.

Related Topics

Abutment Event 3

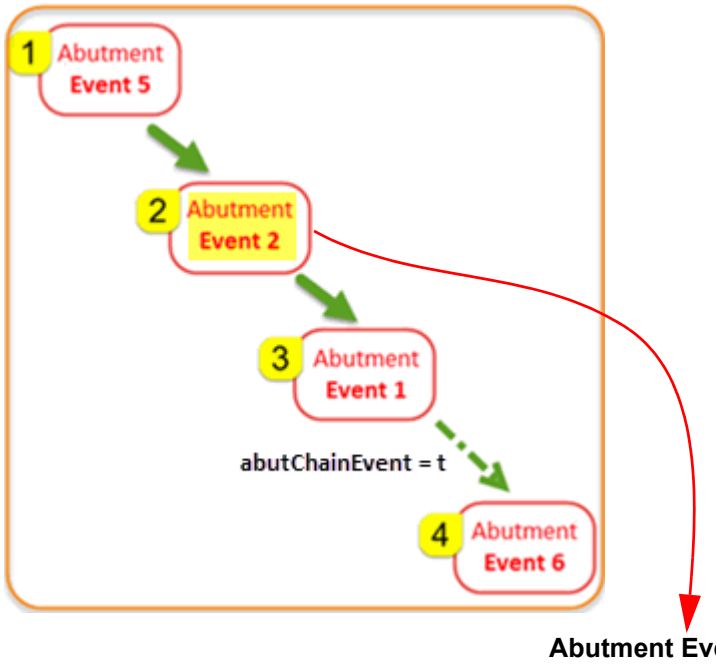
Abutment Event 5

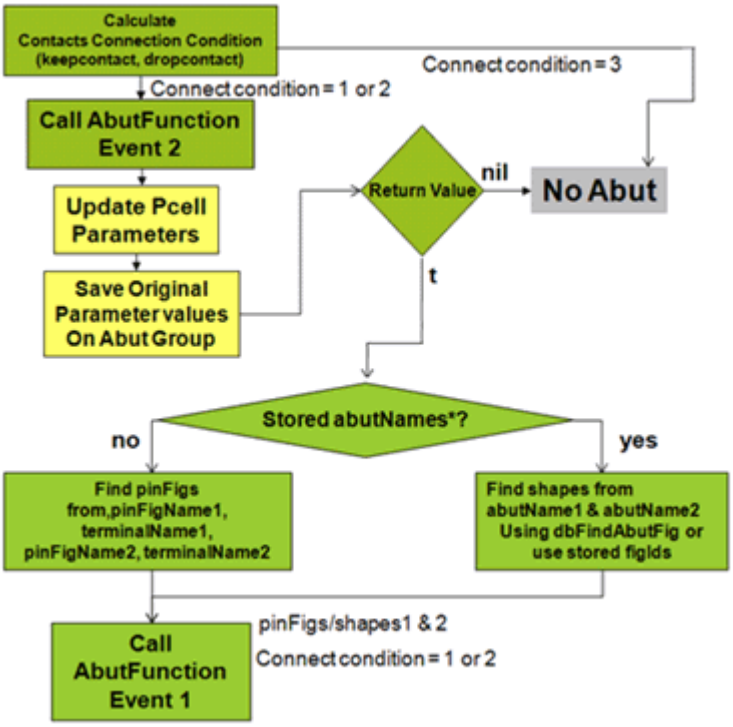
Abutment Event 2

© 2023 Cadence Design Systems, Inc.

Abutment Event 2

Abutment Function Flow





© 2023 Cadence Design Systems, Inc.

In Event 2, the abutment engine updates the Pcell abutment parameters on both the Pcell instances based on the connectivity and physical characteristics of the Pcells. This parameter update may result in creation of new masters, which can replace the old ones.

After being called with Event 2, if `abutFunction` returns `t`, the abutment engine calls the abutment function again, this time with Event 1. On the other hand, if `abutFunction` returns a value of `nil`, the Pcells are considered as unabuttable.

The original Pcell parameter values, prior to the update, are stored to be used later during an unabut call.

Computing Connection Condition for Abutment Event 2

After the parameter adjustment, the `abutFunction` returns the figIDs of the dummy shapes that are actually to be abutted. The returned shapes are used to compute the connection condition for Abutment Event 2 to determine if the connection contacts should be retained or removed during the abutment.

The shapes that can be returned by the abutment function and their contact condition for Event 2 are as given below:

1. *Both figs are pinFigs connected to the same upper-level net*

   In this case, the connection condition—whether the connection contacts should be kept or dropped—is based on the connectivity of the net at the upper level.

2. *Both figs are pinFigs connected to different upper-level nets*

   In this case, because the upper-level net connectivity for both the shapes is different, the abutment is aborted to prevent a short.

3. *Only one fig is a pinFig connected to an upper-level net*

   In this case, the connection condition—whether the connection contacts should be kept or dropped—is based on the upper-level net to which the pinFig is connected.

4. *Neither fig is connected to an upper-level net*

   In this case, the abutment engine returns shapes that are not pinFigs, or returns pinFigs that do not have their terminals connected to instance terminals or nets at a higher level. To determine the connection contact condition for this case, Virtuoso Layout Suite XL supports the following environment variables:

   - `abutWithoutConnectivity`
   - `preserveTerminalContacts`

© 2023 Cadence Design Systems, Inc.

If the abutment Event 5 returns a value of `nil`, the abutment is aborted.

Related Topics

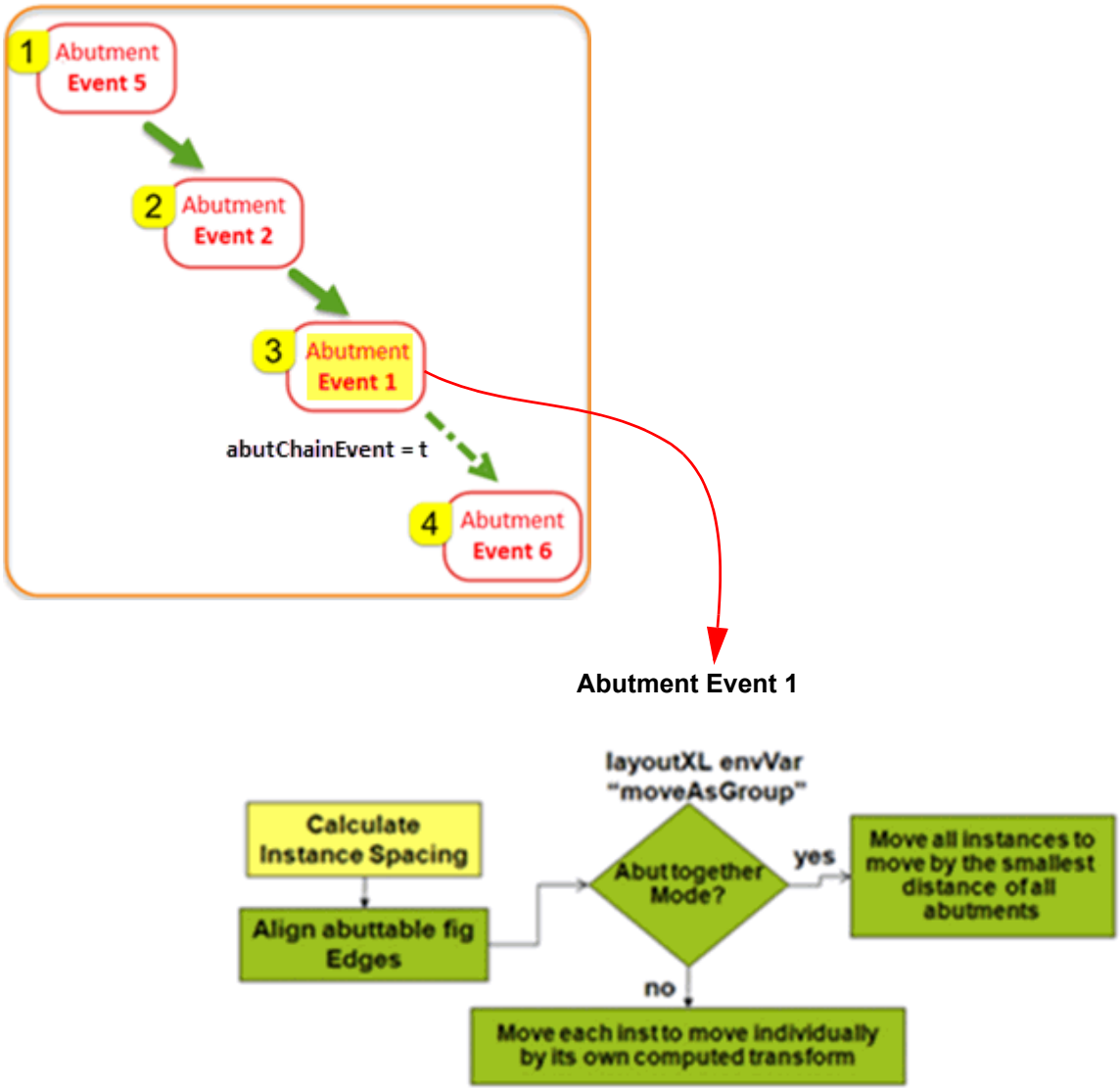
abutWithoutConnectivity

preserveTerminalContacts

© 2023 Cadence Design Systems, Inc.

Abutment Event 1

Abutment Function Flow



During the processing of Event 1, the abutment function calculates the spacing between two figs—pin figures or shapes—and returns the value. The use of `abutNames` takes precedence over the use of pin figure names to find the abuttable pin figures from the new master.

The abutment engine uses the abutment direction to determine how the instances should move so that the reference edges of the figures are properly spaced. If necessary, the perpendicular edges of the abuttable instances are aligned.

© 2023 Cadence Design Systems, Inc.

- If you have `abutTogether` mode selected, which implies that the Virtuoso Layout XL environment variable `moveAsGroup` is set to `true`, the instances move as a group by the smallest distance of all the abutments.
- By default, the abutment group ID is passed to abutment Event 1. You can use the group ID to store the Pcell properties or the Pcell state information at intermediate stages of abutment. You can also use the abutment group ID to retrieve the previously saved Pcell state information.

To prevent the abutment group ID to be passed to Event 1, set the `lxAbutPassGroupIdToEvent1` environment variable to `nil`.

Alignment Support During Abutment Event 1

- Horizontal and Vertical abutment is supported, depending on the abutting pinFigs `accessDir`.
- Based on the value specified during event1, abutting instances offset in x and y direction and determine the snapping mode.

The following data types can be returned during Event1:

- Number: Determines the total offset in the direction of abutment.
- String: Determines the snapping mode to be one of (`dbcAbutSnapTop`, `dbcAbutSnapBottom`, `dbcAbutSnapMiddle`, `dbcAbutSnapLeft`, `dbcAbutSnapRight`, `dbcAbutSnapCenter`) corresponding to the values provided (“`top`”, “`bottom`”, “`middle`”, “`left`”, “`right`”, “`center`”).
- List of two numbers (`x y`): First number decides the total offset in the direction of abutment and the second number decides the perpendicular offset.
- List with one number and one string (`x “left”`): First number decides the total offset in the direction of abutment and the string value decides the snapping offset to be applied.

The supported snapping modes (`dbcAbutSnapTop`, `dbcAbutSnapBottom`, `dbcAbutSnapMiddle`, `dbcAbutSnapLeft`, `dbcAbutSnapRight`, `dbcAbutSnapCenter` are detailed below. These snapping modes are useful when devices are of different heights and widths.

- Snapping modes (`dbcAbutSnapTop`, `dbcAbutSnapBottom`, `dbcAbutSnapMiddle`) are useful for horizontal abutment.
- Snapping modes (`dbcAbutSnapLeft`, `dbcAbutSnapRight`, `dbcAbutSnapCenter`) are useful for vertical abutment.
- `dbcAbutSnapTop`: Snaps the smaller device at the top edge of the bigger device.

© 2023 Cadence Design Systems, Inc.

- `dbcAbutSnaBottom`: Snaps the smaller device at the bottom edge of the bigger device.
- `dbcAbutSnapMiddle`: Snaps the smaller device in the middle of the bigger device in `Y` direction.
- `dbcAbutSnapLeft`: Snaps the smaller device at the left edge of the bigger device.
- `dbcAbutSnapRight`: Snaps the smaller device at the right edge of the bigger device.
- `dbcAbutSnapCenter`: Snaps the smaller device in the middle of the bigger device in the `X` direction.

Related Topics

lxAbutPassGroupIdToEvent1

Abutment Event 6

Abutment Event 6 is the final event triggered for a chain when the `abutChainEvent` environment variable is set to `true`.

Abutment Event 6 is common to both the abutment and the unabutment flows. By default, the event is called at the end of an abutment (or an unabutment) flow. After abutment, the event gets triggered if the size of the chain is more than one. After unabutment, the event gets triggered if the size of the chain is equal to or more than one.

Abutment Event 6 is an optional event that, when triggered, enables you to modify the resultant chain after abutment (or unabutment). For example, the event could be called to finalize the Source-Drain connections for a chain by adding (or removing) the top-level figures. To enable this, a new optional argument, `list of chain instances`, is passed to the PDK SKILL abutment function, `abutFunction`.

```
procedure( AbutFunction (iA iB pA pB pinAside conn event @optional (group nil)
(chain nil)))
```

The return value of the PDK SKILL abutment function for Abutment Event 6 does not get used by the abutment engine, unlike the return values of other abutment events, such as Abutment Event 2 and 5.

The PDK abutment function needs to be modified to support the new optional argument. Between each top-level figure and an instance in the chain, the PDK abutment function establishes a parent-child relationship such that the instance in the chain is the parent and the top-level figure is the child.

© 2023 Cadence Design Systems, Inc.

When abutted instances are part of a figGroup, during abutment Event 6, the top-level shapes are also added to the figGroup of the abutted instances.

Given below is an example that illustrates how the Abutment Event 6 can be used to modify a chain that is formed as a result of a recent abutment (or an unabutment) event.

For the Abutment Event 6 to be triggered, the `abutChainEvent` environment variable must be set to `t`:

```
envSetVal("layoutXL" "abutChainEvent" 'boolean  t)
```

The Abutment Event 6 is triggered using an optional `chain` argument in the abutment function callback, as illustrated below.

```
procedure(abutFunc(instA instB figA figB accessDir conn event @optional (group nil)
(chain nil))
prog( ()
case( event
( 1
return( myMosDoItEvent1(instA instB figA figB accessDir conn group))
)
( 2
return(myMosDoItEvent2(instA instB figA figB accessDir conn group))
)
( 3
return(myMosDoItEvent3(instA instB group))
)
( 4
return(myMosDoItEvent4(instA instB))
)
( 5
return(myMosDoItEvent5(instA instB figA figB accessDir conn group))
)
; event 6 for post-process whenever a chain is modified
( 6
return(myMosDoItEvent6(instA instB figA figB accessDir conn group chain))
)
)
return(t)
)
)


; Event 6 can be used for modifying completed chains, after abutment or unabutment,
to finalize the S-D connections by adding or removing top-level figures.
```


© 2023 Cadence Design Systems, Inc.

```
procedure(myMosDoItEvent6(instA instB figA figB accessDir conn group chain)
let((objs  (inst car(chain)) (propname "myEvent6PostProcPropName" ) )
; destroy user-added shapes
foreach(instId chain foreach(fig instId->children when(dbFindProp(fig propname)
dbDeleteObject(fig))))
; execute event6 processes
let(( (postcb list( 'myCallBackReturnDBObjList ) )  )
foreach(funcobj postcb
when(getd(funcobj)
objs = apply(funcobj list(chain))
if(typep(objs) == 'list   then
when(length(objs) > 0
foreach(obj objs
if(dbIsId(obj)  then
when(inst~>figGroup dbAddFigToFigGroup(inst~>figGroup obj)) ; add user-shapes to
parent figGroup
obj->parent = inst  ; add parent-child relationship
dbReplaceProp(obj propname "string" symbolToString(funcobj))
else
warn("invalid dbobject returned by %L" funcobj)
)
)
)
else
; warn("%L returns value illegal; should return a list of dbobjects." funcobj)
)
);when
); foreach
);let
);let
)


procedure( myCallBackReturnDBObjList(chain)
let((figs)
; modifies completed chains to finalize the S-D connections by adding top-level
figures, adjusting Pcell CDF parameters on chain instance figs, and returns a list
of added instances
))
```

Related Topics

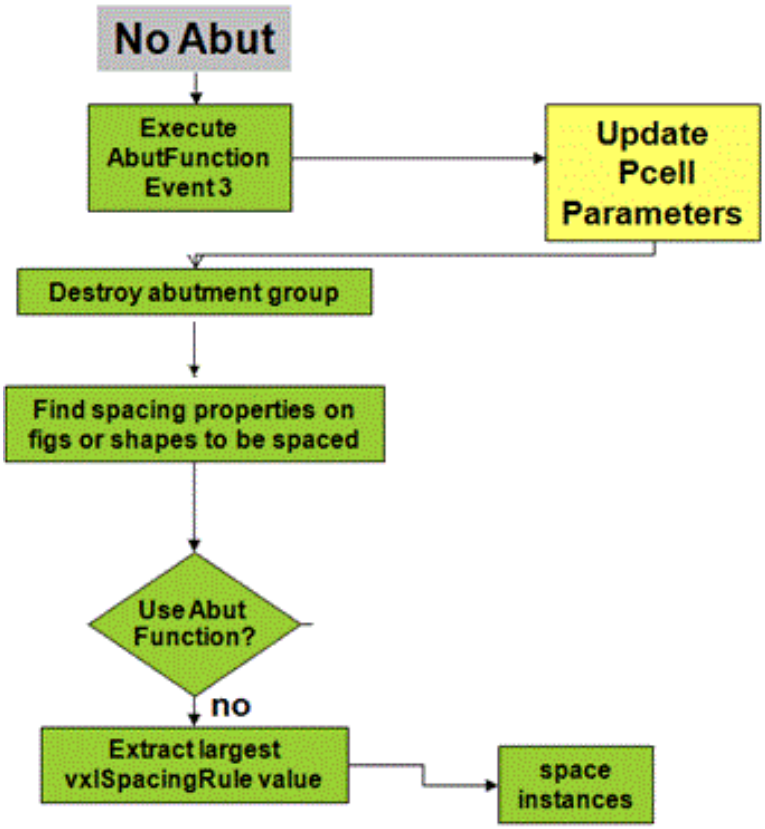
abutChainEvent

© 2023 Cadence Design Systems, Inc.

Unabutment Flow

Dummy Abutment Example

Abutment Event 3



Event 3 is called to perform unabutment, which can be triggered due to one of the following:

- A layout editor command, such as *Move*, which moves an abutted instance relative to another instance to which it is abutted. Or, due to a parameter update of the abutted Pcell instances using commands such as *Update Layout Parameters*, or during a manual parameter update of the Pcell instance using the *Property Editor* or a stretch handle.
- The abutment engine needs to unabut a partially completed abutment that cannot proceed further. For example, if Event 2 returns `nil` or the shapes returned by the event are pin figures on different nets.

In Event 3, the `abutFunction` retrieves the original parameter values that were stored by the function during Abutment Event 2 and Abutment Event 5. These values are then used to update the Pcell parameters to return the Pcell to its preabutment state.

© 2023 Cadence Design Systems, Inc.

If unabutment occurs due to the parameter update of Pcell instances, re-abutment is triggered immediately after the unabutment using the figIDs stored on the current abutment group. The abutment engine is now enhanced to store the figIDs that had triggered the abutment during Abutment Event 5 or Abutment Event 2, as required.

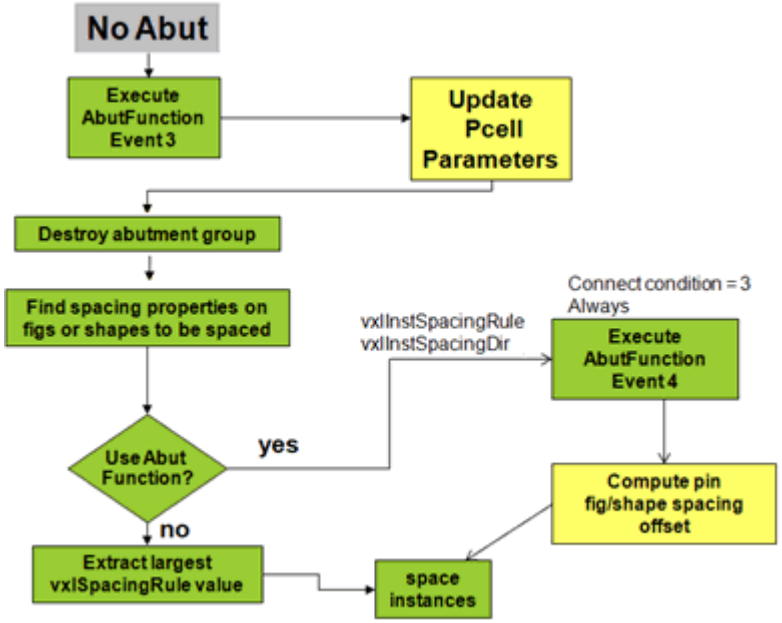
Having the figIDs of the originally abutted shapes available ensures that re-abutment can be performed using the same figIDs and in the same manner as the original abutment. Re-abutment is performed even if the abuttable shapes do not physically overlap after the parameter update.

To determine if explicit re-abutment will be triggered after unabutment, the PDK abutment SKILL Event 3 can call the Boolean SKILL function, `dbAbutIsReAbutGroup`. If the SKILL function returns `true`, the PDK SKILL function can store the relevant information, if needed, from the current abutment group. The stored information could be required later during re-abutment of the Pcell instances to achieve abutment with the same settings as before.

Important

The re-abutment of Pcell instances is triggered when the change in the Pcell instance parameters was triggered by an edit in Virtuoso Layout Suite XL, or an abutted instance was remastered. If the parameter change was triggered due to an edit in Virtuoso Layout Suite L, no re-abutment is triggered after unabutment.

Abutment Event 4



© 2023 Cadence Design Systems, Inc.

In this event, the abutment engine verifies if the pin figures or shapes have a spacing property set. If the value of that property is set to `abutFunction`, the abutment engine executes Event 4 using the “Connect Always” condition. This allows adjusting the wells and guard rings for the non-abutted instances. The pin figures or shapes are moved based on the value returned by the spacing property. If the spacing property is not set, the instances are not moved.

Related Topics

Abutment Event 2

Abutment Event 5

Abutment Event 6

Automatic Abutment Flow

Unabutment Flow

For design situations where abutment is not possible, the `abutFunction` calls the abutment events in a sequence that lead to unabutment. This is the case when the Pcell abutment fails or the dummy shapes are not considered suitable for abutment.

Other than abutment Event 3 that causes the unabutment, other events that are likely to be called during an unabutment flow, include:

- Abutment Event 4
- Abutment Event 6

For abutment Event 6 to be triggered, the abutChainEvent must be set to true.

Related Topics

Abutment Event 4

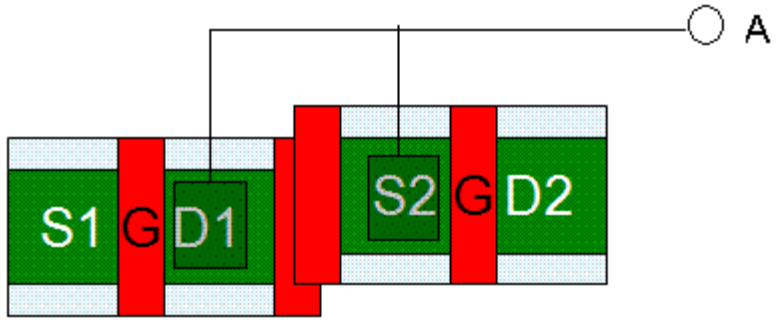
Abutment Event 6

abutChainEvent

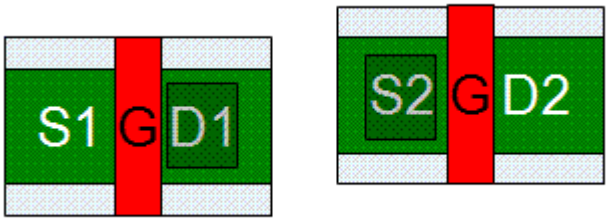
© 2023 Cadence Design Systems, Inc.

Dummy Abutment Example

Let us now consider an example depicting abutment of dummy shapes. Here, `D1` and `S2`, displayed in the figure below, are connected to a top-level net, `netA`, which has only two instance terminals.



The overlap of the two dummy poly shapes triggers abutment Event 5. This results in the parameters of the overlapping Pcell instances to be updated, causing removal of dummy poly on both the Pcell instances, as shown in the figure below.

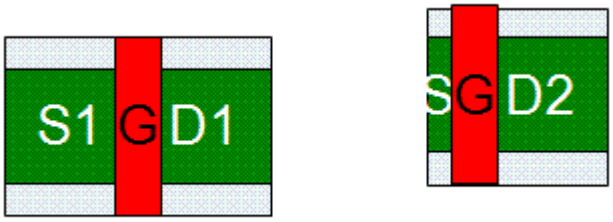


After the dummies are removed, the abutment engine identifies that `D1` and `S2` are connected to the same upper-level net, `netA`. This enables the engine to compute the contact condition to be passed to the abutment Event 2.

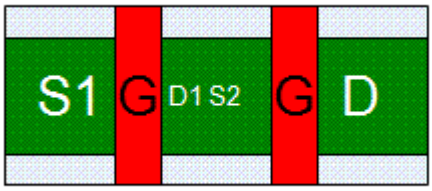
In this case, since both the pinFigs are connected to the same upper-level net, the *contact condition 1* is applicable, indicating that contacts need to be dropped.

© 2023 Cadence Design Systems, Inc.

During Event 2, the Pcell parameters are adjusted further. In this example, the further parameter adjustment results in removal of `S2` from the second instance and dropping the connection contacts, as shown in the figure below.



Finally, the engine returns the alignment offset for aligning the abutted pinFigs in the direction of abutment. In this example, the alignment offset is `0`, so the instances are abutted and aligned, as shown in the figure below.



Related Topics

Abutment Event 2

© 2023 Cadence Design Systems, Inc.

© 2023 Cadence Design Systems, Inc.

5

Layout Generation

You can use the Virtuoso® Layout Suite XL layout editor (Layout XL) to generate a layout and make an initial placement of components in it.

Related Topics

Naming Convention during Layout Generation

Generating a PR Boundary

Generation of Optical Pins and Chains (Virtuoso Photonics Option)

Generation of RF TLine Chains (Virtuoso MultiTech Framework)

Generating All Components from Source

Array Generation

Clone Generation

Synchronous Copy Generation

Dummy Instance Generation

Mosaic Instances Generation

Generating Parameterized Layout Cells (VPLGen)

Generating Master Variants

Transparent Layout Hierarchy Generation

© 2023 Cadence Design Systems, Inc.

Naming Convention during Layout Generation

When generating and manipulating components in the layout view, Layout XL uses the naming conventions outlined in this topic.

Manually changing a system-generated name, or renaming an instance such that its new name conflicts with a system-generated name, might lead to unpredictable behavior.

General Naming Convention

By default, layout instances are generated without the OR bar (|). But, you can set the `prefixLayoutInstNamesWithPipe` environment variable to `t` to delimit hierarchy and to prefix the names of all the layout instances generated by Layout XL.

Layout instance naming convention with `prefixLayoutInstNamesWithPipe` set to `t`.

Schematic	Layout
P0	|P0
NAND0/P0	|NAND0|P0

Layout instance naming convention with `prefixLayoutInstNamesWithPipe` set to `nil`.

Schematic	Layout
P0	P0
NAND0/P0	NAND0|P0

Mfactored Devices and Folded Devices

Devices specified using the `mfactor` property in the schematic and those that are folded in the layout are named *`instName.integer`* in the layout.

For example, if you have a schematic instance named `P0` with

```
mfactor = 3
```


© 2023 Cadence Design Systems, Inc.

the layout instances generated are named as follows.

Schematic	Layout (with OR bar)	Layout (without OR bar)
P0	|P0.1 |P0.2 |P0.3	P0.1 P0.2 P0.3

Series-Connected Devices

Devices specified using the `sfactor` property in the schematic are named *`instName`*`.s`*`integer`* in the layout.

For example, if you have a schematic instance named `R0` with

```
sfactor = 3
```

the layout instances generated are named as follows.

Schematic	Layout (with OR bar)	Layout (without OR bar)
R0	|R0.s1 |R0.s2 |R0.s3	R0.s1 R0.s2 R0.s3

lxCombination Devices

Devices defined using the `lxCombination` property in the schematic are named *`instName`*`.ms`*`integer`* in the layout.

For example, if you have a schematic instance named `R0` with

```
lxCombination = (2k*2)/3+1500/4
```


© 2023 Cadence Design Systems, Inc.

the ten layout instances generated are named as follows:

Schematic	Layout (with OR bar)	Layout (without OR bar)
R0	|R0.ms1 |R0.ms2 |R0.ms3 . . . |R0.ms10	R0.ms1 R0.ms2 R0.ms3 . . . R0.ms10

Nets in Series-Connected and lxCombination Devices

Nets associated with series-connected and `lxCombination` devices are named *`instName`*`.s`*`integer`*`.expandedNet`*`integer`* in the layout.

For example,

```
R0.s1.expandedNet1
```

where "`expandedNet`" signifies that the layout net has no real match in the schematic.

Related Topics

prefixLayoutInstNamesWithPipe

mfactor Property

sfactor Property

lxCombination Property

Constraint Transfer During Layout Generation

Layout XL transfers all the constraints defined in the schematic view to the layout view, correctly mapping the constraints and their members between the two views.

The constraint transfer is triggered:

- Automatically when you start Layout XL.

© 2023 Cadence Design Systems, Inc.

  - Constraints in the schematic are transferred to the top-level layout view. Constraints that have been created or changed in the schematic but not yet saved are also transferred.

- Automatically when you use the *Generate All From Source* command. Previously-transferred constraints are deleted and updated constraints are created in the layout.
- Automatically when you update components and nets.
- Manually when you use the *Update Layout Constraints* command.
- Optionally, when you load information from another physical cellview.

Device correspondence information is maintained during the transfer. Logical and physical name mappings are tracked and constraints are updated appropriately to take account of folded instances in the layout.

Constraint Manager Assistant During Layout Generation

You can manipulate these constraints on both schematic and layout views using the Constraint Manager assistant.

This lets you:

- Enter custom constraints in the schematic to drive layout. These are transferred to layout when Layout XL is initialized.
- Enter additional constraints in the layout, which are supported by the various physical design applications.
- Select a constraint in the Constraint Manager and see the constraint in question highlighted in the canvas.

Constraints transferred from schematic to layout are read-only and cannot be modified in the layout view. However, you can create new constraints in the layout to override the schematic constraints.

© 2023 Cadence Design Systems, Inc.

Constraint Mapping in One-to-Many Relationships

The following table shows how constraints are mapped when there is a one-to-many schematic to layout relationship.

Constraint type	Schematic	Layout

Symmetry pair	`symmetry(A, B)`	`symmetry(A.1, B.1)`
		`symmetry(A.2, B.2)`
		`symmetry(A.<N>, B.<N>)`

Matched orientation	`relOrient(A, B, "orient")`	`relOrient(A.1, B.1, "orient")`
		`relOrient(A.1,.., A.<N>, "R0")`
		`relOrient(B.1,.., B.<M>, "R0")`

Layout structure	`layoutStructure(A, B)`
	`layoutStructure(A, B, B, A)`
		`layoutStructure`
		`(A.1,..,A<N>, B.1,..,B<M>)`
		`SlayoutStructure`
		`(A.1, B.1, B.2, A.2)`

All other	`allOther(A, B)`	`allOther` `(A.1,.., A.<N>, B.1, ..,B.<M>)`

Constraint Mapping in Many-to-One Relationships

Many-to-one relationships are handled by not adding duplicate members to layout constraints.

The following situations cause a warning to be issued and no layout constraint generated.

- A symmetry constraint containing a pair of instances with have different multiplication factors.
- A layout structure constraint with schematic instances specified multiple times but which do not have matching layout instances; for example, the schematic instance is referenced three times in the constraint but has an mfactor of only `2` in the layout.

Creation and Synchronization of Clusters

To promote interoperability with designs that originate in Innovus and allow layout placement tools to use `oaCluster` constraints, the constraint transfer mechanism converts a `ciCon(cluster)` constraint in the schematic to an `oaCluster` constraint in the layout and keeps the two synchronized during subsequent transfer operations.

© 2023 Cadence Design Systems, Inc.

`ciCon(cluster)` constraints and `oaCluster` constraints are synchronized only in a forward direction (from `ciCon(cluster)` to `oaCluster`). A change to an `oaCluster` does not affect its corresponding `ciCon(cluster)`.

`oaCluster` constraints with no corresponding `ciCon(cluster)` constraints are never deleted by constraint transfer operations. If you delete from the layout an `oaCluster` constraint with a corresponding `ciCon(cluster)` constraint, the `oaCluster` is recreated the next time constraints are transferred from the schematic.

Related Topics

The Constraint Manager Assistant

Generating a PR Boundary

When you initialize a layout, Layout XL automatically generates a PR boundary for the design.

To set the default size for the PR boundary, from the layout window menu bar:

1. Choose *Connectivity – Generate – All From Source*.

   The Generate Layout form is displayed.

2. On the *Generate* tab, select the *PR Boundary* option.
3. On the *PR Boundary* tab, set the options to generate the shape and size of boundary you need.
4. When you click *OK* to generate the layout, the system draws the specified boundary in the layout canvas.

To create a design boundary manually:

➡ Use the *Create – P&R Objects – P&R Boundary* command.

To stretch, move, or delete the PR boundary.

➡ Use the layout editor *Edit* commands.

Related Topics

Generate Layout Form

© 2023 Cadence Design Systems, Inc.

PR Boundary

Creating a PR Boundary

Editing Objects

Constraint-Driven PR Boundary Generation

With advanced node licensing, during *Generate All From Source* and *Update Components And Nets,* Layout XL supports the generation of a PR boundary object for which the dimensions are based on the `allowedPRBoundaryDimensions` constraint defined in the technology file. If the dimensions specified in the `allowedPRBoundaryDimensions` constraint require the PR boundary area to be modified, *Generate All From Source* allows increasing or snapping up the PR boundary area to meet the constraint. This also ensures that the generated devices are accommodated after the increase in area.

If the constraint is not defined, the PR boundary is created as usual based on the parameters specified in the *PR Boundary* tab of the *Generate Layout* form.

The `allowedPRBoundaryDimensions` constraint can be applied separately for the horizontal and vertical directions to cover all the dimensions of a PR boundary. In this case, the generated PR boundary should meet both the constraints—horizontal and vertical.

Related Topics

allowedPRBoundaryDimensions

PR Boundary

Moving Generated Components inside the PR Boundary (Placing As In Schematic)

To move all the generated components in the layout and place them inside the design boundary at a position relative to their position in the schematic:

➤ From the layout window menu bar, choose *Connectivity – Generate – Place As In Schematic*.

Alternatively, click the *Place As In Schematic* button *(*  *)* in the Layout XL toolbar.

When you use the *Place As In Schematic* command:

© 2023 Cadence Design Systems, Inc.

- All the instances and pins are moved inside the design boundary, maintaining their position relative to the schematic. However, any unbound instances and pins are positioned below the design boundary.
- Physical-only power and ground pins are placed at the top and bottom center of the design boundary, respectively.
- The hierarchical instances are kept together and the command tries to fit all the instances and pins within the design boundary. If there is no design boundary, Layout XL places the devices within a square the size of the default boundary (25% utilization).
- Factored and iterated devices are kept together and positioned in one or more rows depending on the size and number of the devices.
- When placing iterated pins, the default shape for these pins is determined based on their closest boundary edge. If the pins are too long for placement along the closest boundary edge, they are wrapped. Else, the pins are placed at the exact center of the design boundary.
- If some objects in the design, such as pins, are much smaller than the other components; the small objects might not initially appear in the layout, even though they have been moved. To view these small objects, you need to zoom into the layout.
- Layout XL does not compact the layout, so the placement of components might not meet the Design Rule Checker (DRC) or other design requirements.
- Constraints entered using the Constraint Manager assistant are not honored. If a design has constraints, the command will display a message in the CIW that design has placement constraints and that Analog Auto Placer can be used to satisfy placement constraints.

Although the *Place As In Schematic* command aims at removing any component overlaps to retain the schematic topology, some overlaps might still exist, resulting in shorts. Markers for these shorts are displayed in the Annotation Browser.

To move the I/O pins so that they align to the edges of the design boundary, use the *Place – Pin Placement* command.

Related Topics

Place As In Schematic video

Pin Placement

Annotation Browser Assistant

© 2023 Cadence Design Systems, Inc.

Updating the Placement Status Automatically

To automatically update the placement status of instances and pins when they are moved into or out of the PR boundary,

➤ Type the following in the CIW.

```
envSetVal("layoutXL" "updatePlacementStatus" 'boolean t)
```

This environment variable is honored by all interactive commands and automatic tools which cause instances and pins to be moved into or outside the PR boundary. It also includes situations where the PR boundary is moved or stretched to enclose or exclude an instance or pin.

When switched on, instances and pins with status `unknown` that are wholly inside the PR boundary after the edit are updated to status `placed`. Instances and pins with status `placed` that are wholly outside the PR boundary after the edit are updated to `unknown`.

To update the placement status of instances or pins in designs that are loaded in Layout XL for the first time, or which have been edited outside the Layout XL or GXL environment, do one of the following.

- Choose the *Place – Update Placement Status* menu command.
- Use the `lxUpdatePlacementStatus` SKILL function.

Related Topics

updatePlacementStatus

lxUpdatePlacementStatus

© 2023 Cadence Design Systems, Inc.

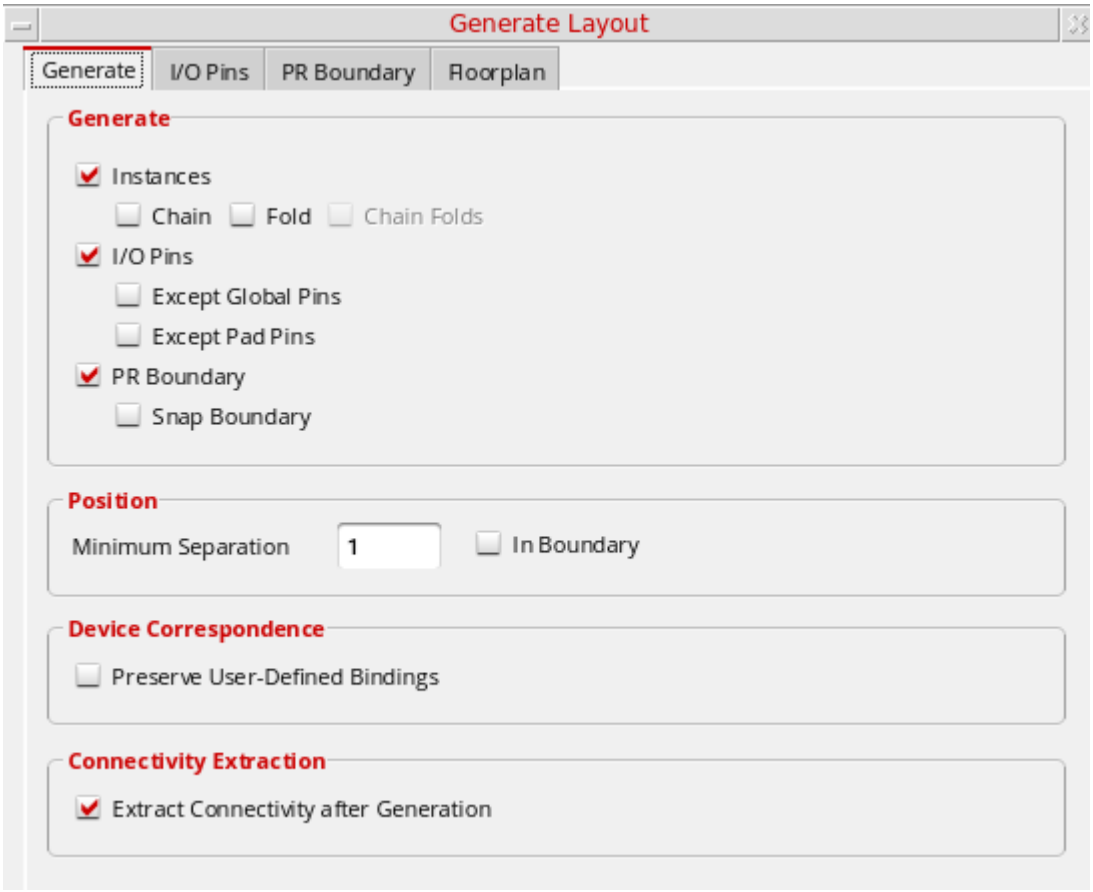
Generating All Components from Source

Use the *Generate All From Source* command to generate layout representations of the schematic design components. *Generate All From Source* deletes any existing components in the layout view and generates everything from scratch.

To generate all components from source:

1. From the layout window menu bar, choose *Connectivity – Generate – All From Source*.

   The Generate Layout form is displayed.



   The form remembers values set previously in the current Virtuoso session. When you open it for the first time, it shows the default values set in your `.cdsenv` file.

   You can also load predefined cellview information from an existing layout cellview.

2. Set the options on the form as needed. See:

© 2023 Cadence Design Systems, Inc.

   a. Specifying the Components to be Generated

   b. I/O Pin Generation

   c. Specifying the Place and Route Boundary During Layout Generation

   d. Preserving Floorplanning Objects and Snap Patterns during Layout Generation

3. Click *OK* in the Generate Layout form.

   If you already have design elements in the layout canvas, a message box warns you that this command deletes all layout instances and wiring in the design.

4. Click *Yes* to dismiss the message and generate the layout view you specified.

   Layout XL generates instances and pins equivalent to the symbols and terminals in the schematic view. Instances and pins are placed below the place and route boundary in the same relative positions as in the schematic. However, you can choose to place the instances and pins within the place and route boundary (design boundary) by setting the `lxGenerateInBoundary` environment variable.

   Iterated pins are sorted and placed together in the layout. Constraints defined in the schematic are transferred to the layout view, as are the signal types assigned to individual nets.

   To transfer the bus terminal and bus bit order (ascending or descending) information from the schematic to the layout, you need to set the `createImplicitBusTerminals` environment variable before running the *Generate All From Source* command. This allows for the bus information to be implicitly annotated on the layout side, removing the need for running VerilogAnnotate.

   Alternatively, you can choose to create bus terminals implicitly by selecting the **Create Implicit bus terminals** option on the *Generation* tab.

Any problems encountered during layout generation are reported in the CIW and in a Layout XL Info text window. Check the design against source to get more information on the problems encountered and how to resolve them.

Related Topics

Generate Layout Form

Flattening a Layout Instance

lxGenerateInBoundary

createImplicitBusTerminals

© 2023 Cadence Design Systems, Inc.

Generation

Design Check Against Source

Schematic Parameters when Generating All From Source

Generate All From Source and schematic parameters:

- *Generate All From Source* copies all the schematic parameters to the new layout instance with the following exceptions.
  - `mfactor` and `sfactor`
  - Parameters that are always ignored.

    ```
    instancesLastChanged
    instNamePrefix
    lxIgnoreParamForCAS
    lxIgnoredParams
    lxMFactorNum
    lxParamsToIgnore
    lxParamsToIgnoreForCheck
    lxPlacementStatus
    lxRounding
    lxStopList
    lxTimeStamp
    lxUseCell
    pin#
    posi
    ```

  - Parameters set using the `lxParamsToIgnore` property or on the *Parameters* tab in the Connectivity form.
- *Generate All From Source* applies all CDF callbacks before any instance is created if `lxEvalCDFCallbacks` is set to `t`. The complete list of parameters found on the schematic instance (including the adjusted folding, `sFactor`, or `lxCombination` parameters) is used to call the CDF; the resulting list is passed to the layout instance.
- *Generate All From Source* does not maintain the value of the *Folding threshold* unless all device widths and all *Width parameter* values are specified consistently; for example, all in meters (10 μm is 1-e-5) or all in user units (10 μm is 10).

Related Topics

lxParamsToIgnore Property

Connectivity Form

Generate Layout Form

© 2023 Cadence Design Systems, Inc.

lxEvalCDFCallbacks

Update Components and Nets Form

Inherited Connections when Generating All From Source

Generate All From Source and inherited connections:

- If you use inherited connections to assign more than one value to a global net, the `netSet` property, which specifies the new value for a global signal, is not copied from the schematic instance to the layout instance.
- Any `netExpression` pin properties that have not been set are copied to the layout terminals. If multiple `netExpression` properties exist for the same terminal name, then you are prompted to select one property to be preserved on the terminal.
- Layout nets inherit the `sigType` attribute from the schematic. This means the signal type set for a net in the schematic is automatically passed to the layout during generation.
- If any changes are made to the schematic signal type, the same can be updated in the layout by using the *Update Net Signal Type* option on the Update Components and Nets form. If the net signal type needs to be manually updated in the layout, you can use the Property Editor assistant. The signal type for the selected net is listed as an *Attribute* in the Property Editor assistant.
- Although it is recommended to use explicit terminals for layout generation, if required, inherited connections from the schematic can be mapped to physical pins in the layout. *Generate All From Source* can create layout terminals for implicit schematic terminals defined by inherited connections, if the layout master terminal has a defined net expression. The net expressions are resolved by the implicit inherited connections, or the net sets in the schematic.
- If a layout instance master terminal has an inherited net expression (an implicit inherited connection) for a global layout net, *Generate All From Source* creates a physical-only layout terminal with no schematic equivalent. The terminal has the `physOnly` property set to `t` so that it can be ignored by Layout XL functions, such as *Update Components And Nets* and *Check Against Source,* and by the digital applications further along the design flow.

Related Topics

lxParamsToIgnore Property

© 2023 Cadence Design Systems, Inc.

Connectivity Form

Generate Layout Form

lxEvalCDFCallbacks

Update Components and Nets Form

Viewing Ungenerated Instances In Navigator

Ungenerated instances are displayed as inactive objects at the end of the object list in the *Default* Navigator filter. The Navigator also provides an *Ungenerated* filter that you can use to view only the ungenerated instances in the design. You can customize the filter in the standard way by using the Customize Navigator Filters form.

The display of ungenerated instances in the Navigator provides you direct access to all the ungenerated instances in a design. In fact, you can quickly select a set of ungenerated instances in the Navigator and place them in the layout—the operation being similar to a *Pick From Schematic* operation.

You can also select ungenerated instances in the layout Navigator to verify the device correspondence, before the physical object and the binding is created. If an ungenerated schematic instance reflects correspondence to a target layout instance that does not exist, a warning icon gets displayed for that instance on top of the ungenerated icon in the Navigator. In addition, a tooltip displays that mentions that the target layout object does not exist.

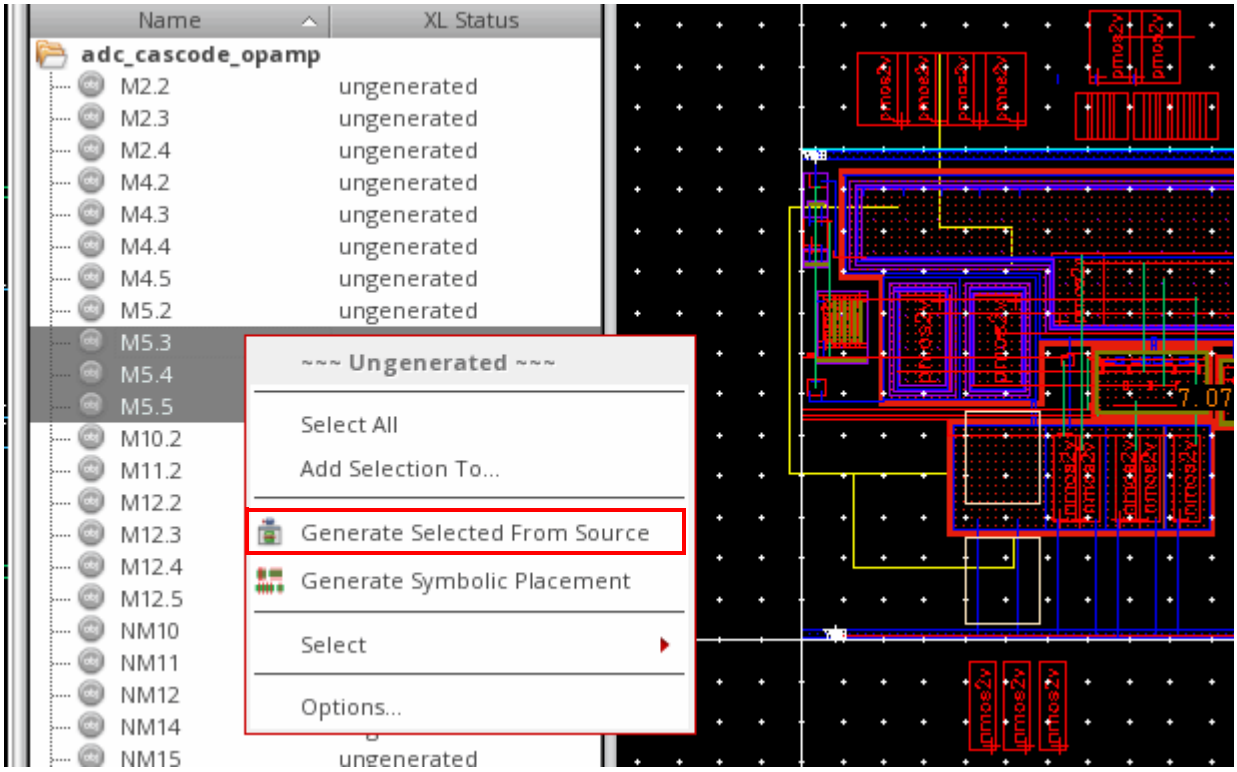
If you select an ungenerated instance in the layout Navigator, the corresponding schematic instance is selected in the schematic Navigator and in the schematic canvas. If you select an mfactored, ungenerated instance in the schematic Navigator or in the canvas, all the corresponding layout instances are selected in the layout Navigator.

To place an ungenerated instance in the layout:

1. Right-click an ungenerated instance in the Navigator and choose *Generate Selected From Source*.

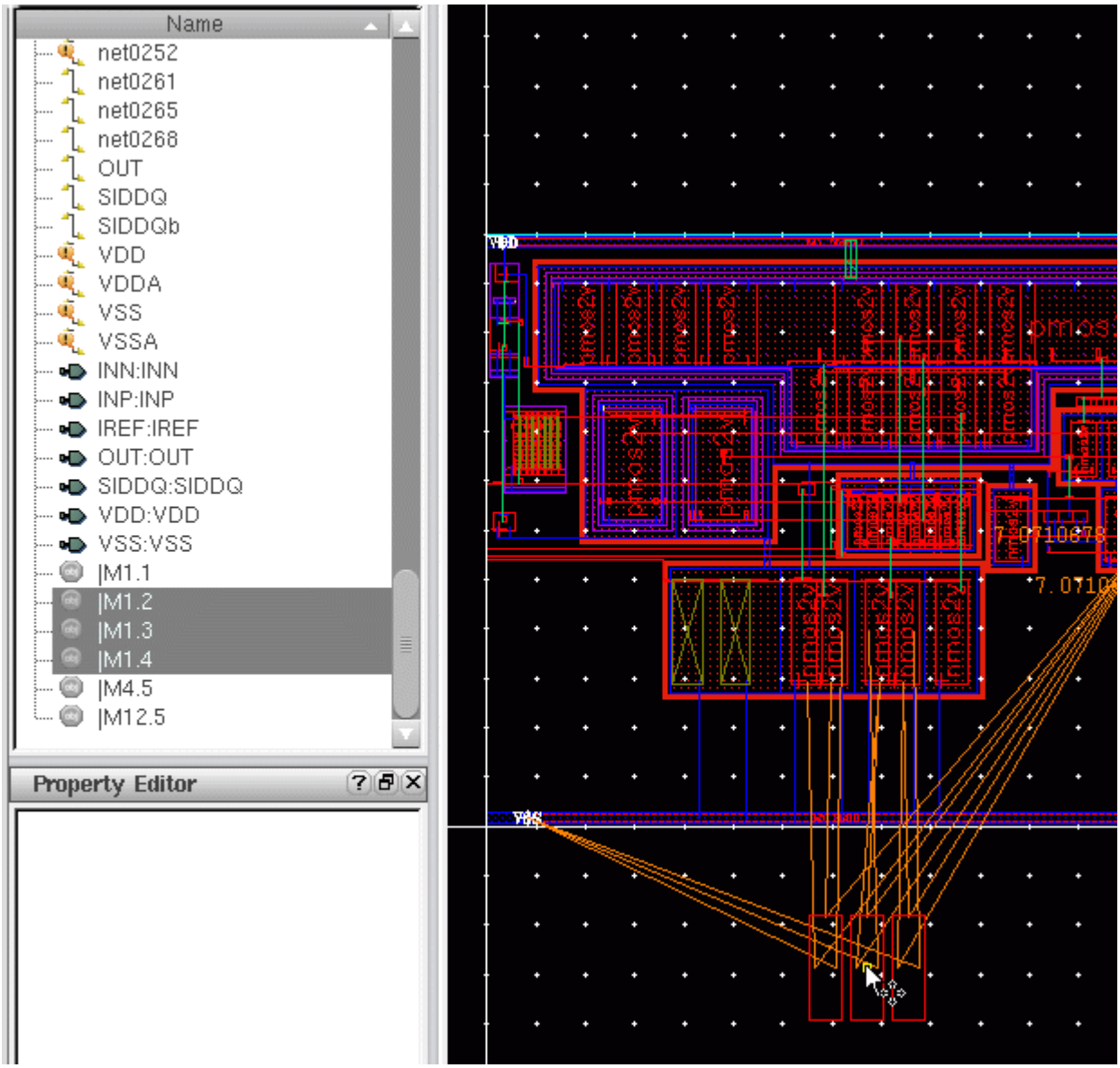
© 2023 Cadence Design Systems, Inc.

You can also select multiple ungenerated instances from the Navigator and choose *Generate Selected From Source* to place all the instances on the layout canvas.



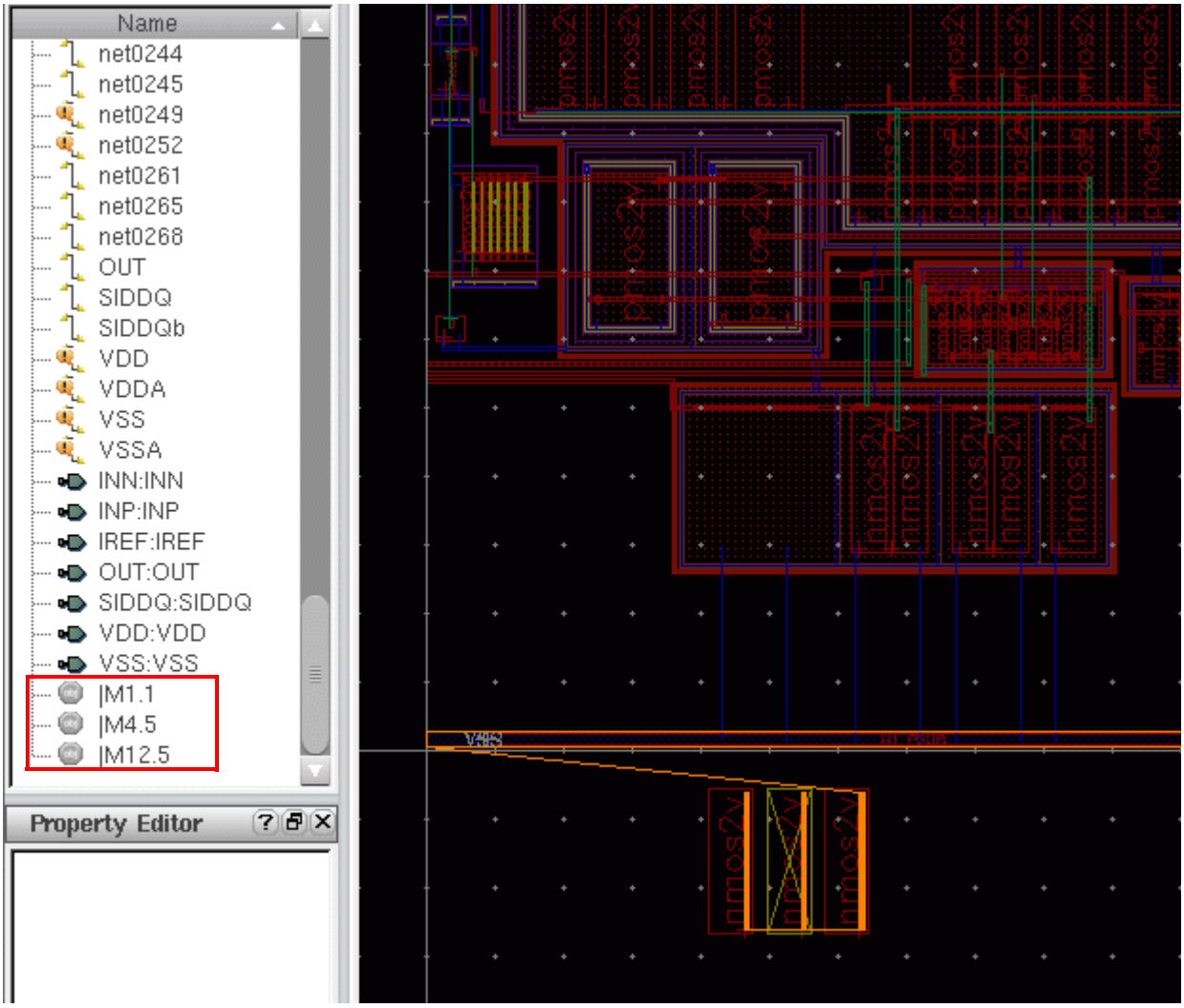
© 2023 Cadence Design Systems, Inc.

2. Select a location on the layout canvas where you want to place the selected instances.



The selected instances are placed in the layout, as displayed in the figure below, and the list of ungenerated instances in the Navigator is instantly updated to display only those instances that are still ungenerated.

© 2023 Cadence Design Systems, Inc.



Related Topics

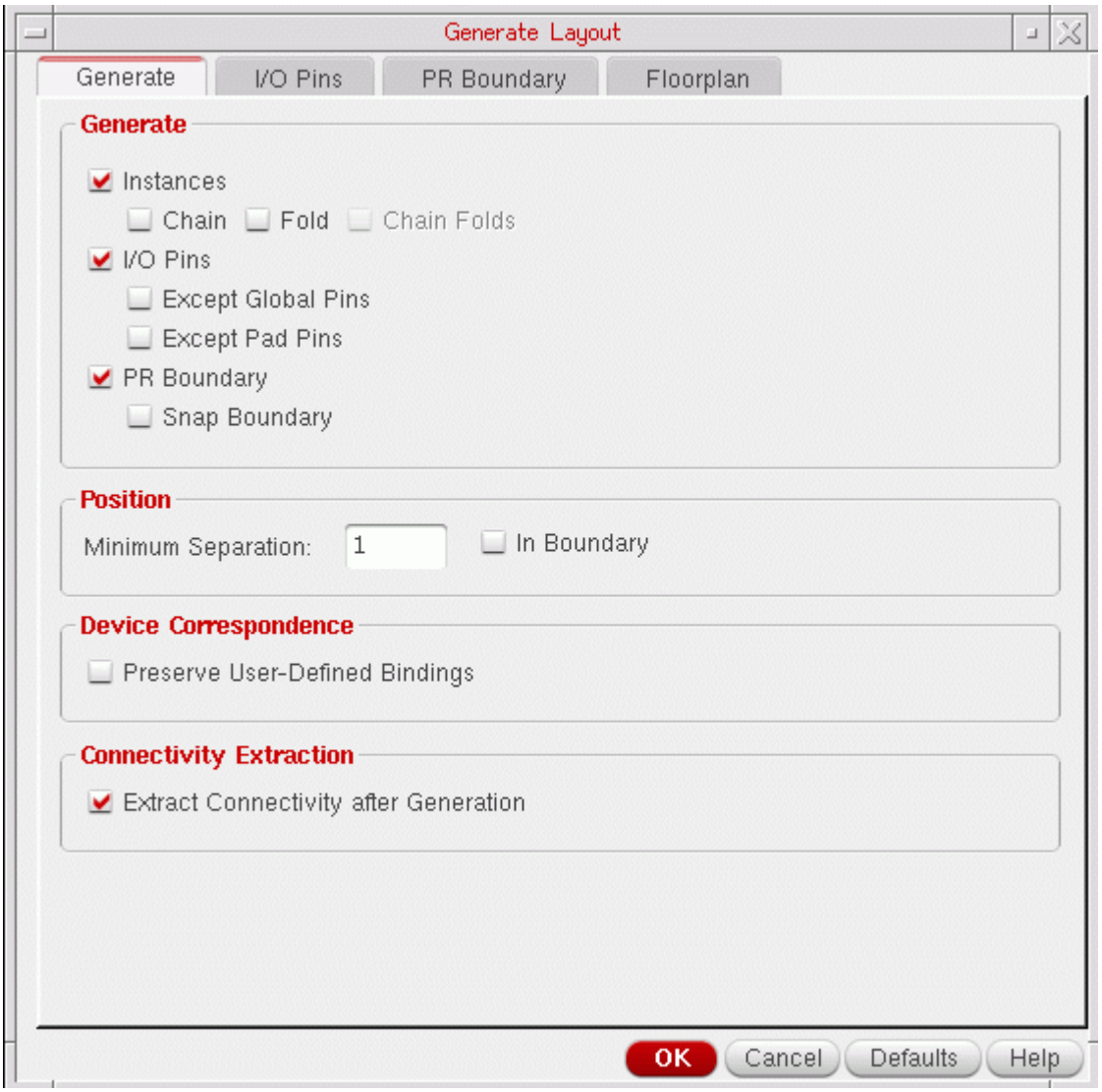
Navigator

© 2023 Cadence Design Systems, Inc.

Specifying the Components to be Generated

To specify the components to be generated:

1. Select the *Generate* tab of the Generate Layout form.



2. In the *Generate* group box, select the *Instances*, *I/O Pins*, and *PR Boundary* options, as required.

   a. To chain transistors, select the *Chain* check box.

   b. To divide transistors in the design into two or more individual folds, select the *Fold* check box.

© 2023 Cadence Design Systems, Inc.

If the `disableFolding` environment variable is set and the folding threshold for the component type is set to `0`, folding is disabled.

c. To chain the individual folds of a transistor, select the *Chain Folds* check box.

- For the *Chain Folds* option to be available, only the *Fold* check box must be selected. If you select the *Chain* check box also, *Chain Folds* is deactivated.
- Any devices to be chained or folded must belong to a component type that has *Component class* set to *PMOS* or *NMOS* and values set for the *Active layer*, *Width parameter*, and *Folding threshold* parameters. To change any of these parameters, update the appropriate value in the Component Types Mode in the CPH Window. Alternatively, update the LAM File Component Type Section.

  If the devices to be chained or folded have their component class set to *NFIN* or *PFIN*, these devices must have their *Number of fins* parameter or the *Width* parameter specified to enable folding.

d. To stop Layout XL from generating layout pins for global nets in the schematic, select *Except Global Pins*.

e. To stop Layout XL from generating layout pins for schematic pins that are connected to I/O pads, select *Except Pad Pins*.

f. To generate a snap boundary, select *Snap Boundary*.

You can generate a snap boundary only if the *PR Boundary* option is switched on.

3. To preserve any user-defined binding of devices between the schematic and the layout, select *Preserve User-Defined Bindings*.

This option preserves only user-defined one-to-one, many-to-many, many-to-one, and one-to-many device correspondence defined in the Define Device Correspondence form. It does not report missing devices or shapes within a bound group.

4. To run connectivity extraction after the layout has been generated, select *Extract Connectivity After Generation*.

5. To create a label specifying the library, cell, and view of your design just inside the center top edge of the boundary, set the `createBoundaryLabel` environment variable.

Related Topics

Generate

disableFolding

© 2023 Cadence Design Systems, Inc.

Configure Physical Hierarchy Window

Component Types Mode in the CPH Window

Hierarchy Configuration Mode in the CPH Window

Soft Block Mode in the CPH Window

Define Device Correspondence Form

Device Correspondence Updates

Device Correspondence Updates

I/O Pin Generation

You specify the pins to be generated on the I/O Pins tab of the Generate Layout form.

For each pin listed, the form shows the parameters that will be used to generate its equivalent in the layout. You can remove or change the specification of any of the listed pins, or add new pins to be generated.

When generating pins, Layout XL uses the same naming convention as the Virtuoso Schematic Editor, allowing you to assign different names to terminals and nets. Where terminal and net names are different in the schematic, Layout XL creates pins with the same terminal name as in the layout, even though the net name associated with the pin might be different.

By default, this section lists all the top-level pins detected in the schematic design, including schematic pins that are connected to I/O pads (cells of type `pad`, `padSpacer`, or `padAreaIO`). To generate only pads but no pins, turn on the *Except Pad Pins* option on the Generate tab on the Generate Layout form (or on the Update tab if you are using the Update Components and Nets form).

Power and ground pins defined at a lower level of the design hierarchy are not listed on the *I/O Pins* tab but are nevertheless generated in the layout view. The system issues an information message to tell you what has happened.

The system generates a pin for a stub net (a non-global `netSet` automatically created by the schematic extractor) only if the stub net is connected to an instance terminal. If it is unconnected, no pin is generated.

Layout XL also correctly propagates connectivity to nets that are connected via patchcords.

© 2023 Cadence Design Systems, Inc.

Related Topics

Specifying Default Values for All Pins During Layout Generation

Specifying Values for Selected Pins During Layout Generation

Adding a New Pin During Layout Generation

Creating Pin Labels During Layout Generation

Ignoring Mismatched Pins

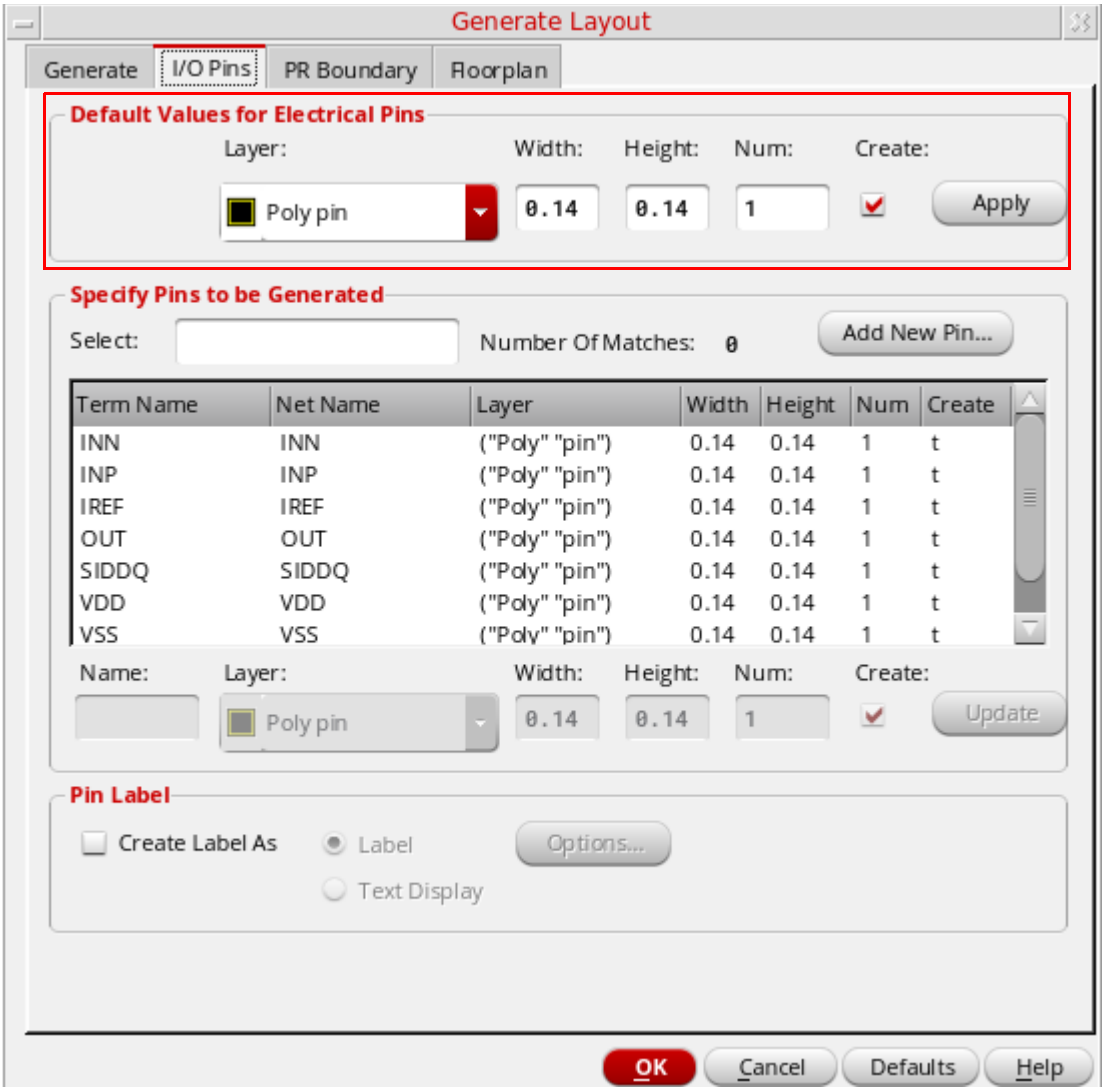
Generate Layout Form

© 2023 Cadence Design Systems, Inc.

Specifying Default Values for All Pins During Layout Generation

To specify default values for all pins:

1. Bring up the I/O Pins tab of the Generate Layout form.



Specify default values for all pins and click *Apply*.

2. Choose the default routing layer from the *Layer* cyclic field.

   The list of pin layers is obtained from the technology information applied to the design.

3. Specify the default *Height* and *Width* of the pins and the *Number* of pins you want to create.

4. Check the *Create* box to specify that the pins are to be created in the layout view.

© 2023 Cadence Design Systems, Inc.

5. Click *Apply*.

   The specified values are applied to all the pins displayed in the list box and the corresponding environment variables (`initIOPinLayer`, `initPinHeight`, `initPinWidth`, and `initPinMultiplicity`) are set appropriately.

   `initPinHeight` and `initPinWidth` are set only if the value you specify is greater than the default minimum width set for the layer in the technology file.

Related Topics

Generate Layout Form

initIOPinLayer

initPinHeight

initPinWidth

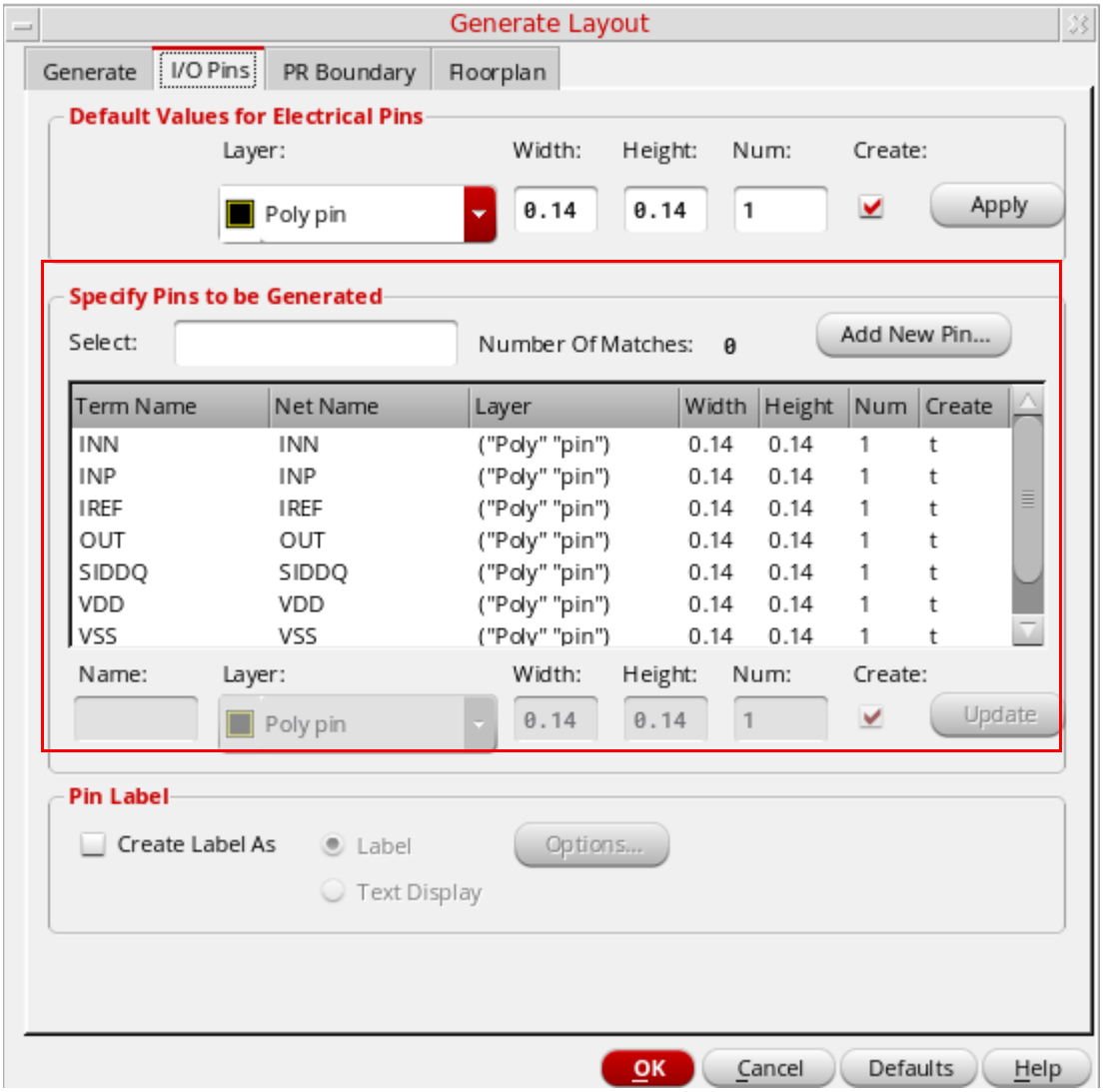
initPinMultiplicity

© 2023 Cadence Design Systems, Inc.

Specifying Values for Selected Pins During Layout Generation

To update the values of selected pins in the list:

1. Bring up the I/O Pins tab of the Generate Layout form.



Select the pins to update

Specify the new values and click *Update*.

2. In the list box, select the pins you want to update.
3. Choose the routing layer for the selected pins from the *Layer* cyclic field.

   The Layer Purpose Pair (LPP) list on the Generate Layout form is filtered based on the values specified for the `initIOPinLayerNames` and `initIOPinPurposeNames` environment variables.

© 2023 Cadence Design Systems, Inc.

4. Specify the default *Height* and *Width* of the selected pins and the *Number* of each pin you want to create.
5. Check the *Create* box to specify that the selected pins are to be created in the layout view; clear it to prevent a pin from being generated in the layout.
6. Click *Update*.

   The selected pins are updated with the new values.

Related Topics

Generate Layout Form

initIOPinLayerNames

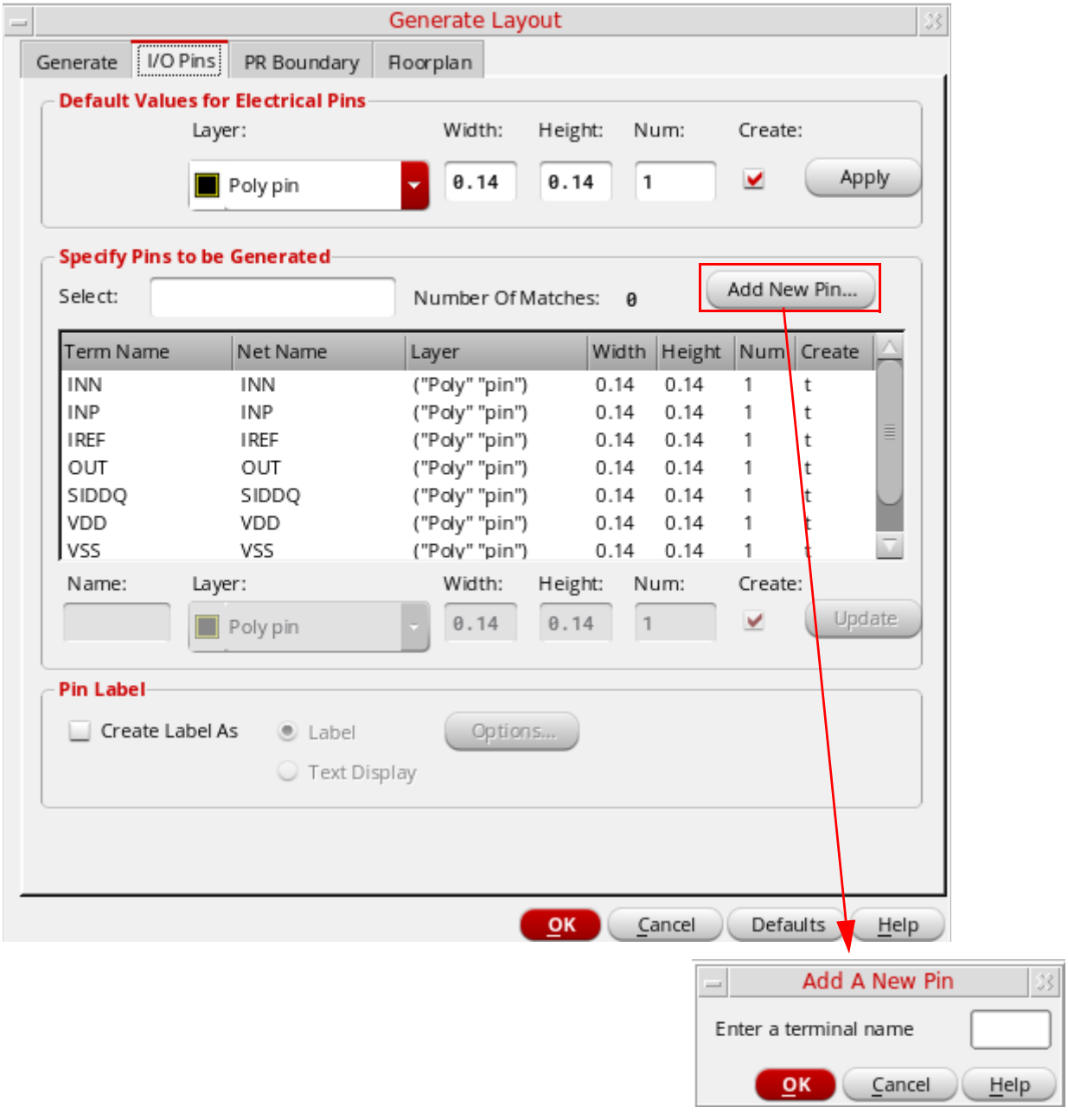
initIOPinPurposeNames

© 2023 Cadence Design Systems, Inc.

Adding a New Pin During Layout Generation

To add a new pin:

1. Bring up the I/O Pinstab of the Generate Layout form, click *Add New Pin*.



2. Type in a pin name and click *OK*.

© 2023 Cadence Design Systems, Inc.

If the pin to be added has the same attributes as any of the existing pins in the pin list (pin name, pin layer, and/or purpose), then the added pin is highlighted. You can change the number of the highlighted pin in order to create the number of pins you require.

Related Topics

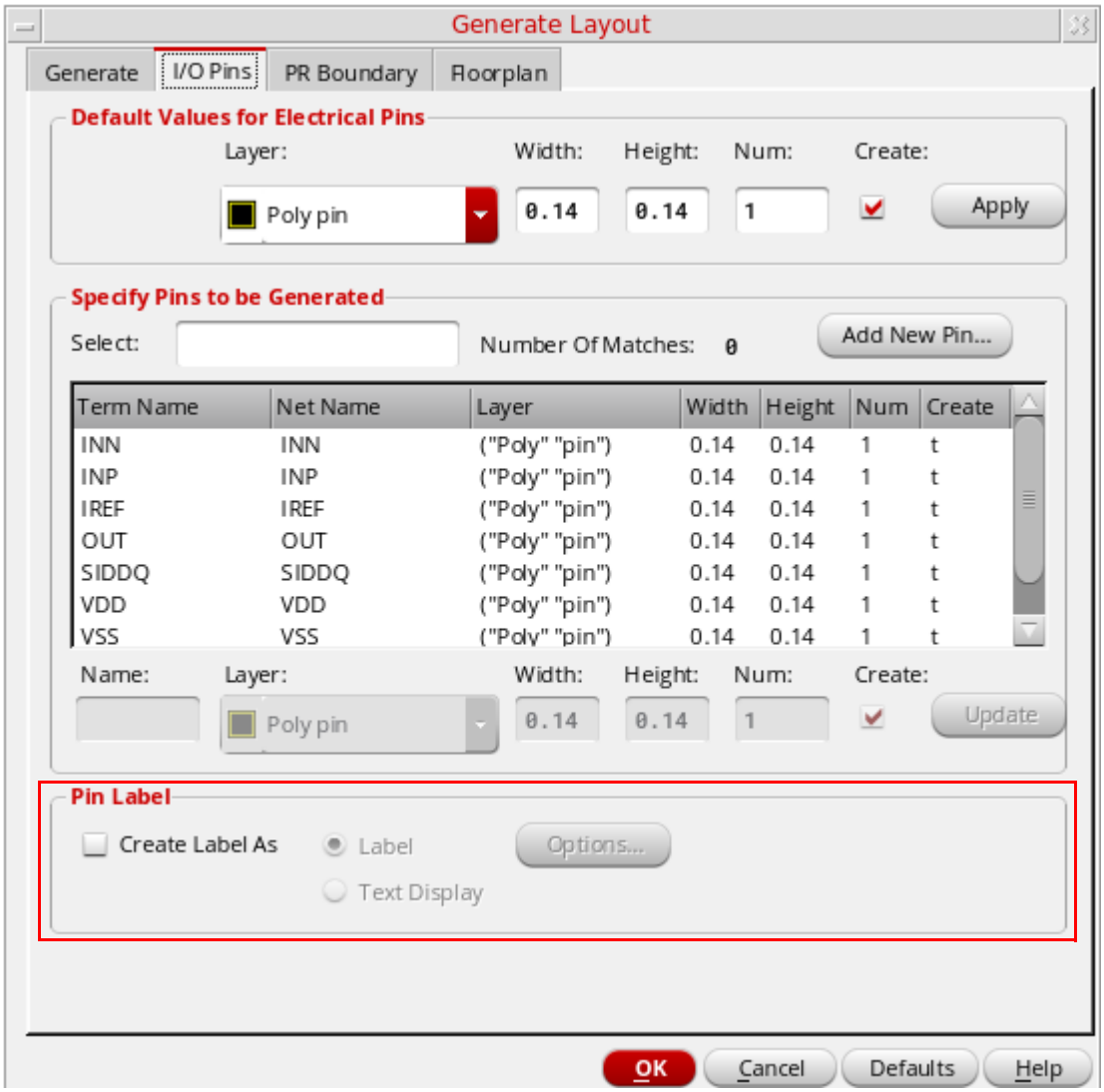
Generate Layout Form

© 2023 Cadence Design Systems, Inc.

Creating Pin Labels During Layout Generation

To generate name labels for each pin:

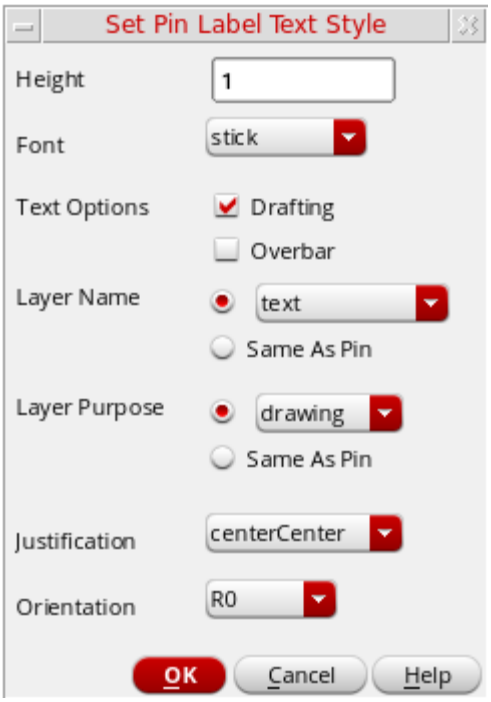
1. Bring up the I/O Pins tab of the Generate Layout form.



2. Turn on the *Create Label As* option and choose whether you want to create a *Label* object or a *Text Display* object.
3. If you choose to create *Label* objects, you can set the label style by clicking *Options*.

© 2023 Cadence Design Systems, Inc.

The Set Pin Label Text Style form is displayed.



4. Choose the options you want and click *OK*.

   The corresponding environment variables are listed below.

   labelDrafting
   labelFontStyle
   labelHeight
   labelJustify
   labelOverbar
   orientation
   pinTextLayer
   pinTextPurpose
   pinTextSameLayer
   pinTextSamePurpose

The default *Layer Name* is `text`. But, if the layer name does not exist as a `validLayer` in the technology file, the first layer in the list of `validLayers` is set as the layer to be used for creating the pin label. The layer purpose then defaults to the purpose associated with the first layer in the `validLayers` list.

© 2023 Cadence Design Systems, Inc.

Related Topics

Generate Layout Form

Set Pin Label Text Style Form

Ignoring Mismatched Pins

If a symbol in a schematic has more pins than the corresponding device on the layout, Layout XL generates a warning about mismatched pins. To ignore mismatched pins,

1. In the schematic window menu bar, choose *Edit – Hierarchy – Descend Edit* to descend into the symbol of the instance.
2. In the schematic window, choose *Edit – Properties – Cellview*.

   The Edit Cellview Properties form is displayed.
3. Click *Add* to display the Add Property form and add a new Boolean property on the pin, for example, `unusedPin`, and set it to `t`.
4. Add the property name (`unusedPin`) as a value in the `propsUsedToIgnoreObjs` environment variable in your `.cdsenv` file.

   ```
   layoutXL propsUsedToIgnoreObjs string "unusedPin"
   ```

   Or type the following in the CIW.

   ```
   envSetVal("layoutXL" "propsUsedToIgnoreObjs" 'string "unusedPin")
   ```

The next time you generate or update a layout or run the *Check Against Source* command, the software does not issue warnings about mismatched or missing pins.

Related Topics

propsUsedToIgnoreObjs

Generation of Optical Pins and Chains (Virtuoso Photonics Option)

If you are using the Virtuoso Photonics Solution and have the `Virtuoso_Photonics_Option` license checked out, the *Generate All From Source* command enables you to create optical pins for the design.

If a pin is associated with an optical net, Layout XL automatically generates optical pins on an optical layer, such as the `waveguide` layer. By default, the layer to be used for the optical pin generation is determined from the value set for the `phoPinLayer` environment variable.

© 2023 Cadence Design Systems, Inc.

If the environment variable does not have a valid layer set, the first photonic layer in the `validLayer` list is used.

Each optical pin can have the following attributes defined:

- **Width**: Width of the waveguide associated with the photonic port.
- **Radius**: Curvature of the waveguide associated with the photonic port.

  By default, the radius is not defined.
- **Angle**: Access direction of the photonic port.

Related Topics

phoPinLayer (Virtuoso Photonics Option)

Photonics in Virtuoso Schematic Editor, ADE, and Layout XL (Virtuoso Photonics Option)

Generation of Optical Pins

Generating Optical Chains

Generation of RF TLine Chains (Virtuoso MultiTech Framework)

If you are using the Virtuoso RF Solution and have the `Virtuoso RF Option` license checked out, you can use the following Layout XL commands to chain TLine instances:

- *Generate All From Source*
- *Update Components And Nets*
- *Generate Selected From Source*
- *Generate Chained Devices*

Layout XL enables you to create two-dimensional TLine chains, unlike the usual MOS chains that are one-dimensional. The TLine chains are generated using the same orientation as in the schematic. The abutment of TLine pinFigs is supported with the `center` accessDir.

You can chain the TLine instances using the Generate Chained Devices form. However, top-level TLine pins cannot be abutted.

The following interactive chaining command options are not supported for TLine devices:

- *Preserve Existing Chains*

© 2023 Cadence Design Systems, Inc.

- *Use Device Order*
- *Interdigitate Chains*
- *Mirror*
- *Permute Pins*
- *Align PMOS*
- *Align NMOS*

Related Topics

lxChain

Generate Chained Devices Form

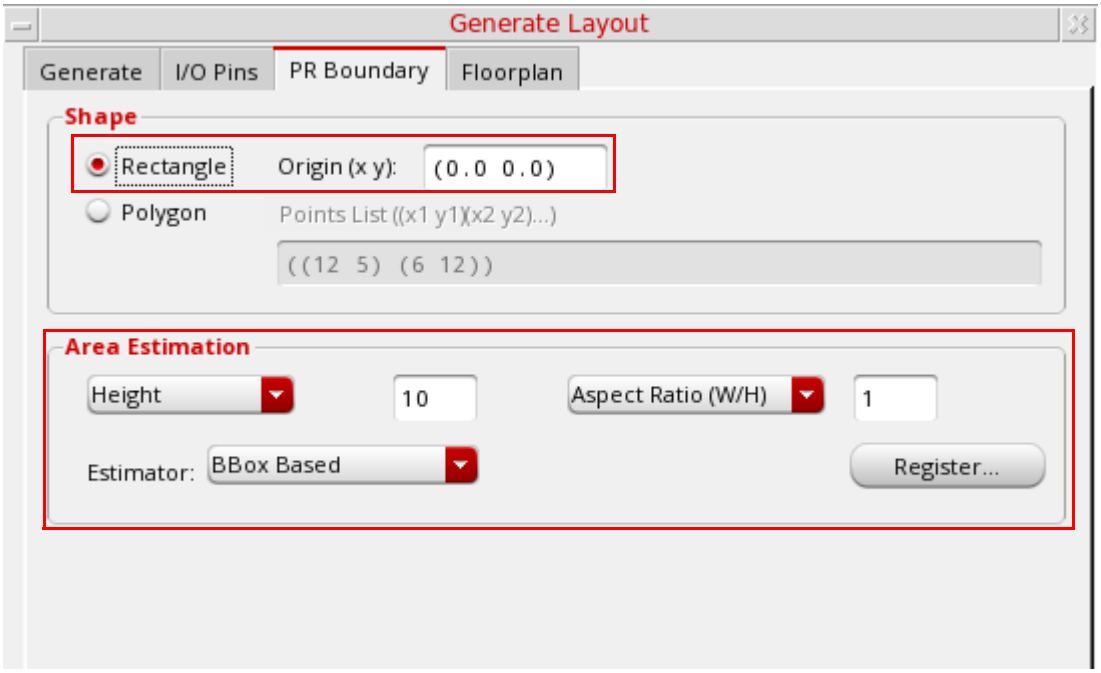
© 2023 Cadence Design Systems, Inc.

Specifying the Place and Route Boundary During Layout Generation

You specify the place and route boundary on the *PR Boundary* tab of the Generate Layout form.

To specify a rectangular place and route boundary:

1. On the *PR Boundary*, choose *Rectangle*.

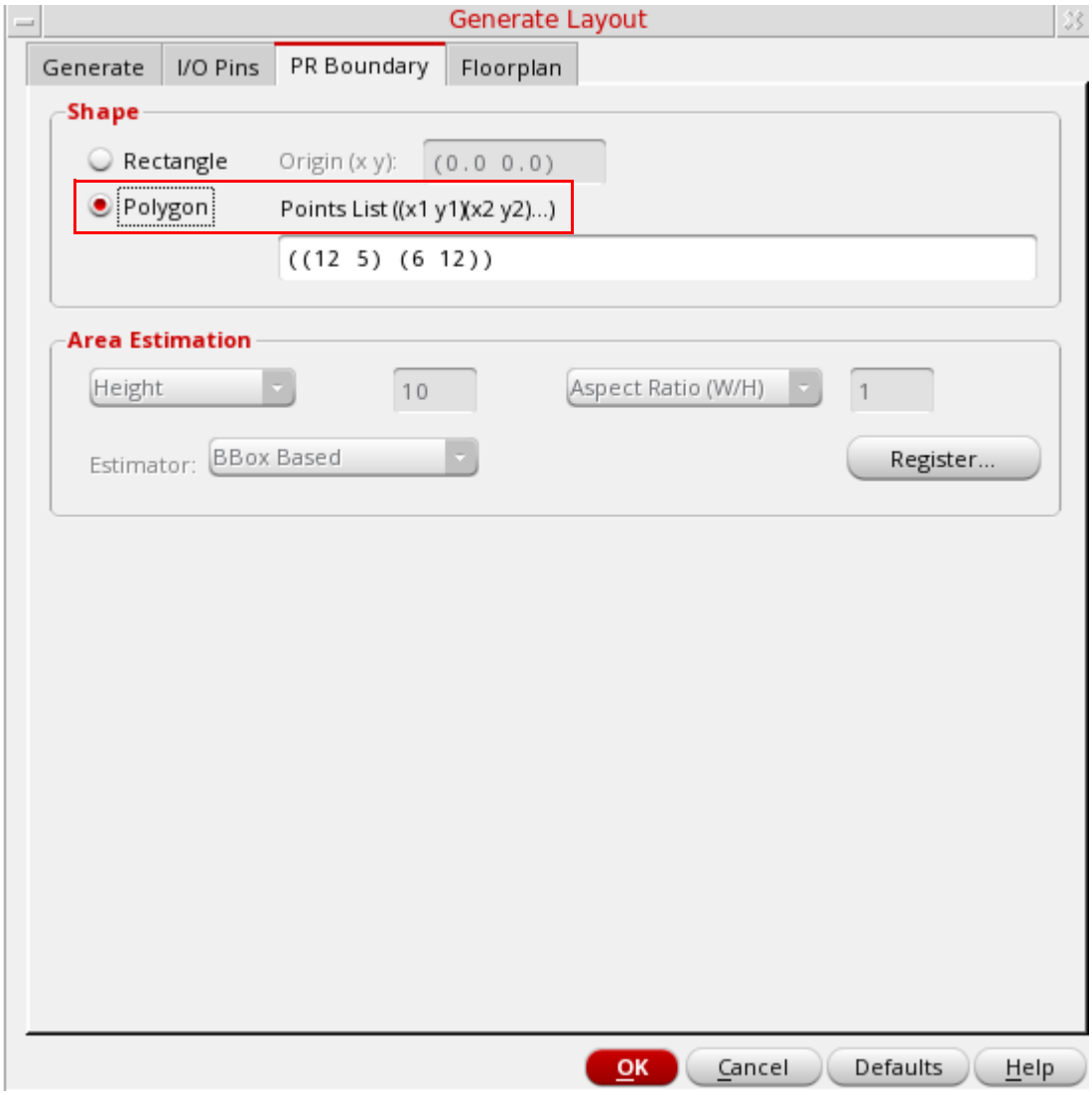


2. Specify the coordinates for the *Origin* of the boundary shape.
3. Use the controls in the *Area Estimation* group box to calculate the dimensions of the boundary.

© 2023 Cadence Design Systems, Inc.

To specify a polygonal place and route boundary:

1. On the *PR Boundary* tab of the Generate Layout form, choose *Polygon*.



2. Type the X and Y coordinates of each vertex of the polygon in the *Points List* field. For example, for a rectangle, type the following.

```
(0 0) (0 10) (10 8) (8 0) (0 0)
```

When you move the cursor into the canvas, the X and Y coordinates of the location of the cursor are displayed in the status banner at the bottom of the layout canvas.

© 2023 Cadence Design Systems, Inc.

Related Topics

PR Boundary

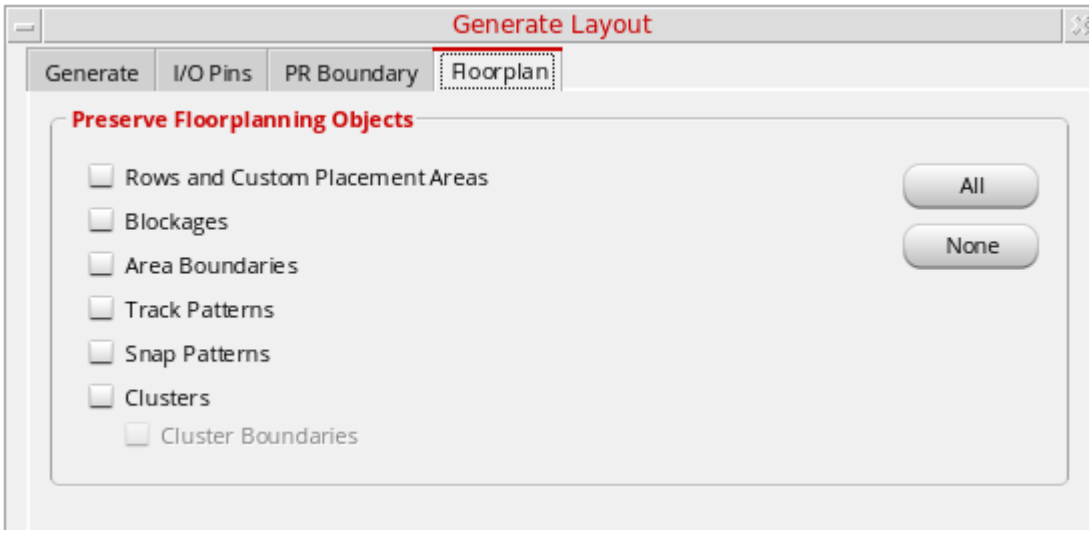
Generate Layout Form

Preserving Floorplanning Objects and Snap Patterns during Layout Generation

Use the controls on the Floorplan tab on the Generate Layout form to preserve existing floorplanning objects during layout generation.

➤ On the *Floorplan* tab, check the objects you want to preserve.

Click *All* to select all the items in the list; *None* to deselect them all.



You can also preserve any *Snap Patterns* and *Width Space Pattern (WSP)* regions already defined in the layout view. The following components and definitions related to a WSP data model are also preserved in the cellview:

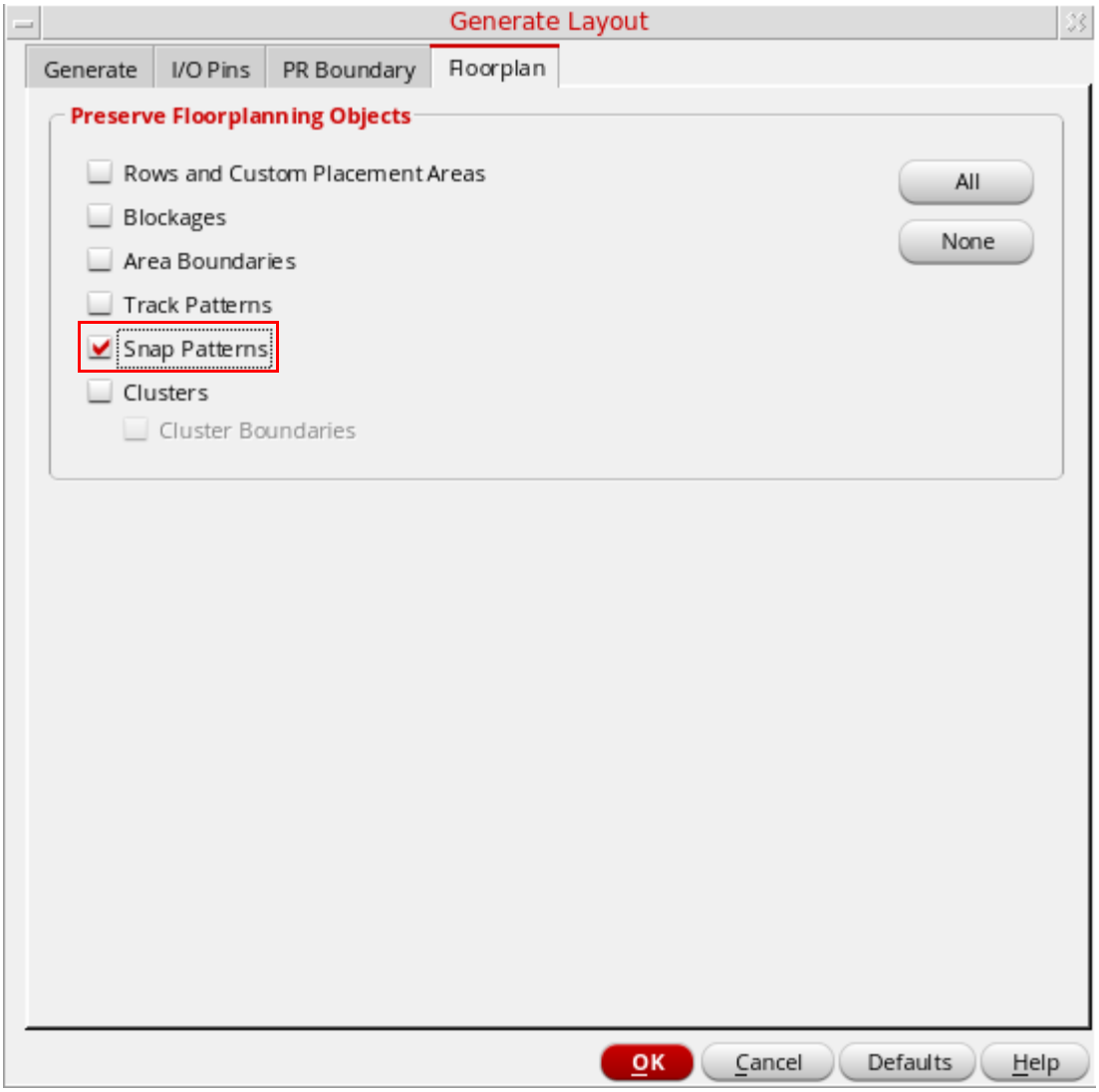
- `widthSpacingPatterns`
- `widthSpacingSnapPatternDefs`
- `widthSpacingPatternGroups`
- `relatedSnapPatternDefs`

© 2023 Cadence Design Systems, Inc.

- constraints for the global grid setup for `snapPatternDefs` and `widthSpacingSnapPatternDefs`
- the global grid offset reference for `snapPatternDefs` and `widthSpacingSnapPatternDefs`

To preserve the predefined patterns:

➡ Select *Snap Patterns* in the *Floorplan* tab of the Generate Layout form.



Alternatively, you can choose to preserve the predefined patterns by using the `preserveSnapPatterns` environment variable.

© 2023 Cadence Design Systems, Inc.

Related Topic

Working with the Binder to stay XL-compliant (RAK)

preserveSnapPatterns

Generate Layout Form

Layout Generation from Source Using SKILL

You can run *Generate All From Source* using one of the following methods:

- Using lxGenFromSource SKILL function.
- Using a combination of the following functions that let you specify what is generated directly using SKILL:

  lxGenerateStart

  lxGetAvailablePinLPPs

  lxGetPinNets

  lxSetBoundaryOptions

  lxSetAreaEstimationOptions

  lxSetGenerateOptions

  lxSetNetPinSpecs

  lxSetPreserveFloorplanningOptions

  lxGenerateFinish

Related Topics

Connectivity Driven Editing Functions

Placement Modes for Generating Selected Components from Source

Use the *Generate Selected From Source* command to incrementally generate schematic instances and pins in the layout view. You can select the components to be generated either

© 2023 Cadence Design Systems, Inc.

before or after you launch the command. You can select the components to be generated either from the schematic canvas or from the ungenerated instances list in the Navigator assistant.

The instances generated using the *Generate Selected From Source* command are automatically bound to their schematic counterparts and have the correct connectivity.

You can generate the components in the layout view using one of the following generation modes:

- *As In Schematic* – Use this mode to generate instances and pins either as a group or as individual components.

  By default, the selected instances are placed as in the schematic. But, you can select a combination of instances and pins and change the alignment, spacing, and orientation of the selected components before you place them in the layout. You can also enable automatic folding for instances.

  See Generating Components As In Schematic.

- *Array* – Use this mode to generate an array of ungrouped instances, a modgen, a mosaic, or a group array.

  Array placement mode cannot be used to place pins.

  See Generating an Array.

- *Individually* – Use this mode to generate instances and pins one at a time.

  When generating components individually, you can change the values of any properties and attributes associated with each object before you generate it. You can also enable automatic chaining and folding for instances.

  See Generating Individual Instances and Generating Individual Pins.

Generate Selected From Source command:

- Honors any constraints entered using the Virtuoso Constraint Manager. It also applies all callbacks before any instance is created. The complete list of parameters found on the connectivity source instance (including the adjusted folding, sfactor Property, or lxCombination Property parameters) are used to call the CDF; the resulting list is passed to the layout instance.
- When it is active, clicking on a component in either the design canvas or the Navigator assistant only highlights the object, it does not select it. Consequently, any information displayed in the Property Editor assistant is not updated.

© 2023 Cadence Design Systems, Inc.

- If a layout instance master terminal has an inherited net expression (an implicit inherited connection) for a global layout net, *Generate Selected From Source* creates a physical-only layout terminal with no schematic equivalent. This terminal has the `physOnly` property set to `t` so that it can be correctly ignored by Layout XL functions such as *Update Components And Nets* and *Check Against Source* and by the digital applications further on in the design flow.

Related Topics

Viewing Ungenerated Instances In Navigator

Generating Components As In Schematic

Generating an Array

Generating Individual Instances

Generating Individual Pins

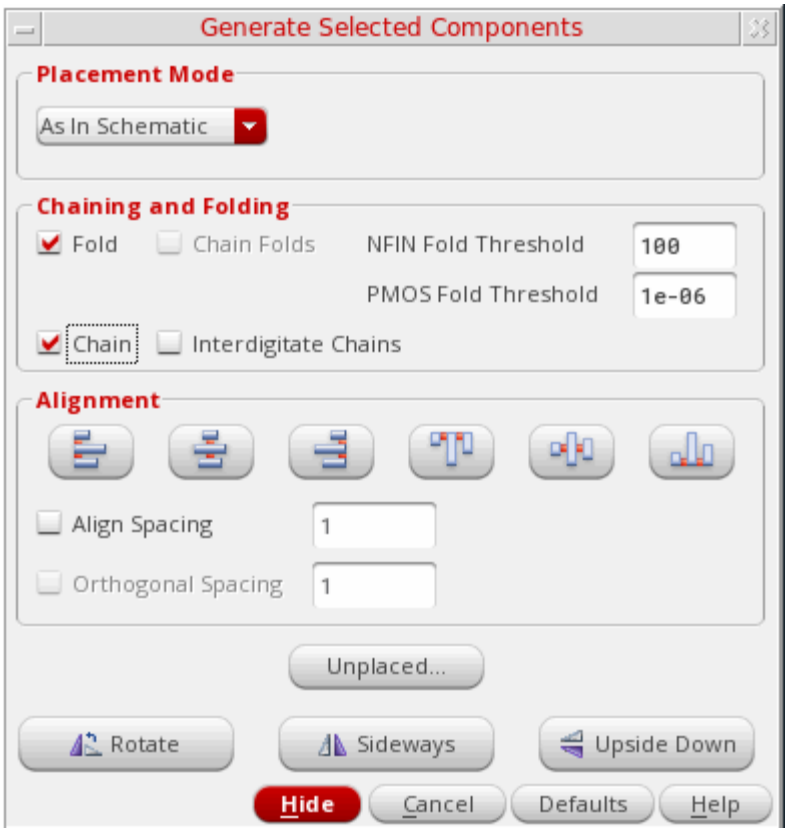
Listing Unplaced Components after Layout Generation

© 2023 Cadence Design Systems, Inc.

Generating Components As In Schematic

To generate schematic instances and pins in a single operation so that the generated components are placed almost as in the schematic view:

1. From the layout window menu bar, choose *Connectivity – Generate – Selected From Source*.
2. Press `F3` to display the Generate Selected Components form.



As In Schematic mode is enabled as the default *Placement Mode.* However, if you change the placement mode, the form opens with the last selected placement mode when opened the next time.

In the *As In Schematic* mode, you cannot change the properties and attributes associated with the selected components. However, you can set pseudoparallel nets. See Connectivity Extraction in Layout XL.

3. In the schematic window, select the instances and pins you want to generate using one of the following methods.

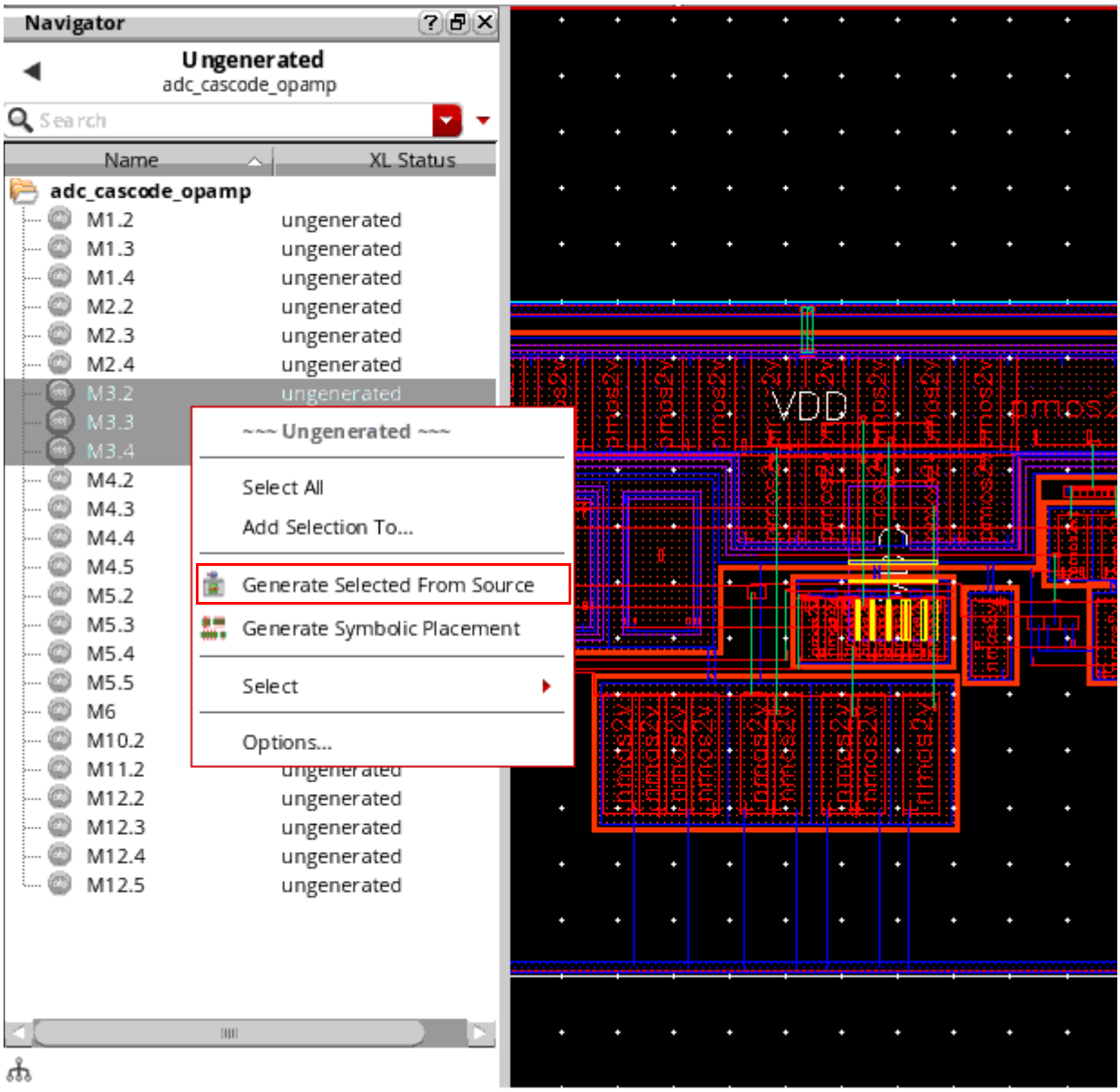
© 2023 Cadence Design Systems, Inc.

- Click the components in the schematic canvas or Navigator assistant. To select multiple components, keep the `Shift` key pressed during selection. Alternatively, drag to enclose an area that has only the components that you want to select. To deselect a component, click the component while the `Ctrl` key is pressed.
- Click *Unplaced* to view the Instance/Pin List form, select the components you want, and click *Apply* to select them in the schematic window. See Listing Unplaced Components after Layout Generation.
- Right-click the ungenerated instances in the layout Navigator assistant and choose *Generate Selected From Source*, as shown in the figure below.

The selected instances remain highlighted until they are either deselected or generated in the layout view. You can select an instance in the schematic if it is not already bound to an instance in the layout. If an existing layout instance with a matching name is not bound to the selected schematic instance, the name of the instance generated in the layout is determined based on the `renameConflictingLayoutInstances` environment variable.

© 2023 Cadence Design Systems, Inc.

Press `F3` if the Generate Selected Components form is not displayed automatically.

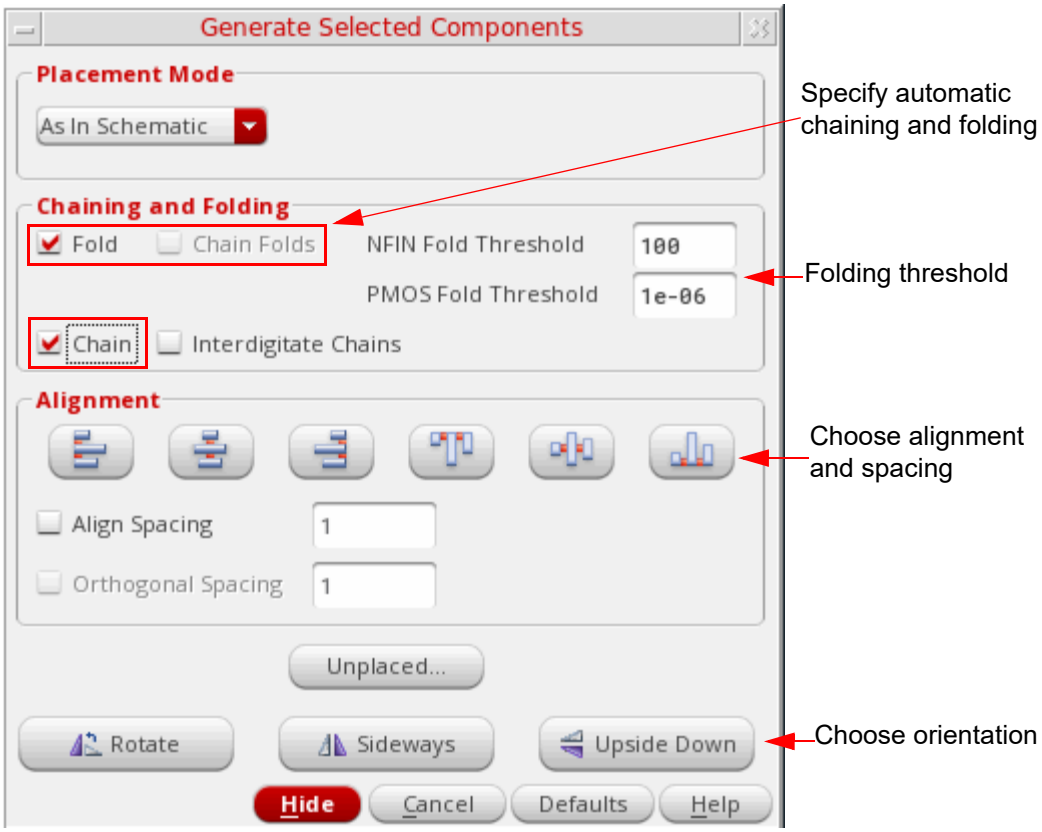


4. Move your pointer into the layout canvas.

   The outlines of the selected components follow the pointer in the canvas.

© 2023 Cadence Design Systems, Inc.

5. Set the options you want for the selected components.



- (*Optional*) Select *Chain* and *Fold* to automatically chain and fold the instances you are generating. To chain the individual folds of a transistor, select the *Chain Folds* check box.
  - If the `disableFolding` environment variable is set and the folding threshold for the component type is set to `0`, folding is disabled.
  - The *Chain Folds* option is active when *Fold* is selected and *Chain* is deselected.
- (*Optional*) Select *Interdigitate Chains to* automatically identify the nodes that qualify as pseudoparallel connections and define them during chaining and abutment.
  - The *Interdigitate Chains* option is active when *Chain* is selected.
  - A pseudoparallel net connects nodes that are always at the same voltage, so current does not pass through the net.
- (*Optional*) Use the *Alignment* options to automatically align and space the components as they are placed. For example, set *Align Spacing* to 2 to

© 2023 Cadence Design Systems, Inc.

automatically space the selected instances by 2 user units when they are generated and then click *Align Bottom* to align the bottom edges.

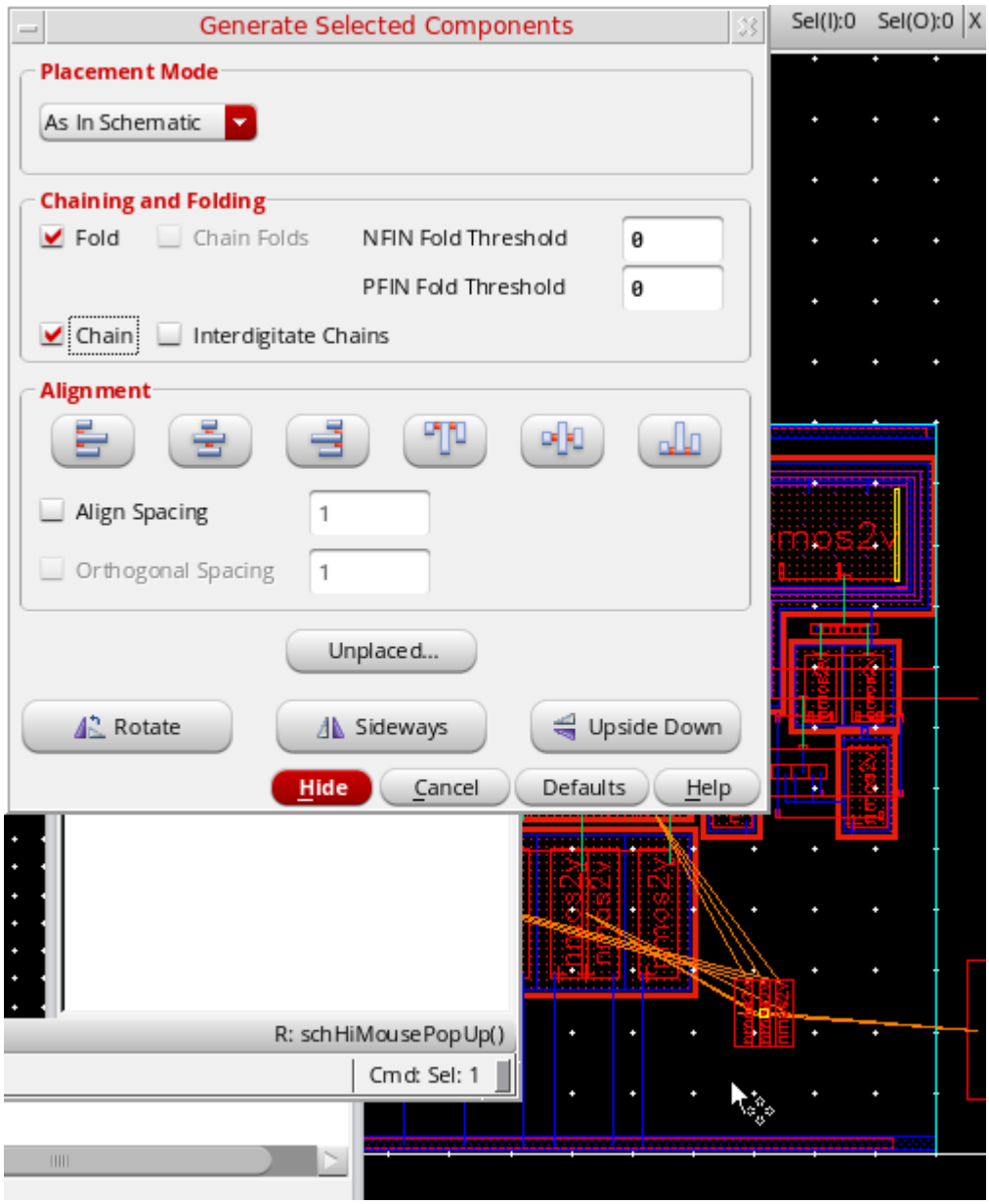
- Choose the orientation you need using the *Rotate*, *Sideways*, and *Upside Down* buttons.

6. Click in the layout canvas to place the selected components at the required location.

Iterated pins placed in the schematic are placed in order (i.e. `Q1`, `Q2`, `Q3`) in the layout.

© 2023 Cadence Design Systems, Inc.

The selected components are generated in the layout view replicating their schematic placement, as shown in the figure below.



Related Topics

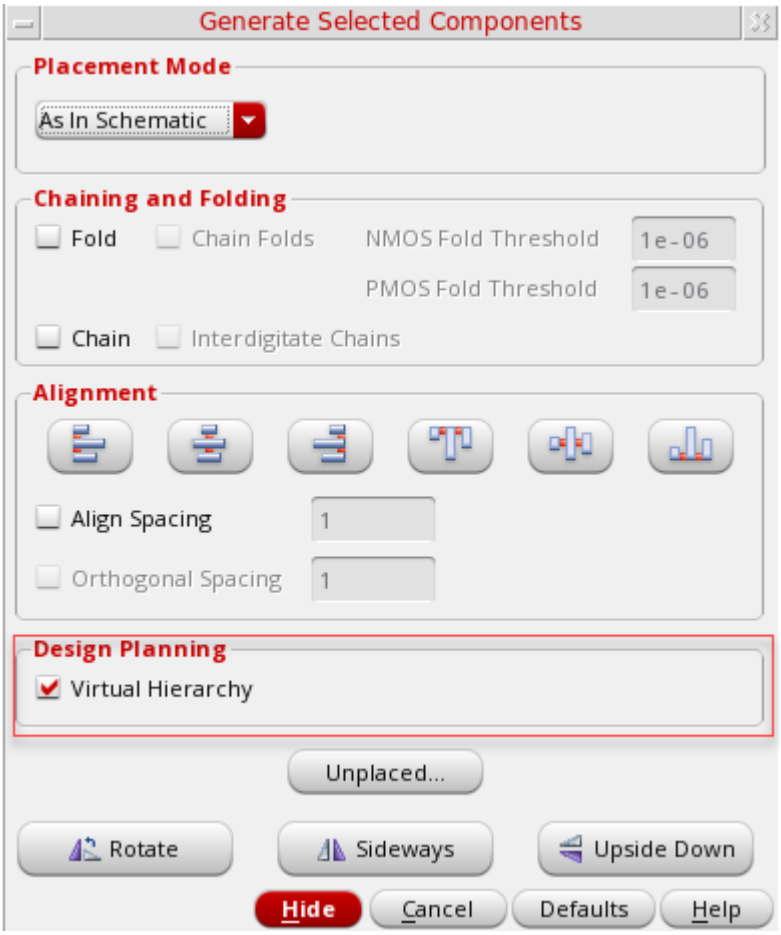
Generate Selected Components Form

Align Objects using the Align Toolbar

© 2023 Cadence Design Systems, Inc.

Generation of Virtual Hierarchy using Selected Schematic Components

In Layout EXL or higher tier, the Generate Selected Components form displays an additional *Design Planning* group box in the *As In Schematic* placement mode.



The *Virtual Hierarchy* check box in the Design Planning group box is OFF by default, unless the form is opened using the Design Planning Toolbar or the Design Planning Menu.

The *Virtual Hierarchy* option allows you to generate a “yet-to-be-realized” hierarchy for the selected schematic components, which can be accessed from the top level using the DESIGN PLANNING category in the Navigator assistant.

For more information on virtual hierarchies and the layout generation capabilities that are offered by Layout EXL or higher tier to work with virtual hierarchies, see the *Virtuoso Design Planner User Guide*.

© 2023 Cadence Design Systems, Inc.

Related Topics

Design Planning Toolbar

Design Planning Menu

Virtual Hierarchy Generation

Generate Selected Components Form

Automatic Snapping of FinFET Devices during Layout Generation

If you are generating a layout for a design that uses FinFET devices, Layout XL correctly snaps these devices to the fin grid defined for the design.

- If you use the *Generate Selected From Source* command in the *As In Schematic* mode, the generated devices are automatically snapped to the fin grid.
- If you move a chain of abutted devices in the layout canvas, all the devices in the chain are snapped together, that is, the snapping does not break existing chains.

Related Topics

Generate Selected Components Form

Instance/Pin List Form

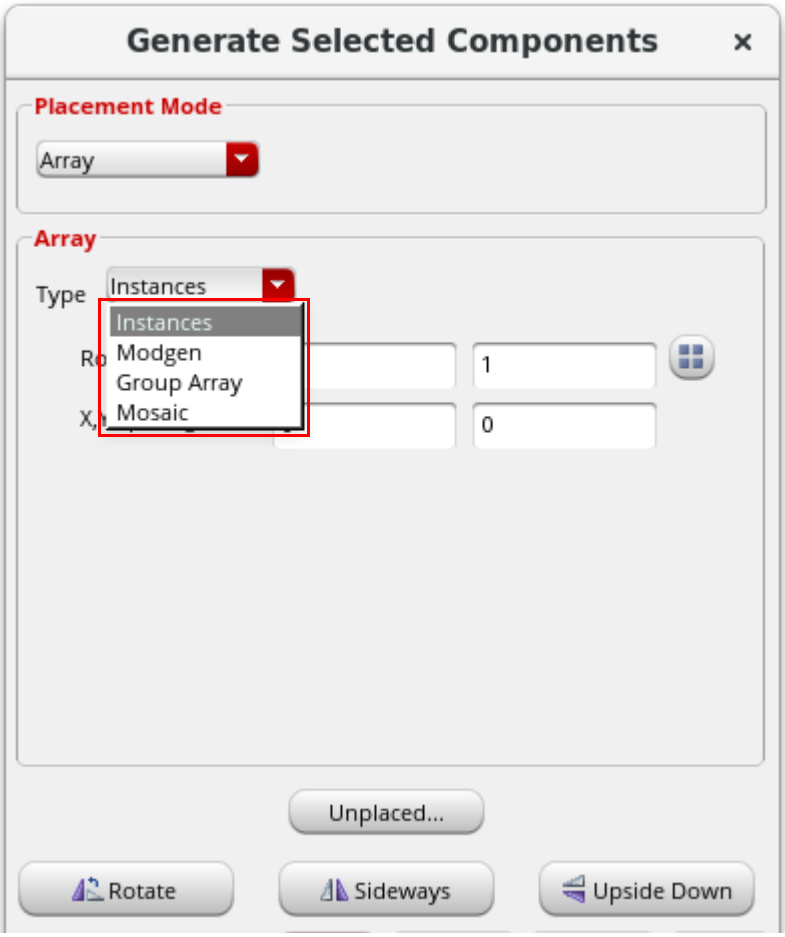
renameConflictingLayoutInstances

disableFolding

© 2023 Cadence Design Systems, Inc.

Array Generation

Layout XL lets you generate selected or unplaced schematic instances as an XL-compliant array in the layout view. You can create an array of instances with the same or different master, parameters, or connectivity.



You can choose to create an array of ungrouped instances, a modgen, a mosaic, or a group array.

During array generation, preselected objects are automatically squared in the array. To square postselected objects, use the button.

Array placement mode cannot be used to place pins.

© 2023 Cadence Design Systems, Inc.

Related Topics

Generate Selected Components Form

Instance/Pin List Form

generateXSpacing

generateYSpacing

Array Generation - Instances

Array Generation - Modgen

Array Generation - Mosaic

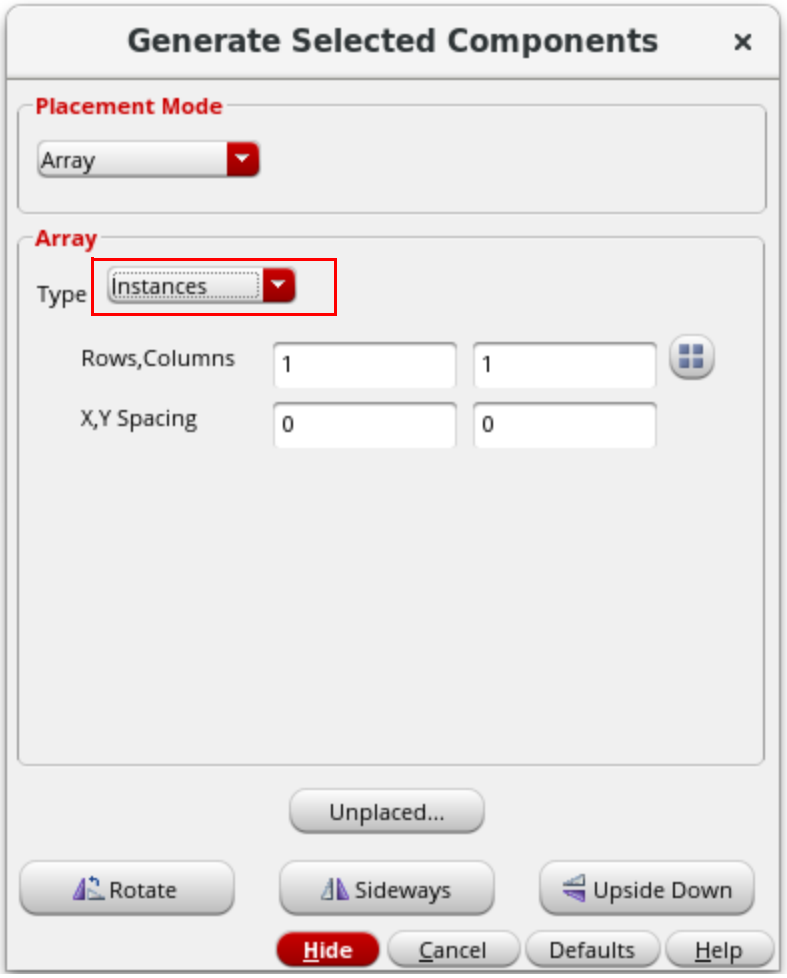
Array Generation - Group Array

© 2023 Cadence Design Systems, Inc.

Array Generation - Instances

An array of ungrouped instances that is generated based on row or column values and the spacing values that you specify.

You can manually type in the X and Y spacing values in the fields provided, or use the `generateXSpacing` and `generateYSpacing` environment variables to specify the spacing values, as appropriate.



Instance arrays are XL-compliant and enable simple array generation for a group of selected components. However, the selected schematic instances must not be members of any top-level modgen constraint.

© 2023 Cadence Design Systems, Inc.

The selected components are generated in the layout as free objects, as displayed in the figure below. If you modify the array or spacing dimensions, you must regenerate the array.



Related Topics

Array Generation - Modgen

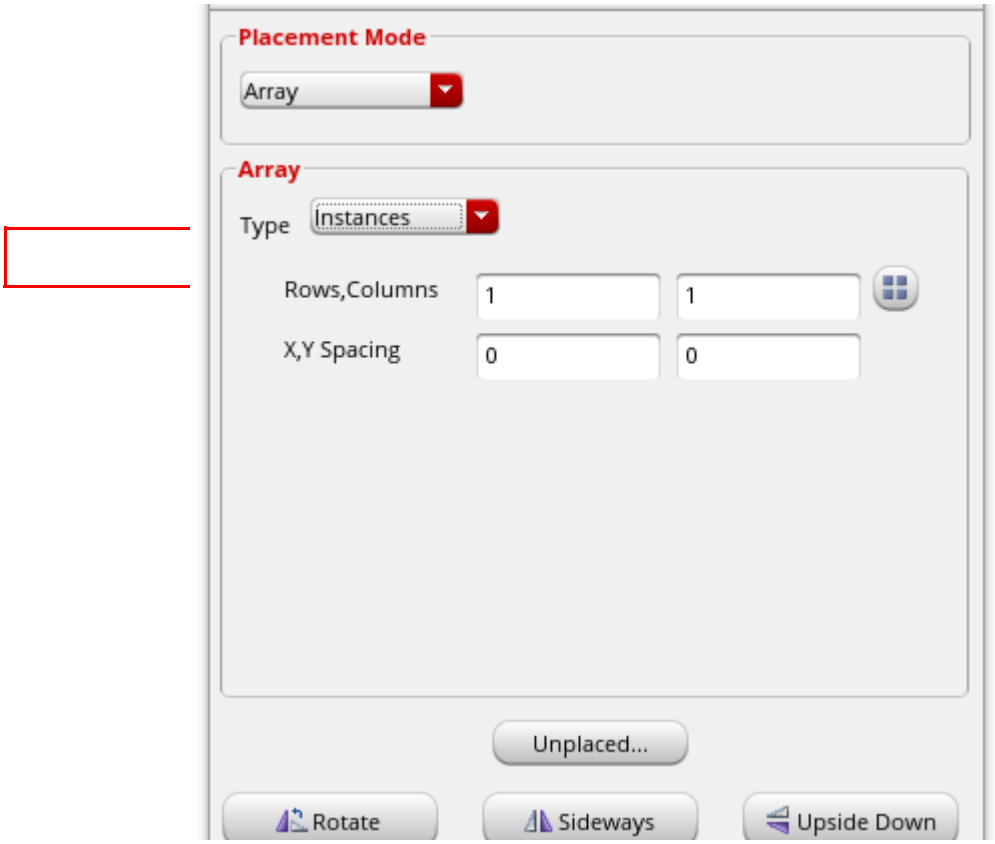
Array Generation - Mosaic

Array Generation - Group Array

© 2023 Cadence Design Systems, Inc.

Array Generation - Modgen

An array of instances that are grouped in a modgen constraint and have same or different connectivity and masters.



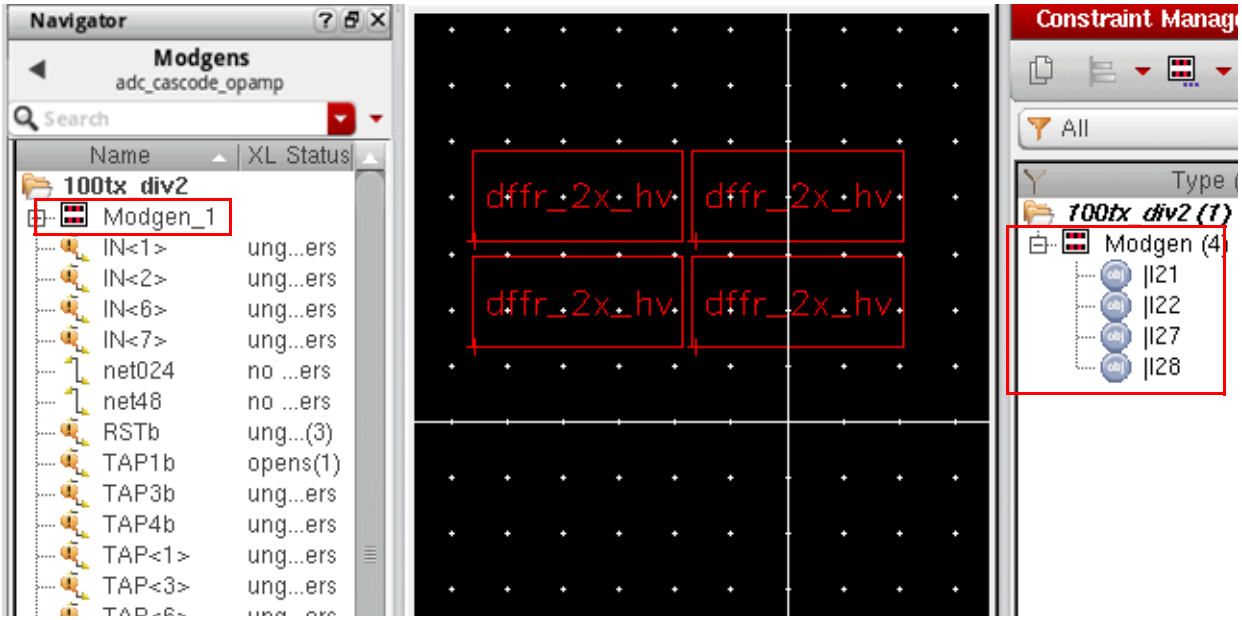
You can choose to abut all the modgen instances to generate a compact placement or specify the spacing values and reference layers for X and Y spacing, as appropriate. By default, the spacing is DRC correct and no reference layers are specified.

Alternatively, you can use the `generateModgens` environment variable to generate a modgen array.

Cadence recommends that you generate a modgen if it is likely that the array will require updating after placement. To update the modgen, run the `Modgen Editor`.

© 2023 Cadence Design Systems, Inc.

The selected components are generated in the layout as a group and are combined in a modgen constraint, as shown in the figure below.



Related Topics

generateModgens

Modgen Editor

generateXSpacing

generateYSpacing

Array Generation

Array Generation - Mosaic

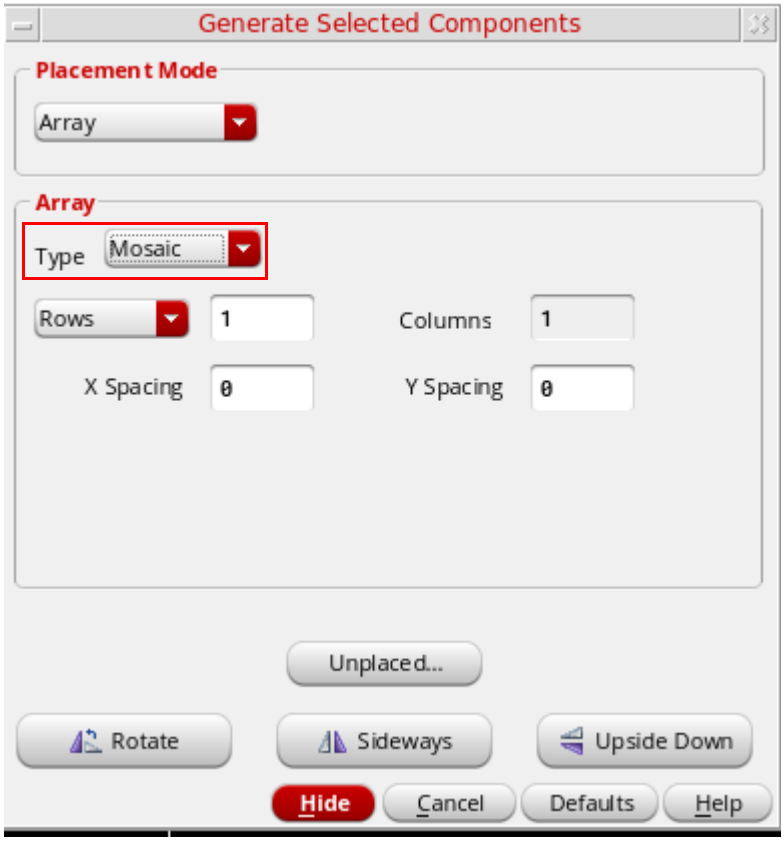
Array Generation - Group Array

Generate Selected Components Form

© 2023 Cadence Design Systems, Inc.

Array Generation - Mosaic

- *Mosaic*: A compact array of instances that belong to the same mfactored instance and, therefore, have the same master and connectivity. In addition, array creation is supported for vectored instances with same master and same connectivity for each bit.

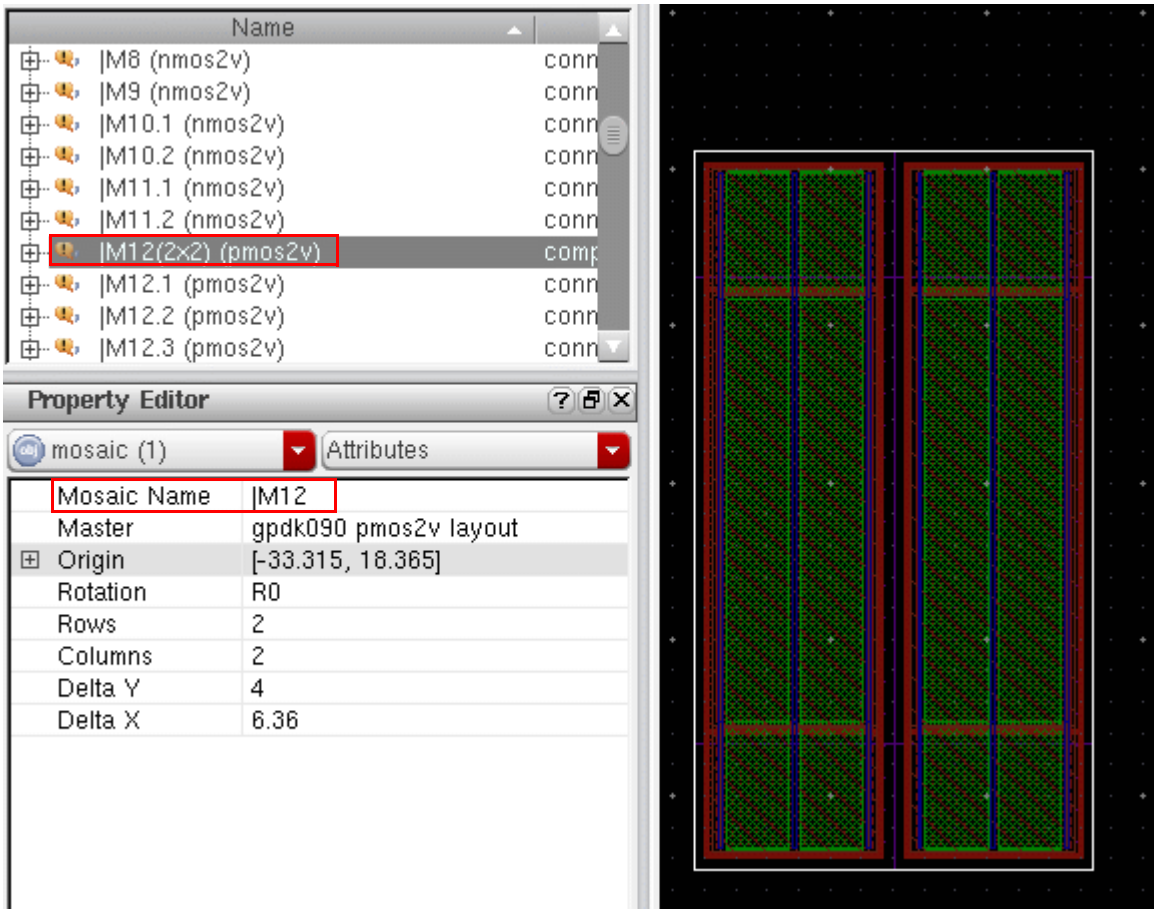


To generate a mosaic array, the selected set must not contain any pins and the selected instances must not be members of any top-level modgen constraint.

Mosaics have restricted routing capabilities and are recommended for generating large mfactored arrays because they consume less memory.

© 2023 Cadence Design Systems, Inc.

The selected components are generated in the layout as a compact mosaic, as shown in the figure below. If required, you can modify the row, column, and spacing values of the generated mosaic by using the Property Editor assistant.

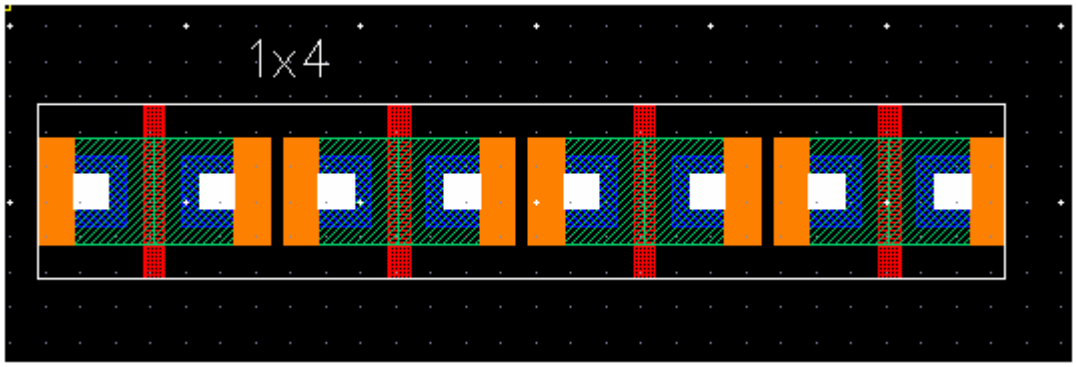


Depending on the row, column, and spacing values that you specify in the Generate Selected Components form, you can control the configuration of the array to be

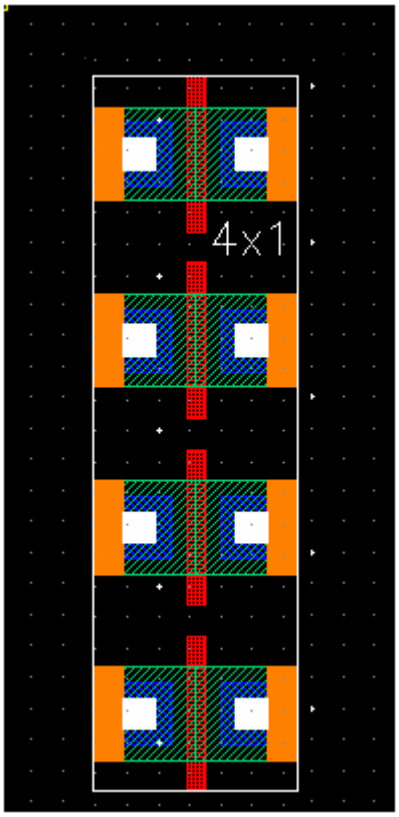
© 2023 Cadence Design Systems, Inc.

generated. The figure below displays the possible configurations in which a set of selected schematic components can be generated as an array in the layout view.

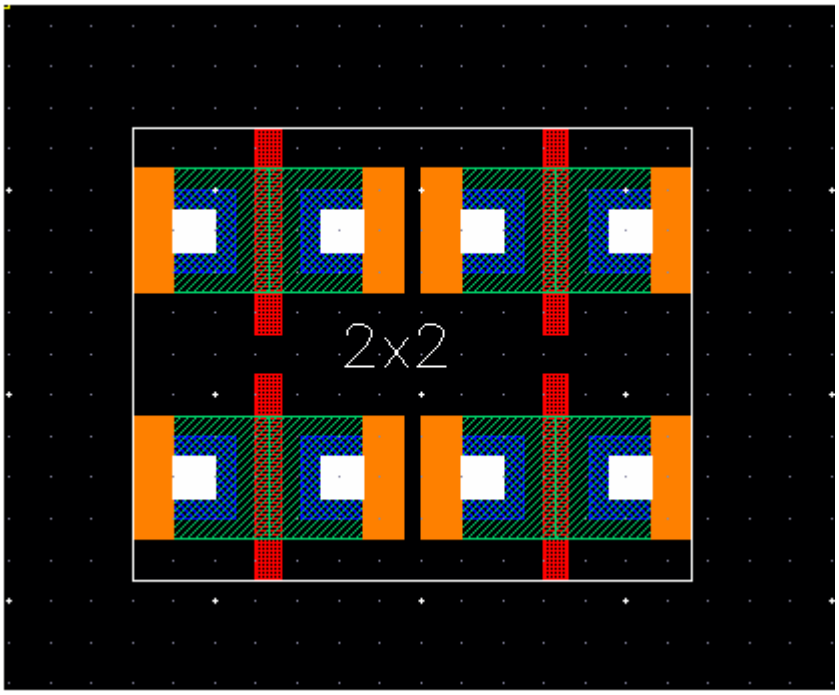
Configuration 1 - (1 X4) Array



Configuration 2 - (4 X1) Array



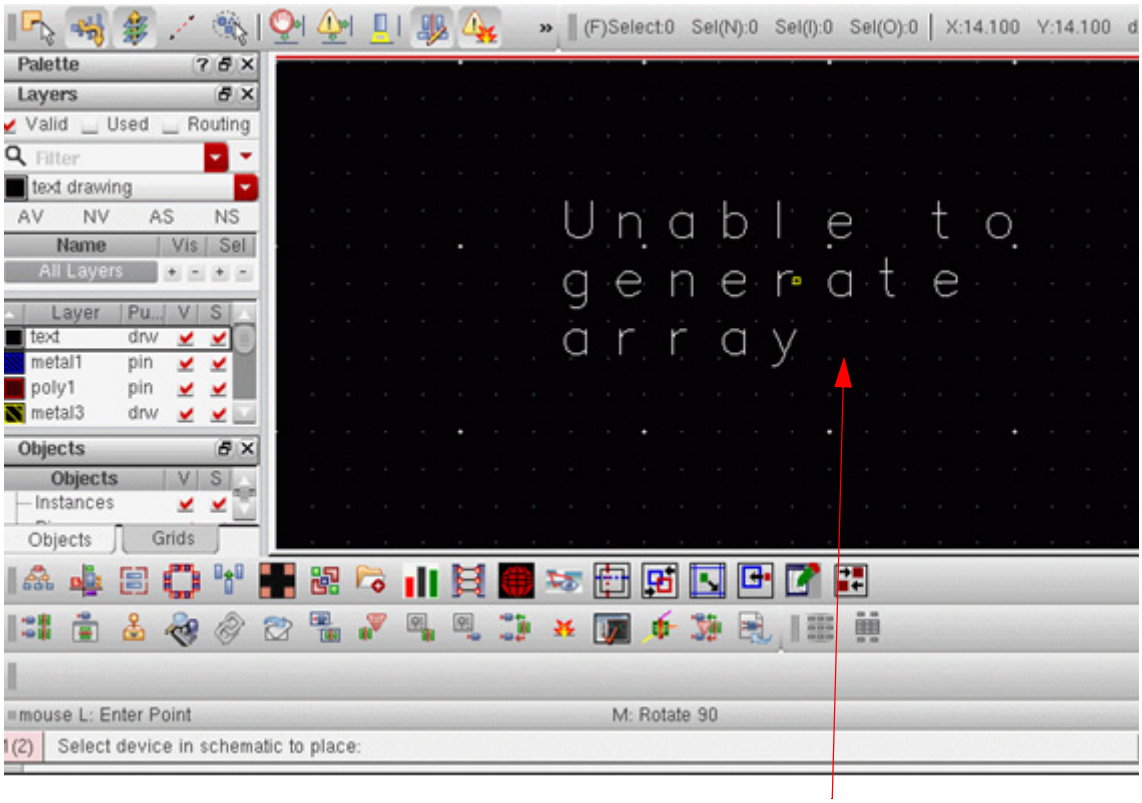
Configuration 3 - (2 X2) Array



By default, the number of rows to be generated is set to `1` and the number of columns to be generated is set to the selected number of instances. However, you can choose to modify either the number of rows or the number of columns to be generated.

© 2023 Cadence Design Systems, Inc.

If the selected set of components cannot be generated as an array, a text label is displayed in the layout canvas to indicate this, as shown in the figure below.



- An array cannot be generated if any of the selected schematic instances are in a top-level modgen constraint.
    - If any of the selected instances are in a top-level symmetry constraint, other members of the constraint cannot be added to the array.

A 1X1 mosaic cannot be generated using Array placement mode in Generate Selected Components form.

Related Topics

Creating a Mosaic Instance

generateXSpacing

generateYSpacing

© 2023 Cadence Design Systems, Inc.

Array Generation

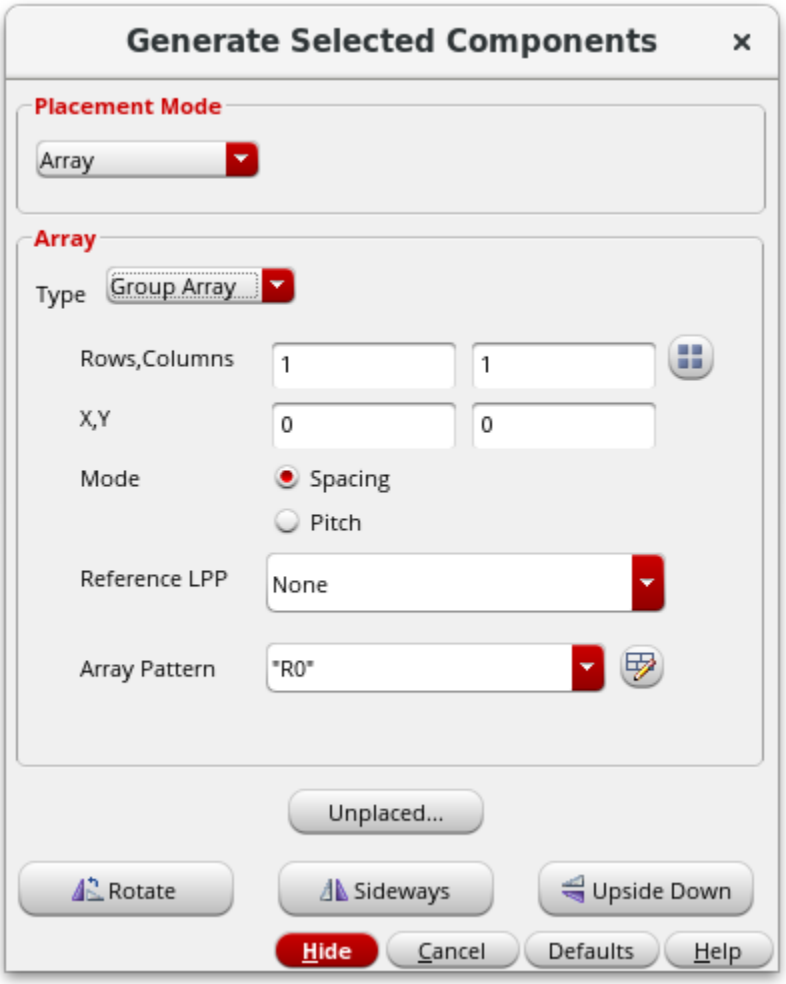
Array Generation - Modgen

Array Generation - Group Array

Generate Selected Components Form

Array Generation - Group Array

Certain custom and digital designs, such as those containing memory arrays and routing channels, might require patterns of objects to be repeated across a design. You can use group arrays to create such repeated patterns in your designs. Group arrays are XL compliant and support mixed connectivity with different masters.

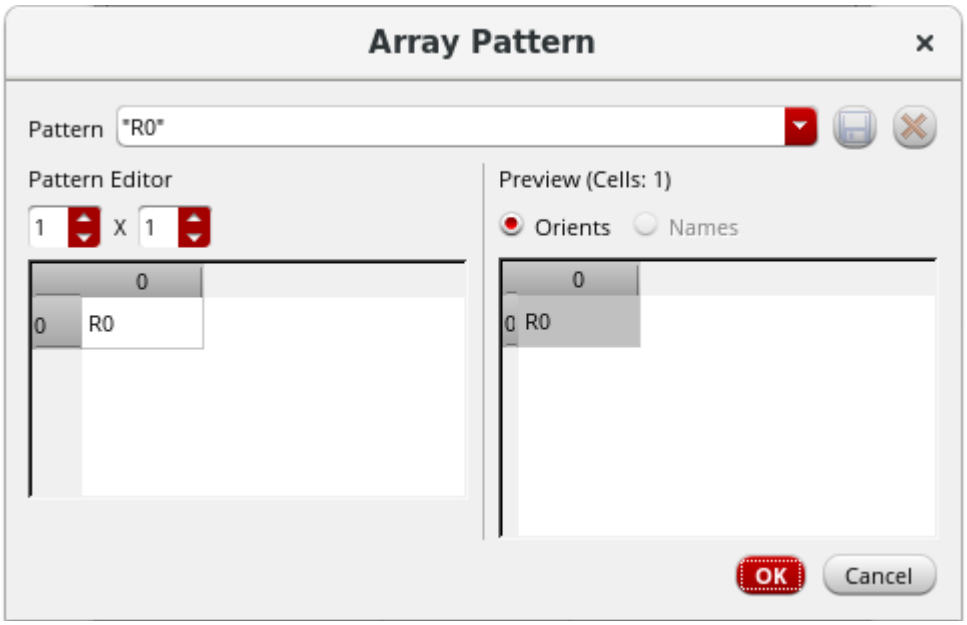


© 2023 Cadence Design Systems, Inc.

To generate a group array, you need to specify number of rows and columns and X and Y coordinates. *Mode* lets you specify the type of spacing for the group array.

Reference LPP is used to specify the layer-purpose pair used for computing the bounding box of the cells in a group array. The combined bounding box of the geometries on the specified reference layer-purpose pair is used for calculating the spacing between cells.

Array Pattern is used to control the orientation pattern for the group array. The Array Pattern form is displayed when you click the *Array Pattern* button on the Generate Selected Components form.

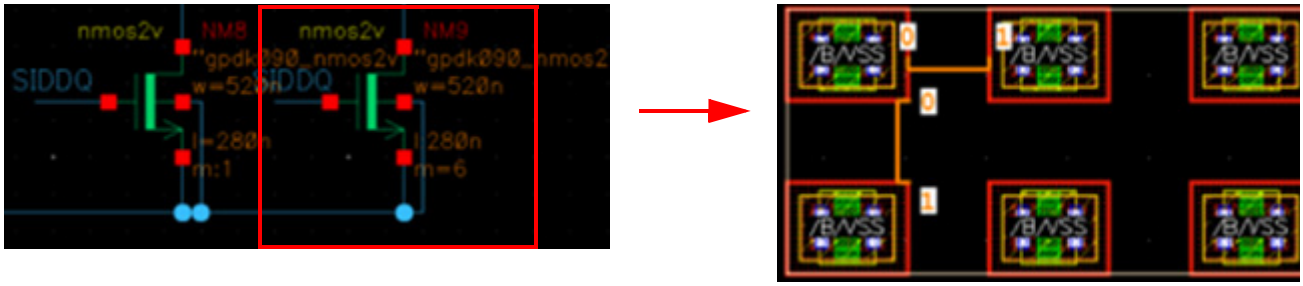


You can also change the orientation pattern of the array as required.

Layout components will appear in the drag set that you can place in the layout canvas. All the instances of the group array are bound to respective schematic instances and their XL status

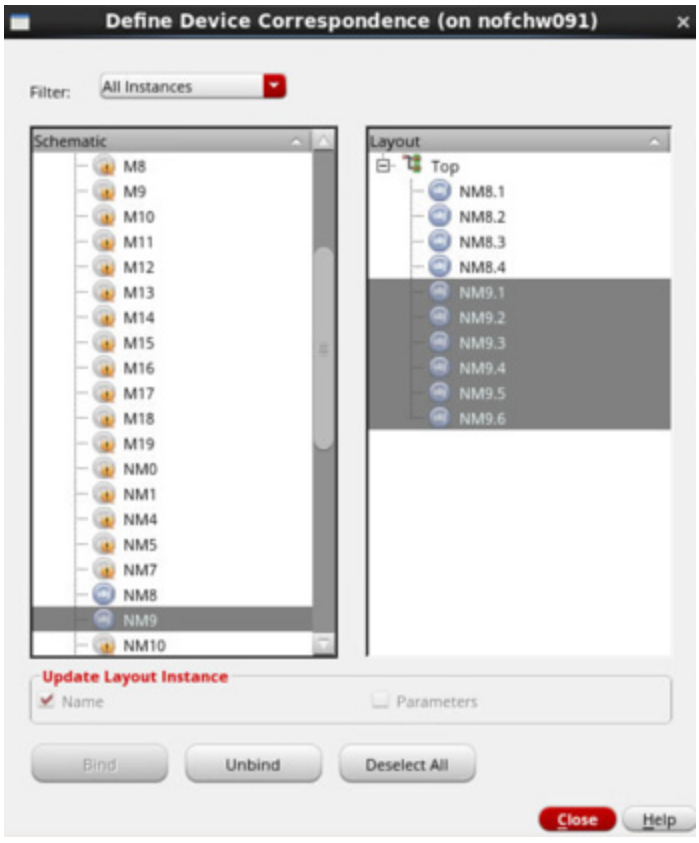
© 2023 Cadence Design Systems, Inc.

is OK for all the instances in the Navigator. Cross selection works between group arrays and schematic instances where members of the group array are highlighted in the canvas.



Cross Selection between Schematic and Layout

You can check or define the binding of instances in a group array using the Define Device Correspondence form.

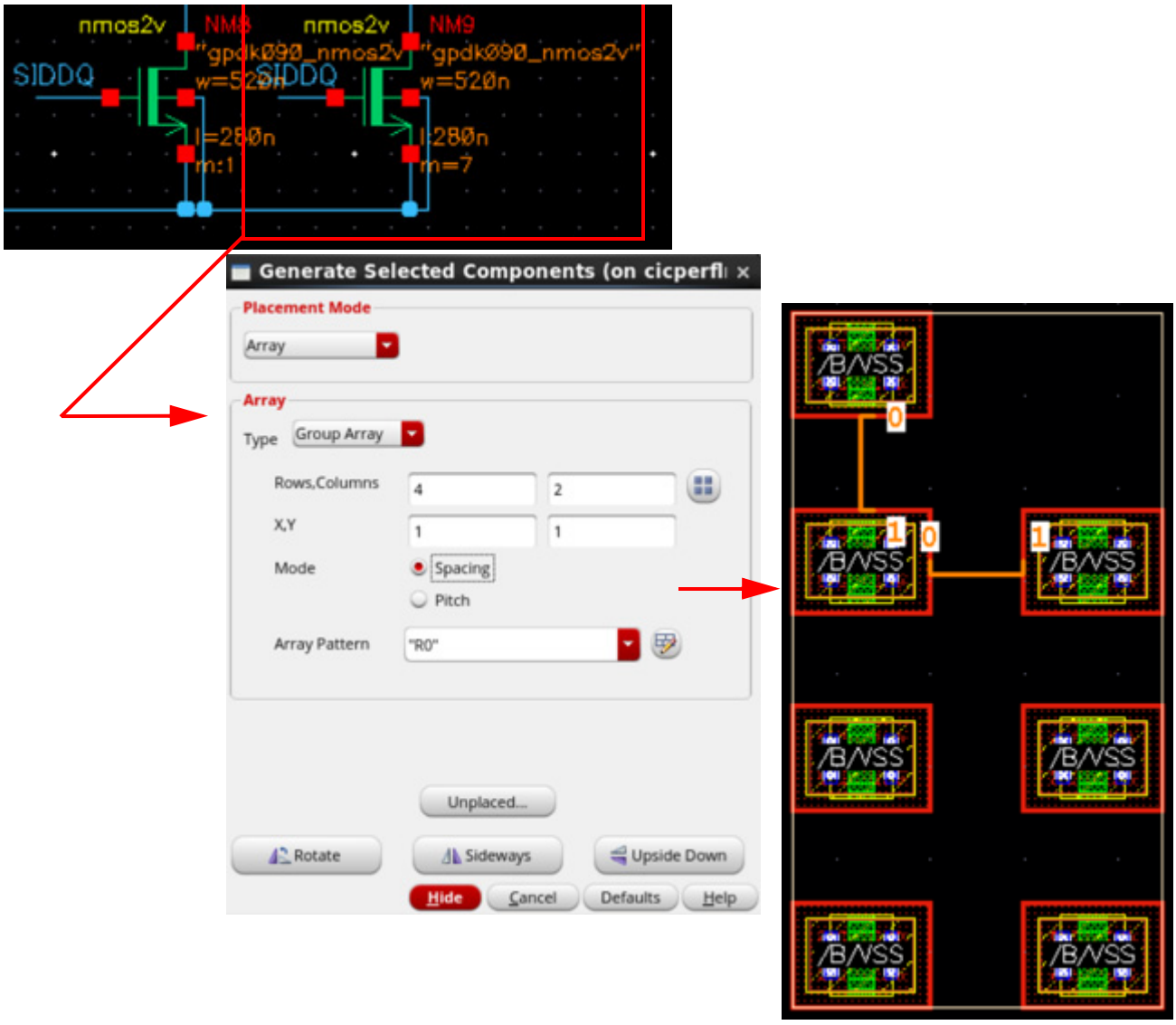


Selected layout instances are members of a group array

In group arrays, you can create holes by specifying different number of rows or columns that result in different sizes.

© 2023 Cadence Design Systems, Inc.

Consider a schematic instance where m=7 and, specify 4 in either the row or column field of the Generate Selected Components form. The other field (row or column) is automatically filled with the value 2. The resultant group array has one hole to complete the product because there are 7 members in the group array.



You cannot generate a group array:

- If instances of different masters or different parameters are selected in the schematic.
- While editing inside a Group Array. You should either choose a different placement mode or create the Group Array outside the existing Group Array.

© 2023 Cadence Design Systems, Inc.

In these cases, you might see the following message in the layout canvas and a warning message is displayed in the CIW.



Related Topics

Generate Selected Components Form

Specifying Group Array Orientation Using the Array Pattern Form

generateXSpacing

generateYSpacing

Group Arrays

Holes in a Group Array

Array Generation

Array Generation - Modgen

Array Generation - Mosaic

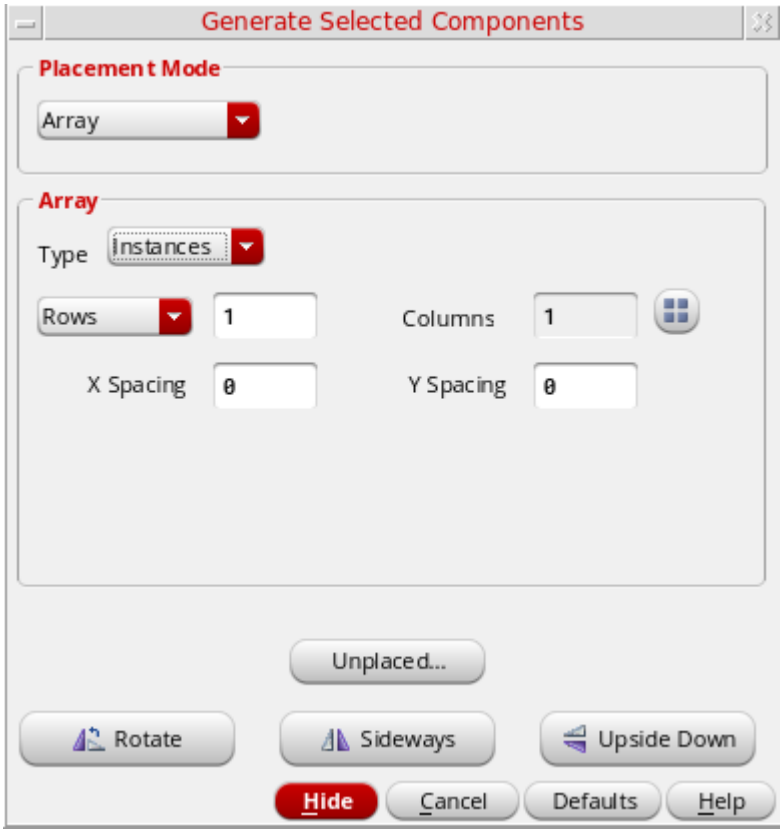
© 2023 Cadence Design Systems, Inc.

Generating an Array

To generate an array:

1. From the layout window menu bar, choose *Connectivity – Generate – Selected From Source*.

   The Generate Selected Components form is displayed (press `F3` if it is not displayed).



2. In the *Placement Mode* group box, choose *Array*.
3. In the schematic window; select the instances you want to generate using one of the following methods:

   - Click the components in the schematic canvas or Navigator assistant. To select multiple components, keep the `Shift` key pressed during selection or area-select the components. To deselect a component, click the component while the `Ctrl` key is pressed.
   - Click *Unplaced* to view the Instance/Pin List form. In the form, select the instances to place and click *Apply* to select them in the schematic window. See Listing Unplaced Components after Layout Generation.

© 2023 Cadence Design Systems, Inc.

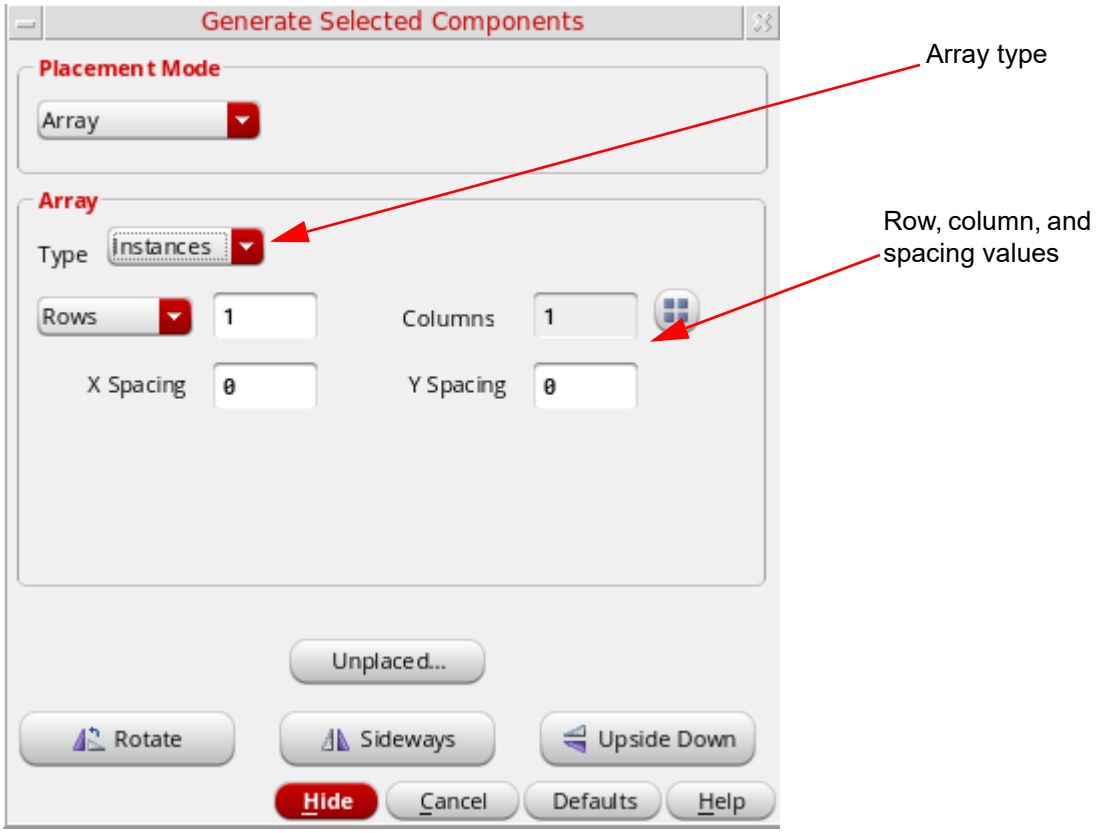
The selected schematic instances remain highlighted until they are either deselected or generated in the layout view. You cannot select an object in the schematic if an object with the same name already exists in the layout.

Alternatively, right-click the ungenerated instances in the layout Navigator assistant and choose *Generate Selected From Source*. Press `F3` if the Generate Selected Components form does not display automatically.

4. Move your pointer into the layout canvas.

   The outlines of the selected instances follow the pointer in the canvas.

5. Set the options you want for the selected components.



6. Click in the layout canvas to place the selected instances at the required location.

Related Topics

Generate Selected Components Form

Instance/Pin List Form

© 2023 Cadence Design Systems, Inc.

Generating Individual Instances

To generate schematic instances one at a time in the layout view:

1. From the layout window menu bar, choose *Connectivity – Generate – Selected From Source*.

   The Generate Selected Components form is displayed (press `F3` if it is not displayed).



2. In the *Placement Mode* group box, choose *Individually* from the drop-down list.
3. In the schematic window, select the instances you want to generate using one of the following methods:

   - Click the components in the schematic canvas or Navigator assistant. To select multiple components, keep the `Shift` key pressed during selection or area-select the components. To deselect a component, click the component while the `Ctrl` key is pressed.
   - Click *Unplaced* to view the Instance/Pin List form. In the form, select the components to place and click *Apply* to select them in the schematic window. See Listing Unplaced Components after Layout Generation.

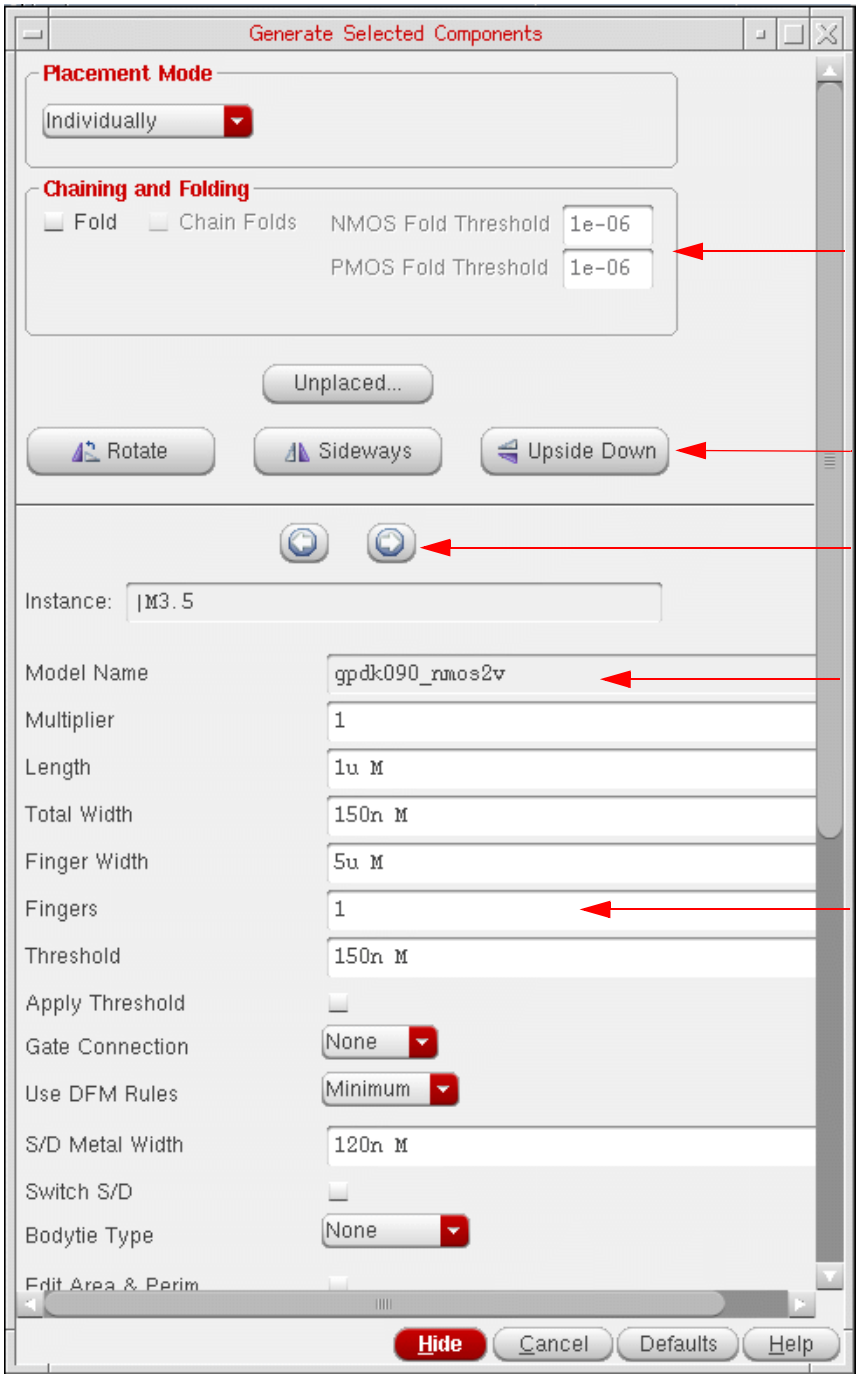
© 2023 Cadence Design Systems, Inc.

The selected instances remain highlighted until they are either deselected or generated in the layout view. You can select an instance in the schematic if it is not already bound to an instance in the layout. If there is an existing layout instance with a matching name which is not bound to the selected schematic instance, the name of the instance generated in layout is controlled by the `renameConflictingLayoutInstances` environment variable.

Alternatively, right-click the ungenerated instances in the layout Navigator assistant and choose *Generate Selected From Source*. Press `F3` if the form does not displayed automatically.

© 2023 Cadence Design Systems, Inc.

The form expands to show the name and master of the first instance in the selected set (the current instance), along with the properties set on that instance, as shown in the figure below.



4. Move your pointer into the layout canvas.

© 2023 Cadence Design Systems, Inc.

The outline of the current instance follows the pointer in the layout window.

5. Set the options and property values you want.

   - Select *Fold* to automatically fold the current instance.

     If the `disableFolding` environment variable is set and the folding threshold for the component type is set to `0`, folding is disabled.

   - Select *Chain Folds* to automatically chain the folded instance.
   - Choose the orientation for the instance using the *Rotate*, *Sideways*, and *Upside Down* buttons.
   - Change the instance property values, if required. The instance properties are not shown when *Fold* is selected. To update a property, you must first turn off *Fold*.

   The *Chaining* and *Alignment* options are disabled because you are placing only one instance at a time.

6. Click in the layout canvas to place the current instance at the required location.

   If the layout library and cellview are different from those of the schematic, the command copies all the parameters (including default parameters) into the layout view. If any of the default parameter values in the schematic is an empty string, this can cause Pcell evaluation to fail.

   If you selected more than one instance, the next instance in the list becomes the current instance in the form. You can use the *Previous* and *Next* arrow buttons at the top of the properties section to quickly move back or forward through the set of selected instances without generating them.

7. Repeat steps **4** through **6** to generate the next instance.

Related Topics

Generate Selected Components Form

Instance/Pin List Form

renameConflictingLayoutInstances

disableFolding

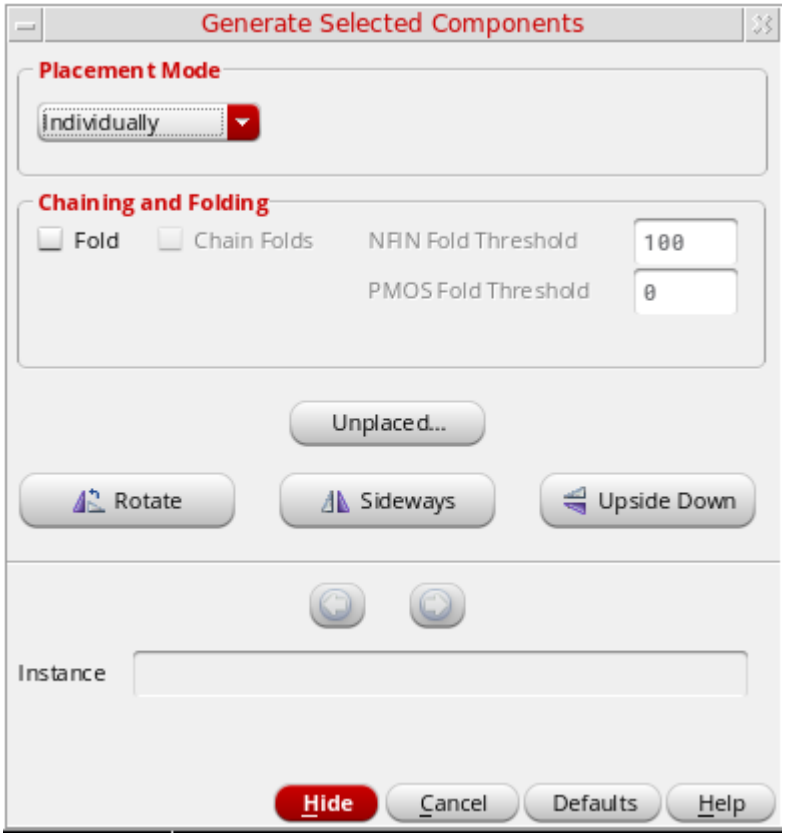
© 2023 Cadence Design Systems, Inc.

Generating Individual Pins

To generate schematic pins one at a time in the layout view,

1. From the layout window menu bar, choose *Connectivity – Generate – Selected From Source*.

   The Generate Selected Components form is displayed (press `F3` if it is not displayed).



2. In the Placement Mode group box, choose *Individually* from the drop-down list to generate the pins one at a time.
3. In the schematic window, select the pins you want to generate using one of the following methods:

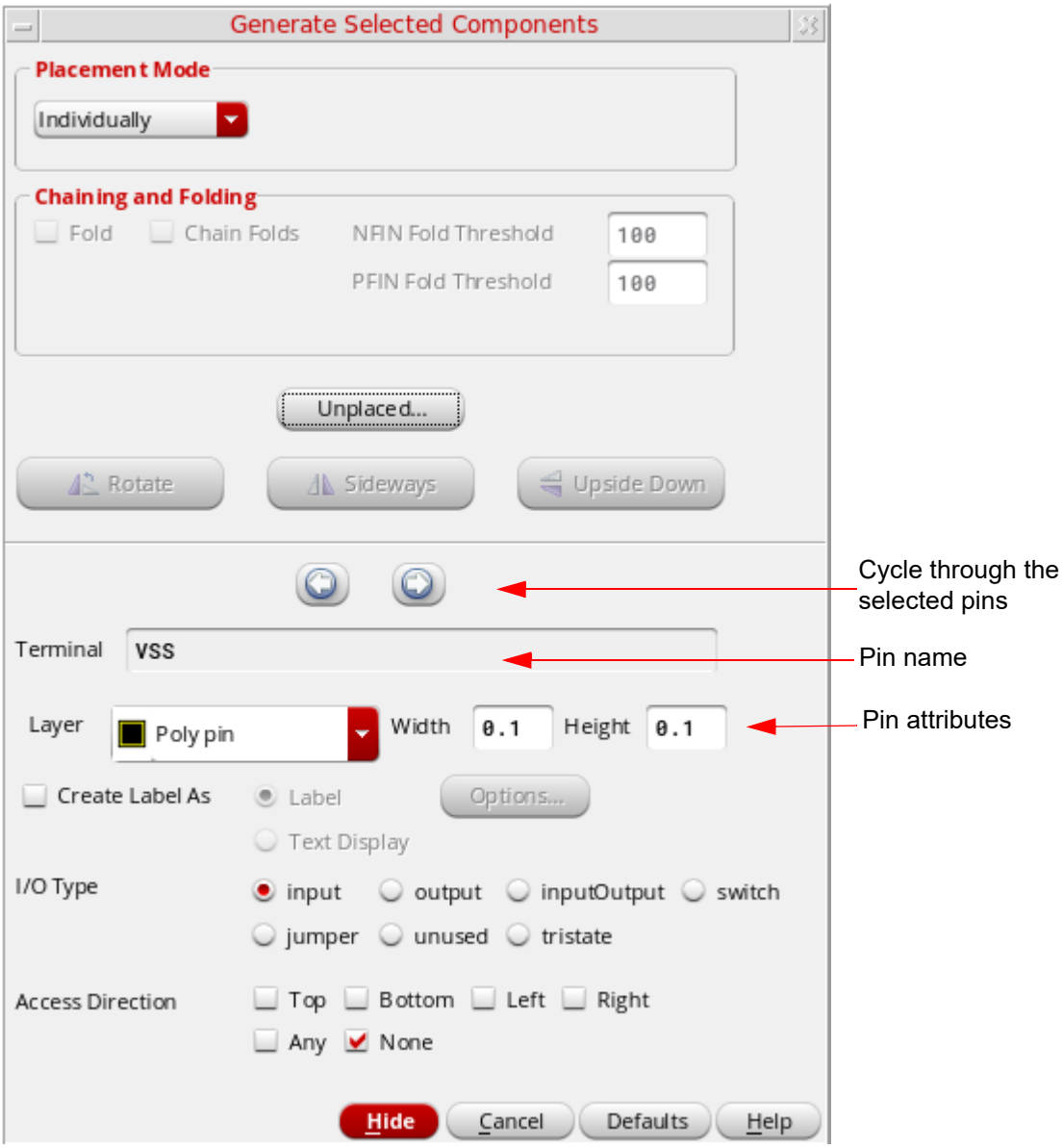
   - Click the components in the schematic canvas or Navigator assistant. To select multiple components, keep the `Shift` key pressed during selection or area-select the components. To deselect a component, click the component while the `Ctrl` key is pressed.

© 2023 Cadence Design Systems, Inc.

- Click Unplaced to view the Instance/Pin List Form. In the form, select the components to place and click Apply to select them in the schematic window. See Listing Unplaced Components after Layout Generation.

The selected pins remain highlighted until they are either deselected or generated in the layout view. You cannot select a pin in the schematic if an instance with the same name exists already in the layout.

The form expands to show the name of the first pin in the list (the current pin), along with the attributes set on that pin.



© 2023 Cadence Design Systems, Inc.

The current pin is highlighted in the schematic in the color assigned to the `Poly pin` layer. Selected pins that have not yet been generated are highlighted in the color assigned to the `Poly pin` layer.

4. Move your pointer into the layout canvas.

   The outline of the current pin follows the pointer in the layout window.

5. Set the options and pin attributes you want.

   - Change the pin attributes if required. Any changes you make are applied to the current pin and also become the defaults for all subsequent pins in the selected set. For more information, see Generate Selected Components - Pin Options Form.

   The *Chaining and Folding* and *Alignment* options and the orientation buttons are disabled because you are placing pins.

6. Click in the layout canvas to place the current pin at the required location.

   The pin has the connectivity indicated in the schematic and is automatically snapped to the placement grid.

   If you selected more than one pin, the next pin in the list becomes the current pin in the form. You can use the *Previous* and *Next* arrow buttons at the top of the pin attributes section to quickly move back or forward through the set of selected pins.

7. Repeat steps **4** through **6** to generate the next pin in the selected set.

Related Topics

Generate Selected Components Form

Instance/Pin List Form

Generate Selected Components - Pin Options Form

Listing Unplaced Components after Layout Generation

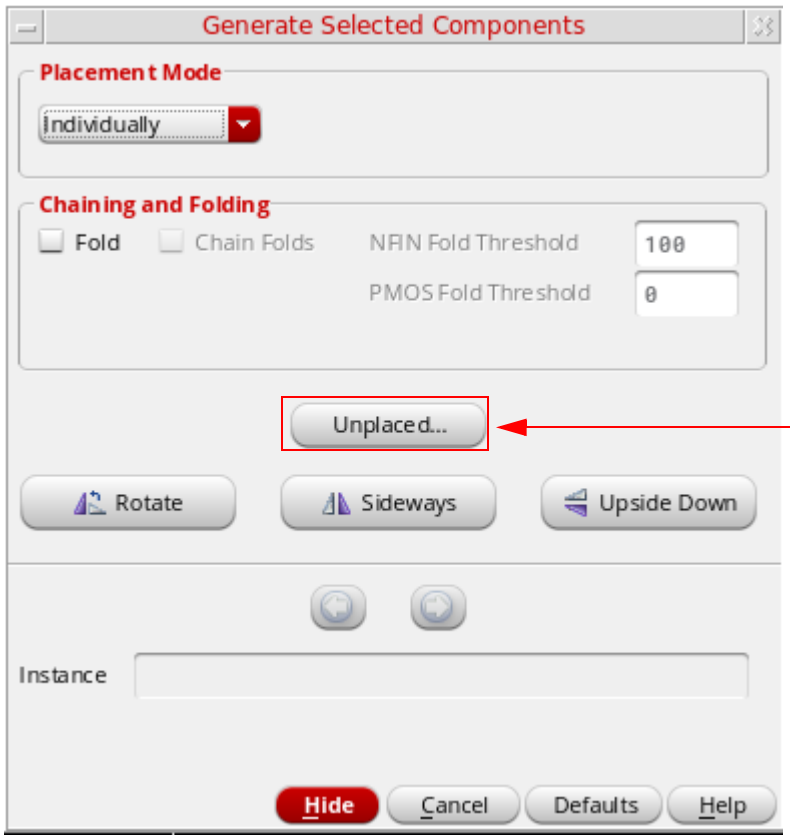
To see a list of all schematic components not yet generated in the layout view:

1. From the layout window menu bar, choose *Connectivity – Generate – Selected From Source*.

   The Generate Selected Components form is displayed (or press `F3`).

© 2023 Cadence Design Systems, Inc.

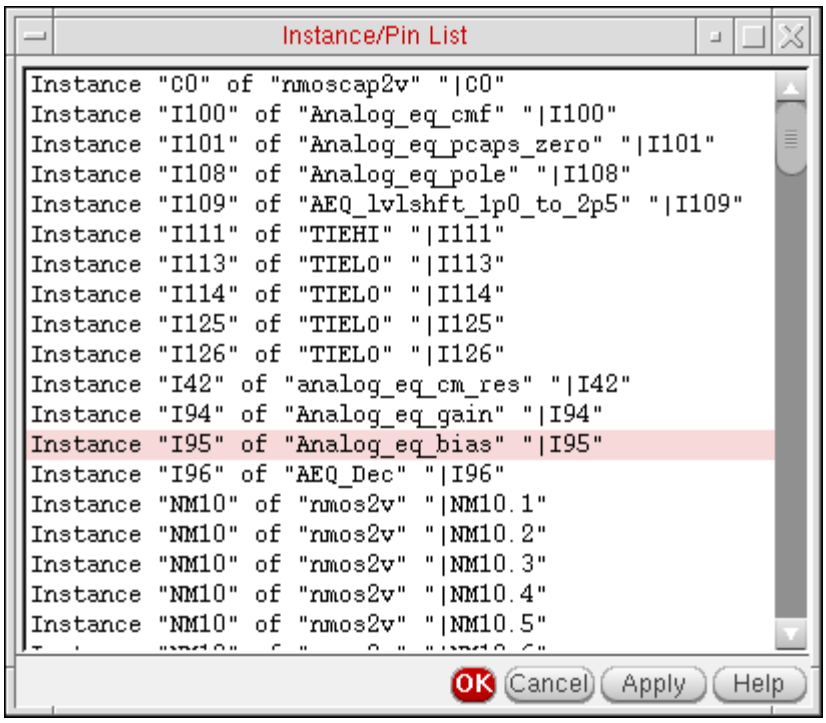
2. Click *Unplaced*.



Click *Unplaced* to show schematic components not yet generated

© 2023 Cadence Design Systems, Inc.

The Instance/Pin List form is displayed.



Wires, labels, text, instances with an `ignore` property, and instances that have already been placed in the layout are not considered available components and are not shown in the list.

3. Choose one or more instances and pins in the list and click *Apply*.

   The Generate Selected Components form changes to show the available information on the first component in the selected set.

Related Topics

Generate Selected Components Form

Instance/Pin List Form

ignore Property

© 2023 Cadence Design Systems, Inc.

Clone Generation

Cloning is the ability to replicate a section of the layout, associated with a section of the schematic, in such a way that the new piece of layout material can be placed at more than one location, with each part preserving the hierarchical structure of the design. Cloning differs from synchronous copying in that the cloned structure maintains its schematic correspondence and inherits the connectivity information, while a synchronous copy does not maintain any schematic correspondence.

Layout XL also supports clone creation by setting a *Clone Family* constraint in the Constraint Manager. For more information, see Instance Cloning Using a Clone Family Constraint and the cloneFamily option described under Constraint Manager Toolbar in *Virtuoso Unified Custom Constraints User Guide.*

Cloning can help you lay out:

- Datapath bit slices, datapath and memory control blocks with identical logic cones, programmable logic structures, and other arrayed and iterated structures in device-level and gate-level custom digital design.
- Groups of analog/mixed signal components with identical constraints (for example, current mirrors, cascode stages, differential pairs, and other isomorphic structures).
- Instances of the same symbol master – or instances with the same logic but with differing parameters and hence a different symbol master – that have a flat layout representation.
- A chained set of devices or a part of the chain.
- Members of a `Clone Family` constraint.
- Transparent instances in the layout such that the generated clone is also a transparent instance.

The section of the schematic or layout that is used as a template for the clone is called the *clone source*. The section of the schematic that is to be implemented by replicating the source is the *clone target*. The result of replicating the clone source to implement the clone target is the clone. For the clone source to be used to generate a clone target, the clone source should be bound.

You can use multiple connectivity sources and multiple layout cellviews, including layout cellviews that are open in read-only mode, to generate clones. The clone source that you select can comprise devices, pins, and (if selected from the layout canvas) interconnect structures such as wires and path shapes. You can also clone components that are implemented as one-to-many relationships defined using the `mfactor` property on a schematic instance.

© 2023 Cadence Design Systems, Inc.

A source structure is considered invalid for cloning if it includes:

- any device that is part of a user-defined many-to-one binding (specified using the Define Device Correspondence form).
- a folded device that has only some of its folds selected for cloning.
- instances that do not exactly match the `Clone Def` of a `Clone Family` constraint.
- An instance that is neither bound nor ignored, which makes the device an invalid clone dummy device.
  - For a device to be considered as a valid clone dummy device, it must either have one of the Layout XL ignore properties or a `physOnly` attribute set.
- One or more group arrays.

In each case, a warning appears in the CIW explaining the error and the corrective action.

Related Topics

Synchronous Copy Generation

Transparent Layout Hierarchy Generation

Mfactored Component Cloning

Exact and Non-Exact Matches for Cloning

Types of Clones

Instance Cloning

Define Device Correspondence Form

Exact and Non-Exact Matches for Cloning

To clone components, you select the structure to be replicated (the clone source) and the software searches the design for target structures that match the source. You create a clone when you generate a target structure in your design.

To be considered a valid target, the instance master referenced by the components in the target must exactly match the instance master referenced by the components in the source. A target that also matches the source in terms of connectivity is considered an *exact match*.

© 2023 Cadence Design Systems, Inc.

A target that tolerates slight differences in connectivity is a *non-exact match*, also known as a mutant clone.

You can use the options in the Generate Clones form to specify which differences are tolerated.

- *Allowed Permutation* considers pin permutability when searching for matching target structures. If an appropriate permuteRule Property is defined for the device in question, and if permuting the pins on a device results in a match, then that match is reported (but only if no matching targets were found without permutation).

  Example

  Two NMOS devices connected by their D terminals will match two NMOS devices connected by their S terminals, provided there is a permute rule in place that allows the S and D pins to be permuted.

  For more information, see Pin Permutation.

- *Exact Parameters* requires that the parameter values on the components in a target structure match exactly the parameter values on the components in the clone source. When switched off, Layout XL also reports target structures that have different parameters and values.

  Example

  When switched off, an NMOS device in the source with w = 200 will match an NMOS device with w = 100 in the target.

  If required, you can update the parameters and values to match when you generate a clone by checking the *Update Layout Parameters to Match Schematic* option in the Generate Clones Form form.

  To prevent any layout-only parameters from getting updated, ensure that the required parameters are set as parameters to be ignored. If any abutment parameters are required for clone generation, they might need to be ignored as well.

  You can set the parameters to be ignored using any of the following methods:

  - Adding the layout-only parameters to the *ignore for generation* list in CPH.

    See Ignore Parameters and Terminals.

  - Add the layout-only parameters to the *Parameters to ignore during generation and update* list in the Connectivity form.

    See Parameters.

  - Add the layout-only parameters to the *Ignore for generation* list in the LAM file.

© 2023 Cadence Design Systems, Inc.

See LAM File Mapping Section.

- *Exact Connectivity* requires that the connectivity of the components in a target structure matches exactly the connectivity of the components in the clone source. When switched off, Layout XL also reports target structures where the set of instances is the same, but they are connected differently.

  Example

  When switched off, two NMOS devices connected by their D and S terminals in the source will match two NMOS devices connected only by their S terminals in the target.

  Searching for non-exact connectivity matches takes significantly longer than searching for exact matches. You can use the options on the Non-Exact Connectivity Matches form to limit the search.

Related Topics

Generate Clones Form

Troubleshoot Cloning

Exact and Non-Exact Matches for Cloning

Types of Clones

The Generate Clones form lets you generate clones as:

- *Synchronized Family*

  If the clone source is selected from the layout cellview in which the clone target is generated, the clone source and the clone target are gathered in a common clone family. The generated clones, in this case, are synchronized to the other members in the clone family.

  See Generating Synchronous Clones.

  If the clone source layout cellview is different from the layout cellview from which the *Generate Clones* command is run, the clone source is not pushed inside a clone group. The clone targets, on the other hand, are pushed inside groups, but a synchronized clone family cannot be created.

- *Grouped Objects*

  If the clone source group is selected from the layout cellview in which the target clone groups are generated, the clone source objects and the clone target objects are pushed

© 2023 Cadence Design Systems, Inc.

inside a group. The groups created are neither synchronized clone groups nor gathered in a clone family.

To clone a group of components, create a cell containing all the devices and then clone the cell using the procedure in Clone Source Selection and Target Search and Generation.

This ensures that all the cells or similar circuit sections have the same layout parasitics associated with them.

Related Topics

Generating Clones as Grouped Objects (video)

Generating Clones as Free Objects (video)

Clone Source Selection and Target Search and Generation

Instance Cloning

This topic covers:

- Cloning Instances with no Schematic Counterpart
- Cloning Iterated Instances
- Cloning Instances with Inherited Connections
- Cloning Synchronous Clones
- Creating Modgens

Cloning Instances with no Schematic Counterpart

The *Generate Clones* command correctly handles layout instances with no schematic counterpart (clone dummy devices), including those that are part of a chain.

Layout XL considers a layout instance to be a dummy device if:

- The instance or its master has at least one property included in the list of *Properties used to ignore objects during check* on the Parameters of the Connectivity form.
- The instance has the `physOnly` attribute set.
- The instance component type is set as FILLER, STSUBCOUNT, or BOUNDARYCELL.

© 2023 Cadence Design Systems, Inc.

When Layout XL detects a dummy device among the objects in the clone source, it generates a similar instance with the same properties in the target structure. This new instance is automatically named `Ix`, where *x* is an appropriate integer value assigned by the database.

Cloning Iterated Instances

Because they all have the same master and same connectivity, cloning considers the individual bits of an iterated instance to be equivalent. Consequently, cloning cannot guarantee the same relative placements of devices in the clone source and the matching targets. If required, you can change the relative placement of instances in the target structure manually by changing the correspondence of the individual instances in the source and the target.

Cloning Instances with Inherited Connections

If a layout instance master terminal has an inherited net expression (an implicit inherited connection) for a global layout net, *Generate Clones* creates a physical-only layout terminal with no schematic equivalent. The physical-only terminal has the `physOnly` property set to `t` so that it can be correctly ignored by Layout XL functions such as *Update Components And Nets* and *Check Against Source* and by the digital applications further on in the design flow.

Cloning Synchronous Clones

You cannot create a synchronous clone that contains a synchronous clone. If the clone source includes groups of type `syncClone`, the groups are created as regular groups in the generated clone.

The only exception to this is if you are generating a synchronized family and the clone source comprises a single synchronous clone. In this case, matching target structures are generated as synchronous clones belonging to the same synchronized family.

Creating Modgens

You can create synchronous clones using Modgen figGroups as the source. The resulting clone family then comprises the existing Modgen figGroups in the clone source, and basic figGroups, called "pseudo-Modgen", in other clones.

A pseudo-Modgen figGroup cannot be edited. However, if you attempt to edit the figGroup, the *Edit Cloned Modgen* pop-up form displays, which suggests that the selected figGroup cannot be edited but the Modgen synchronized with the figGroup can be edited instead.

© 2023 Cadence Design Systems, Inc.

You can choose to hide the *Edit Cloned Modgen* pop-up form by setting the `disableVXLClonedModgenEditPopUp` environment variable to `t`. This results in the modgen synchronized with the pseudo-modgen figGroup to automatically get edited when *Editing In Place* a pseudo-modgen figGroup.

If you were to Edit In Place a synchronous clone and then create a Modgen inside the clone, the clone will be desynchronized. You should create the Modgen first and then use it as a clone source to create a Modgen inside the clone.

Related Topics

Ignoring Components with no Schematic Counterpart

Synchronous Clones and Dummy Devices

Editing a Target Structure Before Generation

disableVXLClonedModgenEditPopUp

Connectivity Form

Clone Source Selection and Target Search and Generation

Clone generation using the Generate Clones form comprises two main procedures—clone source selection and clone target search and generation.

Selecting the Clone Source

To clone a section of your design, you can choose the components to be cloned from:

- a layout cellview from which the *Generate Clones* form is opened
- another open layout cellview, which could be open in read-only mode
- a corresponding schematic cellview

The layout cellview used as the clone source must have a corresponding schematic connectivity reference available.

When choosing the clone source from a layout cellview, the steps for generating clones can vary slightly depending on whether or not the clone source exists in the layout cellview from which the *Generate Clones* command is run.

© 2023 Cadence Design Systems, Inc.

Scenario 1: Generating clones when the clone source is selected from the layout cellview from which the *Generate Clones* command is run, or from the schematic cellview used as the connectivity reference for this layout cellview*:*

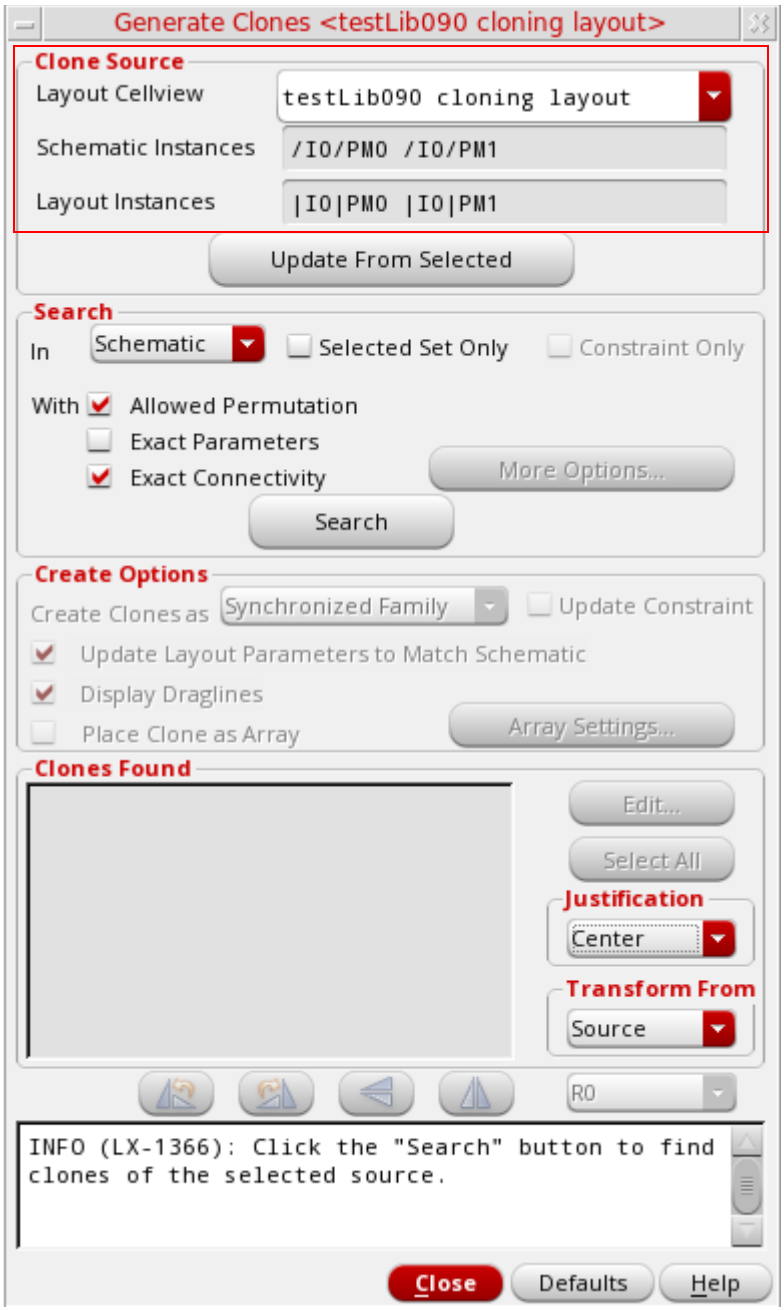
To select the clone source:

- In the layout canvas, select the individual component or components within an area that needs to be cloned and choose the *Connectivity – Generate – Clones* command.

  Alternatively, in the schematic cellview, select the individual component or components within an area that need to be cloned. Then, in the layout cellview in which the clone needs to be generated, choose the *Connectivity – Generate – Clones* command.

© 2023 Cadence Design Systems, Inc.

The Generate Clones form appears.

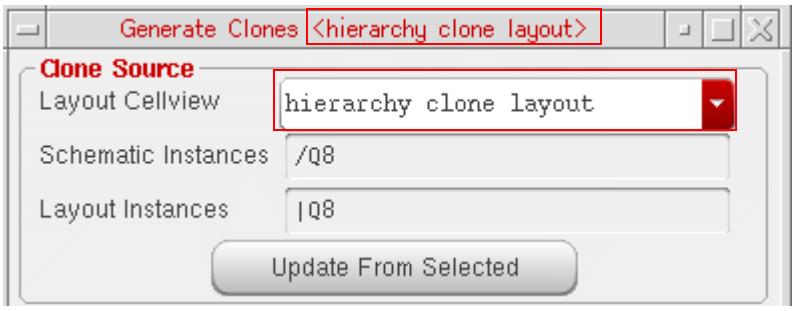


- The clone source selected from the schematic cellview is considered valid only when it is correctly bound.
- If the structure you want to clone is a partially implemented mfactored component, you must select it in the layout canvas.

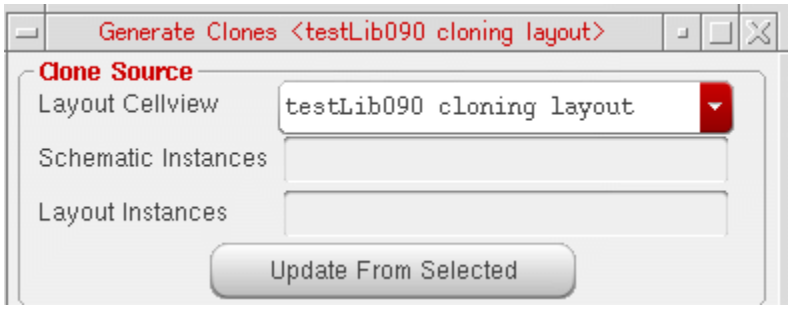
© 2023 Cadence Design Systems, Inc.

If the selected instances and shapes form a valid clone source, the *Clone Source* fields are already filled in when the form opens, as shown in the figure above.

If the clone source is selected from the layout cellview in which the target clone needs to be generated, the name of the layout cellview as displayed in the *Clone Source* group box is the same as that of the current layout cellview. This is illustrated in the figure below.



Instead of opening the *Generate Clones* form with the clone source already selected, you can also open the form with nothing selected. After the form opens, as shown in the figure below, select the clone source in the layout or schematic canvas, and click *Update From Selected*.



If the selected instances and shapes form a valid clone source, the instance names are displayed in the clone source fields. You can use the same method to change the clone source without closing the form.

© 2023 Cadence Design Systems, Inc.

Scenario 2: Generating clones when the clone source is selected from one layout cellview and the clones need to be generated in another layout cellview.

To select the clone source:

a. Choose the *Connectivity – Generate – Clones* command to open the *Generate Clones* form in the layout cellview in which the clones need to be generated.

b. In the *Layout Cellview* drop-down list, select the layout cellview to use as the source cellview.

The *Layout Cellview* drop-down lists all the layout cellviews open in Layout XL that have a corresponding connectivity reference available, including any cellviews that are open in read-only mode.s

The source cellview that you select is automatically brought up. In the source cellview, select the individual component or select the components within an area and choose the *Update From Selected* command in the *Generate Clones* form.

The layout components you selected as the clone source and their schematic counterparts are displayed in the appropriate fields in the *Clone Source* group box.

Related Topics

Generate Clones Form

Cloning between Multiple Cellviews

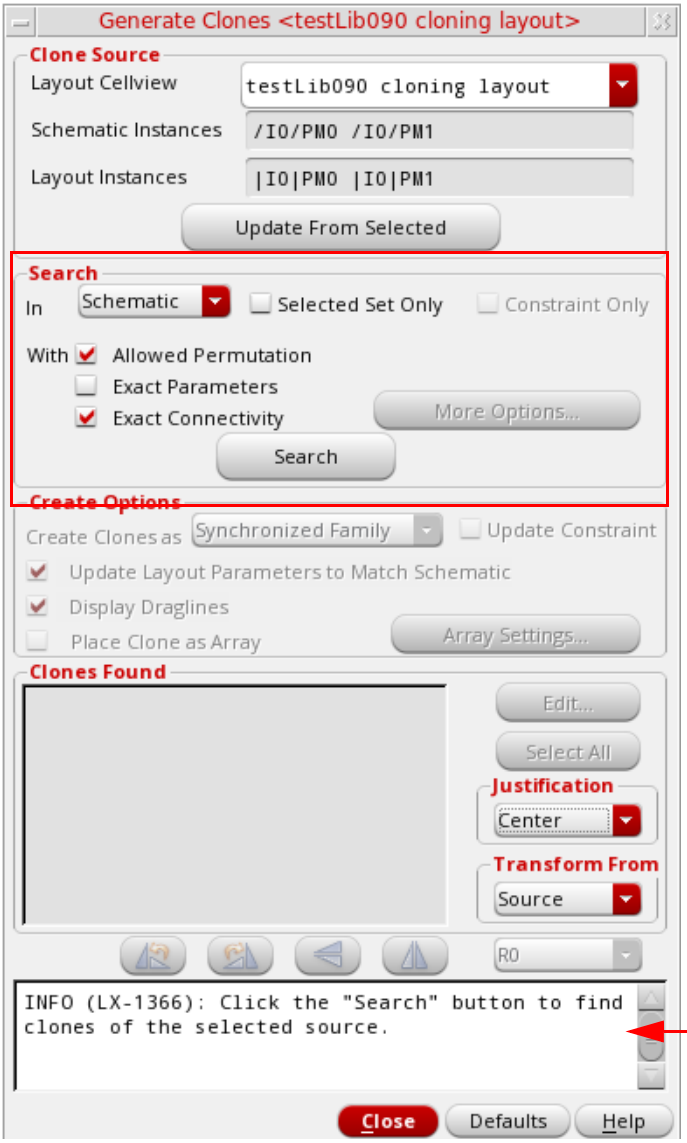
© 2023 Cadence Design Systems, Inc.

Searching for and Generating the Clone Target

After you have selected a suitable clone source, you can use the *Generate Clones* form to search for and identify appropriate clones to generate the target clone structure.

To search for and generate the clone target:

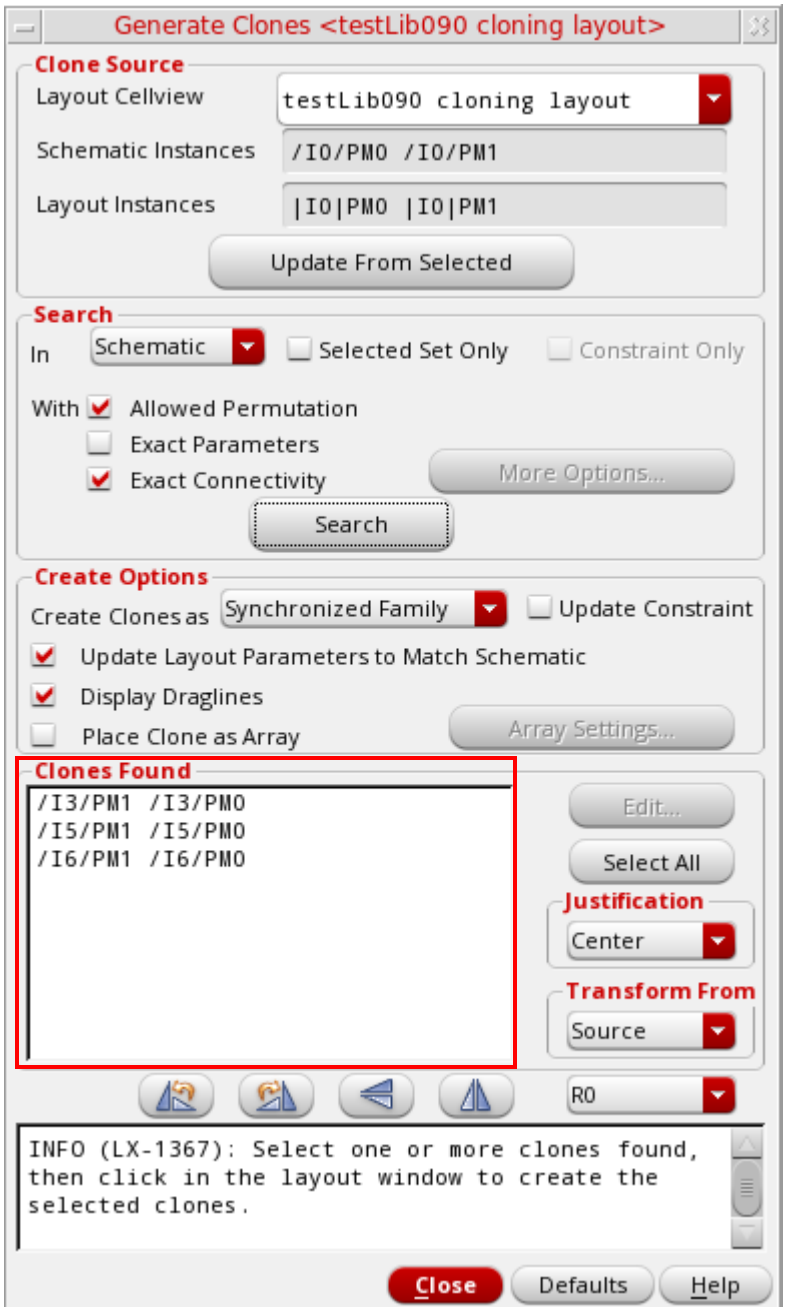
1. In the *Search* group box, choose to search the target clone in the schematic or layout canvas, as appropriate.



2. Choose whether to search for the target clone in the entire *Schematic* or *Layout* or within the *Selected Set Only*.

© 2023 Cadence Design Systems, Inc.

3. Choose the search criteria *Allowed Permutation*, *Exact Parameters*, and *Exact Connectivity,* as appropriate.
4. Click *Search* to find the matching target structures in the design.



Depending on the design situation.

© 2023 Cadence Design Systems, Inc.

- If the search is run in the schematic and *Selected Set Only* is deselected, the clone search ignores any devices that are already implemented in the layout.
- If the search is run in the schematic and *Selected Set Only* is selected, the clone search ignores any devices that are not selected in the schematic or that are already implemented in the layout.
- If the search is run in the layout and *Selected Set Only* is deselected, the clone search ignores any devices that are not yet implemented in the layout. In addition, the search ignores any devices that are implemented in the layout but placed inside the PR Boundary.
- If the search is run in the layout and *Selected Set Only* is selected, the clone search ignores any instances that are not selected in the layout. Selected instances placed inside the PR Boundary are not ignored.
- If the search is run with *Constraints Only* selected, the clone search is confined to the other Clone Defs of the source Clone Family constraint.

  A Clone Def is an ordered list of bound instances contained in a clone.

Irrespective of the design situation, as listed above, devices that are part of a modgen constraint in the schematic are ignored during the clone search.

The matching targets are added to the *Clones Found* list at the bottom of the form.

If the search is likely to take a long time, you will see a pop-up inviting you to do one of the following.

- Click *OK* to continue the search.
- Click *Cancel* to return to the Generate Clones form where you can tighten the criteria either by reducing the search area or by requesting exact matches only.
- Press `Ctrl-c` to stop the search. The system may take a few minutes to respond. The matches that were already found are not removed from the list of *Clones Found*.

5. Choose from *Free Objects, Grouped Objects*, and *Synchronized Family* from the *Create Clones as* pull-down list and select the other *Create Options*, as required.

- If you choose to display draglines, see Changing the Appearance of Draglines f.
- If you choose to place the clones as an array, see Clone Placement as an Array

6. Select one or more devices from the list of *Clones Found* and move your cursor into the layout canvas.

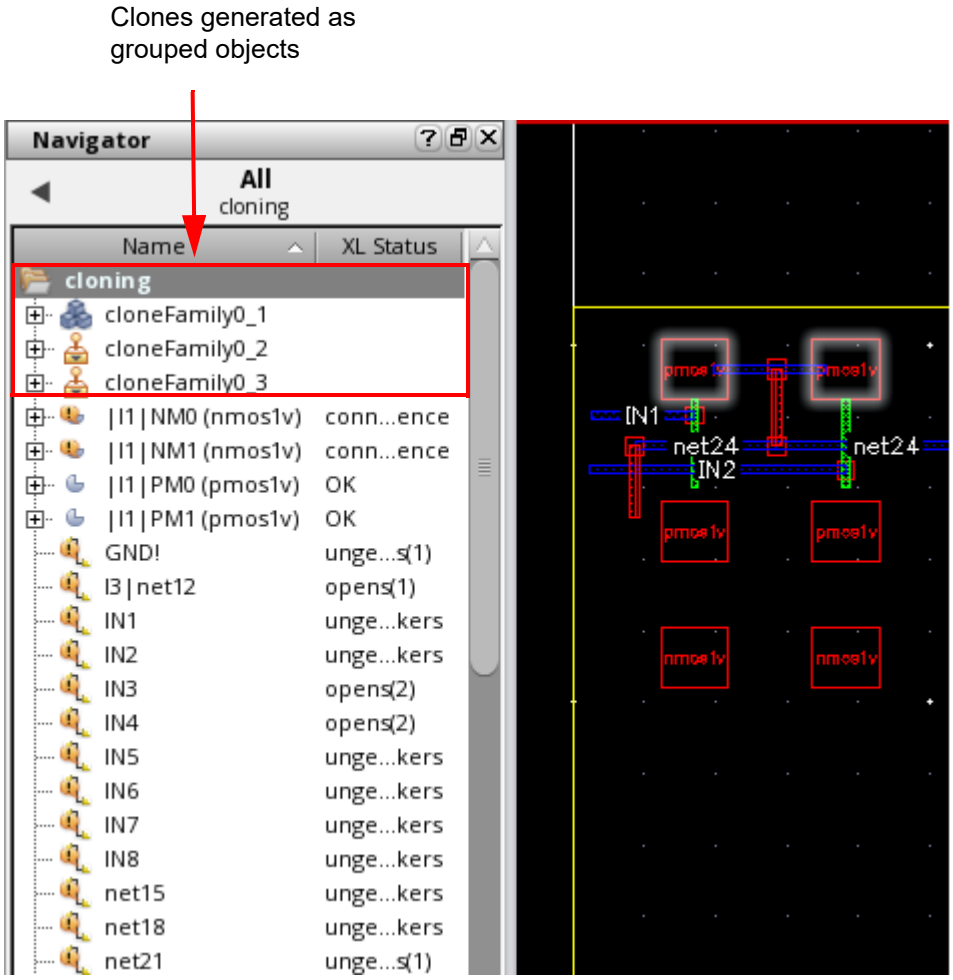
An image of the clones to be generated follows the cursor.

© 2023 Cadence Design Systems, Inc.

7. Select the orientation options you want (*Rotate*, *Flip Horizontal*, *Flip Vertical*) and click at the required location in the layout canvas to place the first clone.

   The target clones are generated in the layout canvas as grouped objects and you can verify this by using the Navigator assistant, as shown in the figure below.

   You can also view the object type of the generated cloneFamily using the Property Editor.



   If you selected more than one target structure from the list and you chose not to place the clones as an array, click again to place the second clone, and so on.

Related Topics

Generate Clones Form

Exact and Non-Exact Matches for Cloning

© 2023 Cadence Design Systems, Inc.

Editing a Target Structure Before Generation

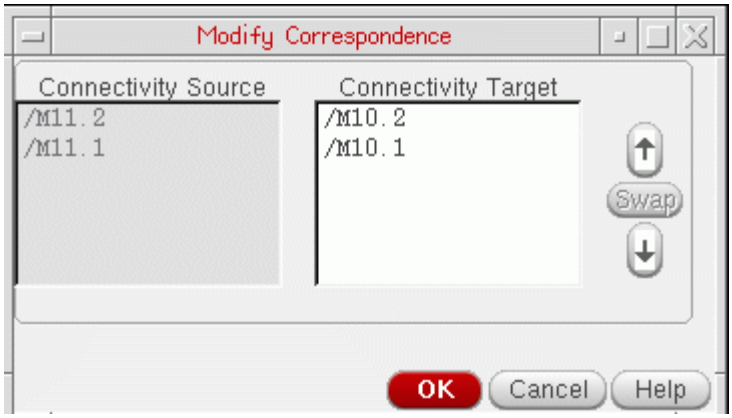
When searching for matching target structures, Layout XL does not consider the relative placement of instances in the source structure as long as swapping these instances does not change the source topology. Additionally, cloning considers the individual bits of an iterated instance to be equivalent because they all have the same master and same connectivity. For these reasons, the relative placement of instances in a target structure can differ significantly from that in the clone source.

The Modify Correspondence form lets you change the relative placement of instances in the target structure manually by changing the correspondence of the individual instances in the source and the target. When you modify the correspondence manually, Layout XL checks that the connectivity of the modified target structure remains valid before allowing you to generate the clone.

To modify the correspondence of source and target instances:

1. Create a list of matching target structures using the steps.
2. Select a structure from the list and click *Edit*.

   The Modify Correspondence form is displayed.



   The instances in the source structure are listed under *Connectivity Source*. The list is read-only. The corresponding instances in the target structure are listed under *Connectivity Target*. When you select an instance in the target structure, the corresponding instance in the source structure is highlighted.
3. To change the correspondence of a single instance, select the instance in the *Connectivity Target* list and use the arrow buttons to move it up or down in the list.
4. To swap the correspondence of two instances, select the instances and click *Swap*.
5. When you have finished modifying the correspondence, click *OK*.

© 2023 Cadence Design Systems, Inc.

Before accepting the new correspondence, Layout XL checks that the connectivity of the target structure remains valid.

6. Generate the clones using the steps described in Clone Source Selection and Target Search and Generation.

Related Topics

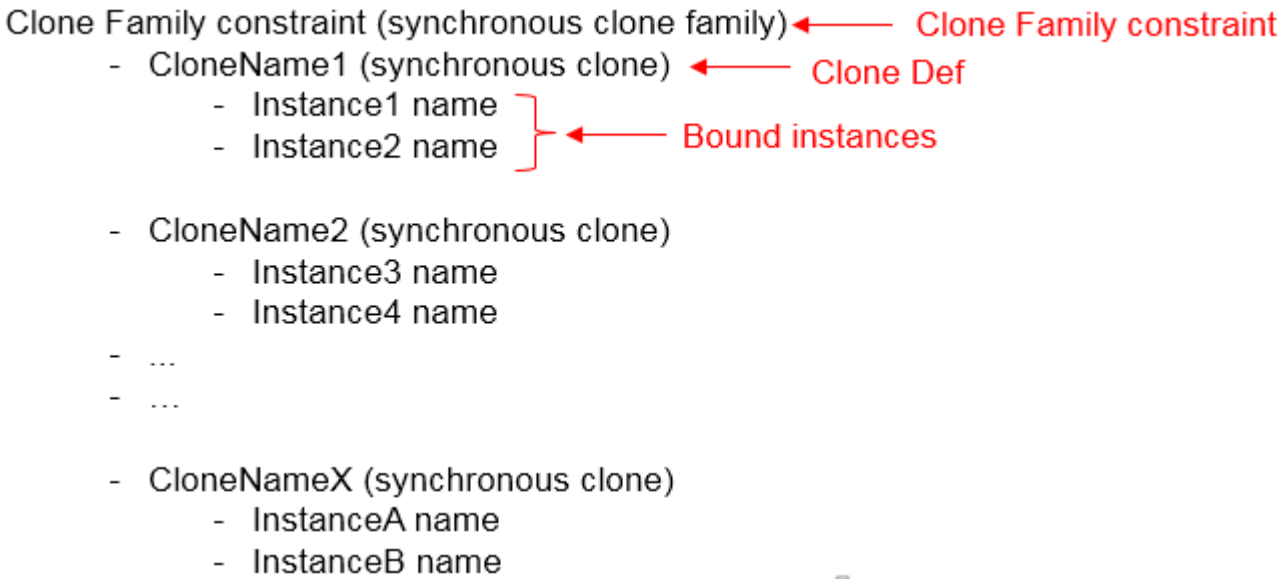
Modify Correspondence Form

Clone Source Selection and Target Search and Generation

Instance Cloning Using a Clone Family Constraint

A Clone Family constraint represents a synchronous clone family, which consists of two or more Clone Defs, where a Clone Def is an ordered list of bound instances that are contained in a clone.

In general, a Clone Family constraint can be represented by the illustration below:



This topic covers:

- Using the Constraint Manager to Generate the Clone Family Constraint
- Using the Generate Clones Form to Generate the Clone Family Constraint

© 2023 Cadence Design Systems, Inc.

- Using Iterated Instances to Generate the Clone Family Constraint
- Using Mfactored Instances to Generate the Clone Family Constraint

If a Clone Family constraint already exists in the schematic, the constraint is transferred to the layout Constraint Manager when the *Generate All From Source* command is run. If possible, Layout XL creates clones using the devices generated by the command. If the instance masters are identical for the generated clones, Layout XL creates synchronous clones.

Using the Constraint Manager to Generate the Clone Family Constraint

To add a Clone Family constraint using the Constraint Manager:

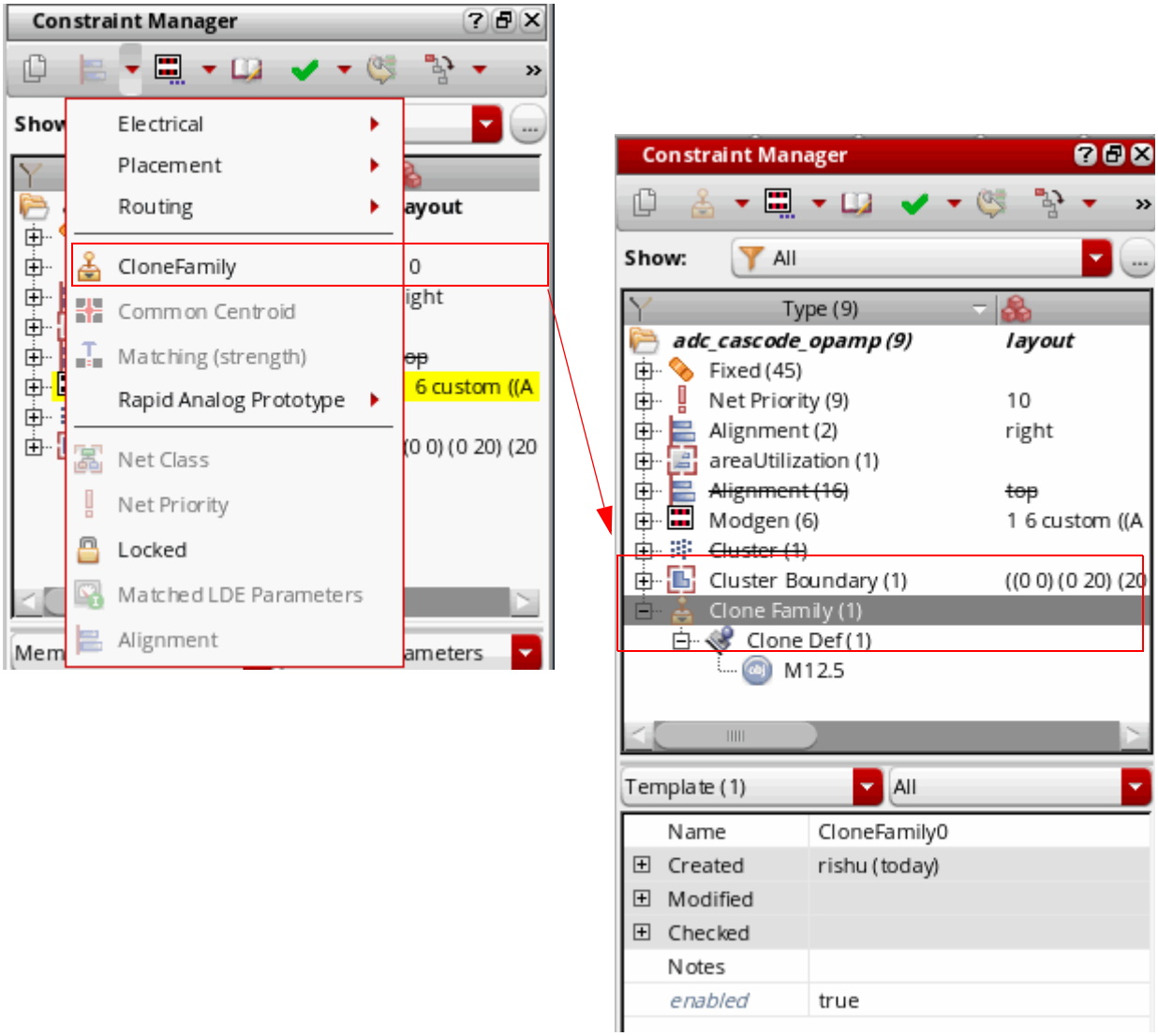
➥ In the schematic or layout canvas, select the instances to be added to the constraint and choose the *cloneFamily* option in the *Constraint Creation* menu in Constraint Manager.

A new *Clone Family* constraint is added to the Constraint Manager and the selected instances are added to the constraint as members of the Clone Def associated with the constraint.

- If the same instances end up being in two Clone Family constraints, the new Clone Family constraint is not created. For example, if a Clone Family constraint comprises instances `IM0` and `IM1`, Layout XL does not allow creating another Clone Family constraint that will contain these two instances.

© 2023 Cadence Design Systems, Inc.

- If the instances selected for creating a Clone Family constraint contain any unbound instances, the unbound instances are added to the Clone Def that is generated.



You can also use the Constraint Manager to add new clones to an existing clone family. To do this, use the *Add Clone* option in the shortcut menu of the selected Clone Family constraint.

To delete an existing Clone Family constraint or a Clone Def, select the *Delete* option in the associated shortcut menu. Deleting a clone from a Clone Family constraint does not desynchronize the clone family.

© 2023 Cadence Design Systems, Inc.

Using the Generate Clones Form to Generate the Clone Family Constraint

In the layout view, the Generate Clones form provides the following options to generate and update the Clone Family constraint:

- *Search – Constraint Only*

  Confines the clone target search to the Clone Family constraint within the selected search area—schematic or layout. See Constraint Only.

  Environment variable: cloningSearchInConstraintOnly

- *Create – Update Constraint*

  Depending on what the clone source includes, the option creates a new Clone Family constraint, adds a new Clone Def to an existing constraint, or leaves the existing constraint unaltered. See Update Constraint.

  Environment variable: cloningUpdateConstraint

Constraint Only

Constraint Only searches for matching target structures only within the other Clone Defs that belong to the same Clone Family constraint as the source. The option is enabled only when the clone source belongs to a Clone Def. The Clone Defs found during the search are ordered in the same manner as they exist in the constraint.

If *Constraint Only* is deselected, the clone target search is first conducted within the source Clone Family constraint, followed by a search within the remaining clone target search space.

If the clone source devices are not connected together, but belong to a Clone Def, the *Constraint Only* option is automatically selected, but disabled to prevent any manual change in state. In this case, the regular clone search cannot be performed because that requires the clone source devices to be connected. Therefore, only other Clone Defs from the Clone Family constraint will be listed as clone targets.

Listed below are some conditions that determine whether a clone source can be considered as valid and what determines if a Clone Def can be found as a clone target:

- If a Clone Family constraint is already present in the layout Constraint Manager, for the clone source to be considered valid, if an instance from the clone source belongs to a Clone Def, all the bound instances in the clone source should exactly match the Clone Def instances. The clone source is rejected if it has either less or more bound instances compared to those in the source Clone Def. In addition, the clone source can include shapes and/or ignored instances.

© 2023 Cadence Design Systems, Inc.

- If the clone source is a perfect Clone Def, instances that are members of another Clone Family constraint are not considered as clone targets.

Let us now consider some case scenarios that illustrate the conditions in which a clone source is considered valid. Let us assume that the Clone Def for these cases comprises instances `I1`, `I2`, and `I3`.

Scenario 1: Clone source consists of instances `I1`, `I2`, and `I3`, and the instances are interconnected.

The clone source is considered valid as it exactly matches the Clone Def. The clone search will find other Clone Defs from the same clone family but not the Clone Defs from other clone families.

Scenario 2: Clone source consists of instances `I1`, `I2`, and `I3`, but the instances are not interconnected.

Search option *Constraint Only* is automatically selected and grayed out. Only the instances in the other Clone Defs belonging to the same Clone Family constraint are considered as candidates.

In this case, the *Exact Connectivity* option is deselected and grayed out because an exact connectivity search cannot be triggered on instances that are not interconnected.

Scenario 3: Clone source consists of instances `I1`, `I2`, `I3`, and `I4` – the clone source contains an additional instance compared to the Clone Def.

In this case, the clone source is rejected and a message displays suggesting that the clone source must be an exact match of the Clone Def for it to be considered valid.

To use the clone source anyway, remove or disable the associated Clone Family constraint in the Constraint Manager and trigger the clone search again.

Scenario 4: Clone source consists of `I1` and `I2` – the clone source has one less instance compared to the Clone Def.

In this case, the clone source is rejected and a message displays suggesting that the clone source must be an exact match of the Clone Def for it to be considered valid.

To use the clone source anyway, remove or disable the associated Clone Family constraint in the Constraint Manager and trigger the clone search again.

© 2023 Cadence Design Systems, Inc.

Update Constraint

If the Clone Family constraint is not already present for the selected instances, *Update Constraint* creates a new Clone Family constraint and the associated Clone Def in the Constraint Manager, based on the selected synchronous clone figGroups. If a Clone Family constraint already exists, *Update Constraint* either leaves the constraint as is, or adds a new Clone Def, if appropriate.

The option is available even if the clone source does not belong to a Clone Def.

By default, *Update Constraint* is deselected. Therefore, no constraints are created or updated unless the option is selected.

If *Update Constraint* is selected and:

- The clone source does not belong to a Clone Def, a new constraint is created in the layout Constraint Manager that comprises the clone source and the clone target(s).
- The clone source belongs to a Clone Def, no update is made to the Clone Family constraint, if the clone found is already part of a Clone Family constraint. Also, no new Clone Family constraint is created. But, if the clone found does not belong to a Clone Family constraint, the constraint is updated. A new Clone Def is added to the clone source Clone Family constraint in the layout Constraint Manager.

Using Iterated Instances to Generate the Clone Family Constraint

When the instances selected for generating a Clone Family constraint are iterated instances, the Constraint Member Select form displays to confirm which instances to use for the constraint. If the selected instances belong to a Clone Def, the instances in the form are sorted according to the iterated instance order. For example, in a Clone Def comprising instances `IM14.1` to `IM14.4`, IM14.1 is the first instance in the Clone Def.

Using Mfactored Instances to Generate the Clone Family Constraint

An mfactored instance is considered a single instance in the schematic view. Therefore, the instances are not split in the schematic Constraint Manager.

Related Topics

cloneFamily

Constraint Manager

© 2023 Cadence Design Systems, Inc.

Generate Clones Form

Generating All Components from Source

Clone Constraint (RAK)

Cloning between Multiple Cellviews

You can generate a clone in a layout different from the one in which you selected the clone source.

To clone components across multiple layout cellviews:

1. Open both the source and the target layout cellviews in Layout XL.

   The layout cellview that you want to use as the clone source can be opened in read-only mode.

2. In the target layout cellview, choose the *Connectivity – Generate – Clones* command to open the *Generate Clones* form.

3. In the *Layout Cellview* drop-down list, select the layout cellview to use as the source cellview.

   The *Layout Cellview* drop-down list shows all the layout cellviews open in Layout XL that have a corresponding connectivity reference available, including any cellviews that are open in read-only mode.

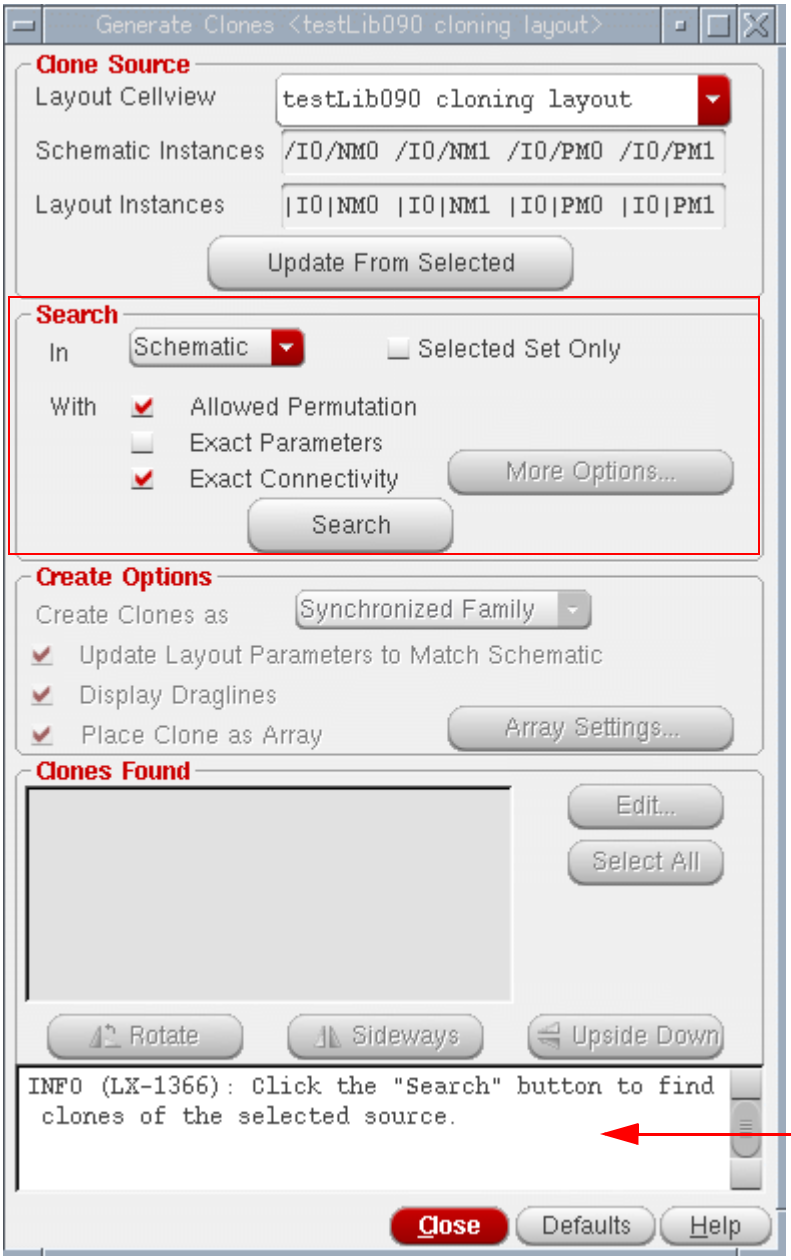
   The source cellview that you select is automatically brought up.

4. In the source cellview, select the individual component or select the components within an area and choose the *Update From Selected* command in the *Generate Clones* form.

   The layout components you selected as the clone source and their schematic counterparts are displayed in the appropriate fields in the *Clone Source* group box.

© 2023 Cadence Design Systems, Inc.

5. In the *Search* group box, choose whether to search for the target clone in the entire *Schematic* or *Layout* or within the *Selected Set Only*.



6. Choose the search criteria *Allowed Permutation*, *Exact Parameters*, and *Exact Connectivity,* as appropriate.
7. Click *Search* to find the matching target structures in the design.

   Depending on the design situation.

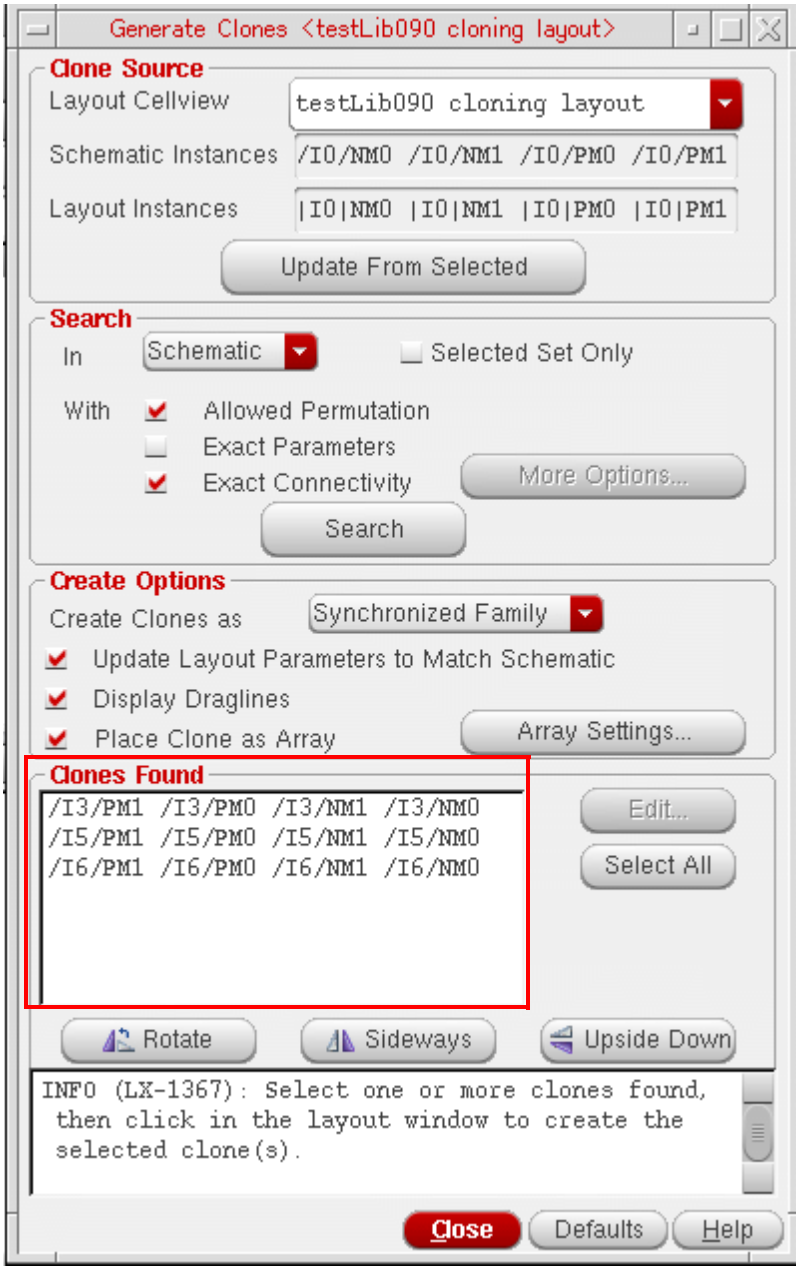
© 2023 Cadence Design Systems, Inc.

- If the search is run in the schematic and *Selected Set Only* is deselected, the clone search ignores any devices that are already implemented in the layout.
- If the search is run in the schematic and *Selected Set Only* is selected, the clone search ignores any devices that are not selected in the schematic or that are already implemented in the layout.
- If the search is run in the layout and *Selected Set Only* is deselected, the clone search ignores any devices that are not yet implemented in the layout. In addition, the search ignores any devices that are implemented in the layout but placed inside the PR Boundary.
- If the search is run in the layout and *Selected Set Only* is selected, the clone search ignores any instances that are not selected in the layout. Selected instances placed inside the PR Boundary are not ignored.

  Irrespective of the design situation, as listed above, devices that are part of a modgen constraint in the schematic are ignored during the clone search.

© 2023 Cadence Design Systems, Inc.

The matching targets are added to the *Clones Found* list at the bottom of the form, as shown in the figure below.



If the search is likely to take a long time, you will see a pop-up inviting you to do one of the following.

- Click *OK* to continue the search.
- Click *Cancel* to return to the Generate Clones form where you can tighten the criteria either by reducing the search area or by requesting only exact matches.

© 2023 Cadence Design Systems, Inc.

- Press `Ctrl-c` to stop the search. The system may take a few minutes to respond. The matches that were already found are not removed from the list of *Clones Found*.

8. Choose from *Free Objects, Grouped Objects*, and *Synchronized Family* from the *Create Clones as* pull-down list and select the other *Create Options*, as appropriate.

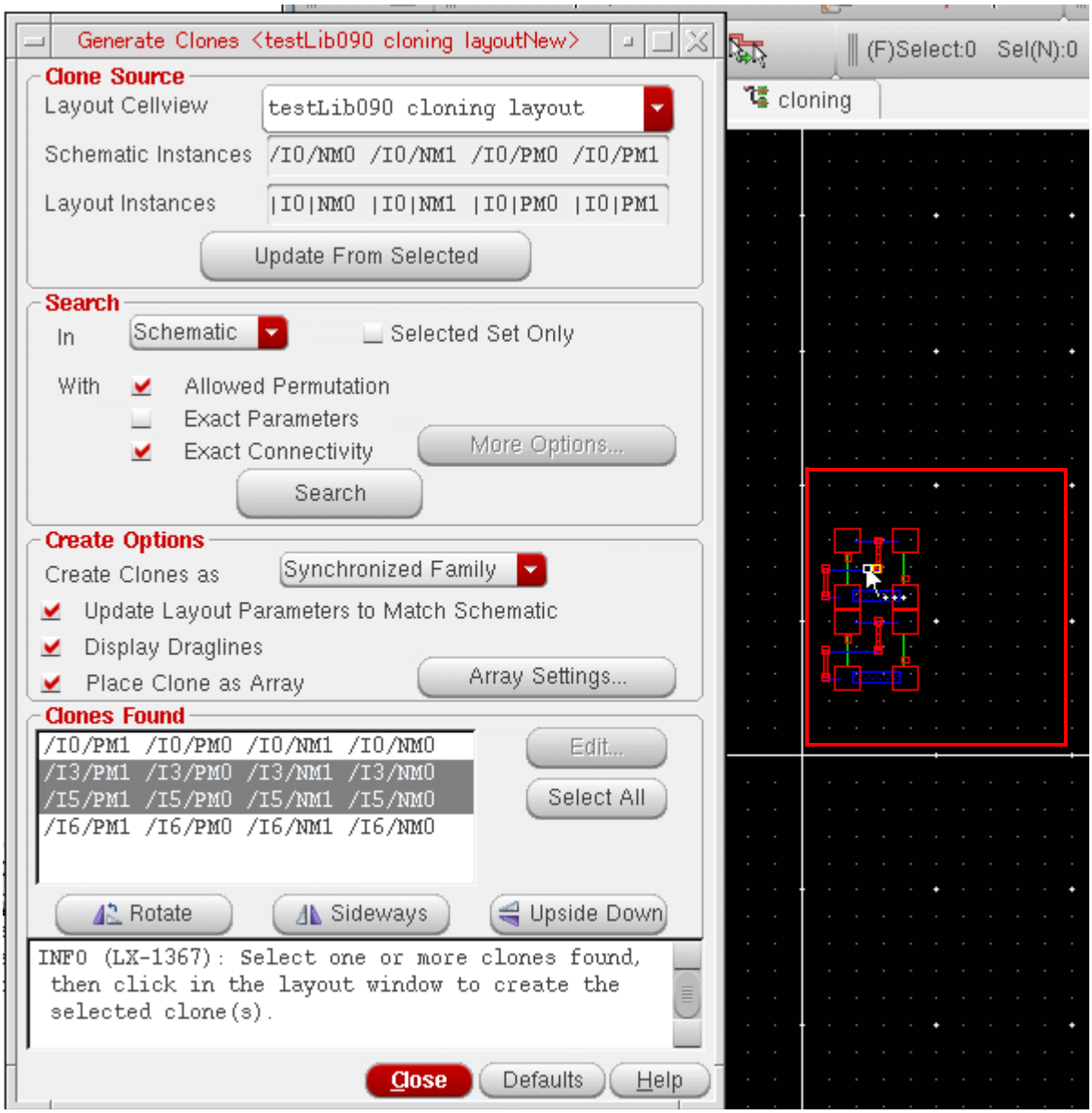
If you choose to display draglines, see Changing the Appearance of Draglines.

If you choose to place the clones as an array, see Clone Placement as an Array.

9. Select one or more devices from the list of *Clones Found* and move your cursor into the layout canvas.

© 2023 Cadence Design Systems, Inc.

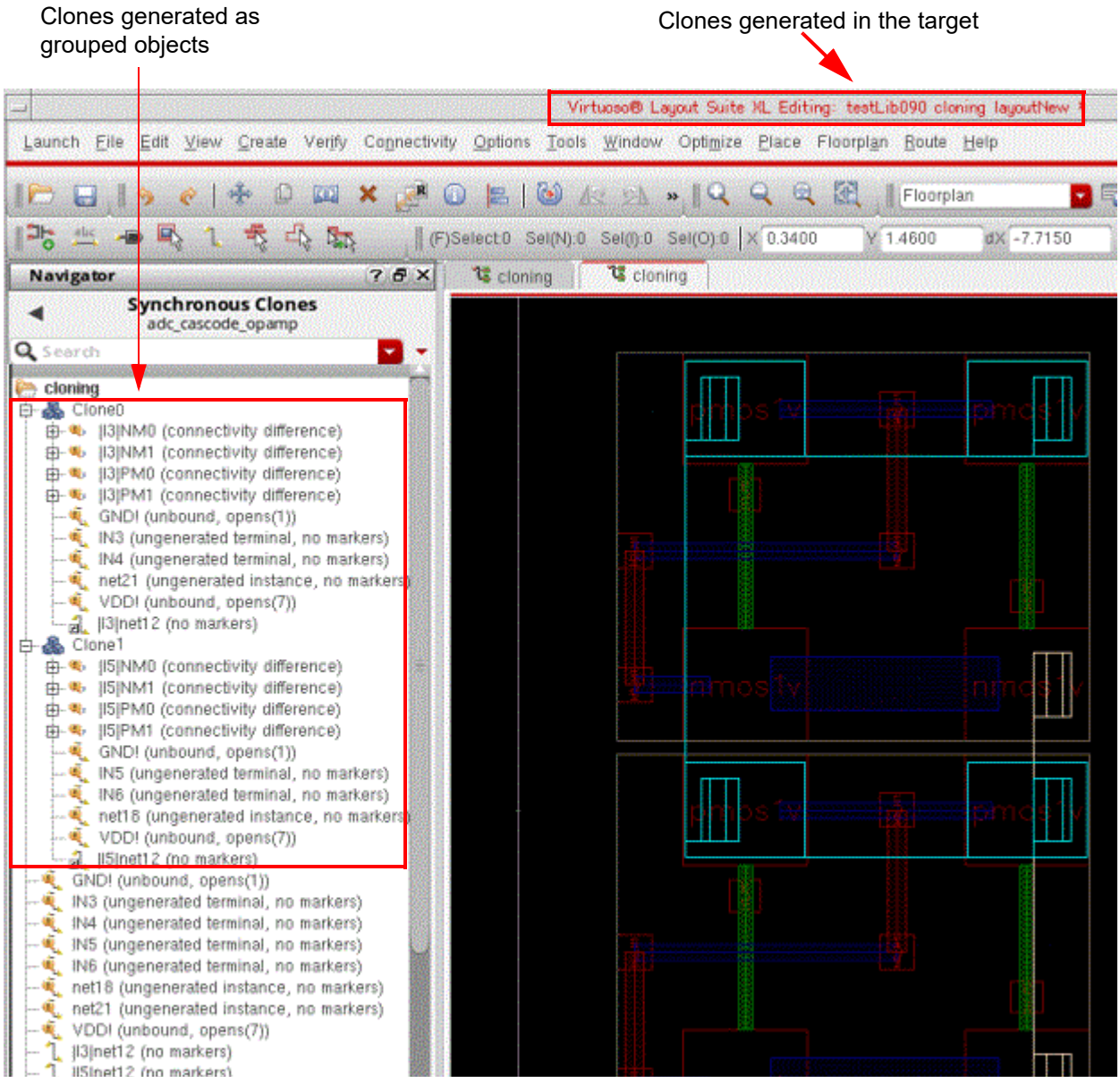
An image of the clones to be generated follows your cursor, as shown in the figure below.



10. Select the orientation options you want (*Rotate*, *Flip Horizontal*, *Flip Vertical*) and click in the layout canvas to place the clone at the required location.

© 2023 Cadence Design Systems, Inc.

The target clones are generated in the target layout canvas as grouped objects and you can verify this by using the Navigator assistant, as shown in the figure below.



Related Topics

cloneFamily

Constraint Manager

Generate Clones Form

© 2023 Cadence Design Systems, Inc.

Generating All Components from Source

Exact and Non-Exact Matches for Cloning

Clone Constraint (RAK)

Mfactored Component Cloning

Layout XL handles one-to-many relationships defined using the `mfactor` property on a schematic instance.

For example, if you have a component with an mfactor of 100, you can choose ten of these components, wire them together in the layout canvas, and then clone the resulting structure ten times to generate all the mfactored instances.

You can clone mfactored components both partially and in their entirety across multiple hierarchy levels, including source structures that include folded and abutted devices. The following limitations apply.

- The selected set must not include folded mfactored devices generated in a previous release; these must be regenerated in the current release.
- When cloning a folded instance, all the folds of the regular device or mfactored device part must be included in the clone source selection.
- When cloning mfactored devices that are partially implemented in the layout, you must select the clone source from the layout canvas.

The examples that follow illustrate what the cloning functionality considers to be valid targets for a number of different mfactored source structures.

- Example 1: Mfactored component instance
- Example 2: Mfactored component instance partially generated in the layout
- Example 3: One or more of multiple mfactored instances partially generated in the layout
- Example 4: Mfactored component instance with other component instances
- Example 5: One or more of an mfactored component instance partially generated in the layout along with other component instances
- Example 6: Although clone source contains only non-mfactored devices, clone target can use mfactored devices
- Example 7: Clone target uses parts of a device with an mfactor value greater than that on the clone source

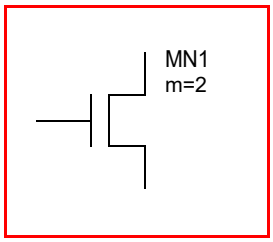
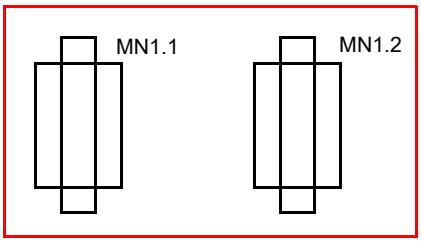
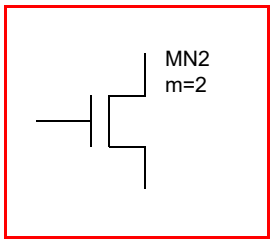
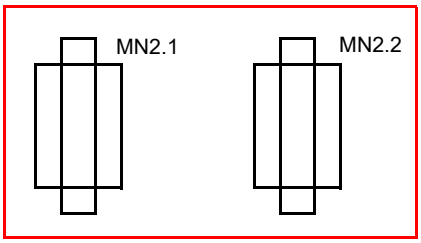
© 2023 Cadence Design Systems, Inc.

- Example 8: No valid clone targets

Example 1: Mfactored component instance

If the clone source is an mfactored component instance (for example, an NMOS instance with m=2), then the following are considered valid clone targets.

- Another mfactored component instance; for example, another nmos instance also with m=2 (see example below).
- Two non-mfactored NMOS instances connected in parallel.

	Schematic	Layout
Clone Source		
Clone Target		

Example 2: Mfactored component instance partially generated in the layout

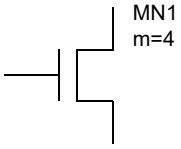
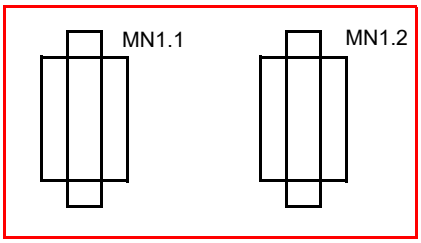
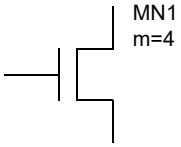
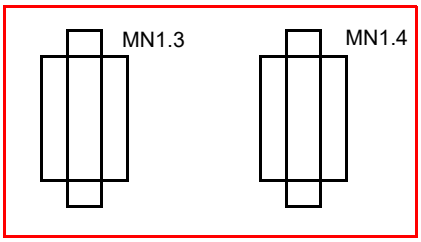
If the clone source is one or more parts of an mfactored component instance that is partially generated in the layout, then the following are considered valid clone targets.

- The remaining components in the mfactored instance (see example below).
- The appropriate number of non-mfactored NMOS instances connected in parallel.

© 2023 Cadence Design Systems, Inc.

Important

You must select the clone source in the layout canvas because of the one-to-many relationship.

	Schematic	Layout
Clone Source		
Clone Target		

Example 3: One or more of multiple mfactored instances partially generated in the layout

If the clone source is one or more of multiple mfactored component instances that are partially generated in the layout, then the following are considered valid clone targets.

- The remaining mfactored component instances that follow the same pattern (see example below).
- The appropriate number of non-mfactored NMOS instances connected in parallel.

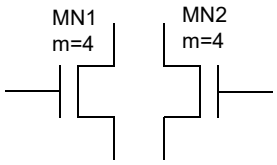
Important

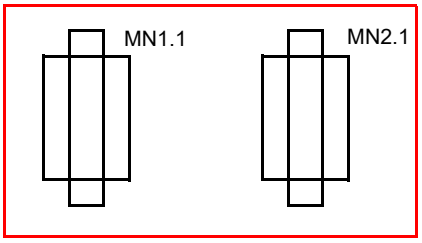
You must select the clone source in the layout canvas because of the one-to-many relationship.

	Schematic	Layout

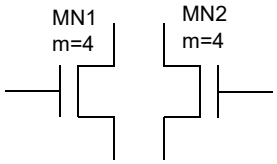
© 2023 Cadence Design Systems, Inc.

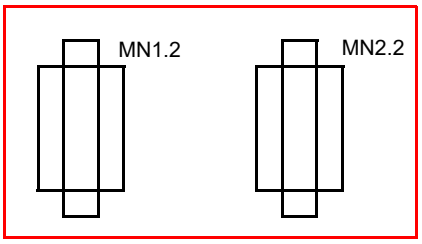
Clone Source

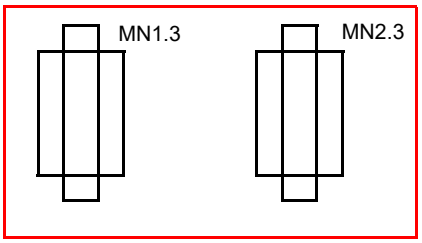




Clone Target







And other such targets

Example 4: Mfactored component instance with other component instances

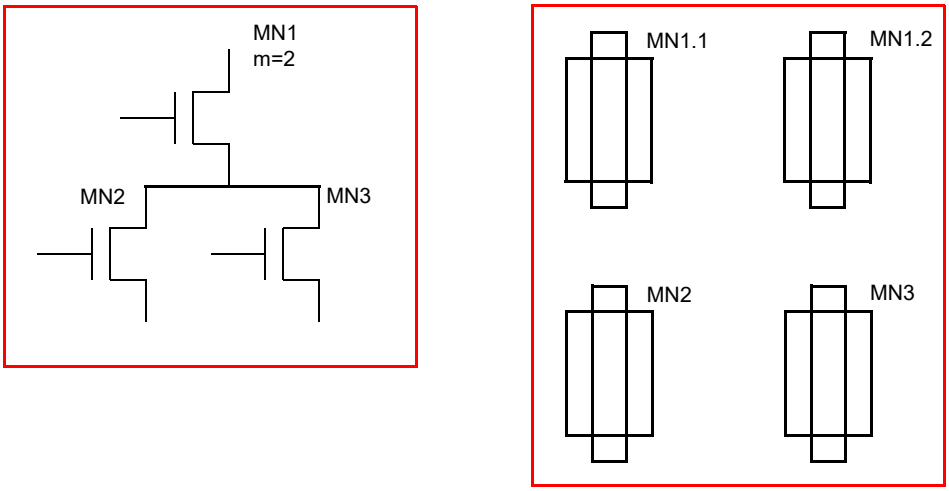
If the clone source is an mfactored component instance along with other component instances, then the following are considered valid clone sources.

- Any other identical configuration with mfactored component instances which have the same mfactor value.
- The appropriate number of non-mfactored NMOS instances connected in parallel.

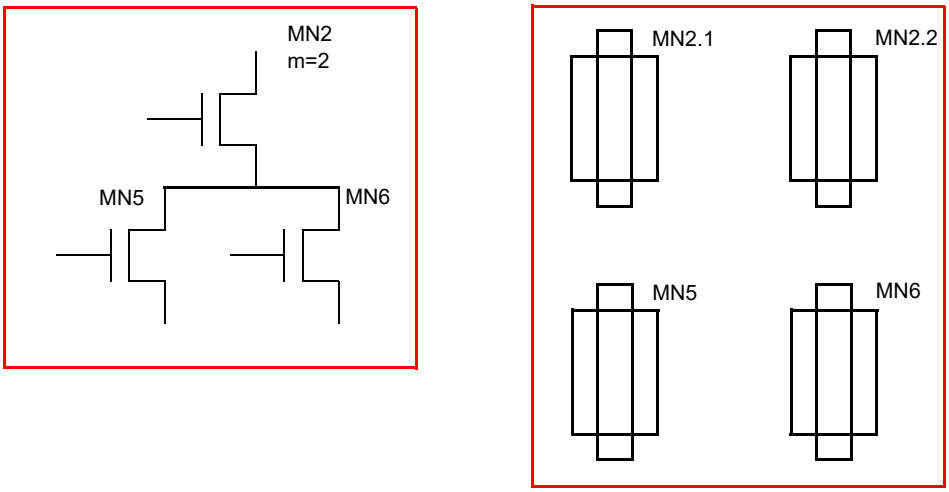
Schematic	Layout

© 2023 Cadence Design Systems, Inc.

Clone Source



Clone Target



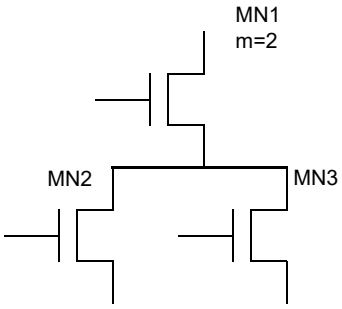
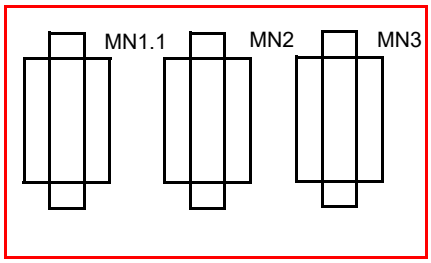
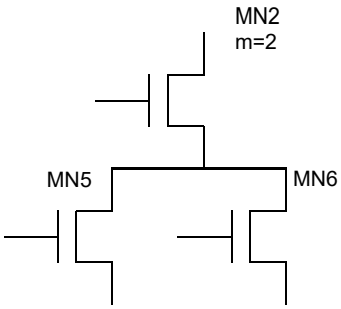
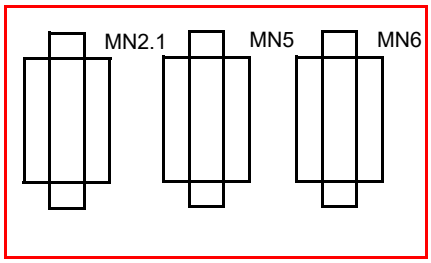
Example 5: One or more of an mfactored component instance partially generated in the layout along with other component instances

If the clone source is one or more of an mfactored component instance that is partially generated in the layout along with other component instances, then any configuration with the same topology but which only partially implements the mfactored components in the same way as the clone source is considered a valid clone target.

© 2023 Cadence Design Systems, Inc.

Important

You must select the clone source in the layout canvas because of the one-to-many relationship.

	Schematic	Layout
Clone Source		
Clone Target		

Example 6: Although clone source contains only non-mfactored devices, clone target can use mfactored devices

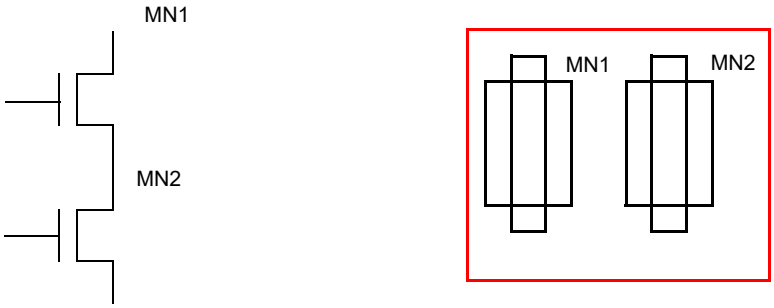
If the clone source is two non-mfactored NMOS instances connected in parallel, then the following can be considered valid clone targets.

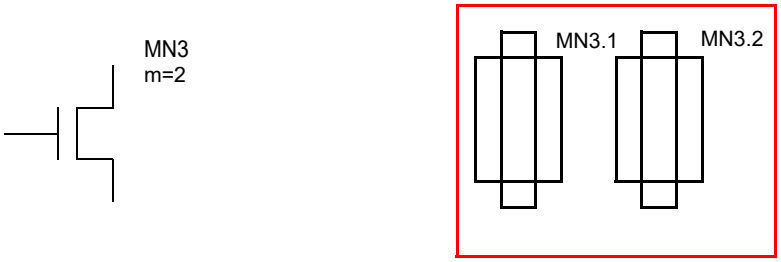
- An mfactored NMOS instance with m=2
- Parts of an mfactored NMOS mfactored instance with m>2

	Schematic	Layout

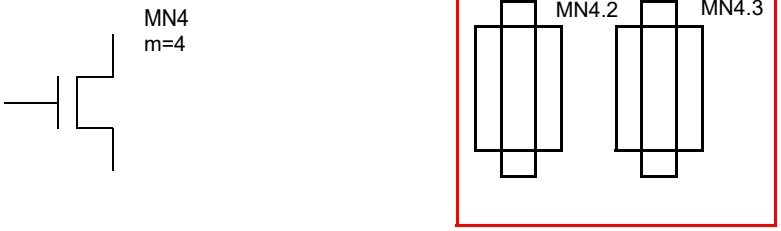
© 2023 Cadence Design Systems, Inc.

Clone Source





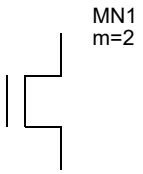
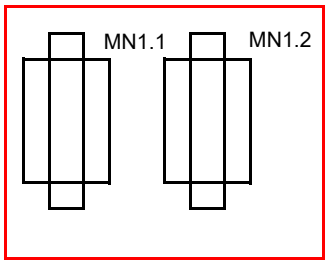
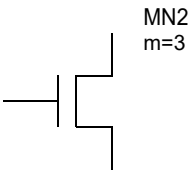
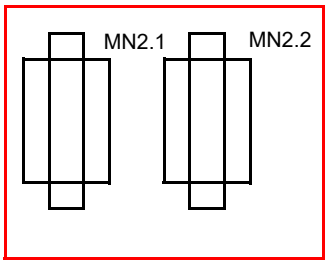
Clone Targets



© 2023 Cadence Design Systems, Inc.

Example 7: Clone target uses parts of a device with an mfactor value greater than that on the clone source

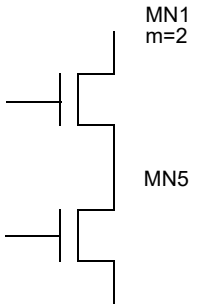
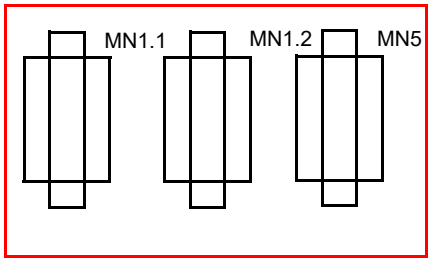
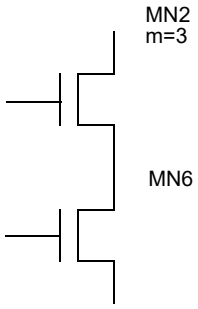
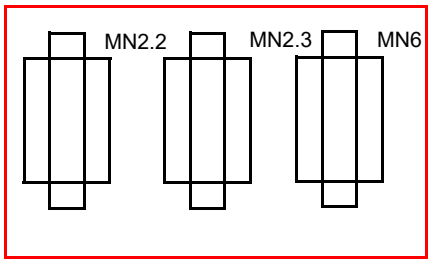
If the clone source is an mfactored component (m=m1), which is fully realized in the layout, then the appropriate number of parts of an mfactored component (m>m1), is considered a valid clone target.

	Schematic	Layout
Clone Source		
Clone Targets		

The same principle applies when devices with different mfactor values are combined with other devices.

	Schematic	Layout

© 2023 Cadence Design Systems, Inc.

Clone Source		
Clone Targets		

Example 8: No valid clone targets

The clone source is a fully-implemented mfactored component (m=m1) and the schematic has the following other components that are not yet realized in the layout:

1. Another mfactored component with the same master (where m<m1).
2. Another non-mfactored component with the same master.

For example,

The clone source is = MN1.1, MN1.2, and MN1.3 (m=3 in schematic)

The schematic also contains MN2.1 and MN2.2 (m=2 in schematic) and a non-mfactored instance MN3.

In this case there are no matching targets for the clone source.

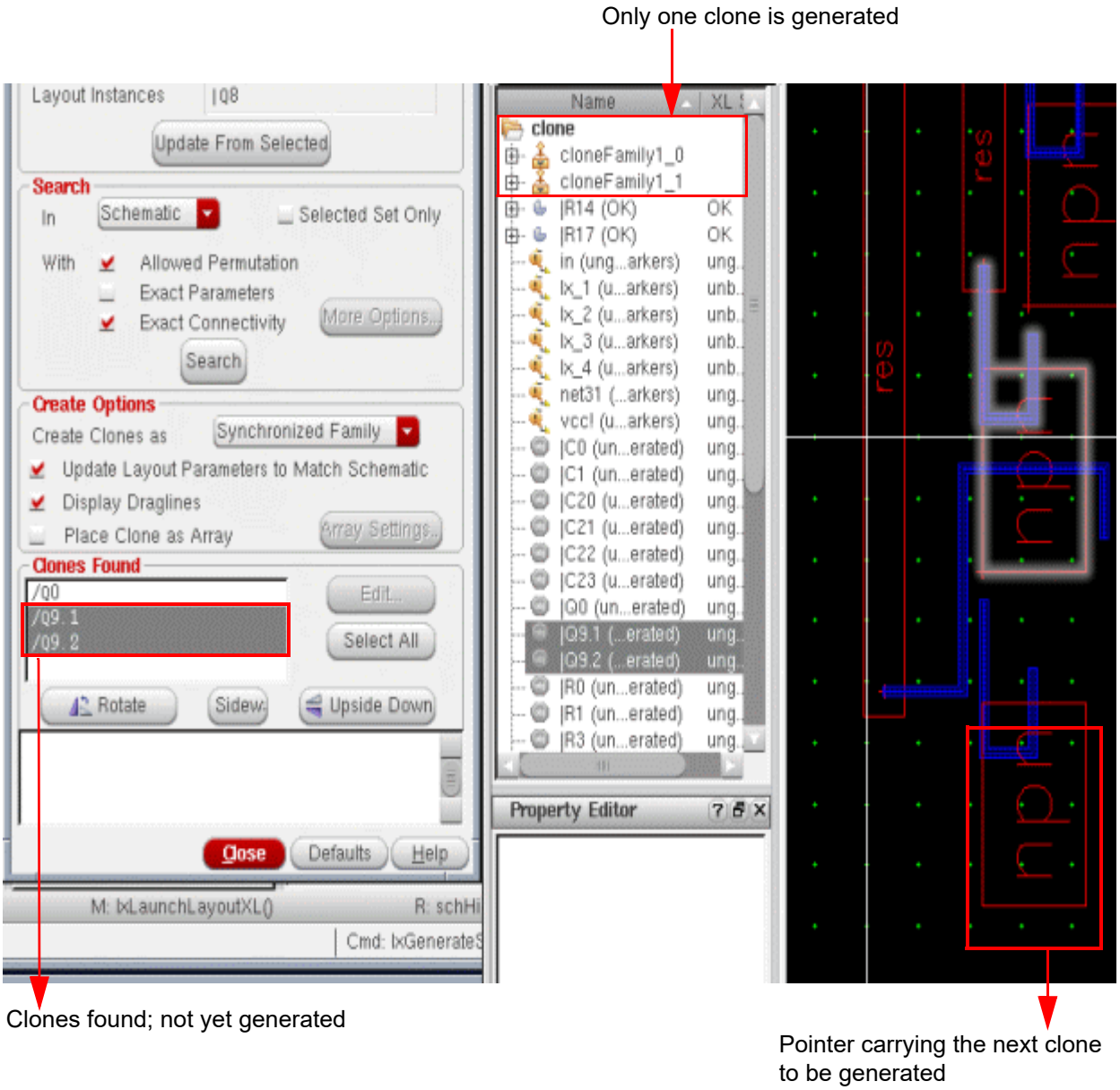
Related Topics

mfactor Property

© 2023 Cadence Design Systems, Inc.

Clone Placement as an Array

By default, if you select multiple clones from the *Clones Found* list, you can place only one selected clone at a time in the layout canvas. You can verify this in the Navigator assistant by observing that only one clone group, such as *cloneFamily1_1*, in the figure below gets generated, which maps to the first clone from the selected list.



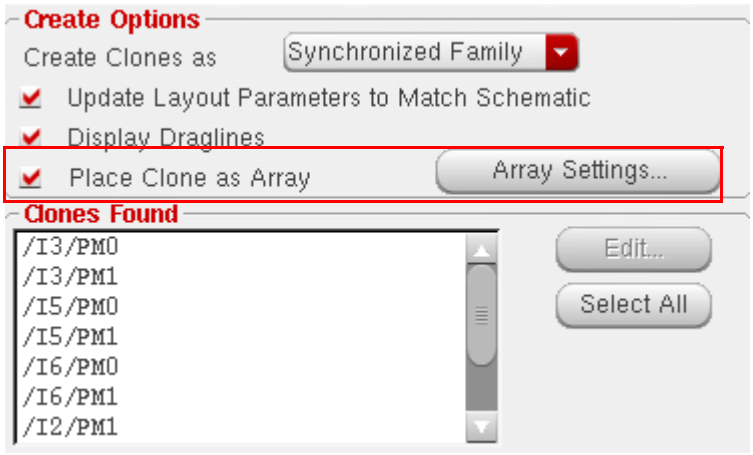
The other two selected clones that are not yet placed, still appear selected in the *Clones Found* list, as shown in the figure above. Also notice that the pointer in the layout canvas is now followed by another clone, which corresponds to the next clone in the list of selected

© 2023 Cadence Design Systems, Inc.

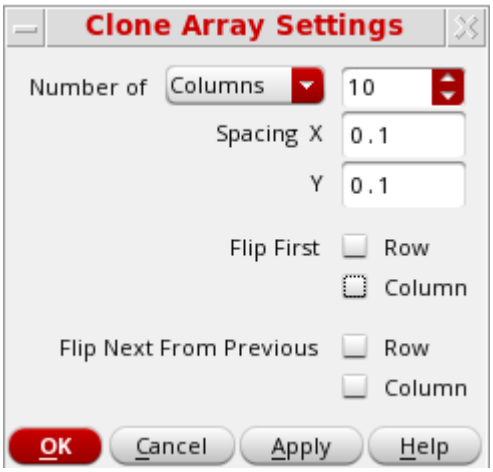
clones. To place this clone and any remaining selected clones, you need to click in the canvas as many times as the number of selected clones to place.

Instead of placing the selected clones one by one, as described above, you can choose to place all the selected clones in a single click as a clone array. To do this, choose the *Place Clone as Array* check box in the Generate Clones form, as shown in the figure below.

Alternatively, you can set the `placeCloneAsArray` environment variable to `t`.



When the *Place Clone as Array* check box is selected, the *Array Settings* button is enabled. Clicking the *Array Settings* button opens the *Clone Array Settings* form that you can use to define the specifications for the clone array to be generated.



Related Topics

Generate Clones Form

© 2023 Cadence Design Systems, Inc.

placeCloneAsArray

Specifying Clone Array Settings

In the Clone Array Settings Form form:

1. Specify the number of clones to be generated per *Row* or *Column*.
2. Specify the *Clone X Spacing* to be used when placing the clones in columns.

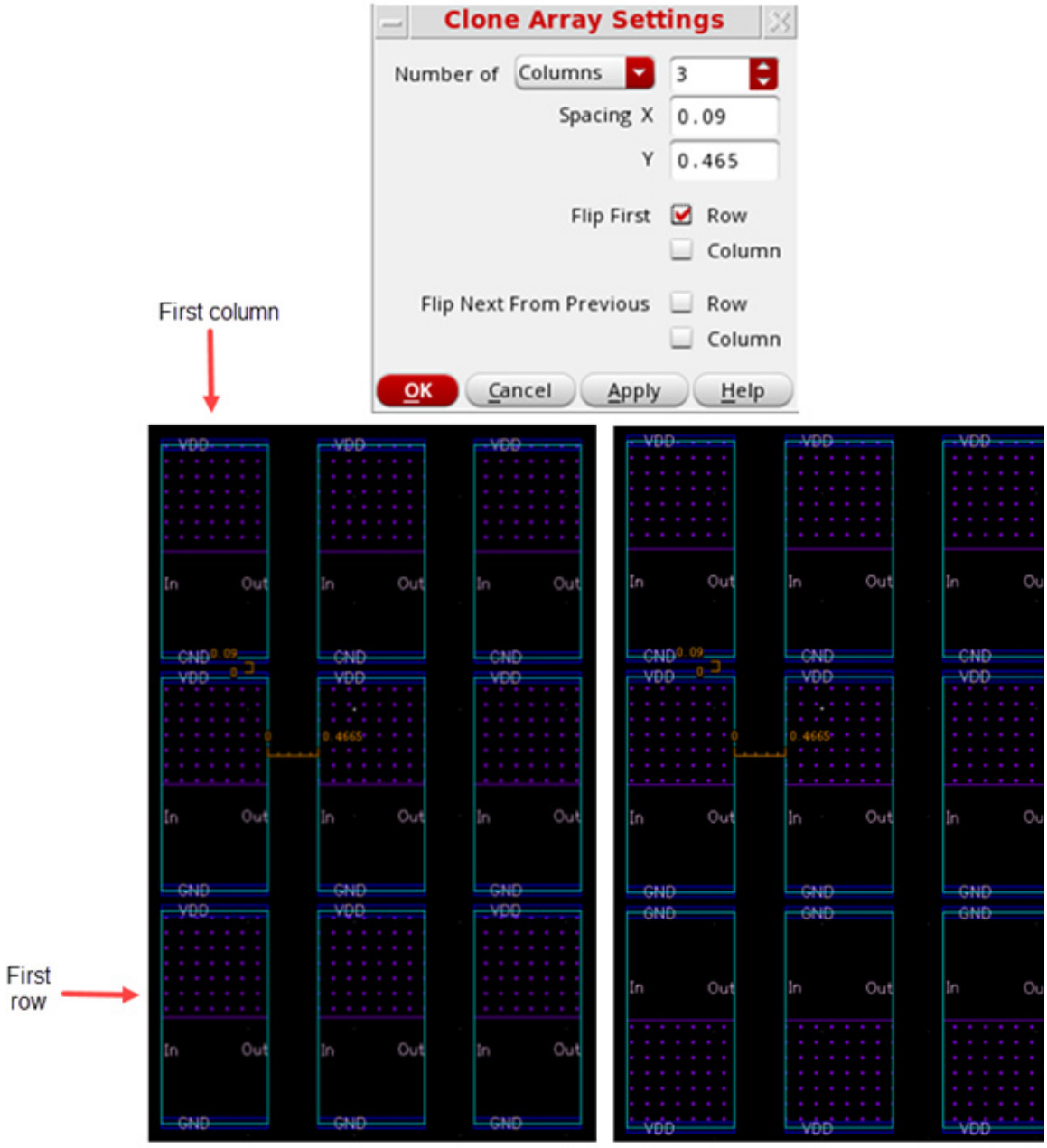
   For information about how the array spacings are applied, see Determining the Clone Array Spacings.
3. Specify the *Clone Y Spacing* to be used when placing the clones in rows.

   For information about how the array spacings are applied, see Determining the Clone Array Spacings.
4. (*Optional*) Specify that the clones from the first row and/or the first column in the array need to be flipped from the clone source.

   a. Click *Row* in the *Flip First* check box to flip the first row of the clone array you want to generate on the layout.

© 2023 Cadence Design Systems, Inc.

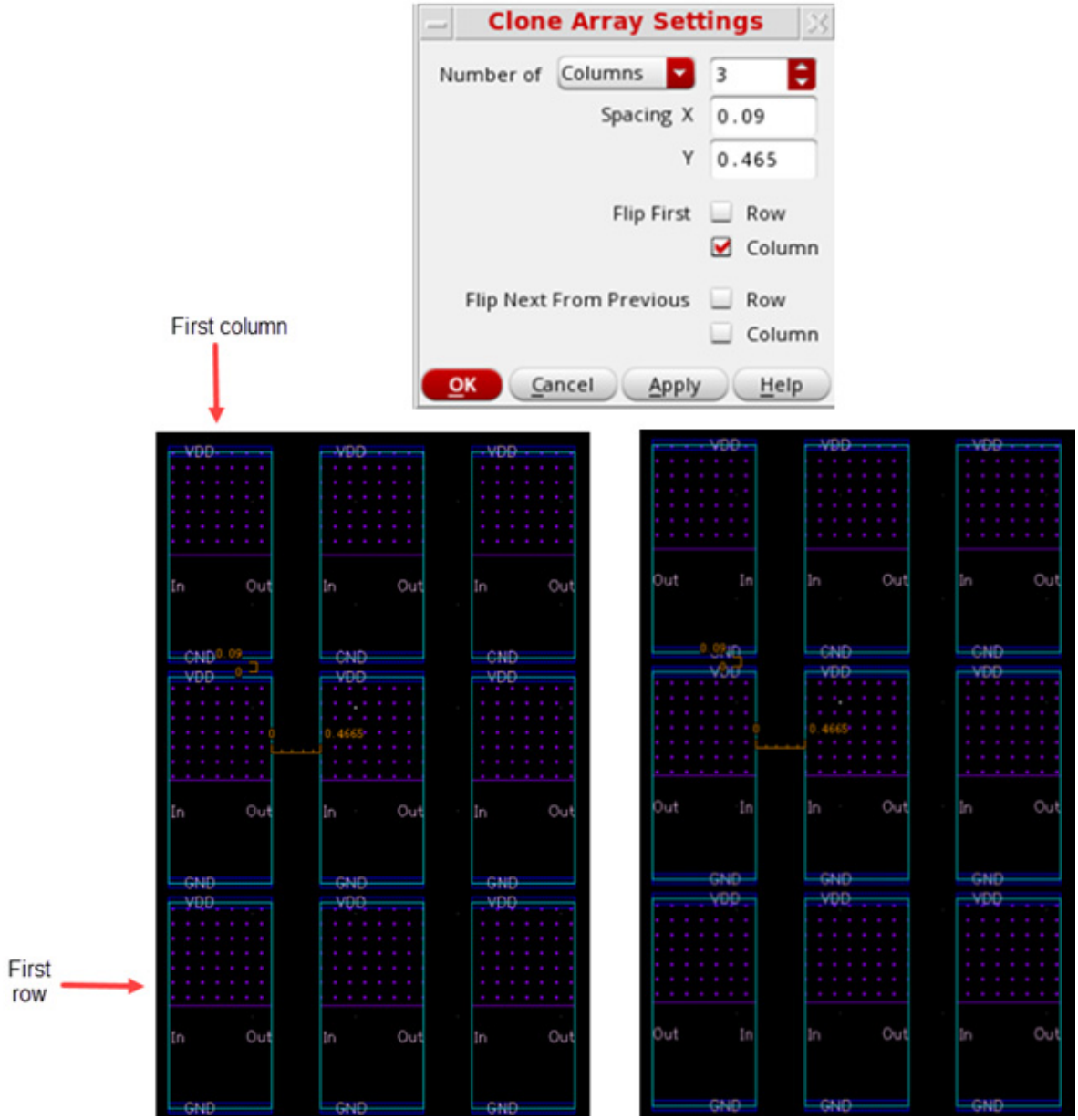
When the value of `cloneArrayFlipFirstRow` is set to `t`, each clone in the first row (bottom-most row) is flipped along the X-axis.



b. Click *Column* in the *Flip First* check box to flip the first column of the clone array you want to generate on the layout.

© 2023 Cadence Design Systems, Inc.

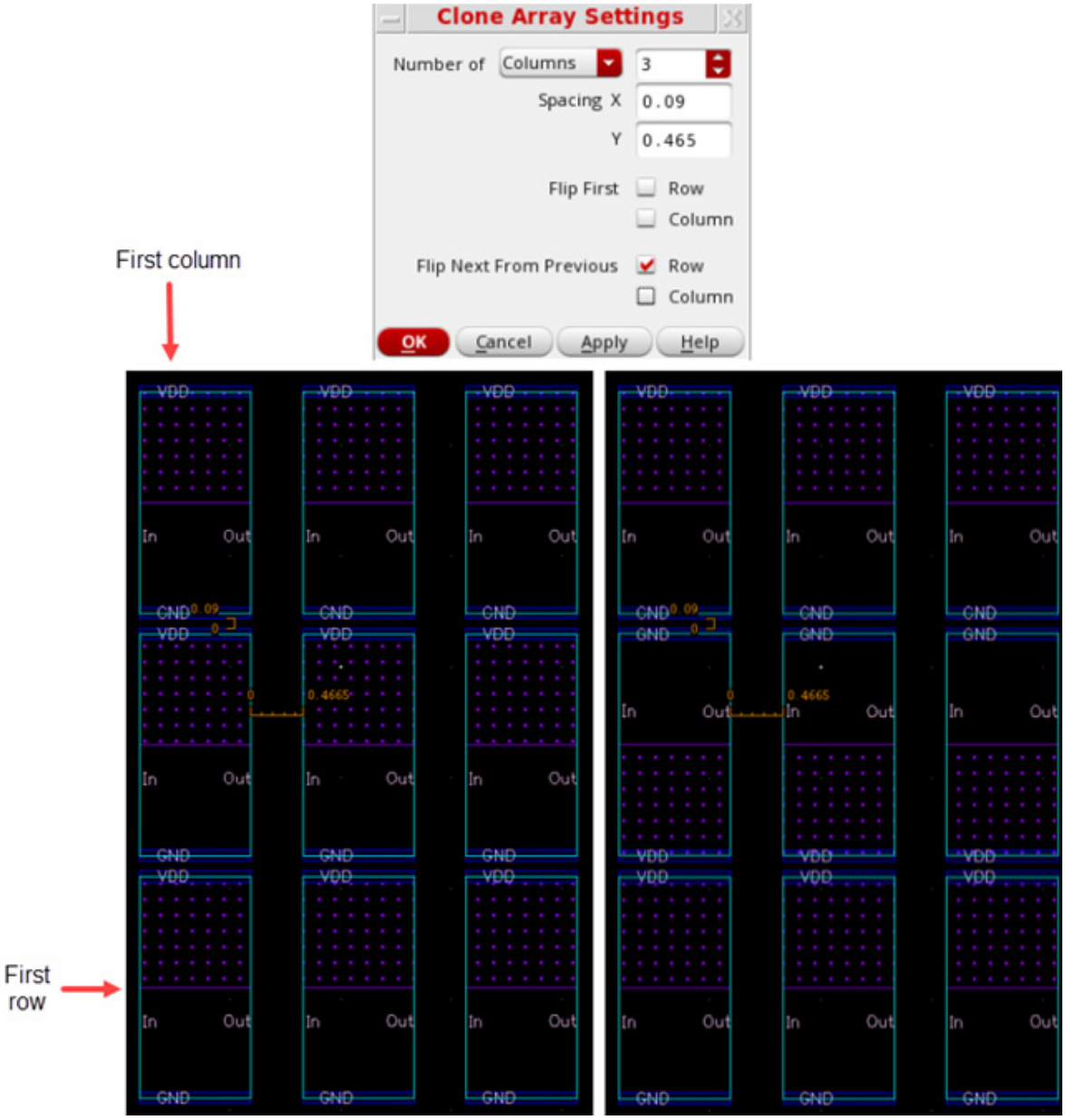
When the value of `cloneArrayFlipFirstColumn` is set to `t`, each clone in the first column (leftmost column) is flipped along the Y-axis.



5. (*Optional*) Specify whether the clones need to be flipped alternatively in each row and/or column.
6. Click *Row* and *Column* in the *Flip Next From Previous* check box to flip the next row and column of the clone array you want to generate in the layout.

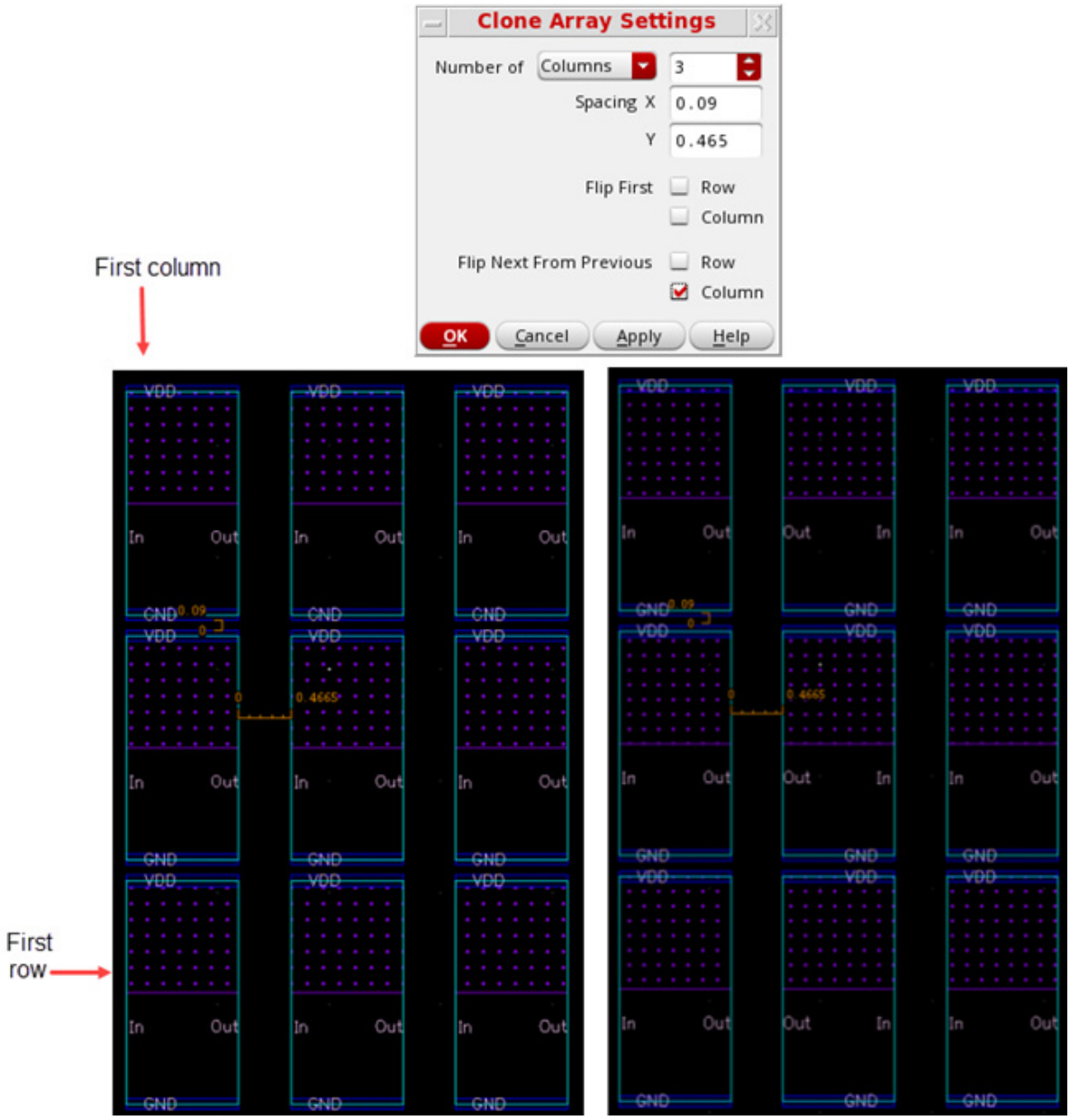
© 2023 Cadence Design Systems, Inc.

When the value of `cloneArrayFlipNextFromPreviousRow` is set to `t` in each column, the clone from row *N* is flipped along the X-axis in relation to the clone from row *N-1*.



© 2023 Cadence Design Systems, Inc.

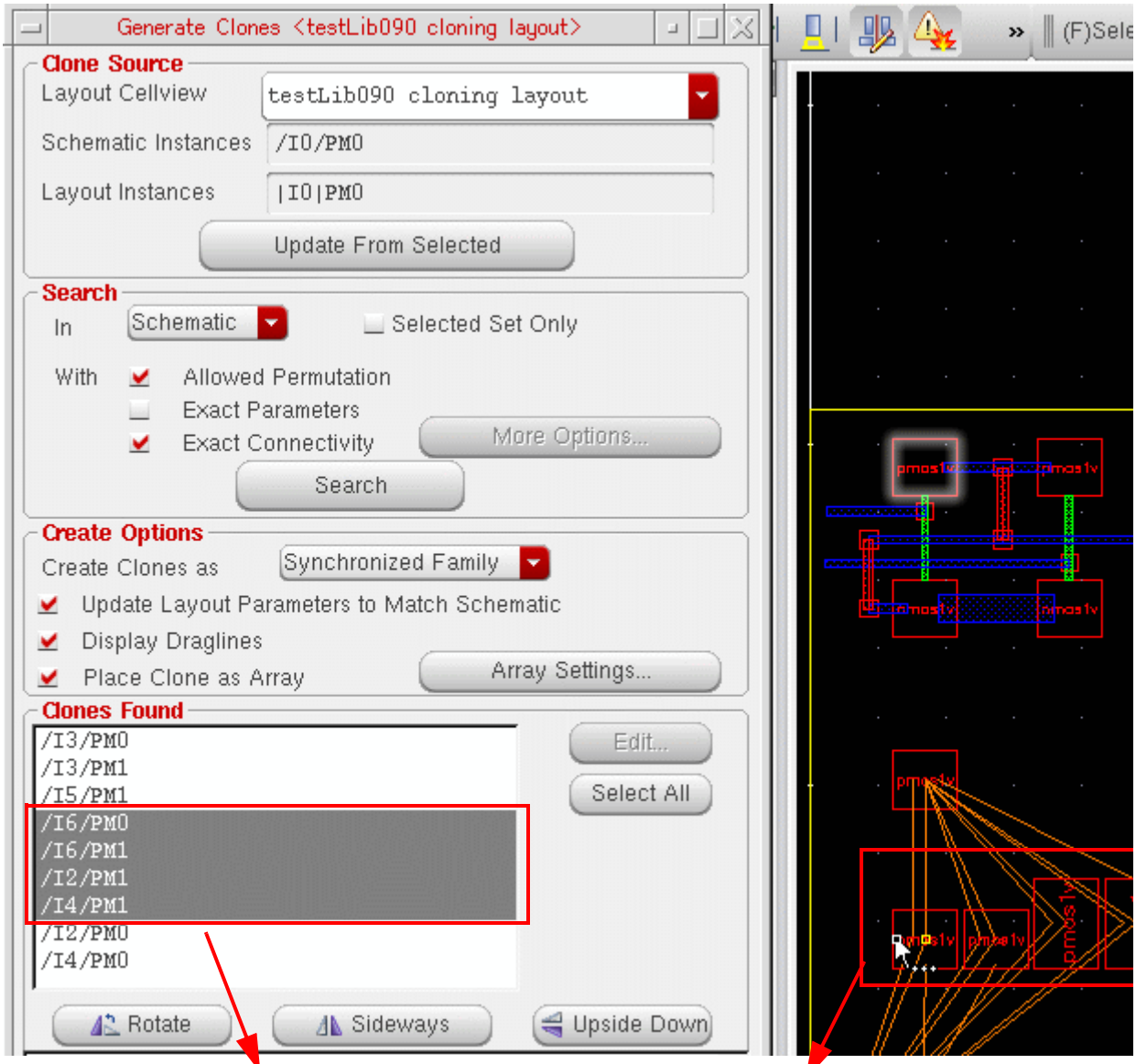
When the value of `cloneArrayFlipNextFromPreviousColumn` is set to `t` in each row, the clone from column *N* is flipped along the Y-axis in relation to the clone from column *N-1.*



7. Click *OK* to save the clone array settings, which will be used the next time you generate clones.

© 2023 Cadence Design Systems, Inc.

All the clones that you selected from the *Clones Found* list follow the pointer in the layout canvas, and are placed together on a single click, as shown in the figure below.

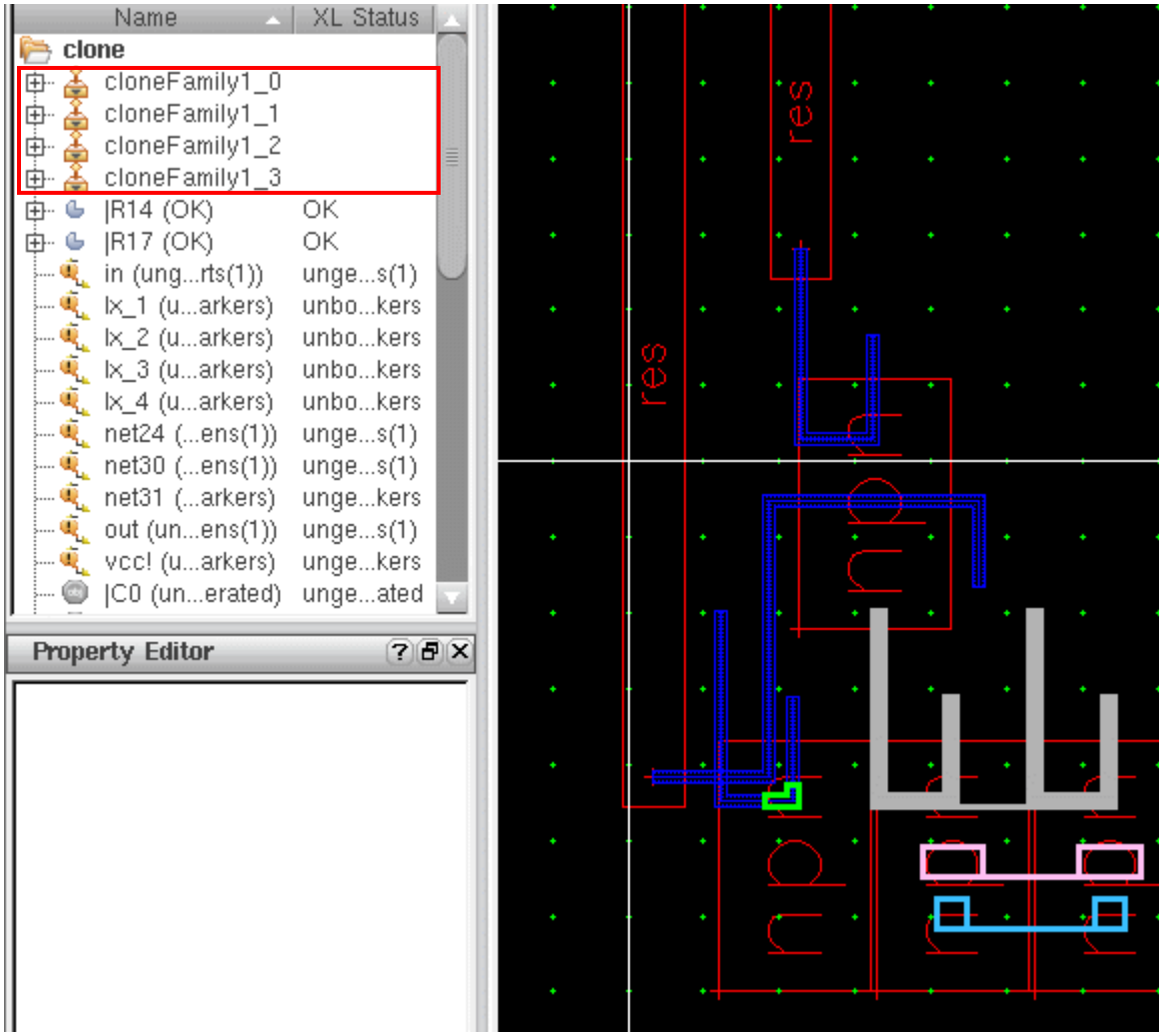


Clones found; selected for generation

Pointer carrying all the selected clones for generation

© 2023 Cadence Design Systems, Inc.

You can also verify this using the Navigator assistant, as shown in the figure below. Notice that the Navigator assistant displays new clone groups that have been created for all the clones that were selected for generation.



Determining the Clone Array Spacings

To determine the array spacings (both X and Y) to be applied between the clone bBoxes, the clone bBoxes are first computed from the merged bBoxes of clone figs (shapes + instances).

For determining the spacing between the instances, the displayed bBoxes are used. Therefore, the clone bBox computation depends on the options selected on the Display Options form.

© 2023 Cadence Design Systems, Inc.

- If the *use True BBox* option is selected, the true bBoxes of the instances are used to compute the clone bBox.
- If the *Instance Drawing Mode* option is set to *P&R Boundary*, the instance PR boundaries are used to compute the clone bBox.
- If the *Instance Drawing Mode* option is set to *Snap* Boundary, the instance snap boundaries are used to compute the clone bBox.

Specifying Transform From Settings

To specify the Transform From settings:

1. From the Transform From drop-down menu, choose the appropriate clone transformation option—*Previous* or *Source*—and choose the appropriate orientation from the Transform combo box.
2. Click at the required location in the layout window to place the transformed clone.
   - If you choose the *Source* transformation option from the *Transform From* drop-down menu**,** the transform displayed in the *Transform* group box is applied from the clone source.
   - If you choose the *Previous* transformation option from the *Transform From* drop-down menu**,** the transform displayed in the *Transform* group box is applied from the previously generated clone.

   If you choose the *Previous* transformation option from the Transform From drop-down menu and create the first clone using no previous clone, the transform displayed in the Transform combo box is applied from the clone source.
3. Click anywhere in the layout window to place the next clone found.
   - If you choose the *Source* transform, the next clone is also rotated by the specified transform value, which is relative to the clone source. This means that the second clone has the same transform as the previous one.
   - If you choose the *Previous* transform, the second clone is rotated by the specified transform value, which is relative to the previously generated clone. This means that the second clone is rotated twice relative to the clone source.

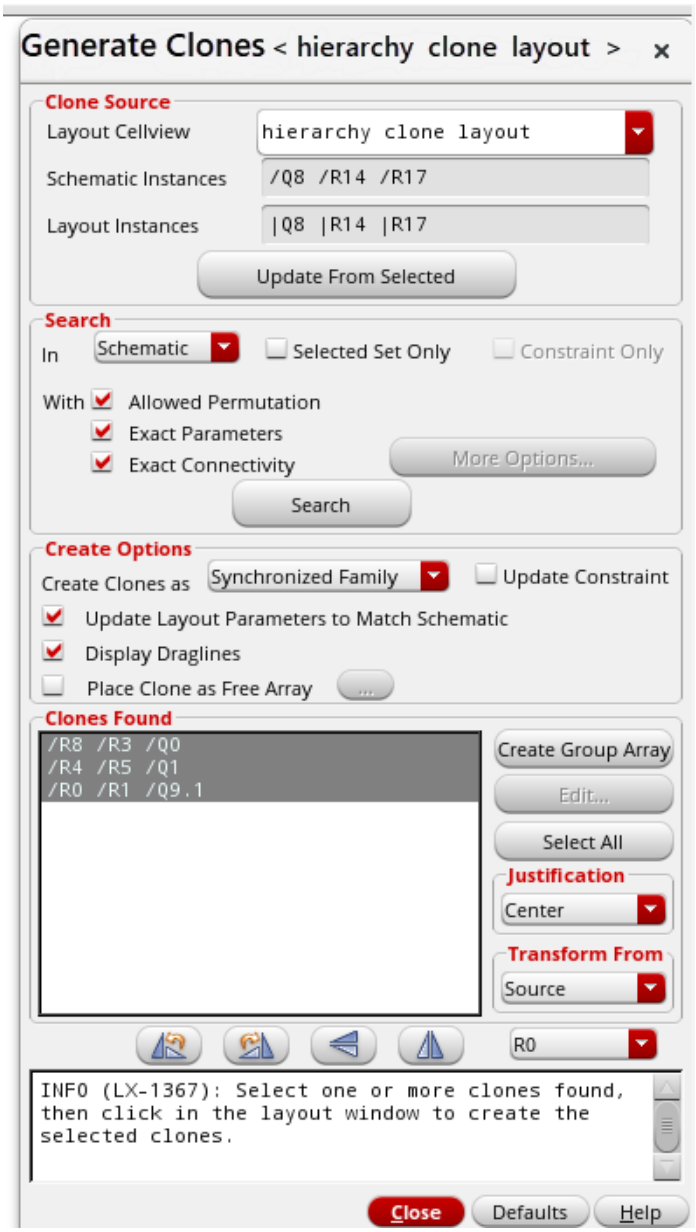
     For example, if you choose the *Previous* transform and select an R90 orientation in the Transform combo box, then the first clone is rotated by 90 degrees relative to the clone source. The second clone is rotated by 90 degrees relative to the previous clone. This means that the second clone is rotated twice to be effectively rotated by 180 degrees relative to the clone source.

© 2023 Cadence Design Systems, Inc.

Creating a Group Array when Generating Clones

To create a group array when generating clones:

1. In the layout canvas, select the individual component or components within an area that need to be cloned and choose the *Connectivity – Generate – Clones* command.
2. Click the *Search* button to find clones in schematic.
3. Select the clones found.



© 2023 Cadence Design Systems, Inc.

4. Click the *Create Group Array* button.

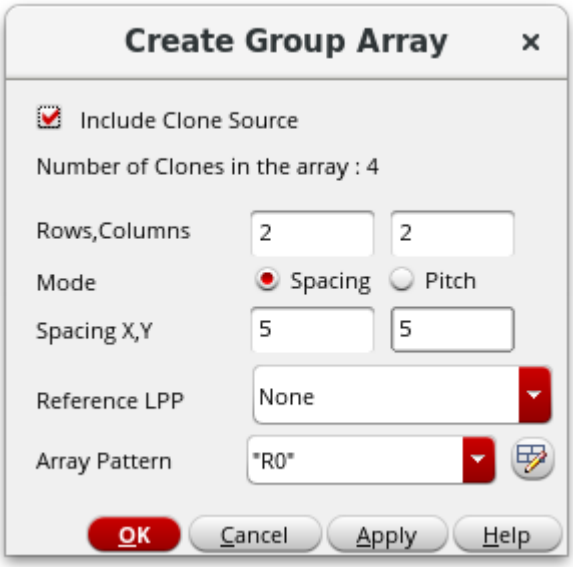
   The Create Group Array form is displayed.

5. Choose to create a group array by:

   - Including the clone source

     a. Select *Include Clone Source*.

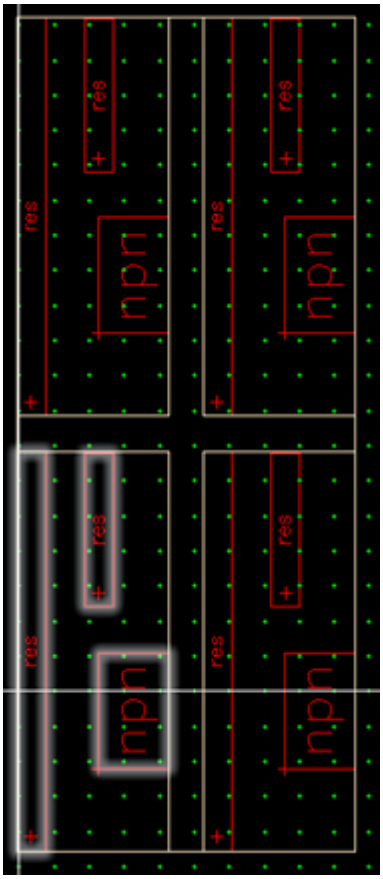
     b. Set different fields in the form based on your design requirements.



     c. Click *Apply* or *OK* to place the group array in the canvas

     The group array is created automatically as a batch operation without any drag set. Each clone is a cell of the group array. The clone source is used as the lower-left cell of the group array and then other newly created clones are placed relatively to the existing clone source, respecting the group array settings such as spacing and number of rows and columns.

© 2023 Cadence Design Systems, Inc.



- Without including the clone source

  a. Set different fields in the form based on your design requirements.

  b. Click *Apply* or *OK*.

  A group array is created in the cloning drag set with all the new clones but not the clone source.

  c. Click in the layout canvas to place the group array where needed.

  A group array is created in the cloning drag set (and not in the canvas).

Related Topics

Create Group Array Form

Specifying Group Array Orientation Using the Array Pattern Form

© 2023 Cadence Design Systems, Inc.

Predefined Orientation Patterns for a Group Array

Generating Synchronous Clones

Synchronous clones stay synchronized during editing— changes made to one member of a family are automatically reflected in all the other members of the clone family.

You can use the *Synchronized Family* option on the Generate Clones form to create multiple clones as members of a single synchronized family. The clone source and the individual clones are each generated as a group in the layout view and linked to the other members in the same synchronized family, with each group containing all the physical shapes, vias, instances, and groups in the layout implementation of the target structure.

Labels in the canvas can be defined either as *text displays* or as *label* objects. Text displays in the clone source are synchronized in the generated clones; label objects are not.

If you add a new instance to an existing synchronous clone using either the *Create Instance* or *Generated Selected From Source* command, or by removing an ignore property from an existing instance inside the clone, the clone in question is not desynchronized from its family. Note, however, that it may no longer form a valid clone source, in which case you will be unable to use the *Generate Clones*, *Create Family*, or *Add to Family* commands on that structure.

- **Synchronous Clones and Instance Creation**

  If you add a new instance to an existing synchronous clone using the *Create Instance*, *Copy*, or *Generate Selected From Source* commands, the system automatically adds a similar instance to the other clones in the family, ensuring that the family remains synchronized. If you add or remove an ignore property from an existing instance inside the clone, the clone is not desynchronized from its family, but the change to the property is not replicated in the other clones in the family.

  In both cases, the clone in question may no longer form a valid clone source, which means you will be unable to use the *Generate Clones*, *Create Family*, or *Add to Family* commands on that structure.

- **Synchronous Clones and Dummy Devices**

  Layout XL considers a layout instance to be a dummy device if either the instance or its master has at least one property included in the list of *Properties used to ignore objects during check* on the Parameters tab of the Connectivity form or the instance has the `physOnly` attribute set.

  When Layout XL detects a dummy device among the objects in the clone source, it generates a similar instance with the same properties in the target structure. This new

© 2023 Cadence Design Systems, Inc.

instance is automatically named `Ix`, where *x* is an appropriate integer value assigned by the database.

Layout XL considers a dummy device as a simple shape in the clone structure. Therefore, dummy devices are synchronized in the same way as any other shape in a family of synchronous clones. Any physical editing, such as move, rotate, or delete performed on a dummy device (including those used in chains of devices) is reflected in every other member of the synchronized family.

To add a previously defined dummy device to a synchronous clone:

a. Create the instance for the dummy device outside the clone family.

b. Add the appropriate ignore-for-check property to the instance.

c. See Use the *Edit – Group – Add To Group* command to push the dummy device inside the clone family.

All the other occurrences of the synchronous clone are populated with a similar dummy device. These new devices are named automatically and are connected as required by the Layout XL connectivity extractor.

When using the *Add To Family* and *Create New Family* commands on the Update Clone Families form, any dummy devices present in the reference group are copied in other occurrences of the clone.

■ Synchronous Mutants

Mutants are clones that may not exactly match the source in connectivity. So, when searching for clone targets on the **Generate Clones** form with the option **Exact Connectivity** set to false, you may find mutant targets in your search result. These mutants will be displayed as *`mutant (number) instanceName1 instanceName2`*, and so on.

As the resulting mutants do not exactly match the source in connectivity, they have missing or extra nets called partial nets. However, these mutants can be considered as copies of the layout clone source with the connectivity of the schematic targets. Therefore, all the routing shapes present in the clone source are copied to the generated mutant, even if the shape net is a partial net, which is missing in the mutant target. As a consequence, the generated target clone is physically identical to the clone source, even if the clone is a mutant. This allows for synchronous mutants to be created as the cloning synchronization mechanism does not require any logical aspects to be met as long as the clones are physically identical.

As the logical connectivity of the mutant instances (inherited from the clone schematic target) may be different from the physical connectivity of the routing shapes (inherited

© 2023 Cadence Design Systems, Inc.

from the clone layout source), the XL connectivity extractor may generate short markers when processing the mutants.

- **Synchronous Clones and Device Abutment**

  You cannot abut two devices if one of the devices is in a synchronous clone and the other is not in the same synchronous clone.

- **Synchronous Clones and Pin Permutation**

  You cannot permute two pins if one of the pins is on a device in a synchronous clone and the other pin is on a device that is not in the same synchronous clone.

  If you permute two pins on an instance in a synchronous clone, the change is not propagated to the other members of the family, nor is the clone desynchronized from the other members of the family, because pin permutation is a logical rather than a physical update.

- **Remastering Clone Instances**

  When an instance is remastered in a clone, the clone synchronizer automatically remasters the equivalent instances in other clones.

  In addition, the physical binding in the Configure Physical Hierarchy window is updated for the remastered instance in the current clone, and for all equivalent instances.

Related Topics

Generate Clones Form

Connectivity Form

Ignoring Components with no Schematic Counterpart

Update Clone Families Form

Creating a New Clone Family

Adding a Group to an Existing Family

Generating a Family of Synchronous Clones

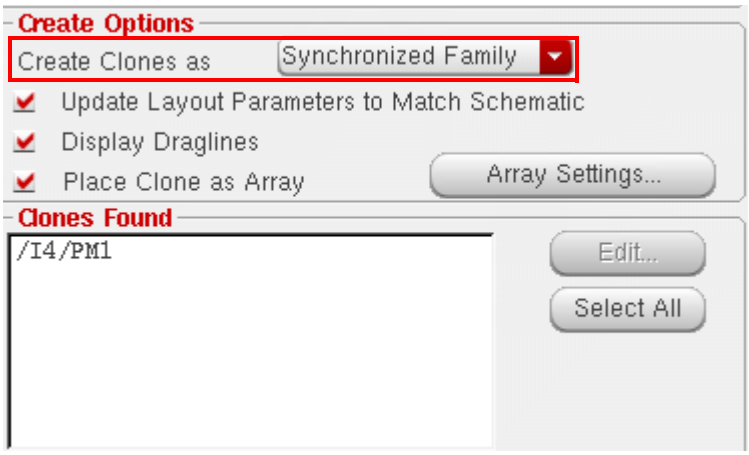
To generate a family of synchronous clones:

1. Create a list of matching targets using the steps described in Clone Source Selection and Target Search and Generation.

© 2023 Cadence Design Systems, Inc.

The matching targets are displayed at the bottom of the Generate Clones form in the *Clones Found* pane.

2. Choose *Synchronized Family* from the *Create Clones as* drop-down list.

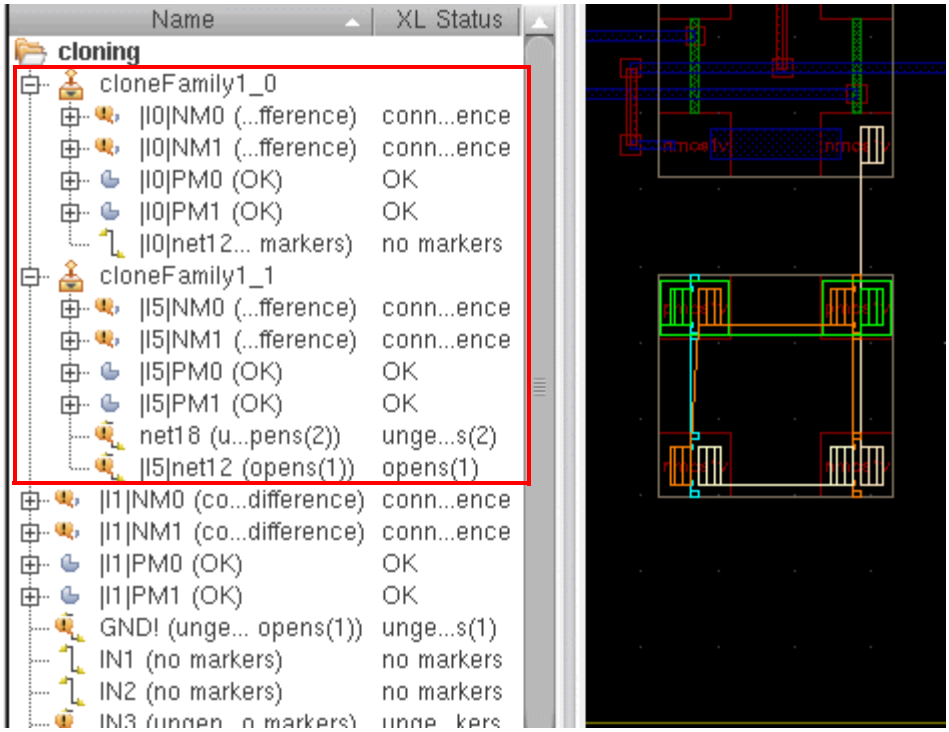


3. From the list of clones found, select all the target structures you want to include in the synchronized family.
4. Move your cursor into the layout canvas and click once to generate each of the clones you selected.

   If the synchronous family you are creating contains only one group, Layout XL uses this existing group to implement the synchronous clone. This prevents the creation of an unnecessary level of hierarchy.

© 2023 Cadence Design Systems, Inc.

The clone source and the target clones are generated as two synchronized clone groups that can be identified in the Navigator assistant by their unique names, such as `cloneFamily1_0` and `cloneFamily1_1` in the figure below.



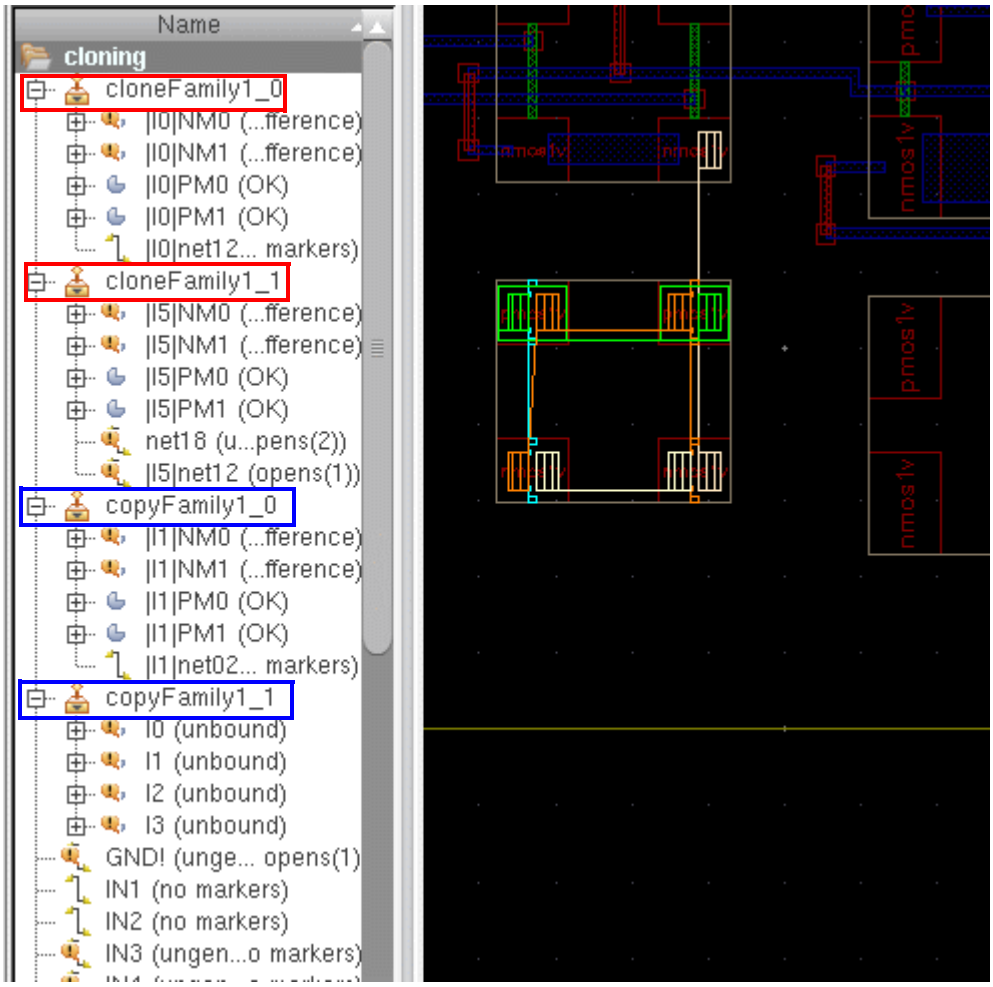
- Each synchronous clone name begins with the prefix *cloneFamily,* unless the source is a group. In that case, the generated synchronized clones have names that carry the name of the source group. If the source group is named as `myDiffPair`, the generated clones are named as `myDiffPair_1`, `myDiffPair_2`, and so on.
- A synchronous clone group created using the Generate Clones form can be differentiated from a synchronous copy that is created using the Create Cluster Boundary Form form by its name. A synchronous clone gets a *cloneFamily* prefix,

© 2023 Cadence Design Systems, Inc.

while a synchronous copy gets as a *copyFamily* prefix, as displayed in the figure below.

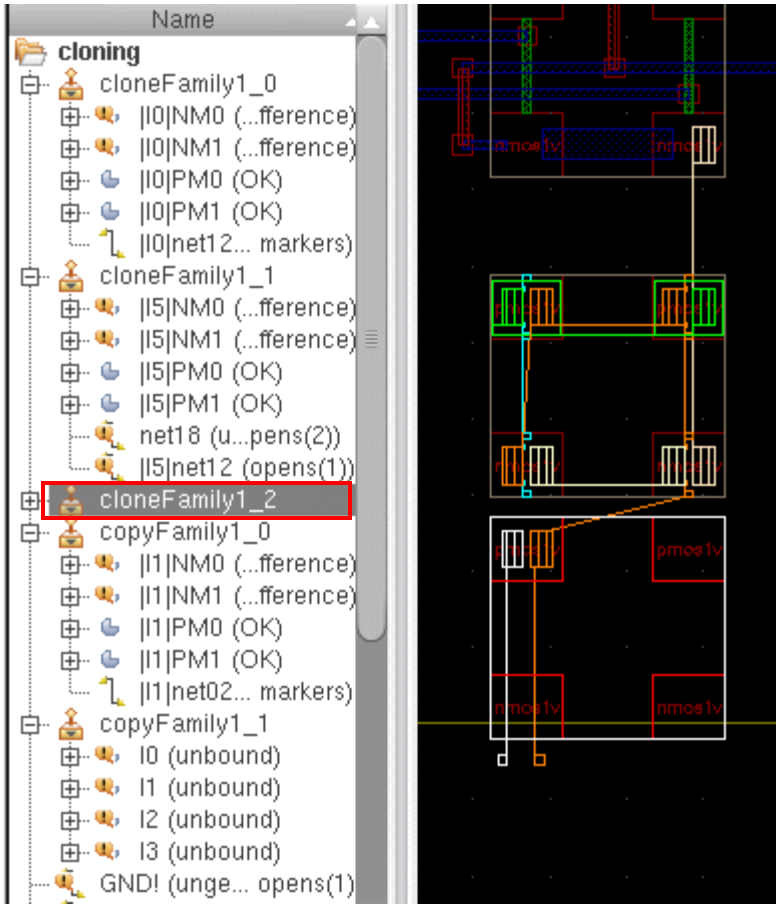
Synchronized clone

Synchronized copy



© 2023 Cadence Design Systems, Inc.

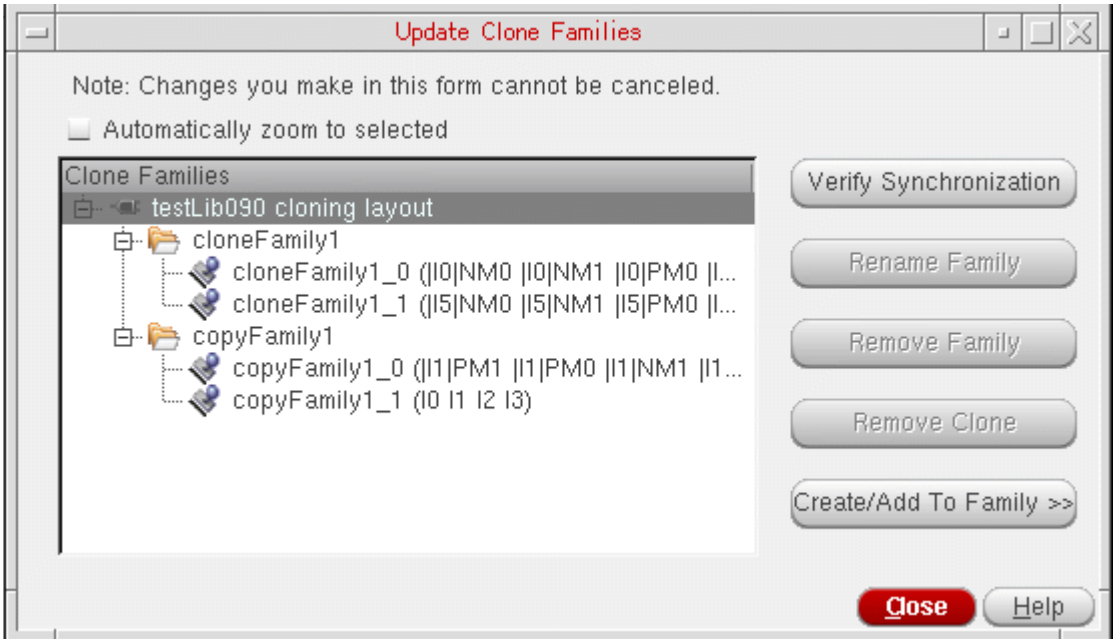
If you create a copy of a synchronous clone, the copy target gets the basename of the synchronous clone, such as *cloneFamily1_2*, in the figure below, if the copy source is *cloneFamily1_0*.



5. To verify whether the generated clone families are correctly synchronized, choose the *Connectivity – Update – Clone Families* command to display the Update Close Families form, as shown in the figure below. Select the clone family to check and click the *Verify Synchronization* button. To check synchronization for all the clone families, leave the clone families deselected and click *Verify Synchronization*,

© 2023 Cadence Design Systems, Inc.

You cannot undo the changes you make using the Update Clone Families form.



The CIW issues a message indicating whether or not the verified clone families are synchronized.

You can also use the Update Clone Families form to add or remove clone members, rename the clone family, or delete the clone family.

Related Topics

Generate Clones Form

Update Clone Families Form

Verifying a Clone Family

Exceptions to Synchronous Clone Generation

If you are experiencing any of the following design situations, synchronous clone generation using the *Generate Clones* command is not possible:

- *Scenario 1: When the selected clone source includes only the parent, not the child*

© 2023 Cadence Design Systems, Inc.

If two figs exist in a parent-child relationship and you select only the parent fig, the child fig automatically gets added to the clone source figGroup but the clone target figGroup only has the parent fig. Therefore, the synchronous clone cannot be created as the clone source and the clone target do not have the same number of figs.

- *Scenario 2: Some clone source figGroups are only “partially selected”*

  If the schematic window is used to cross-select layout instances that are in a layout figGroup, and if only some of the instances from the layout figGroup are selected, only the “selected” layout instances are cloned. Therefore, the synchronous clone cannot be created as the clone source and the clone target do not have the same number of instances.

- *Scenario 3: The clone source and target are not in the same cellview*

  Layout XL does not support synchronous clone generation when the clone source and the clone target belong to different cellviews.

- *Scenario 4: The clone source is a clone figGroup from a family that is not "verified"*

  If the clone source contains a figGroup from a family that is not verified, the *Generate Clones* command cannot use it for generating a synchronous clone. To verify the clone families before they can be used as a clone source, either *Edit In Place* one of the clones or manually run *Verify Synchronization* using the Update Clone Families form.

- *Scenario 5: A mismatch exists between the layout instance master of the clone source and the clone target*

  This is likely when the clone source and the clone target members are processed differently for abutment, causing a mismatch that prevents them from getting generated as synchronous clones.

Related Topics

Update Clone Families Form

Generate Clones Form

Editing a Clone in a Synchronized Family

Because synchronous clones are implemented using groups, you cannot edit the components in the clone from the top level of your design. To edit a synchronous clone, you must use the *Edit In Place* command.

© 2023 Cadence Design Systems, Inc.

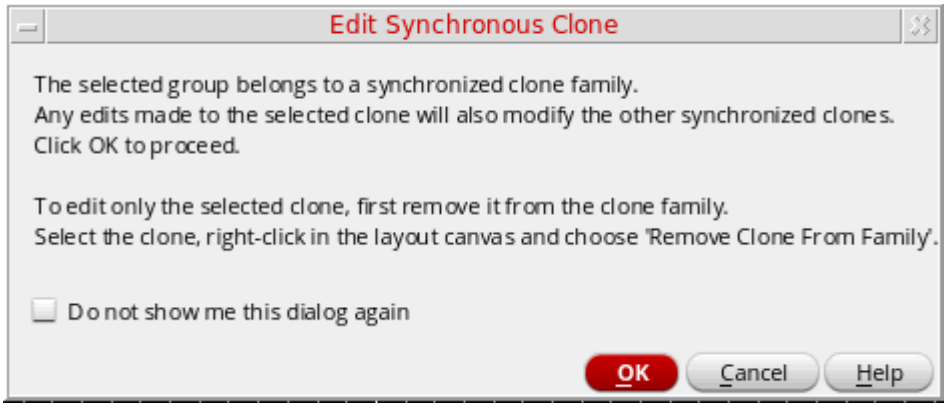
If you make a change to the physical topology of one member of a synchronized family, that change is propagated to all the other clones in the family.

To edit a member of a synchronized family:

1. From the layout window menu bar, choose *Edit – Hierarchy – Edit In Place* and select the clone you want to edit.

   Alternatively, in the layout window, select the clone you want to edit and choose the *Edit – Hierarchy – Edit In Place* command.

   The Edit Synchronous Clone dialog box pops up, warning that the clone you are about to edit is a member of a synchronized family.



2. Click *OK* to proceed with editing the entire clone family.

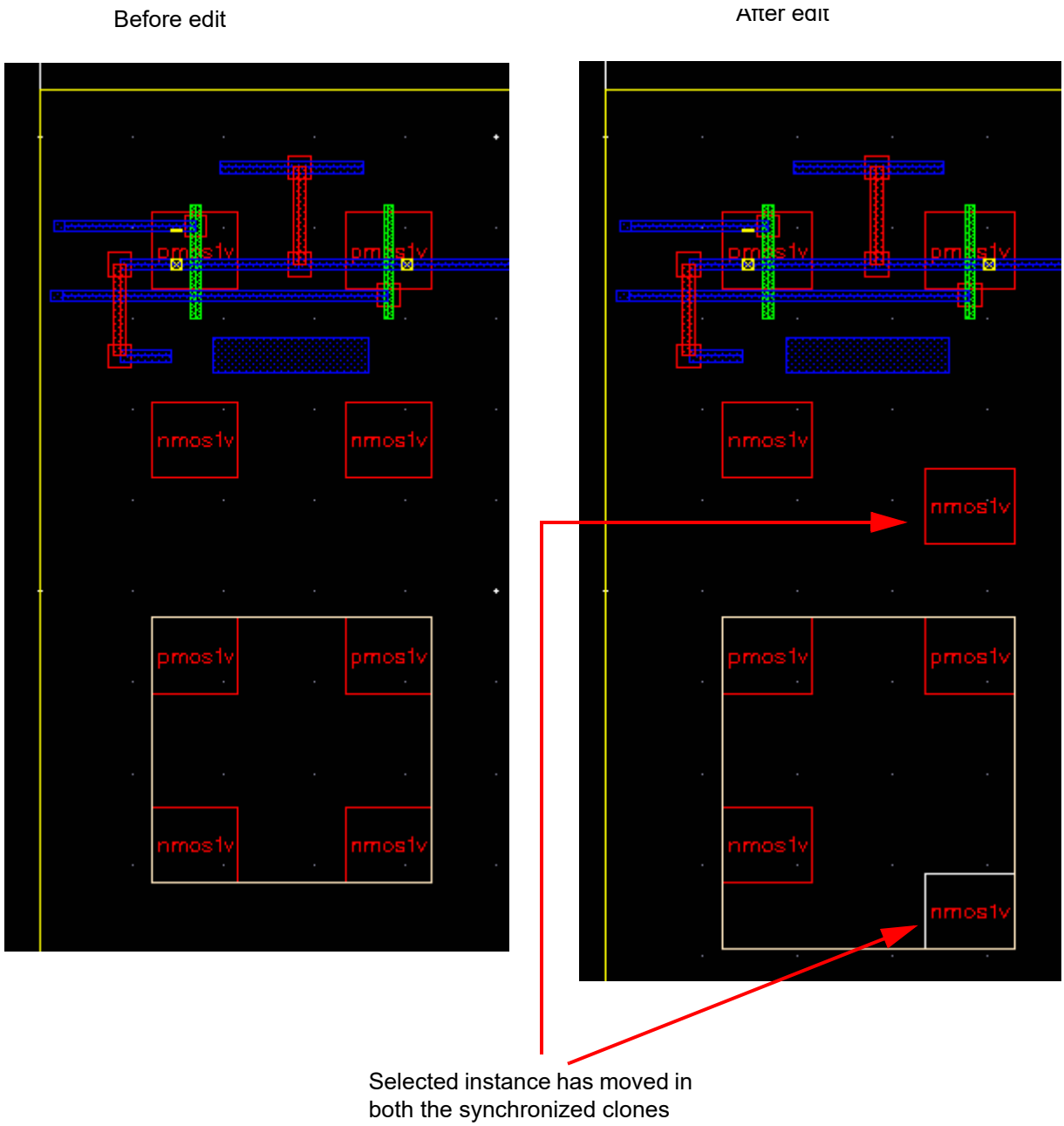
   If you want to edit only the selected clone, first remove the clone from the clone family.

   You can also choose not to see this dialog again.

3. The layout window is displayed. Select one of the instances in the clone you are editing, and choose an appropriate *Edit* command, such as *Move*.

© 2023 Cadence Design Systems, Inc.

The corresponding instance is moved in each of the other clones in the synchronized family, as shown in the figure below.



4. From the layout window menu bar, choose *Edit – Hierarchy – Return* to exit the Edit In Place mode.

© 2023 Cadence Design Systems, Inc.

Related Topics

Editing In Place

Coloring Synchronization in Synchronous Clones

Layout XL supports coloring synchronization for synchronous clones. This means that if the mask color on one of the synchronous clones is manually updated, corresponding shapes in the other synchronous clones are automatically updated and synchronized. For example, if a `mask1Color` color-locked shape in a synchronous clone has its mask color updated to `mask2Color`, all the corresponding `mask1Color` color-locked shapes in other synchronous clones also change their mask color to `mask2Color`.

To prevent the color state and color updates on color-locked shapes from being carried to the corresponding shapes in other synchronous clones, set the `enableCloneColoringSupport` environment variable to `nil`.

Consider some design situations that illustrate the extent of color state or color synchronization possible, depending on the manual edit made to a shape in a synchronous clone:

- *Scenario 1 — The color state of a shape in the synchronous clone is switched from* `unlock` *to* `lock`

  Because Layout XL supports fully synchronizing those shapes that are locked on a color, in this case, both the color state and color are synchronized on equivalent shapes in other synchronous clone(s).

- *Scenario 2 — The color state of a shape in the synchronous clone is switched from* `lock` *to* `unlock`

  In this case, the color state is synchronized on equivalent shapes in other synchronous clones but the color is not synchronized by default if `cloneSynchronizeUnlockedColor` is set to the default = "`no`". For the color to be synchronized, the environment variable must be set to "`shape only`" or "`cut and shape`".

- *Scenario 3 — The color of a “color locked” shape is changed*

  In this case, the color is synchronized on equivalent shapes in other synchronous clones. No change is made to the color state.

- *Scenario 4 — The color of a “color unlocked” shape is changed*

© 2023 Cadence Design Systems, Inc.

No synchronization takes place by default if `cloneSynchronizeUnlockedColor` is set to "`no`". For the color to be synchronized, the environment variable must be set to "`shape only`", "`cut and shape`", or "`cut only`".

To change the color state of a shape in a synchronous clone from `lock` to `unlock` or vice versa, you must Edit in Place the synchronous clone. Such updates to shapes in a synchronous clone are not supported when operating from the top cellview. Conversely, if the shape in question is not a member of a synchronous clone family, color and other updates are possible from the top cellview just as for other regular shapes.

By design, Layout XL keeps the clone families synchronized by preventing any coloring updates from the top cellview that can impact the shapes inside synchronous clones.

By default, only locked colors are synchronized, but in a lockless flow, you can synchronize unlocked colors by setting the value of `cloneSynchronizeUnlockedColor` environment variable to "`cut only`", or "`shape only`", or "`cut and shape`".

When you do this, all locked and unlocked changes are synchronized with other clones during Edit in Place. For example, if `cloneSynchronizeUnlockedColor` is set to "`cut and shape`" and you change an unlocked cut color to `mask3` in one clone, the cut from the other via in the other clone will also be updated to `mask3`.

The pre-existing unlocked color differences are still skipped if the clone checker detects a difference in pre-existing clones about unlocked color, the clones are not desynchronized.

Related Topics

enableCloneColoringSupport

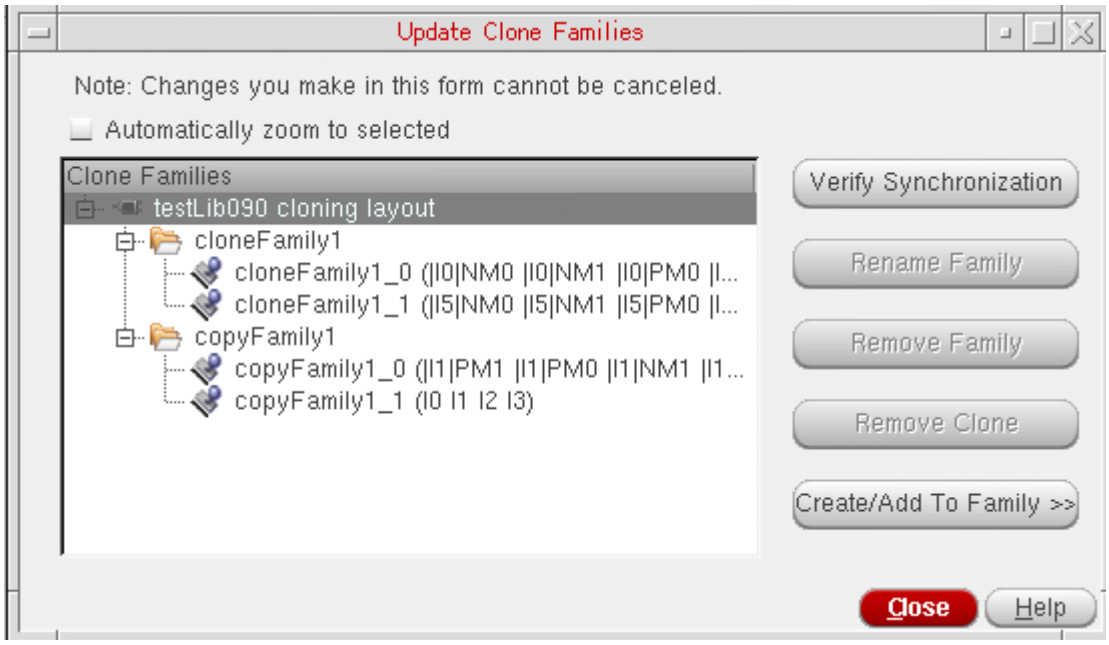
cloneSynchronizeUnlockedColor

Editing In Place

© 2023 Cadence Design Systems, Inc.

Updating Clone Families

The Update Clone Families form lets you manage the families of synchronized clones in your design. To see this form, choose *Connectivity – Update – Clone Families*.



The entries in the form are highlighted depending on what is selected in the layout window. The updates you make in the layout window are reflected immediately in the Clone Families pane.

The *Automatically zoom to selected* option zooms in on the clone or clone family that is currently selected in the *Clone Families* pane, or on the layout group selected in the *Layout groups* pane .

Clicking on the top level fits the whole cellview in the layout window. If you select more than one layout group, the software zooms in on the bounding box that encloses the selected groups.

This option is considered only when you make the selection in the Update Clone Families form; selecting a synchronous clone in the canvas does not zoom the display to the selected clone or group.

For more information on the tasks you can perform in this form, see

- Verifying a Clone Family
- Removing a Clone from a Family

© 2023 Cadence Design Systems, Inc.

- Removing a Clone Family from Design
- Creating a New Clone Family
- Adding a Group to an Existing Family

Related Topics

Update Clone Families Form

Creating a New Clone Family

Adding a Group to an Existing Family

Verifying a Clone Family

If you edit one or more members of a clone family in an application other than Layout XL, you might need to verify the family to ensure its members are consistent with each other.

You can verify one or more clone families with a single button push. Layout XL checks the consistency of the clones in each family and takes the following actions.

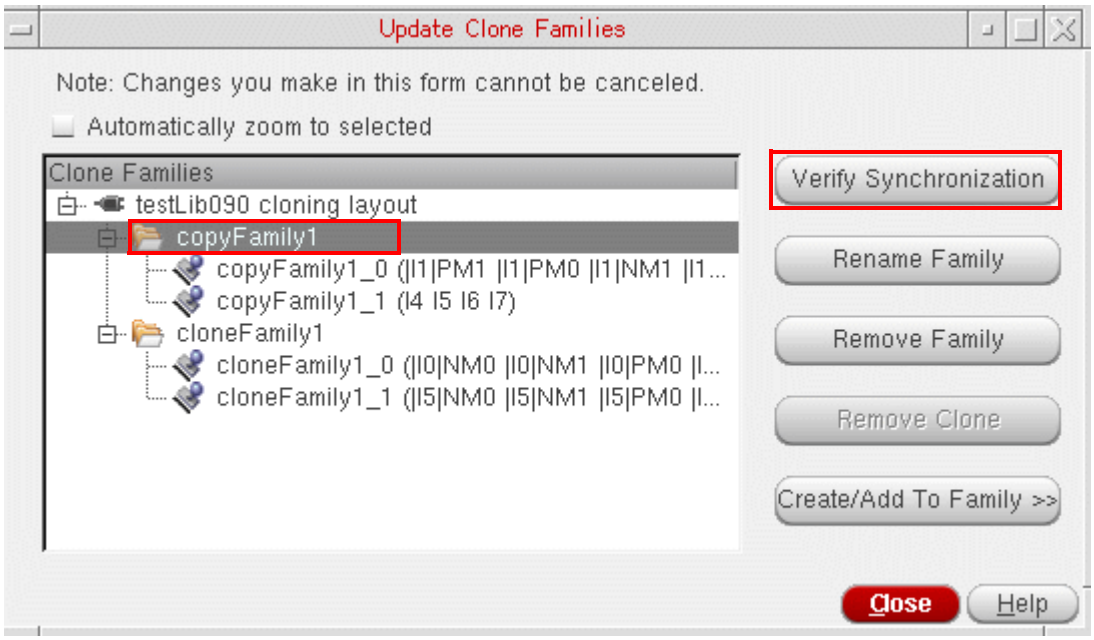
- Any clone that is out of sync with the other clones in the same family is removed from the family.
- If two or more clones are out of sync with the family but in sync with each other, they are moved to a new clone family. In this case, both the original family and the new family are renamed to follow the `orginalFamilyName_index` nomenclature.
- If all the clones in the family are different, the family is removed.

To verify that a clone family is synchronized:

1. From the layout window menu bar, choose *Connectivity – Update – Clone Families*.

© 2023 Cadence Design Systems, Inc.

The Update Clone Families form is displayed.



2. Select the clone family you want to verify and click *Verify Synchronization*.

   To verify that all the clone families are synchronized, click *Verify Synchronization* with none of the families selected.

   Layout XL modifies the clone families as outlined above and prints the results to the CIW.

3. Click *Close* to dismiss the Update Clone Families form.

Related Topics

Update Clone Families Form

Updating Clone Families

Renaming a Clone Family

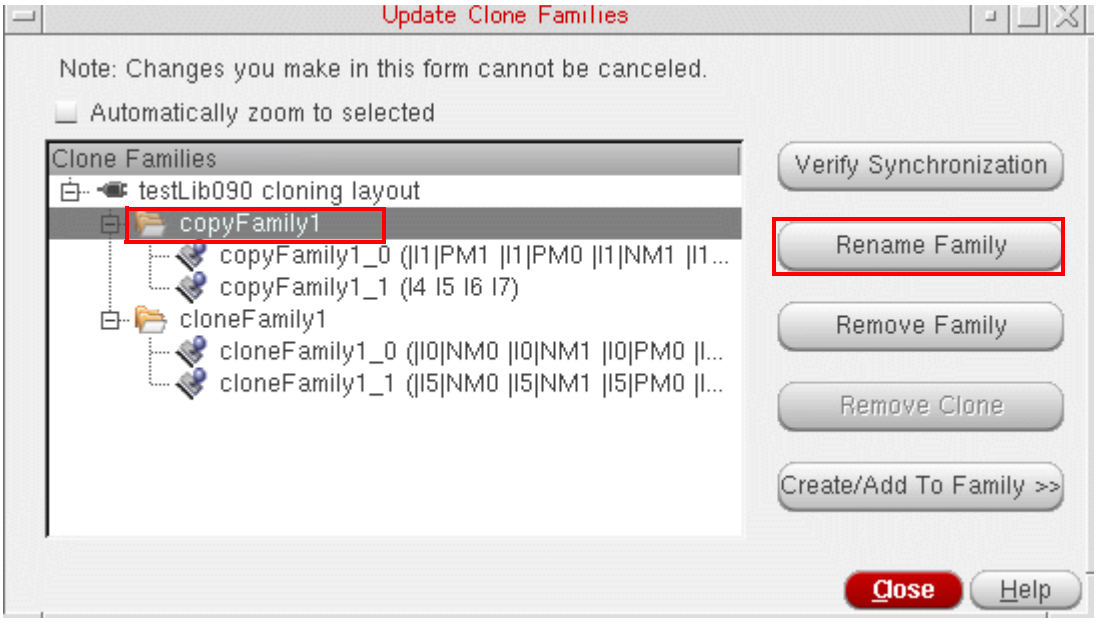
To rename a clone family,

1. From the layout window menu bar, choose *Connectivity – Update – Clone Families*.

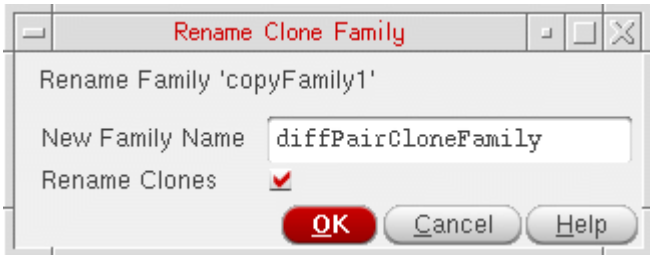
   The Update Clone Families form is displayed, as shown in the figure below.

© 2023 Cadence Design Systems, Inc.

2. Select the clone family to be renamed and click *Rename Family*.



The Rename Clone Family form is displayed, as shown in the figure below.

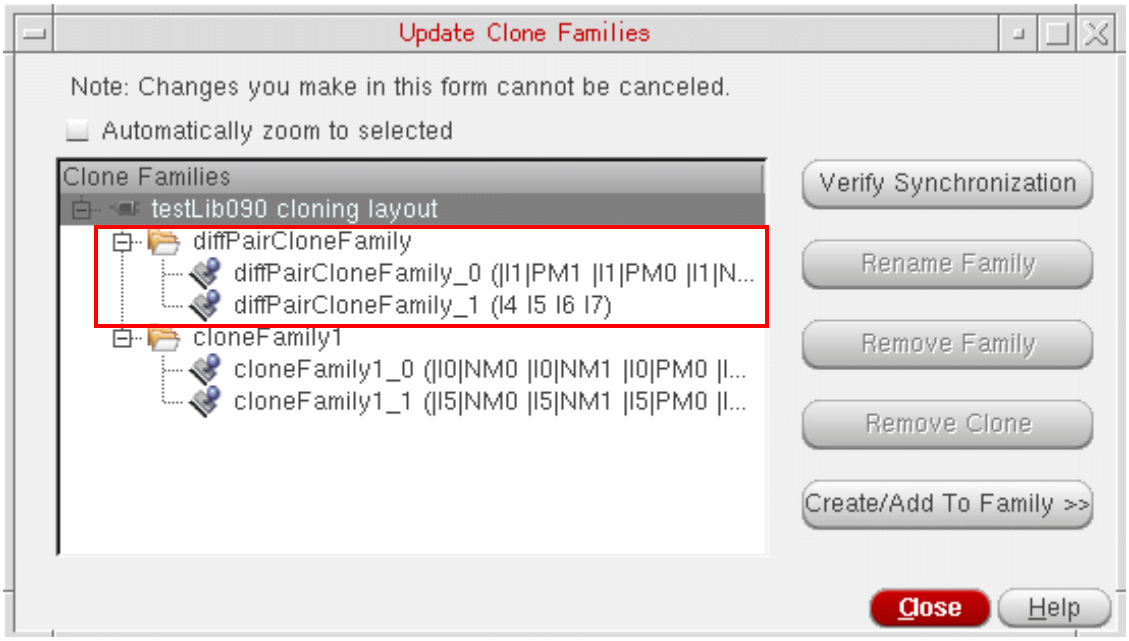


3. In the *New Family Name* field, type a new name for the clone family.
4. Select the *Rename Clones* check box if you also want to rename all the clone members using the new family name,

   Alternatively, set the environment variable, `renameClonesWhenFamilyIsRenamed`, to `t`.
5. Click *OK* to rename the clone family.

© 2023 Cadence Design Systems, Inc.

The selected clone family and the clone members (if set for renaming), display the new clone family base name, as displayed in the figure below.



Click *Close* to dismiss the Update Clone Families form.

Related Topics

Update Clone Families Form

renameClonesWhenFamilyIsRenamed

Updating Clone Families

Removing a Clone from a Family

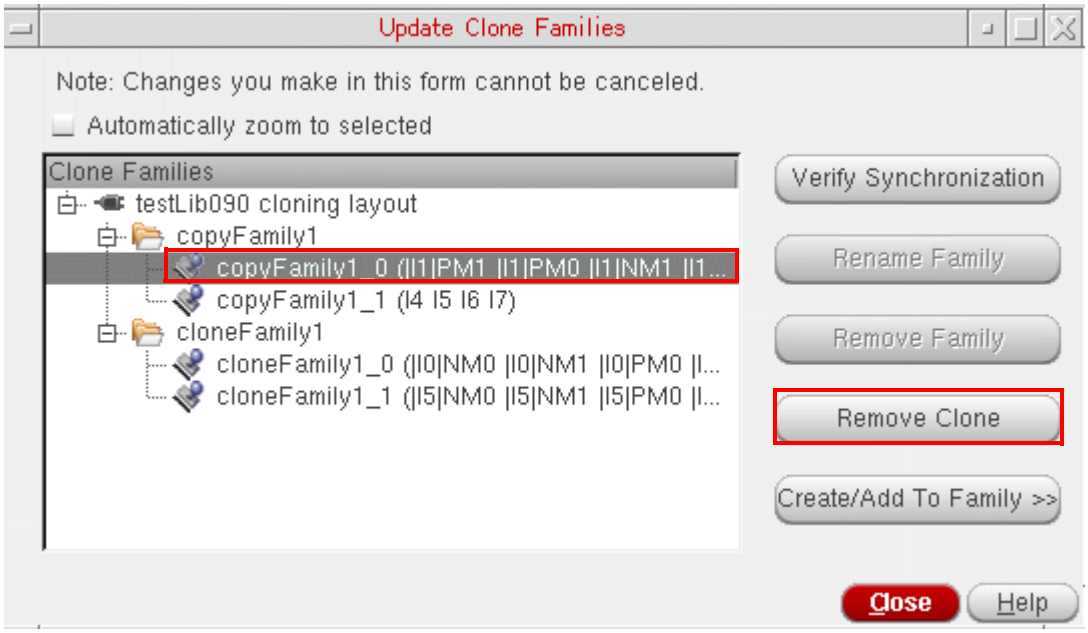
To remove a clone from a family:

1. From the layout window menu bar, choose *Connectivity – Update – Clone Families*.

   The Update Clone Families form is displayed.

© 2023 Cadence Design Systems, Inc.

2. Select the clone to be removed and click *Remove Clone*.



The selected clone is removed from the family. It is still present in the layout but is no longer synchronized with the other members of the family.

If you remove all except a single clone from a family, the family is automatically removed as well.

You cannot remove a clone if a member of its clone family is currently being edited in place. You need to exit the *Edit In Place* command first.

3. Click *Close* to dismiss the Update Clone Families form.

Removing a Clone Family from Design

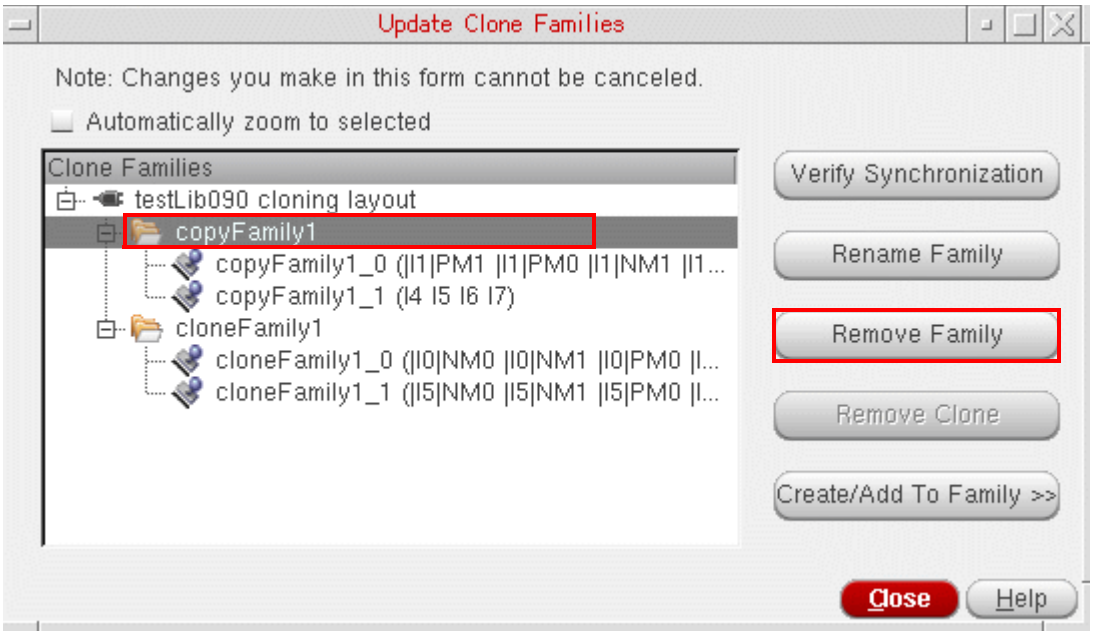
To remove a clone family from your design,

1. From the layout window menu bar, choose *Connectivity – Update – Clone Families*.

   The Update Clone Families form is displayed.

© 2023 Cadence Design Systems, Inc.

2. Select the family you want to remove and click *Remove Family*.



The clone family is removed. The clones it contained are still present in the layout but they are no longer synchronized.

You cannot remove a family if one of its members is currently being edited in place. You need to exit the *Edit In Place* command first.

3. Click *Close* to dismiss the Update Clone Families form.

Related Topics

Update Clone Families Form

Updating Clone Families

Creating a New Clone Family

To create a new synchronized clone family containing two or more of the layout groups currently in your design,

1. From the layout window menu bar, choose *Connectivity – Update – Clone Families*.

   The Update Clone Families form is displayed.

2. Click *Create/Add To Family*.

© 2023 Cadence Design Systems, Inc.

The form is expanded to include the options.



3. Ensure there is no existing family selected in the *Clone Families* pane, and then select at least two groups from the list of *Layout groups*.

   The *Synchronization reference* drop-down and the *Create Family* button are enabled.

   The *Layout groups* pane lists all the regular layout groups not currently assigned to a clone family. It does not list any other type of group. When you remove a clone from an existing group, its type is change to `none` and it is added to the list of layout groups. However, its name is left unchanged, which means that the list might contain names like `SynchronousClone123`. Such groups are **not** synchronous clones.

© 2023 Cadence Design Systems, Inc.

4. From the *Synchronization reference* drop-down, choose the reference group with which the others are to be synchronized.
5. Click *Create Family*.

   As long as the selected groups are valid clones of each other, the software creates the new family, synchronizes it against the reference group, and adds it to the *Clone Families* pane.

   Labels are not synchronized. Any dummy devices present in the reference group are copied in other occurrences of the clone.
6. Click *Close* to dismiss the Update Clone Families form.

Related Topics

Update Clone Families Form

Updating Clone Families

Adding a Group to an Existing Family

To add a layout group to an existing family,

1. From the layout window menu bar, choose *Connectivity – Update – Clone Families*.

   The Update Clone Families form is displayed.
2. Click *Create/Add To Family*.

   The form is expanded to include the options.
3. In the *Clone Families* pane, select the family you want to update,
4. In the *Layout groups* pane, select the group to be added to the family.

© 2023 Cadence Design Systems, Inc.

The *Synchronization reference* drop-down and the *Add To Family* button are both enabled.



The *Layout groups* pane lists all the regular layout groups not currently assigned to a clone family. It does not list any other type of group. When you remove a clone from an existing group, its type is change to `none` and it is added to the list of layout groups. However, its name is left unchanged, which means that the list might contain names like `SynchronousClone123`. Such groups are **not** synchronous clones.

5. From the *Synchronization reference* drop-down, choose one of the following.

   - The reference group with which all other members of the family are to be synchronized.

© 2023 Cadence Design Systems, Inc.

- The existing family, if the new group is to be changed to match existing family members.

6. Click *Add to Family*.

   As long as the selected group can be considered a clone of the other members in the family you selected, it is added to the family. A group containing only shapes and no instances can also be added.

   The family is updated in accordance with the synchronization reference, and the *Clone Families* pane is updated.

   Labels are not synchronized. Any dummy devices present in the reference group are copied in other occurrences of the clone.

7. Click *Close* to dismiss the Update Clone Families form.

Related Topics

Update Clone Families Form

Updating Clone Families

Synchronous Copy Generation

Synchronous Copy is generated when an object or a set of objects that exist in the layout is copied and the synchronization between the copy source and the copy target is maintained. The copy source and the copy target are each generated as a group in the layout view. Each group is called a synchronous copy and a set of such groups is called a synchronized family. Any changes made to the copy source are automatically reflected in the copy target and vice versa. For example, if you edit the routing in the copy source or the copy target, the change is replicated in the other group.

- Labels are not synchronized because these are *clone non-checked objects*.
- *Synchronous Copy* command is not supported in Transparent Group mode because in this mode, the figs inside the basic figGroups are selected, not the figGroup itself.

You use the *Create Synchronous Copy* check box available in the Create Cluster Boundary form or Save Short Locator form to create synchronous copies.

The synchronous copy generated by using the Create Cluster Boundary form or Save Short Locator form cannot be differentiated from the synchronous clones generated using the Generate Clones form. These clones differ only in the way they are generated.

© 2023 Cadence Design Systems, Inc.

Unlike the mechanism used when generating clones by using the Generate Clones form, *Create Synchronous Copy* provides a synchronous grouping feature that does not involve searching the schematic for an appropriate match. Therefore, you can create a synchronous group of objects that exist only in the layout. For example, you can synchronously copy some routing objects, which are objects that exist only in the layout, to create a synchronous routing group. You can also use a single device, a set of devices, or a chained set of devices that exist in the layout, as the copy source. The copy source can contain the entire chain or a part of it.

Copying from One Cellview to Another

If the target cellview is different from the source cellview, synchronous copy cannot be created.

If you run the *Copy* or *Repeat Copy* command with *Create Synchronous Copy* selected to create a copy target in a cellview other than the cellview in which the source is selected, synchronous copies are not created. However, a regular copy of the source is created in the source cellview and the following warning message is generated:

“Cannot create synchronous clones because the source and the target clones are not in the same cellview. The clone is created as a free object instead.”

Related Topics

Clone Non-Checked Objects

Using Transparent Group Mode

Create Cluster Boundary Form

Save Short Locator View Form

Generate Clones Form

Generating a Synchronous Copy

To generate a synchronous copy:

1. In the layout canvas or Navigator, select the copy source for which you want to create a synchronous copy.

© 2023 Cadence Design Systems, Inc.

2. Choose *Edit – Copy*, and then press `F3`.

   The *Copy* form displays.

   Alternatively, you can right-click the copy source in the layout canvas and choose *Copy*. Press `F3` to display the *Copy* form.

3. Select the *Create Synchronous Copy* check box to create a copy that is synchronized with the source.

   - By default, the *Create Synchronous Copy* check box is deselected every time you open the *Copy* form.
   - If you do not select the *Create Synchronous Copy* check box, a copy of the selected source is created, but it is not synchronized with the source. The only exception to this is observed when you choose a synchronous clone or a synchronous copy as the copy source. In this case, the copy target is a synchronous copy even if the *Create Synchronous Copy* check box is not selected.

4. Select other copy options, as appropriate.
5. Click *Hide* to close the form.
6. Select the area on the layout canvas where you want to place the copy target.

   The copy source and the copy target are added to a synchronized family, which can be accessed by using the Navigator.

Important

You can also generate a synchronous copy by using the *Create Synchronous Copy* check box on the *Repeat Copy* form. To open the form, choose *Edit – Repeat Copy*, and then press `F3`.

Related Topics

Synchronous Copy with syncClone Source Group

Save Short Locator View Form

Generating Synchronous Copy (video)

© 2023 Cadence Design Systems, Inc.

Managing a Clone Family

You can perform the following tasks to manage a clone family:

- Selecting a Clone Family
- Updating a Clone Family
- Renaming a Clone Family
- Removing a Clone from a Family
- Removing a Clone Family

Selecting a Clone Family

To select a clone family:

➡ In the layout canvas or Navigator, right-click a clone and choose *Select Clone Family.*

All the clones in the family of the chosen clone are selected in the layout canvas and also in the Navigator.

Alternatively, you can use the `lxSelectSynchronousFamily` SKILL function to select a clone family.

Updating a Clone Family

To update a clone family:

➡ In the layout canvas or Navigator, right-click a clone and choose *Update Clone Families*.

Alternatively, in the layout window, choose *Connectivity – Update – Clone Families*.

The Update Clone Families form is displayed. You can use this form to rename the family of the selected clone, verify the synchronization of the family members, or to make other similar updates to the family of the selected clone.

The *Create Family* and *Add To Family* commands are not available when updating a synchronous copy.

Any changes that you make by using the Update Clone Families form are irreversible.

© 2023 Cadence Design Systems, Inc.

Renaming a Clone Family

To rename a clone family:

1. From the layout window menu bar, choose *Connectivity – Update – Clone Families*.

   The Update Clone Families form is displayed.

2. Select the clone family to be renamed and click *Rename Family*.

   The Rename Clone Family form is displayed.

3. In the *New Family Name* field, type in a new name for the clone family.

4. Select the *Rename Clones* check box if you also want to rename all the clone members with the new family name as prefix, such as `<Clone Family Name>_index`,

   Alternatively, set the environment variable, `renameClonesWhenFamilyIsRenamed`, to `t`.

5. Click *OK* to rename the clone family.

   The selected clone family and the clone members (if set for renaming), display the new clone names.

6. Click *Close* to dismiss the Update Clone Families form.

Removing a Clone from a Family

To remove a clone from its family:

➡ In the layout canvas or Navigator, right-click the clone that you want to remove from its family and choose *Remove Clone From Family*. The option is available only when the cellview is opened in Edit mode.

The selected clone is removed from its family, which means that the clone is no longer synchronized with the other family members.

Alternatively, you can use the `lxRemoveSynchronousCloneFromFamily` SKILL function to remove the selected clone from its family, or use the *Remove Clone* button available in the Update Clone Families form.

© 2023 Cadence Design Systems, Inc.

Removing a Clone Family

To remove a clone family:

1. In the layout canvas or Navigator, right-click a clone that belongs to the clone family you want to remove and choose *Update Clone Families*. The option is available only when the cellview is opened in Edit mode.

   The Update Clone Families form is displayed.

2. Select the clone family from the list of families and click the *Remove Family* button.

   The selected clone family is removed, which means that the family members are no longer synchronized.

   Alternatively, you can use the `lxDeleteSynchronousCloneFamily` SKILL function to remove the selected clone family.

Relate Topics

lxSelectSynchronousFamily

lxRemoveSynchronousCloneFromFamily

lxDeleteSynchronousCloneFamily

renameClonesWhenFamilyIsRenamed

Update Clone Families Form

Synchronous Copy with syncClone Source Group

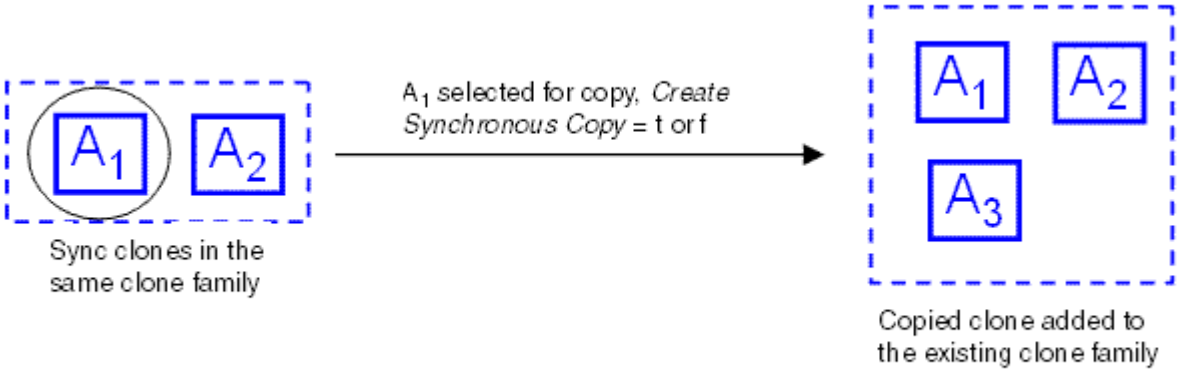
You can use a synchronous clone as the copy source to create a new clone. The copy target is a group of type “syncClone” and is added to the same synchronized family as the copy source. However, depending on the number of synchronous clones you choose to copy, the result of the *Copy* command varies.

Copying a Single Synchronous Clone

If you select a single synchronous clone as the copy source, the resulting copy is a synchronous copy, irrespective of the state of the *Create Synchronous Copy* check box. The copy target is added to the synchronized family to which the copy source belongs.

© 2023 Cadence Design Systems, Inc.

For example, you have two synchronous clones, A_1 and A_2, in a clone family. If you copy either of the synchronous clones, a new synchronous clone, A_3, is added to the synchronized family even if the *Create Synchronous Copy* check box is not selected. The result is the same if the *Create Synchronous Copy* check box is selected.



Converting a Synchronous Clone Group to a Basic Group

To convert a synchronous clone group to a basic group—a group of type “none”:

➡ Right-click the synchronous clone and choose *Remove Clone From Family* from the shortcut menu.

Alternatively, you can remove a clone from its family by using the *Remove Clone* button on the Update Clone Families form.

The selected synchronous clone is removed from its family. As a result, the group type changes from “syncClone” to “none” and the group is no longer synchronized with the other members of the family.

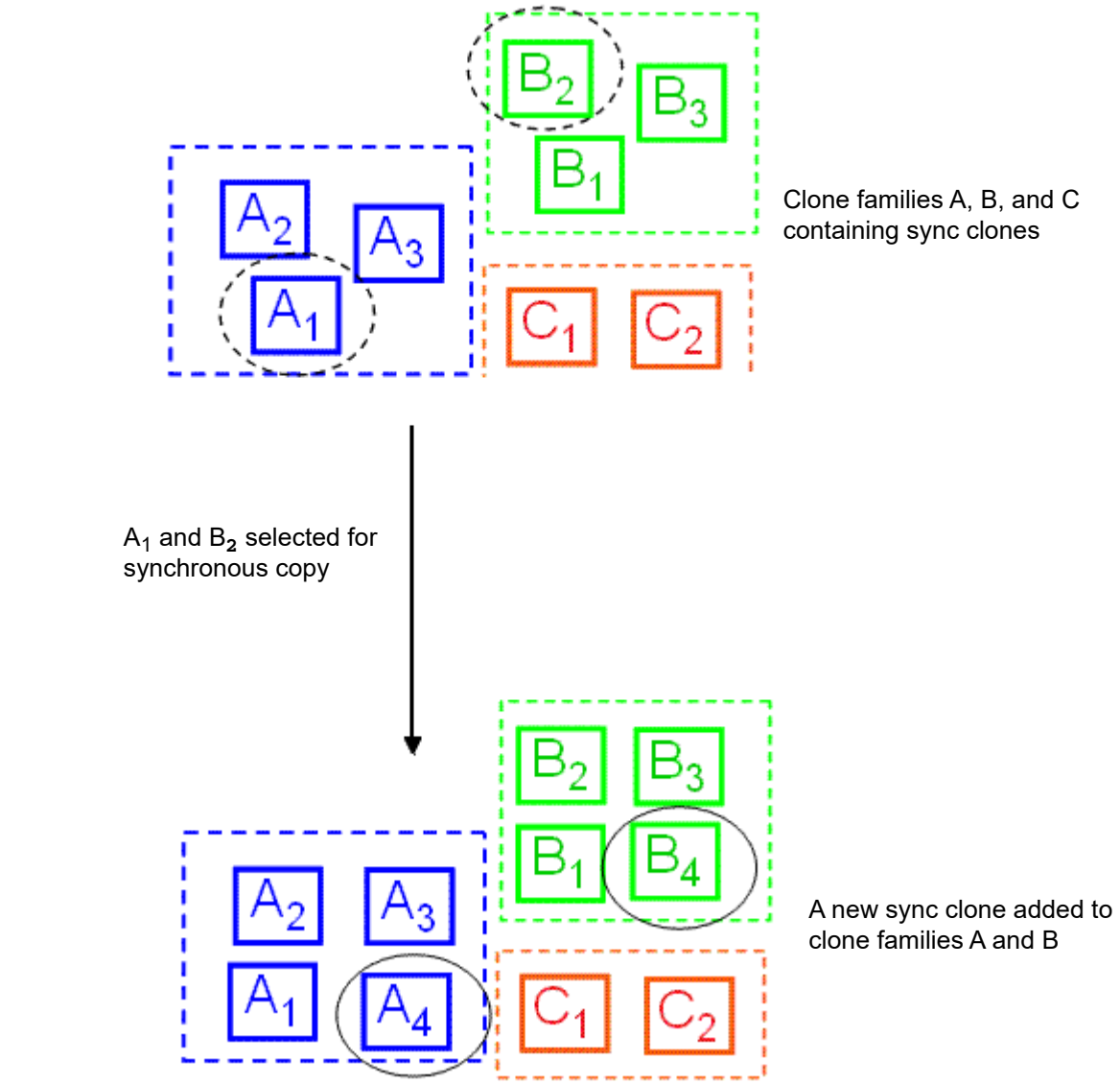
Copying Multiple Synchronous Clones

If you select multiple synchronous clones from a single synchronized family or from multiple families, Layout XL processes the request based on the state of the *Create Synchronous Copy* check box.

Case 1: Copying Multiple Synchronous Clones with Create Synchronous Copy Deselected

© 2023 Cadence Design Systems, Inc.

Consider three clone families: A, B, and C. If you copy the synchronous clones A_1 and B_2, as shown in the figure below, with the *Create Synchronous Copy* check box deselected, a copy of each selected clone is created in its own family.



Case 2: Copying Multiple Synchronous Clones with Create Synchronous Copy Selected

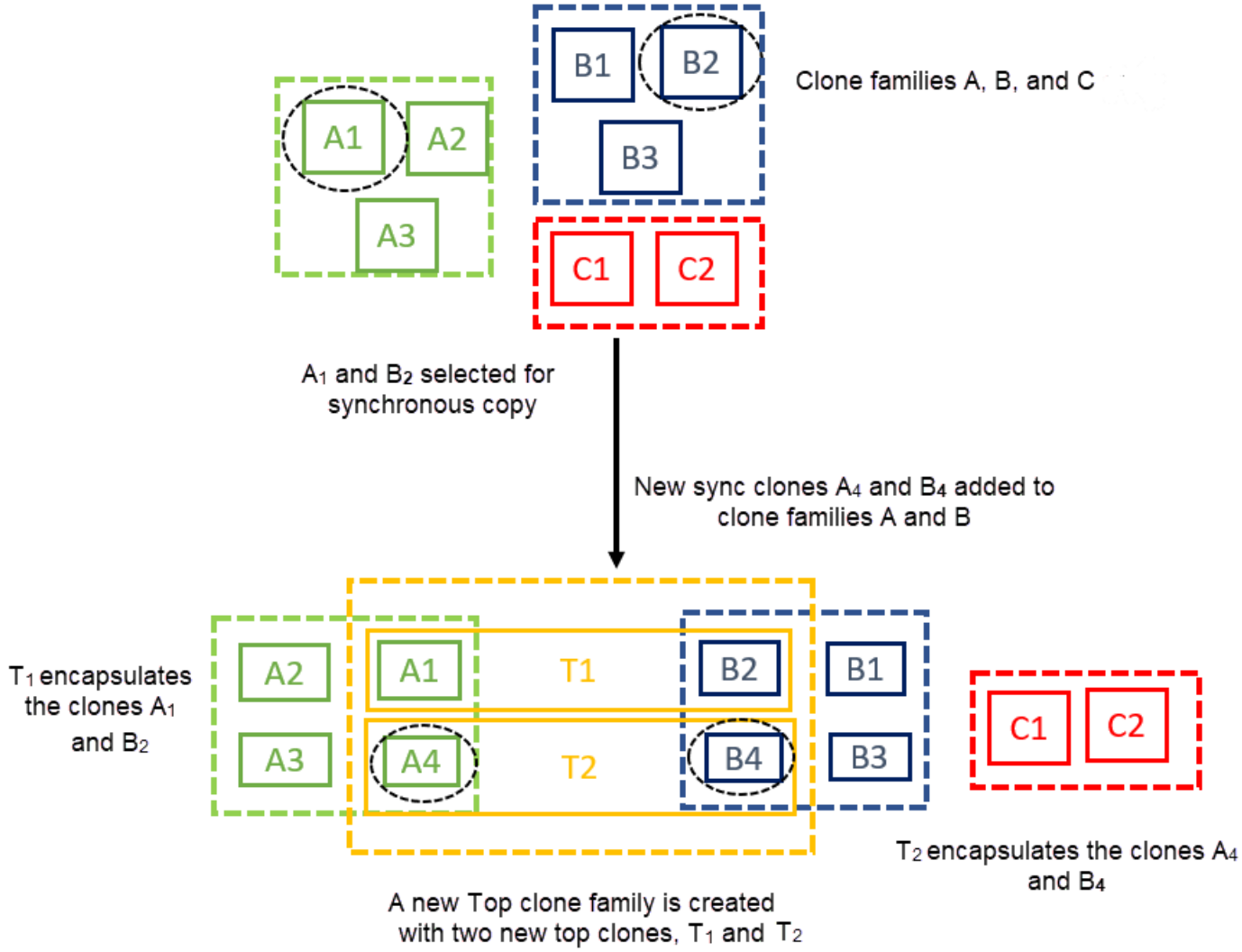
If you select the *Create Synchronous Copy* check box while copying multiple synchronous clone sources, a copy of each selected source is added in its own family.

© 2023 Cadence Design Systems, Inc.

In addition, a new Top clone family is created with two new top clones T_1 and T_2.

A clone T_1 is created and it encapsulates the clones A_1 and B_2. This means that, a clone is created around the copy source.

A clone T_2 is created and it encapsulates the clones A_4 and B_4. This means that, a clone is created around the copy target.



Related Topics

Removing a Clone from a Family

© 2023 Cadence Design Systems, Inc.

Update Clone Families Form

Updating Clone Families

Synchronous Copy with Other Source Groups and Different Source Objects

Depending on the group type that is selected for generating a synchronous copy, the generated copy may or may not be synchronous with the source. However, a basic copy of the copy source is always created.

- **Basic Groups**

  If you select a basic group—a group of type “none*”*—as the copy source and *Create Synchronous Copy* is not selected, a regular copy of the object is created and the resultant object is also of group type “none*”*. However, if *Create Synchronous Copy* is selected, both the copy source group and the copy target group become synchronous copies and their group type changes from “none” to “syncClone”. The copy source and the copy target are gathered in a new synchronized *family*.

- **Non-Basic Groups**

  Non-basic groups include groups such as custom row and placement area. If the copy source contains a non-basic group and the *Create Synchronous Copy* check box is selected, synchronous copy fails and a regular copy of the group is created instead. The following warning message is generated indicating that the selected copy source is not supported for synchronous copying:

“The copy source contains a group with an unsupported type; the copy cannot be synchronized.”

Synchronous copy with different source objects:

- **Microwave Objects**

  If the copy source is a microwave object and the *Create Synchronous Copy* check box is selected, the copy source and the copy target are generated as synchronous copies. However, synchronization of microwave objects is currently not supported. Therefore, edits made on one microwave object may not be reflected in the other copies.

- **PinFig Objects**

  If the copy source is a pinFig and the *Create Synchronous Copy* check box is selected, the copy source and the copy target are generated as synchronous copies. Any edits made to one of the pinFigs is replicated in all the other copies.

© 2023 Cadence Design Systems, Inc.

- **Mosaics**

  If the copy source is a mosaic and the *Create Synchronous Copy* check box is selected, the copy source and the copy target are generated as synchronous copies. Any edits made to one of the mosaics is replicated in all the other copies.

- **Rod Objects**

  If the copy source is a rod object and the *Create Synchronous Copy* check box is selected, the copy source and the copy target are generated as synchronous copies. Any edits made to one of the objects is replicated in all the other copies.

- **P&R Objects**

  P&R objects include blockages, rows, placement area, clusters, PR boundary, snap boundary, and area boundary. Most of these objects, except some, such as PR boundary and snap boundary, can be synchronously copied using the *Copy* or *Repeat Copy* commands. If the copy source contains an unsupported object and the *Create Synchronous Copy* check box is selected, a warning message is generated and the object is not copied. Only those objects that are effectively copied are added to the source synchronous copy.

- **Bound Instances**

  If the copy source is a bound instance and you run the *Copy* command with the *Create Synchronous Copy* check box selected, a synchronous copy is created. If you run the *Copy* command with the *Create Synchronous Copy* option deselected, a regular copy of the source is created.

  When generating a synchronous copy using the *Create Synchronous Copy* option, no binding is created between the copy target instances created in the layout and the existing devices in the schematic. To create synchronous copies in the layout that are bound to the existing schematic devices, you must use the Generate Clones form instead.

- **Steiners and Guides**

  Steiners and guides are Clone Non-Checked Objects. These objects are not supported by the *Copy* and *Repeat Copy* commands.

  If the copy source contains a steiner or a guide, the copy operation fails and an appropriate warning message is generated, indicating that the selected source type is unsupported.

© 2023 Cadence Design Systems, Inc.

Clone Non-Checked Objects

Clone non-checked objects are non-synchronized objects such as labels. When only clone non-checked objects are selected as the copy source, synchronous copies are not created. However, these objects, except some, such as PR boundary, are copied by the *Copy* and *Repeat Copy c*ommands.

For example, if you select an instance and a label as the copy source, both the objects are synchronously copied when you run a *Copy* or *Repeat Copy* command with the *Create Synchronous Copy* check box selected. However, if you edit, move, or delete the label in one copy, the changes are not reflected in the corresponding label in the other copy because the labels in the two copies are not synchronized.

Related Topics

Generate Clones Form

Clone Non-Checked Objects

Synchronous Cloning SKILL Functions

The following is a list of SKILL functions that are supported by the Synchronous Cloning engine.

SKILL Function Name	Description
lxDeleteSynchronousCloneFamily	Deletes the specified family of synchronous clones. As a result, the member clones are desynchronized and they behave as basic groups.
lxGetCloneFamilyNam	Returns the family name of the specified clone.
lxGetOtherClonesInFamily	Returns a list of clones in a family other than the clone specified.
lxRemoveSynchronousCloneFromFamily	Removes a specified synchronous clone from its family. As a result, the clone is desynchronized and it behaves as a basic group.

© 2023 Cadence Design Systems, Inc.

SKILL Function Name	Description
lxSelectSynchronousFamily	Selects all the clones in the layout canvas and the Navigator that are members of the specified clone family. If all the family members are already selected, no more clones are selected on the canvas or in the Navigator. If any new clones are identified, they are selected on the canvas and in the Navigator.
lxSetCloneFamilyName	Sets a new name for the family to which the specified clone belongs.
lxVerifyCloneFamily	Runs a consistency check on the specified synchronous clone family to verify if the family members are identical to each other.

Dummy Instance Generation

Dummy instances are created in the layout to counter electrical effects that are observed at small geometries. These instances get their connectivity from the layout net to which they are tied.

All the terminals of a dummy instance, except its bulk terminal, are connected to the tie net. The bulk terminal, wherever present, is connected to the same bulk net as that used by the instance from which the dummy is created.

Depending on the purpose of the net to which a dummy instance is tied, the dummy instances can be categorized as:

- **Active dummies –** dummies that are connected to a net other than power or ground
- **Inactive dummies –** dummies that are tied to a power or ground net

Because dummies get their category—active or inactive—depending on the net to which they are connected, it is necessary to specify which nets are power and ground nets before you create dummies.

If you choose to add active dummies to your design, it is advisable to take these dummies to the schematic—also called backannotation—to ensure no mismatches are reported during a layout versus schematic run. Inactive dummies, on the other hand, can be backannotated optionally.

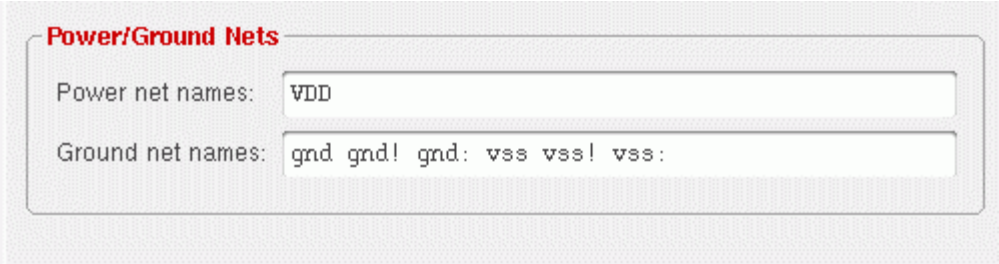
© 2023 Cadence Design Systems, Inc.

Power and Ground Nets Specification for Dummy Instances

To assign a dummy instance to an appropriate net during generation, you must specify which nets should be used as power and ground nets. You can do this by setting the `lxSupplyNetNames` and `lxGroundNetNames` environment variables, respectively.

Alternatively, you can specify the power or ground nets by using one of the following:

- The *Power net names* and *Ground net names* fields in the *Connectivity form*.



- SKILL commands that search for a regular expression on the substrings in net names and then assign values to `lxSupplyNetNames` and `lxGroundNetNames`.

The SKILL example below demonstrates the use of regular expression style matching of substrings when specifying the power and ground nets in a design.

You must run the SKILL code below after the design is loaded to initialize the power and ground net environment variables.

```
procedure( companyPowerNet(net)
    ciIsNet(net "company_power")
)
procedure( companyGroundNet(net)
   ciIsNet(net "company_ground")
)
procedure( companyPowerNetNames(cv)
      let( (res)
          res = ""
          foreach(net cv~>nets
              if( companyPowerNet(net)
                  sprintf(res "%s %s" res net~>name)
              )
)
          res
      )
)
```


© 2023 Cadence Design Systems, Inc.

```
procedure( companyGroundNetNames(cv)
      let( (res)
          res = ""
          foreach(net cv~>nets
              if( companyGroundNet(net)
                  sprintf(res "%s %s" res net~>name)
              )
          )
          res
     )
)
procedure( initPowerAndGroundNetNames(cv)
      ciRegisterNet( "company_ground" nil
                ?regexNetNames '("vss" "VSS" "gnd" "GND" "ground" "GROUND" ) )
      ciRegisterNet( "company_power" nil
                    ?regexNetNames '("vdd" "VDD!" "vcc!" "VCC!") )
envSetVal("layoutXL" "lxSupplyNetNames" 'string companyPowerNetNames(cv))
envSetVal("layoutXL" "lxGroundNetNames" 'string companyGroundNetNames(cv))
)
```

After the SKILL code is loaded, set up the power and ground nets by calling:

```
initPowerAndGroundNetNames(cv)
```

To verify if the SKILL code was successfully initialized, use the following commands:

```
envGetVal("layoutXL" "lxSupplyNetNames")
envGetVal("layoutXL" "lxGroundNetNames")
```


© 2023 Cadence Design Systems, Inc.

Dummy Instance Properties

The following properties are associated with Layout XL dummy instances.

Property	Description
`lxBackAnnotated`	Added to the schematic dummy instances that are created as a result of backannotation. During Check Against Source, if regular or modgen dummies have `lxBackAnnotated` property specified in the schematic, they are skipped when names are matched. See Dummy Instances Backannotation.
`lxDummy`	Identifies an instance as a dummy instance.
`lxDummyNet`	Name of the power or ground net that is used to connect all the instance terminals during the creation of a dummy instance. This net is also used to restore the connectivity when a device is unabutted.

Related Topics

Creating Dummy Instances

Support for Flexible Connectivity in Dummy Instances

Dummy Instances Backannotation

Net Voltages BackAnnotation

lxSupplyNetNames

lxGroundNetNames

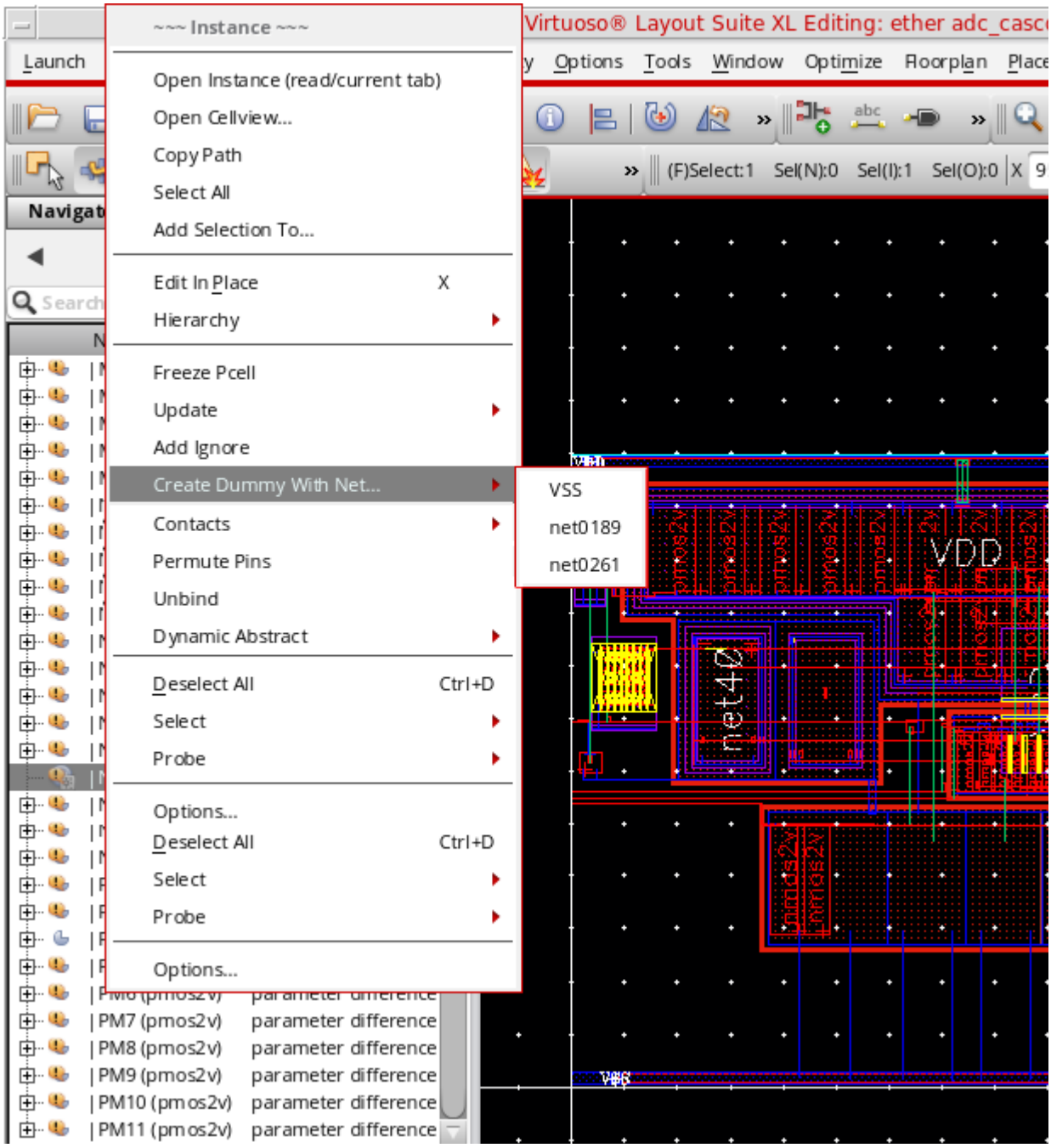
Creating Dummy Instances

Layout XL enables you to create dummies for layout instances of different types, such as MOS devices, FinFET devices, resistors, capacitors, and mosaics. You can create a dummy instance using the *Create Dummy with Net* shortcut command. You can then backannotate the dummies to the schematic, if required.

© 2023 Cadence Design Systems, Inc.

To create a dummy instance of a selected layout device:

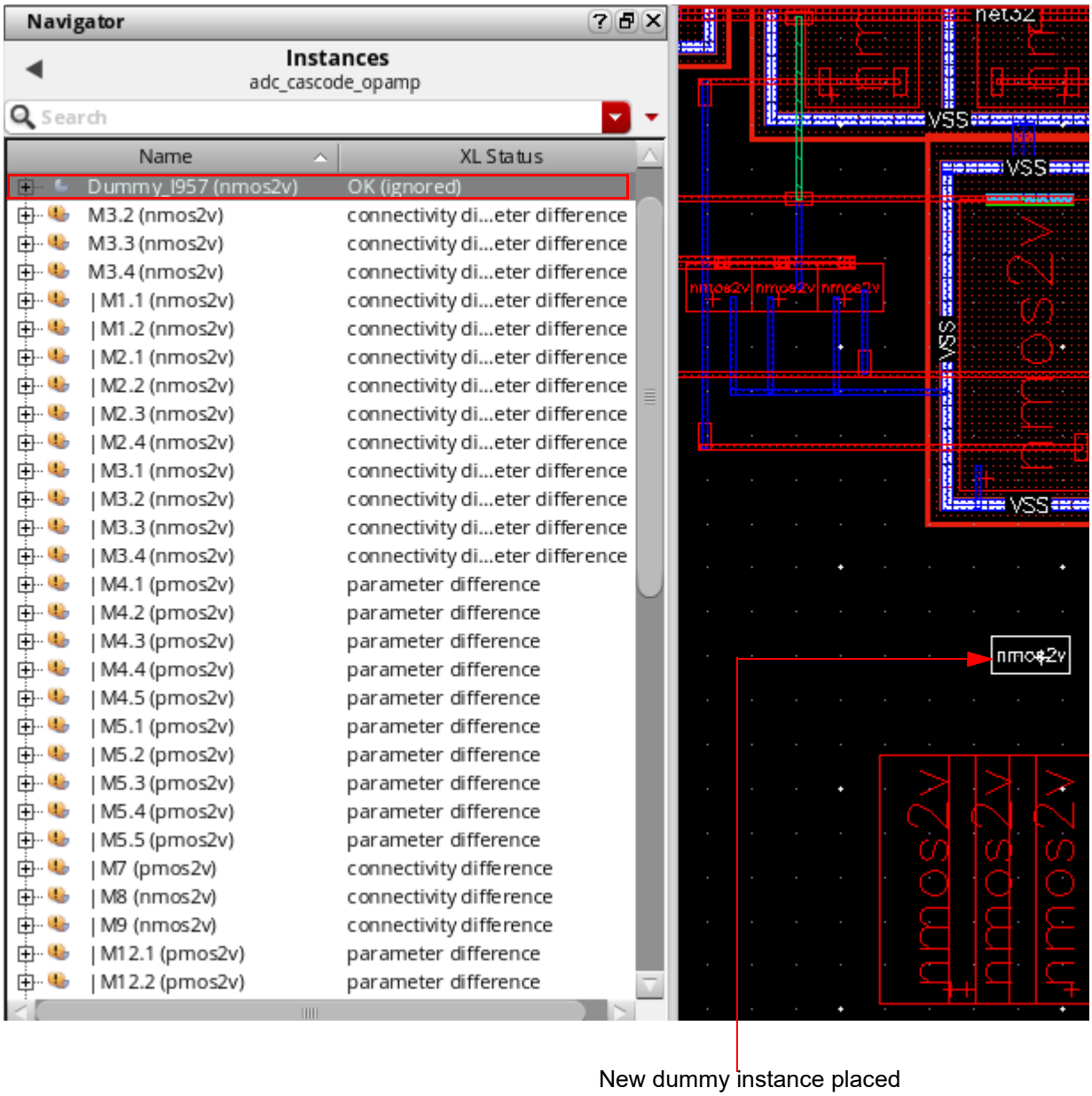
1. In the Layout XL canvas, right-click the selected instance (or multiple instances) for which you want to create a dummy and choose the *Create Dummy with Net* command. Then, select the net with which you want to associate your dummy. The list of nets contains power and ground nets and nets connected to the selected instances.



© 2023 Cadence Design Systems, Inc.

2. Select the reference point on the selected instance that you want to use for creating the dummy instance on the layout canvas.
3. Select the area on the layout canvas where you want the dummy instance to be created. If you select the *Infix* mode, you need to specify only the final position for the dummy instance, not the reference point.

   A dummy instance is created, as shown below, at the target location that you indicated.



© 2023 Cadence Design Systems, Inc.

Alternatively you can use the `lxMakeDummy` SKILL function to create a dummy with a net.

The new dummy that you created can be quickly identified in the Navigator by the prefix, *Dummy*, in its name.

Converting an Instance to Dummy

To convert a selected instance to a dummy instance, right-click the selected instance and choose *Add Dummy Props*.

Converting a Dummy to an Instance

To convert a selected dummy instance to a instance, right-click the selected instance and choose *Remove Dummy Props.*

Related Topics

Dummy Instance Generation

lxMakeDummy

Support for Flexible Connectivity in Dummy Instances

Flexible connectivity in the context of dummy instances means that these devices freely abut to instance terminals, irrespective of the net to which they were originally connected. If unabutted, the device restores its original connectivity.

The flexible connectivity feature supported by dummy instances makes them useful design objects for the following:

- Abutment
- Chaining

Abutment

Dummy instances can change their source, drain, or bulk connectivity according to that of the instance terminal that overlaps them. These instances can also restore their original connectivity by unabutting from the instance terminal that was overlapping them. This flexible connectivity feature makes dummies available for abutment (and unabutment).

© 2023 Cadence Design Systems, Inc.

Chaining

Chaining allows you to control whether a dummy instance can change its net on one side or both. The default behavior is to allow the dummy instances to change their nets on one side through Abutment.

To enable the nets on both sides of a dummy instance to be changed, you must set the `chainDummyAbutment` environment variable.

Layout XL also supports chaining dummy instances with floating S/D nets.

Related Topics

Device Abutment

Chaining Dummy Instances

To chain dummy instances:

1. Choose *Connectivity – Generate Chained Devices*.

   The *Generate Chained Devices* form is displayed.



2. Select the *Use Device Order* check box. This ensures that the dummy instances are treated as regular devices during the chaining operation.

   If you do not select the *Use Device Order* check box, the dummy instances are ignored during chaining, and therefore, are left unchained.

3. Select the other appropriate options on the form, as required.
4. Click the *Show Device List* button and select the devices to be chained.

© 2023 Cadence Design Systems, Inc.

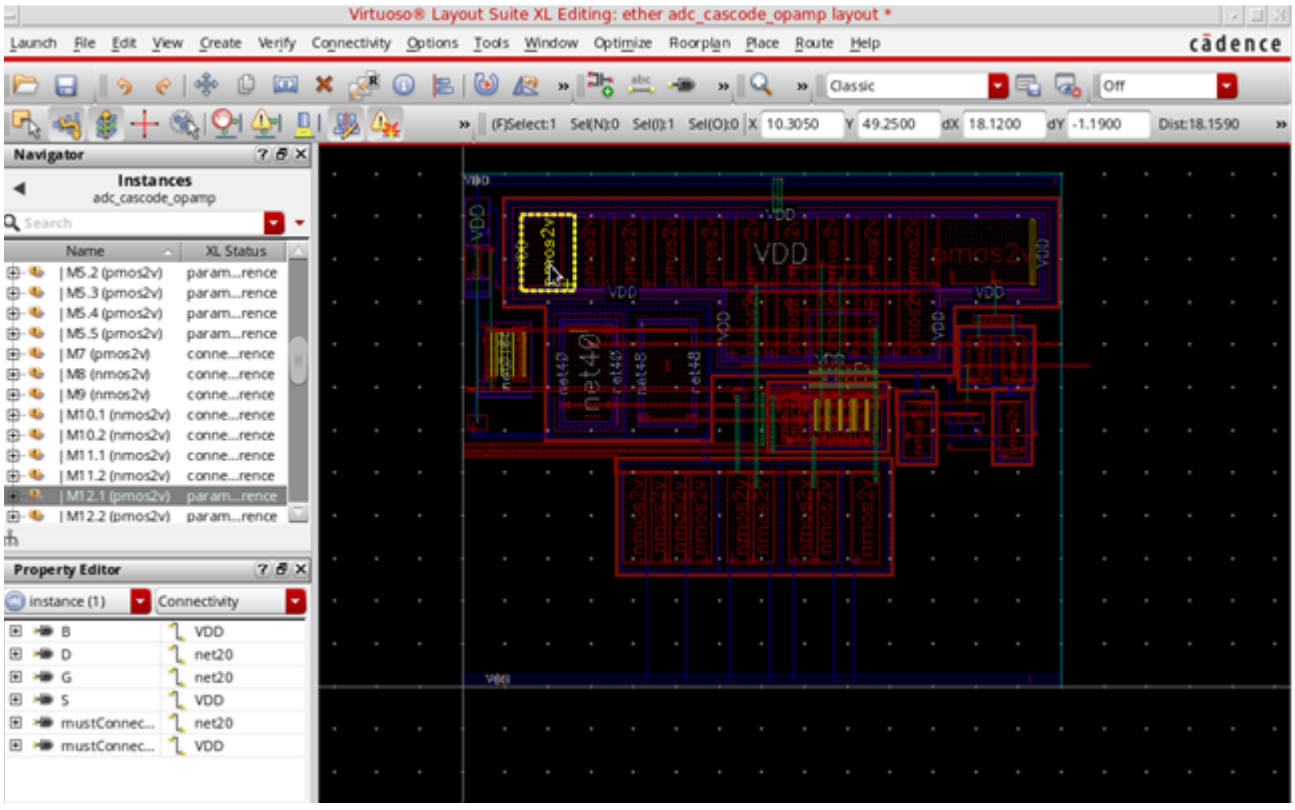
To select multiple devices, hold the *Ctrl* button while you select the devices.

5. Click *Apply*.

   The chained devices are available for placement in the layout.

Example – Demonstrating Flexible Connectivity on Dummy Instances

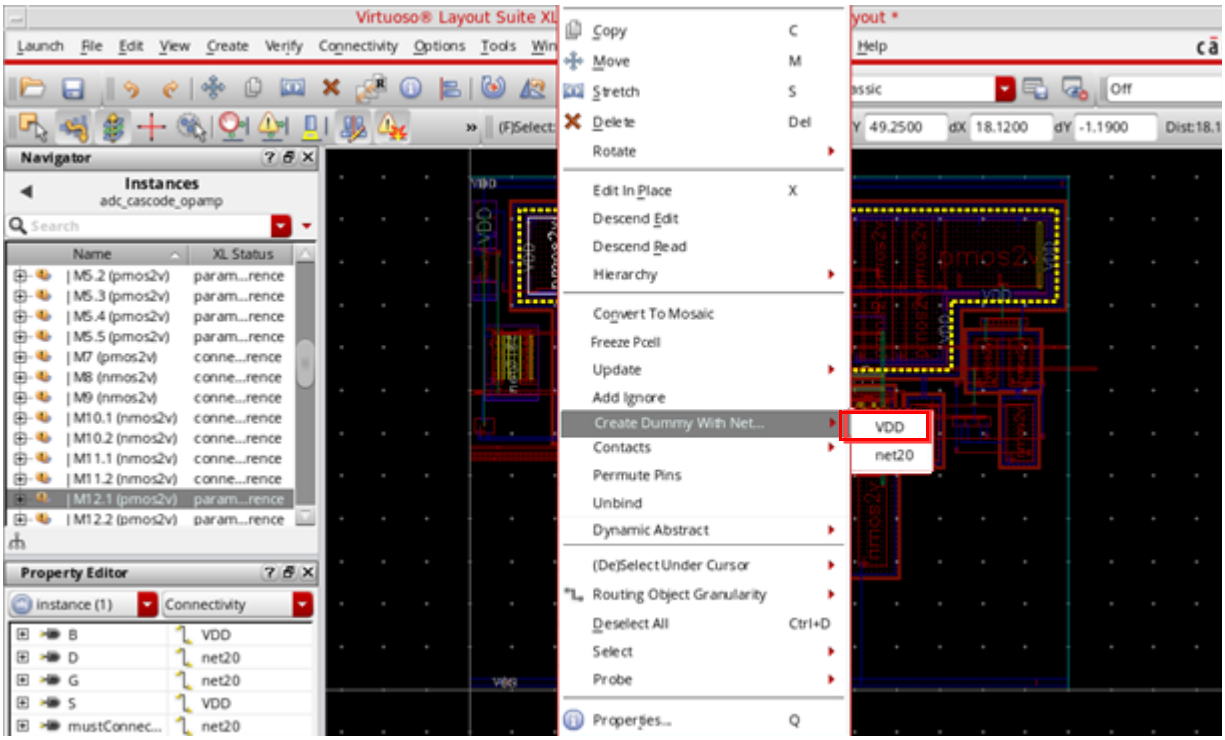
Let us consider the scenario displayed in the figure below to see how dummy instances support flexible connectivity. Let us first create a dummy instance for the pmos device highlighted in the figure below. For this scenario, we will create a dummy for pmos device, `|M12.2`, using net `VDD`.



1. Select the pmos device and use the *Property Editor – Connectivity* property category selector to view the original connectivity of the device. As displayed in the figure above, the selected device has its Drain, *D*, connected to `net20`.

© 2023 Cadence Design Systems, Inc.

2. Right-click the highlighted pmos device and choose *Create Dummy With Net – VDD* from the shortcut menu, as displayed in the figure below.



3. Click the pmos device in the layout canvas again to derive the reference point for dummy generation.

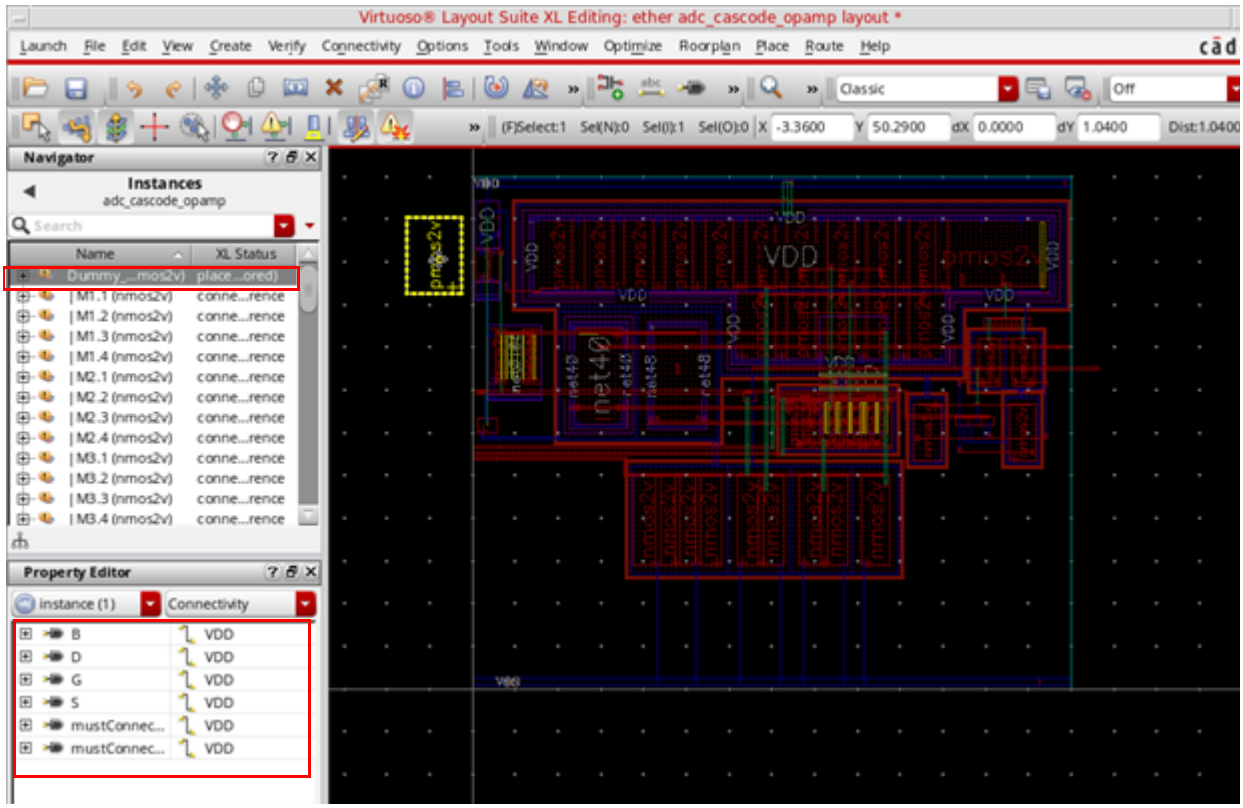
   A copy of the device now follows the mouse pointer.

   If you select the *CIW – Options – User Preferences – Infix (No Click Necessary for First Point)* check box prior to invoking the dummy generation command, a copy of the selected device follows the mouse pointer as soon as the net for creating the dummy is selected.

4. Click anywhere in the layout to place the dummy instance.

© 2023 Cadence Design Systems, Inc.

A dummy instance of the selected device is now generated. For this scenario, the dummy instance is listed in the Navigator assistant as displayed in the figure below.



5. In the Navigator assistant or the layout canvas, select the dummy instance to view its connectivity.
6. In the Property Editor assistant, choose the *Connectivity* property category to view the connectivity of the selected dummy instance. For this scenario, the dummy instance appears connected to *VDD*—the net that was used to create the dummy instance.
7. Move the dummy instance around such that it overlaps another layout device and abuts to it.
8. In the Property Editor assistant, the connectivity of the dummy instance now appears changed to that of the device that the dummy has abutted.
9. Move the dummy instance away from the device it had abutted with such that the dummy instance is not abutted any more.
10. In the Property Editor assistant, the connectivity of the dummy instance now appears same as that before abutment.

© 2023 Cadence Design Systems, Inc.

The example above illustrates that dummy instances take the connectivity of the device that they abut with. When a dummy instance is no longer abutted, it restores its original connectivity.

Related Topics

Device Abutment

chainDummyAbutment

Demonstrating Flexible Connectivity Support of Dummy Instances (video)

© 2023 Cadence Design Systems, Inc.

Dummy Instances Backannotation

Dummy instances are backannotated to the schematic to keep the two views synchronized. The criteria for backannotation are that the layout dummy instance does not have a corresponding schematic instance and the layout instance has the `lxDummy` property set to `TRUE`. Once backannotated, the `ignore` property on the layout dummy is removed. If there are multiple layout dummies and they have the same master, parameters, and connectivity, they can optionally be backannotated as single devices or as m-factored or iterated instances. To do this, you can set the `backAnnotateInstances` environment variable to `"mfactored"` or `"iterated"`, which applies to the interactive backannotation command, and set the `lxDummyBackAnnotateMFactor` environment variable to `t`, which applies to the BackAnnotate All Dummies command.

For each dummy instance that is backannotated, a corresponding symbol is created in the schematic view. However, if the backannotated instances have the same master or connectivity, you can choose to backannotate them as mfactored dummy symbols. To do this, you must set the `lxDummyBackAnnotateMFactor` environment variable to `t`.

Backannotating and Customizing All Dummy Instances

To backannotate and customize all dummy instances:

1. Choose *Connectivity – Back Annotate – Dummy Instance.*

   Alternatively, click the  toolbar button and select *Back Annotate Dummy Instances*.

   a. Press `F3` to display the Back Annotate Dummies form.

      Default values for dummy instances are displayed in the form. Instances are cross selected based on the layout canvas

      If the pointer is moved in the schematic canvas, symbols for dummy instances appear based on the default values in this form.

   b. Modify the numbers of rows and columns and horizontal X and Y spacing if needed.

      The drag set in the schematic canvas is updated based on the changes you make.

   c. Click in the schematic canvas to drop the dummies symbol according to the drag set

   d. The XL status of the backannotated devices is shown as *OK* in the layout Navigator assistant.

© 2023 Cadence Design Systems, Inc.

Backannotating All Active and Inactive Dummy Instances

To backannotate all the active and inactive dummy instances:

➡ Choose *Connectivity – Back Annotate – All Dummy Instances*.

Alternatively, click the  toolbar button and select *Back Annotate All Dummy Instances*.

This calls the `lxHiBackAnnotateAllActiveDummies` SKILL function.

To backannotate selected active and inactive dummy instances:

➡ Choose *Back Annotate – Selected Dummy Instances*.

This calls the `lxHiBackAnnotateSelectedDummies` SKILL function.

To delete all schematic dummy instances:

➡ Choose *Back Annotate – Delete All Schematic Dummy Instances*.

This calls the `lxDeleteSchematicDummies` SKILL function.

Although it is not recommended to edit your layout after you have backannotated the dummies, if you do make edits, it is advisable to run *Check Against Source* to check if any instance mismatches exist. If any discrepancies are reported, you must make the appropriate updates manually.

Let us consider that a symbol that has been backannotated from the layout is represented as a three-terminal device. After backannotation, you add a fourth terminal to the layout representation of the device.

Now, if you were to run *Check Against Source* to check for any instance mismatches, you will observe that a missing instance is reported in the schematic. To fix this issue, in the schematic view, you need to manually add an `ignore` property to the symbol that ignores the fourth or the bulk terminal of the layout device. The same property must also be added to the `ignore` property list in the layout.

After the `ignore` property is set, running *Check Against Source* reports no mismatches.

Related Topics

Back Annotate Dummies Form

lxDummyBackAnnotateMFactor

backAnnotateInstances

© 2023 Cadence Design Systems, Inc.

ignore Property

lxHiBackAnnotateAllActiveDummies

lxHiBackAnnotateSelectedDummies

lxDeleteSchematicDummies

Net Voltages BackAnnotation

To keep the voltage values on nets across the layout and the schematic views synchronized, Layout XL enables backannotating the *Min Voltage* (MinV) and the *Max Voltage(MaxV)* value from the layout to the schematic view.

To backannotate the voltage values, choose *Connectivity – Back Annotate – Net MinV/MaxV.* This backannotates the *MinV* and *MaxV* values for *all* the nets from the layout to the schematic view.

For the case of an expanded bus, if the same *MinV and MaxV* value pair has been set on each bit of the layout bus, during backannotation, each bit of the expanded schematic bus gets updated with the voltage value pair as that set on the expanded layout bus bits.

However, if all the bits of the layout bus are not set to the same *MinV* and *MaxV* voltage values and the schematic bus is not expanded, backannotation works differently. In this case, the maximum voltage value and the minimum voltage value set on the layout bits is backannotated across all the bits of the unexpanded schematic bus.

Related Topics

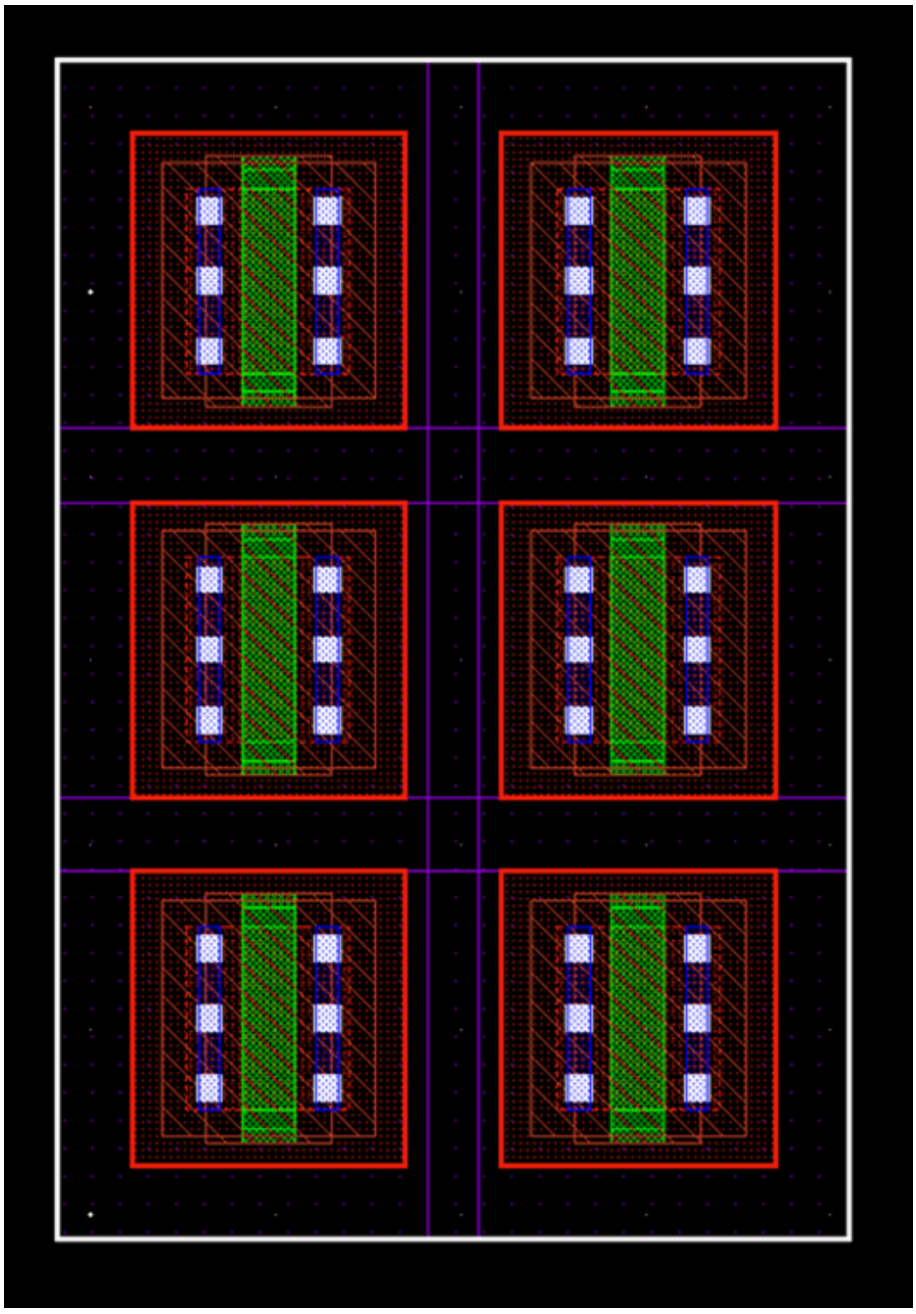
Backannotating Voltage Values from Layout to Schematic

© 2023 Cadence Design Systems, Inc.

Mosaic Instances Generation

A mosaic is an array of instances that belong to the same master. Mosaics are created to allow faster layout generation and editing.

A mosaic is represented in the database as an `oaArrayInst` object. Each mosaic comprises several evenly spaced identical tiles, as shown in the figure below:



© 2023 Cadence Design Systems, Inc.

If you create a 3x2 array of instances, as shown in the figure above, you have created a “mosaic instance” with 6 (3x2) tiles.

Virtuoso Layout Suite XL supports the following types of mosaics:

- **Physical-only mosaics**: These are “physical-only” mosaics that have a master, which has no terminal.
- **Same-connectivity mosaics**: These are mosaics that share the “same connectivity” across all their tiles because each terminal of the mosaic master is mapped to an edited cellview instance terminal.

The master of same-connectivity mosaics can also contain unassigned shapes that are treated as “unassigned shapes” in the master of scalar instances.

Layout XL enables you to create dummy mosaic instances that can be tied to an existing net on the mosaic. These dummy mosaics, if active—tied to a net other than the power or ground net—can be backannotated to the schematic as mfactored instances.

Dummy mosaic instances can be used as fillers in the layout to cover large design areas. When backannotated to the schematic, these instances can be represented as single, mfactored instances.

Dummy mosaic instances in the layout do not need the schematic counterpart to be always available, but backannotating these dummy mosaics to the schematic ensures that no instance mismatches are reported during a layout versus schematic run.

See Creating Dummy Instances and Dummy Instances Backannotation.

Related Topics

Creating a Mosaic Instance

Viewing a Mosaic Instance in the Navigator Assistant

Layout XL Commands Supported for Mosaic Instances

Layout XL Commands Currently Not Supported

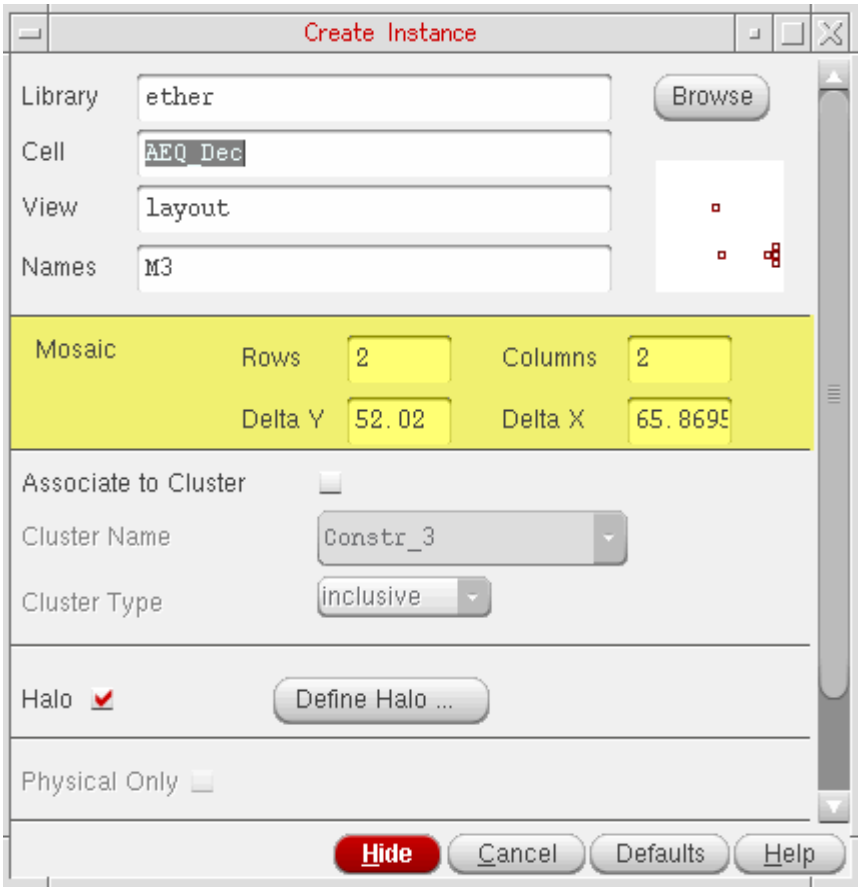
Mosaics do not support abutment at all—neither automatically nor interactively.

© 2023 Cadence Design Systems, Inc.

Creating a Mosaic Instance

You can create a mosaic in Layout XL using one of the following methods:

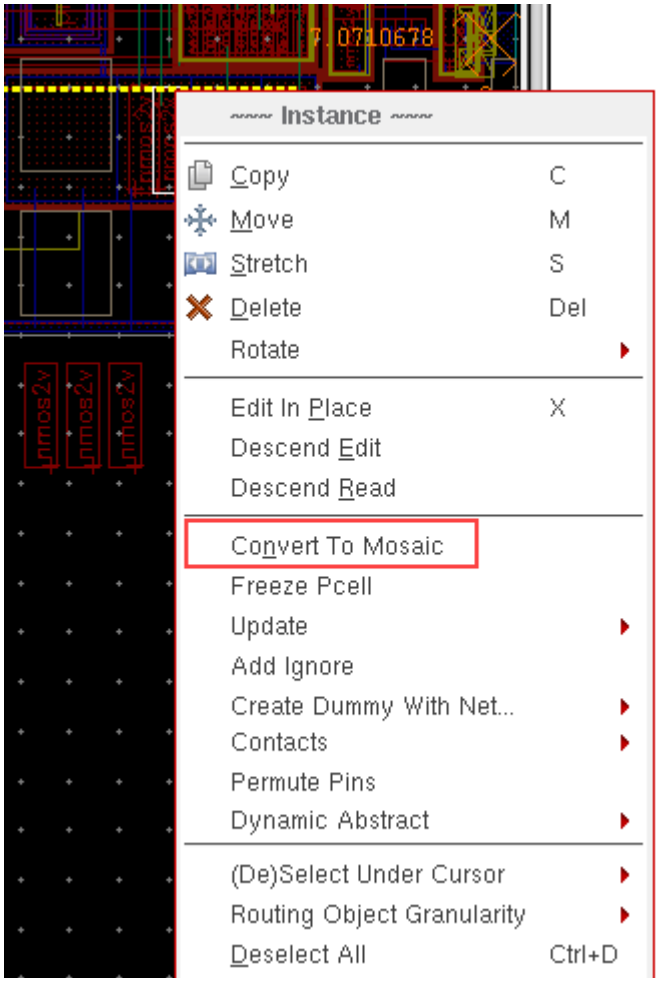
- Using the *Create Instance* form.



- Using the *Convert To Mosaic* command from the *Instance* shortcut menu.

© 2023 Cadence Design Systems, Inc.

To display the *Convert To Mosaic* command in the *Instance* shortcut menu, right-click the instance in the layout canvas.



- Using the *Edit – Convert – To Mosaic* command.

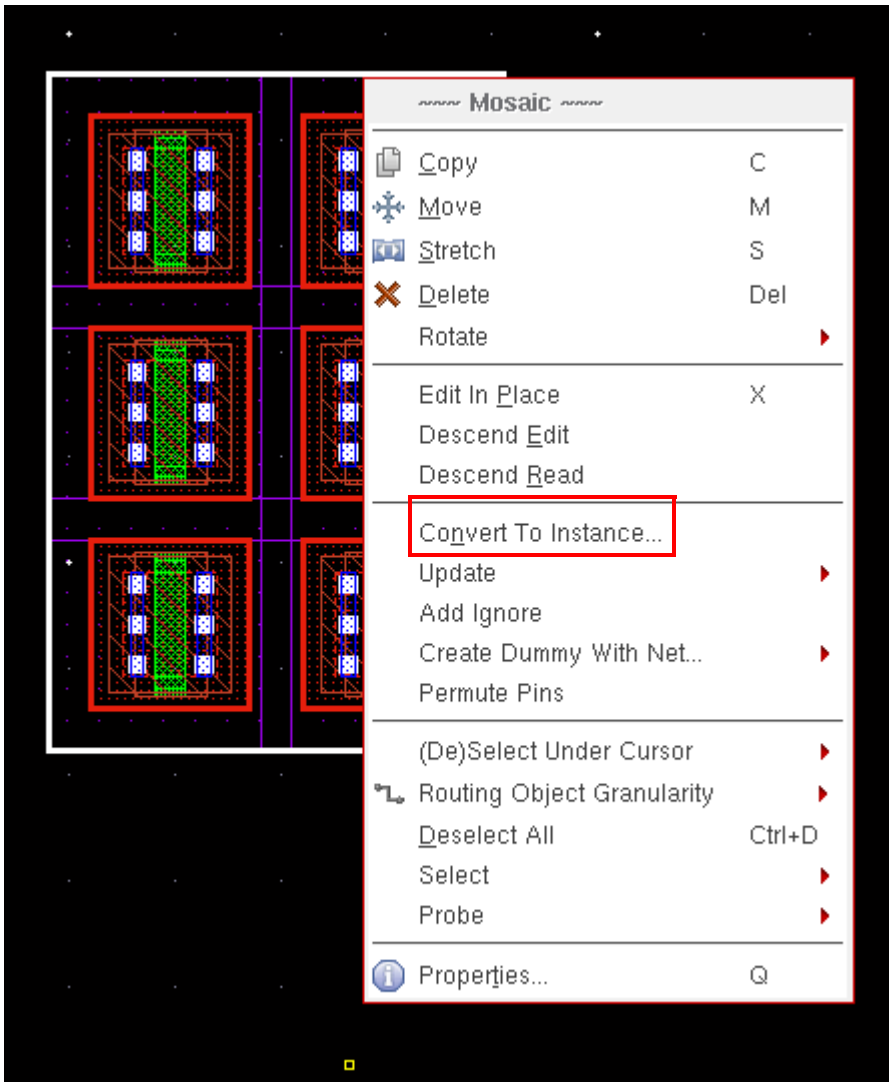
  Select the layout instance in the canvas or the Navigator assistant and choose the command.

- By generating a mosaic array in the layout view based on the instances selected in the schematic view.

Irrespective of the method that you use for creating a mosaic instance, you can convert the mosaic instance back into a regular layout instance by using the *Convert To Instance* command. Any existing bindings and connectivity on the mosaic are retained during the conversion.

© 2023 Cadence Design Systems, Inc.

The *Convert To Instance* command is available in the shortcut menu of a mosaic, as displayed in the figure below, when it is right-clicked in the layout canvas.



Related Topics

Create Instance Form

Instances to Mosaics Form

Converting Instances to Mosaics

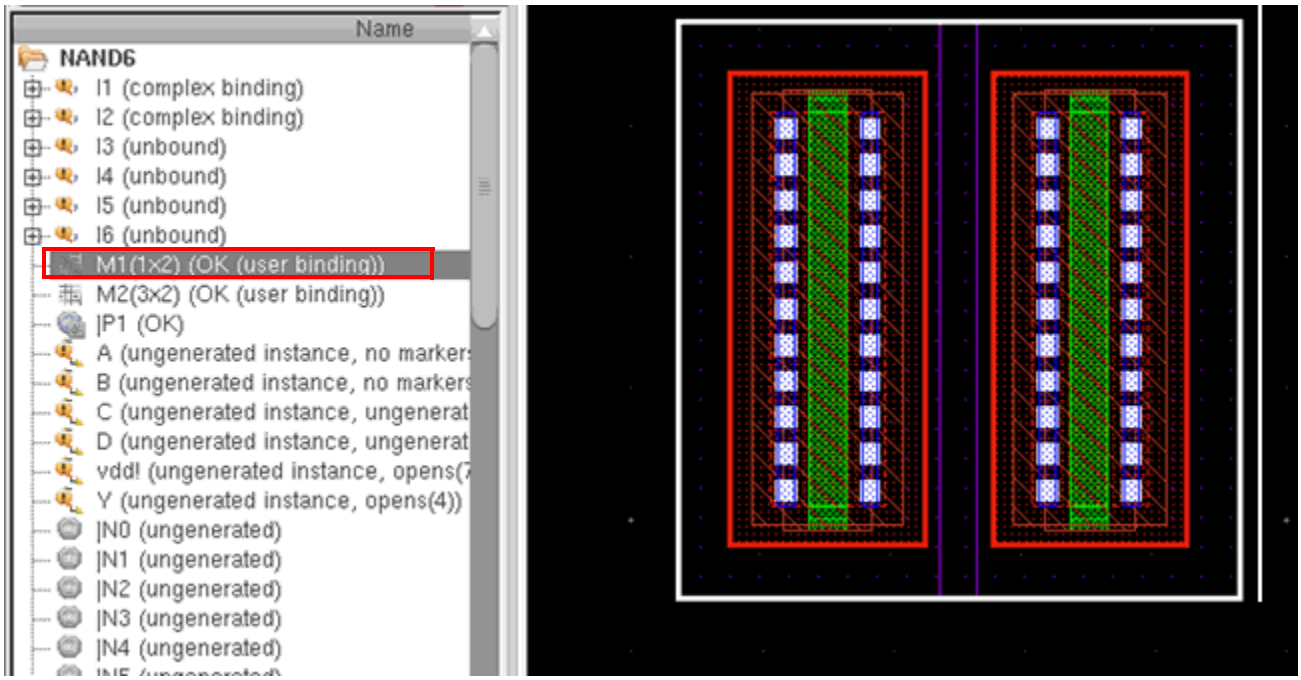
Array Generation - Mosaic

© 2023 Cadence Design Systems, Inc.

Viewing a Mosaic Instance in the Navigator Assistant

A mosaic instance appears in the Navigator assistant along with the other layout instances. You can use any of the instance filters to view the mosaic instances in the design. A mosaic instance can be differentiated from a regular layout instance by its name.

A mosaic instance name carries information about the number of its rows and columns in the (row x column) format. For example, the mosaic instance `M1` in the figure below indicates that it is a (1x2) mosaic with 1 row and 2 columns.



A mosaic instance can also be differentiated in the Navigator assistant by its unique icon, .

However, for the mosaic icon to appear visibly in the Navigator assistant, the mosaic instance must be bound *OK*, with no reported differences.

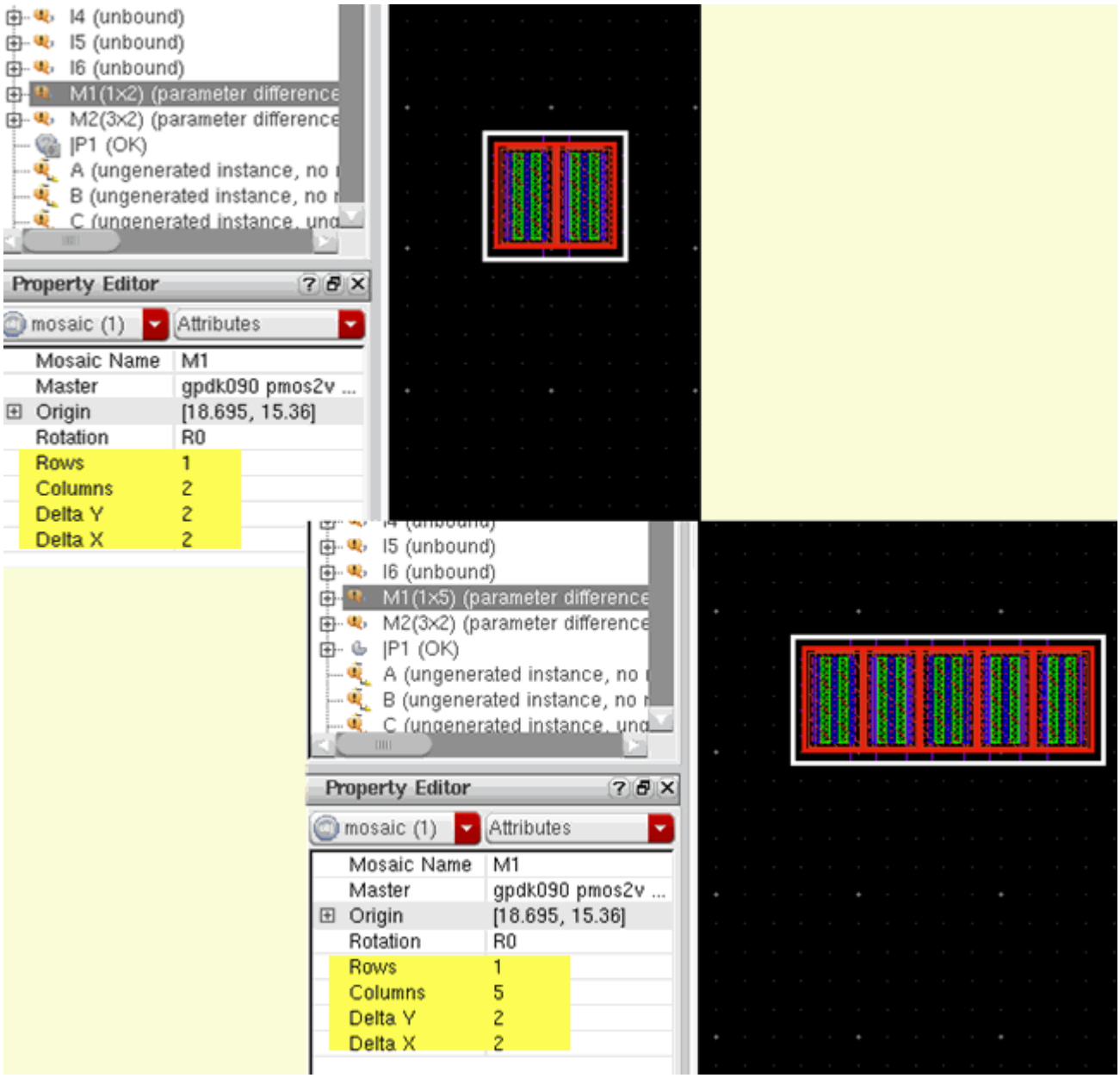
The binding of a mosaic to a schematic instance is reported as *OK* if the size of the mosaic is equal to the mfactor of the schematic instance. Else, the binding is reported as *complex binding.*

To help place and route mosaics in the layout view, Layout XL displays flightlines, if enabled, to the closest element of the mosaic.

As in the case of regular layout instances, you can use the Property Editor assistant to interactively update the properties of a mosaic instance, including the number of rows and

© 2023 Cadence Design Systems, Inc.

columns it supports. Any changes you make to the mosaic properties are instantly reflected in the layout canvas and the Navigator assistant, as displayed in the figure below.



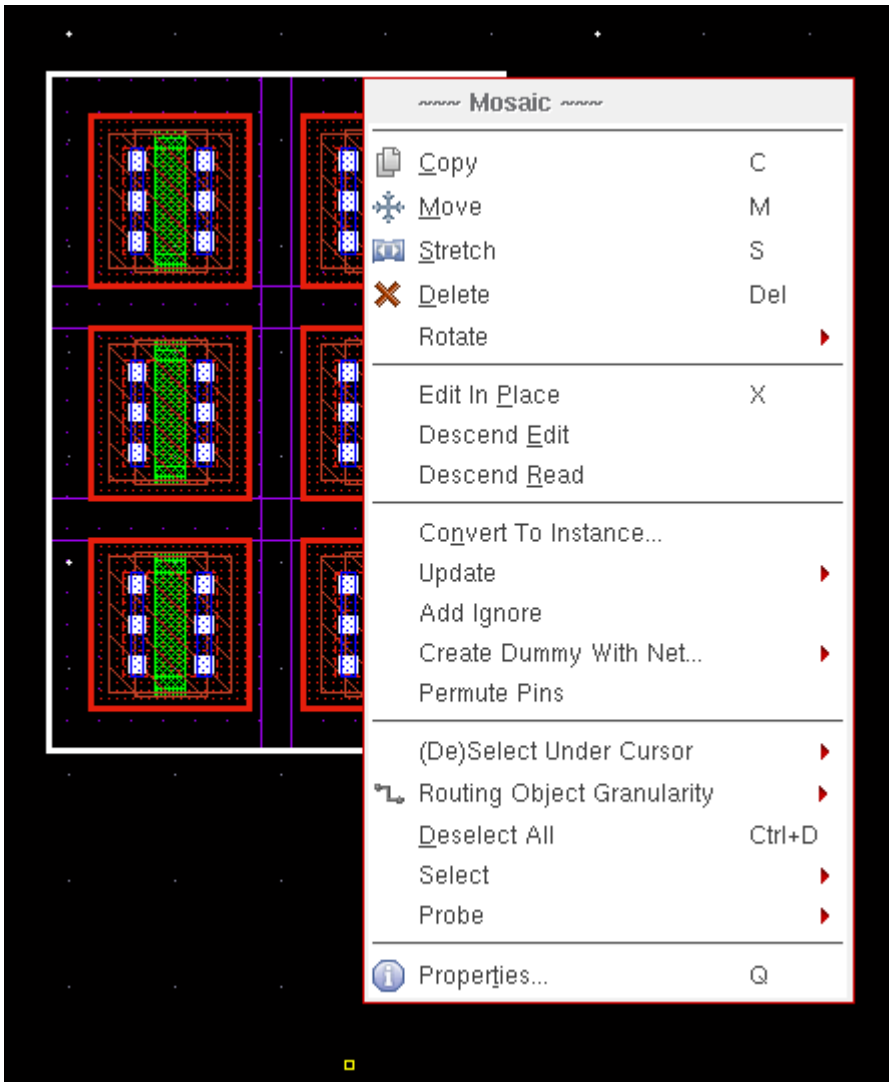
Related Topics

The Navigator Assistant

© 2023 Cadence Design Systems, Inc.

Layout XL Commands Supported for Mosaic Instances

Most Layout XL commands are applicable on mosaic instances and can be used to generate the same results with mosaics as with regular layout instances. The mosaic shortcut menu displayed in the figure below suggests some of the Layout XL operations that are supported by mosaics.

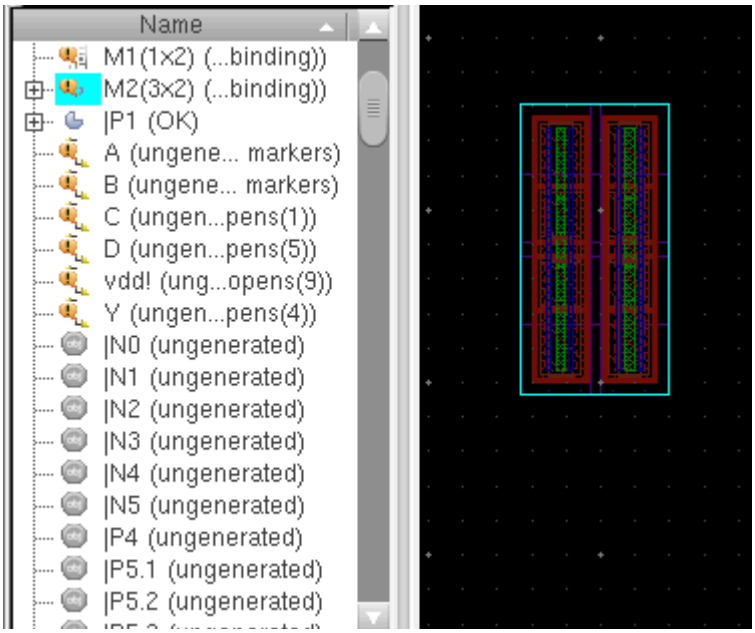
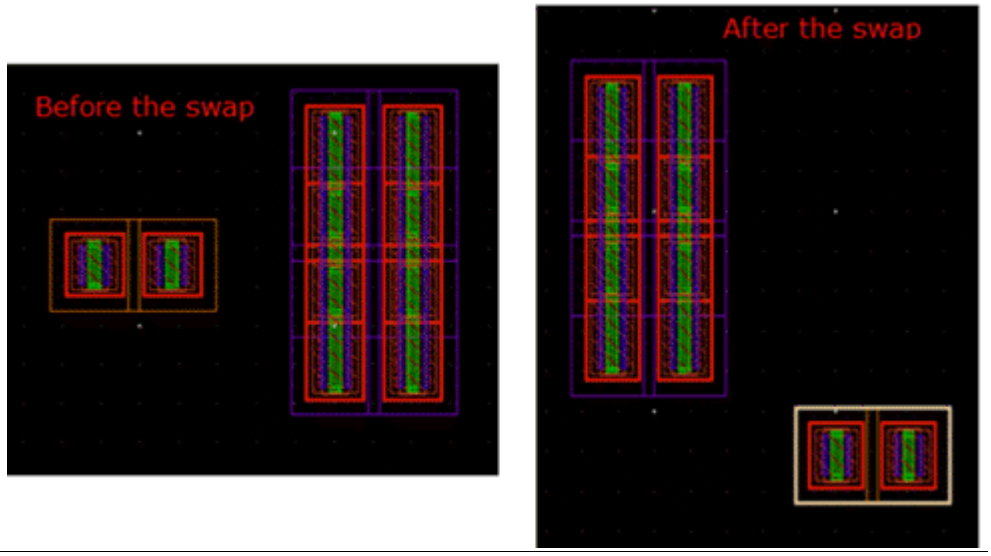


Command	Description
Mosaic – Smart Display	Layout XL supports displaying the abstracted view of mosaics using the Smart Display toolbar.

© 2023 Cadence Design Systems, Inc.

Command	Description
Mosaic – Update	Mosaics retain their connectivity, if any, during their conversion from or to a regular instance. Any updates that you make to the mosaic connectivity or parameters are reflected in the layout using the following Layout XL commands: ■ *Define Device Correspondence* ■ *Update Components And Nets* ■ *Update Layout Parameters* ■ *Update Schematic Parameters* The *Update* commands listed above, except *Define Device Correspondence*, are applicable to mosaics only when there is a one-to-one correspondence between the selected mosaic and the schematic instance bound to it. For the mosaic and the bound schematic instance to have a one-to-one correspondence, the size of the mosaic must be equal to the schematic mfactor value. For example, if the size of a mosaic instance is 6 (3x2), and the mfactor of the corresponding schematic instance is 6, a one-to-one correspondence exists between the layout and schematic instances. In this case, the XL Status reports the correspondence as *OK* and *Update* commands, such as *Update Layout Parameters* and *Update Schematic Parameters* work. But, if the mfactor of the schematic instance is 4 instead of 6, the *XL Status* reports the correspondence between the schematic and layout instance as *complex binding*. In this case, *Update* commands do not work.
Mosaic – Check	Layout XL enables you to check the mosaic instances for connectivity and parameter mismatches, if any, by running the *Connectivity – Check – Against Source* command. The *Check Against Source* command checks only those mosaic instances for which the *XL Status* reports an *OK*.

© 2023 Cadence Design Systems, Inc.

Command	Description
Mosaic – Probes	Layout XL supports probing of mosaics and the probe you apply displays in the Navigator assistant and in the layout canvas, as shown in the figure below. 
Mosaic - Swapping Components	If you have two mosaics in a design, you can use the *Edit – Advanced – Swap Components* command to swap their components, as displayed in the figure below. 

© 2023 Cadence Design Systems, Inc.

Related Topics

Display of Mosaics in Abstracted Mode

Design Check Against Source

Layout XL Design Data Updates.

Design Probing

Swapping Components

Layout XL Commands Currently Not Supported

Mosaics do not currently support the following Layout XL features:

- Auto Abut
- Auto Permute
- Auto Mirror/Rotate

Although mosaics do not automatically support the operations listed above, you can manually perform the following operations on mosaics:

- Interactive Device Permutation
- Interactive Pin Permutation
- Interactive Mirroring/Rotation

Mosaics do not support abutment at all—neither automatically nor interactively.

Generating Parameterized Layout Cells (VPLGen)

VPLGens are Cadence-supported hierarchical Pcells that require no SKILL coding and can be edited using all the standard Virtuoso tools, including Layout XL, floorplanner, automatic placers, routers, and so on.

You can use VPLGens when you need a hierarchical Pcell but there are no resources available to create the required SKILL code, or in any situation where you want a single schematic symbol to create a range of similar but slightly different standard layouts and you expect to re-use the same set of inherited parameters on each.

© 2023 Cadence Design Systems, Inc.

For example, you want to accelerate the process of creating a standard cell library of inverters, nand gates, and so on. For a single schematic pmos or nmos cell, you might want to specify different numbers of devices (mfactors) or different widths, lengths, and numbers of fingers.

You want to specify different layout variants for a single schematic cell, for example, with different routing to connect the devices slightly differently. You can add a (non-inherited) routing parameter to the schematic instance and use this to switch between the different variants of the instance in the layout.

In the absence of schematic instances, VPLGens can also be created for symbol views.

You specify that a schematic cell is to be implemented as a VPLGen in the Configure Physical Hierarchy window, and use Layout XL to generate an instance of that schematic cell in the layout and edit that instance to create a *core layout* that meets your specifications.

The core layout can then be saved either in your design library or a separate library and re-used later in other designs using the same process node.

You can optionally set the core layout to be the *default layout* for a particular schematic cell. Whenever you generate an instance for that cell, Layout XL copies the default layout into the design as a VPLGen Pcell instance, which you can use as a starting point for further editing. If the default layout does not meet your requirements, you can map the VPLGen Pcell instance to a different layout in your design library and use that instead.

If you have a Cadence Support portal account, you can download the Virtuoso Parameterized Layout Generators VPLGen rapid adoption kit that comprises a workshop database and a guided lab manual to help you experience the productivity gain from using VPLGens for creating reusable layouts in a test environment.

This section describes the Virtuoso parameterized layout generator (VPLGen) functionality, which lets you create and re-use predefined layouts for schematic instances that differ only in terms of their inherited parameter values.

VPLGen Environment Variables

The VPLGen functionality is supported by the following environment variables:

- vplGenCreateCells
- vplGenLibs
- vplGenParams
- vplGenSetDefaultLayout

© 2023 Cadence Design Systems, Inc.

VPLGen SKILL Functions

The VPLGen functionality is supported by the following SKILL functions:

- cphDeleteCellVPLGen
- cphGetCellVPLGen
- cphGetCellVPLGenParams
- cphSetCellVPLGen
- cphSetCellVPLGenParams
- dbIsVPLGen
- dbRegVPLGenCreateCellName
- dbUnregVPLGenCreateCellName

This section covers the following topics:

- Basic VPLGen Flow
- Setting Cells as VPLGens
- Creating a VPLGen Core Layout
- Setting the Default Layout
- Generating VPLGen Pcell Instances
- Updating VPLGen Pcell Instances
- VPLGen Environment Variables
- VPLGen SKILL Functions

Related Topics

LAM File Logical Section

Defining Parameters,

© 2023 Cadence Design Systems, Inc.

Basic VPLGen Flow

The section outlines the basic VPLGen flow and provides links to more information.

1. Decide which schematic cells are to be implemented as VPLGens.
2. Use *Configure Physical Hierarchy* to specify that the schematic cells are to be implemented as VPLGens.

   You can also prevent specific instances from being generated as a VPLGen, or remove the VPLGen designation from the schematic cell altogether. See Overriding the Physical Binding for a Specific Instance and Removing the VPLGen Designation from a Schematic Cell.
3. Use *Generate Selected From Source* to generate an instance of the schematic cell as a VPLGen Pcell instance in the layout view, edit the VPLGen in Layout XL to design the layout you require, and save it to create a VPLGen core layout.
4. (Optional) Set the VPLGen core layout as your preferred default layout to be re-used each time you generate a new VPLGen Pcell instance for that schematic cell.
5. Although optional, this step is recommended, so that you re-use as much as possible of the core layout (placement, routing, guard-rings, and so on) the next time you generate that cell in the layout.
6. Use the standard Layout XL commands to generate and edit VPLGen Pcell instances in your layout.
7. Use the standard Layout XL commands to update existing VPLGen Pcell instances in your layout.

Related Topics

Setting Cells as VPLGens

Creating a VPLGen Core Layout

Setting the Default Layout

Generating VPLGen Pcell Instances

Updating VPLGen Pcell Instances

© 2023 Cadence Design Systems, Inc.

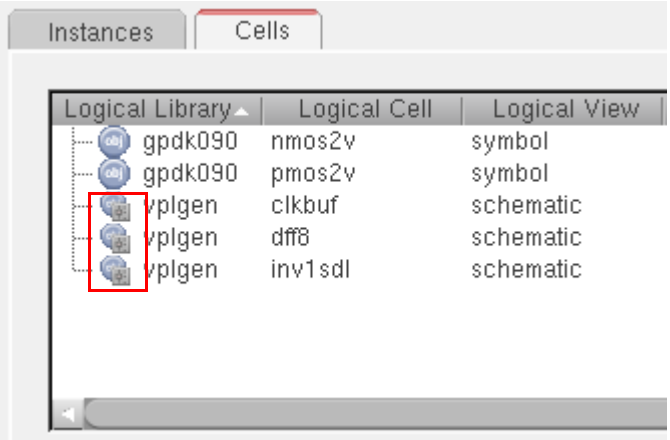
Setting Cells as VPLGens

You use the Configure Physical Hierarchy window to mark a schematic cell as a VPLGen. This specifies that predefined layouts will be generated automatically for variants of this schematic cell. See

To set a cell as a VPLGen,

1. From the layout window, click *Launch – Configure Physical Hierarch*y and ensure you are in the Hierarchy Configuration mode.
2. Click the *Cells* tab and select the cells that you want to implement as VPLGens.
3. Click the right mouse button and select *Create/remove VPLGen*.

   The selected cells are marked as VPLGens and the icon updated to denote this.

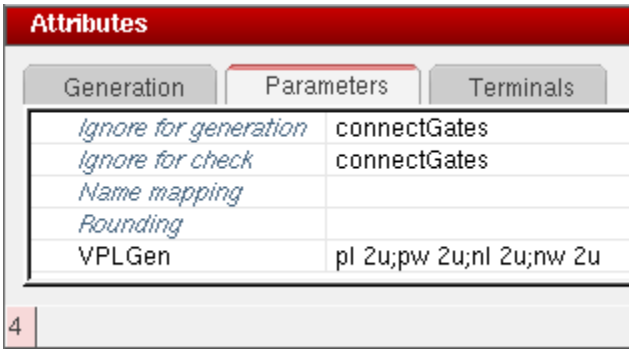


For performance reasons, CPH does not search on disk for VPLGen variants of the same cell that might exist in other designs. If this cell has a VPLGen definition in a different library, you will still be able to create it again here.

You can use the LAM file to designate the schematic cell as VPLGen and map it to the VPLGen variant you require.

© 2023 Cadence Design Systems, Inc.

1. Click the *Parameters* tab in the *Attributes* pane and type in the list of parameters in the *VPLGen* field.



   List the parameter names and default values, each separated by a semicolon; for example pl 2u;pw 2u;nl 2u;nw 2u.

   Although shown here, you do not need to specify default values for CDF parameters; the default value set for each CDF parameter in the schematic is used automatically. You can also populate this field automatically using the vplGenParams environment variable. If you do not specify any parameters or set this environment variable, no VPLGens will be created.

2. Click *File – Save* in the Configure Physical Hierarchy window.

   The specified schematic cells will be generated as VPLGen Pcell instances in the layout view.

Overriding the Physical Binding for a Specific Instance

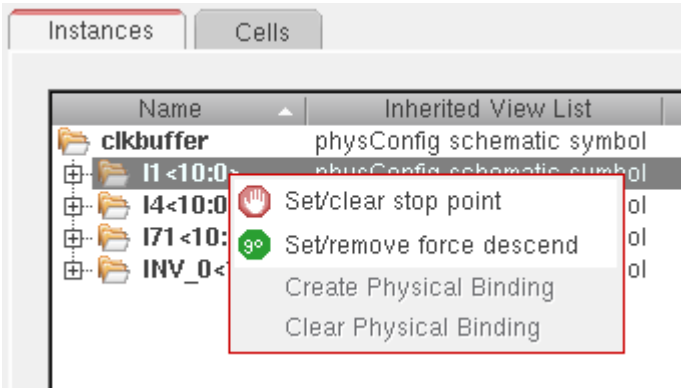
If you have a specific schematic instance that you do not want to be instantiated as a VPLGen, you can override the physical binding for that instance in the Instances tab.

To do this:

1. Click the *Instances* tab in the Configure Physical Hierarchy window.

© 2023 Cadence Design Systems, Inc.

2. Right-click the instance you want to override and choose *Set/remove force descend*.



The selected instance will be generated as normal in the layout, even though it is still set as a VPLGen at the cell level.

Removing the VPLGen Designation from a Schematic Cell

You can also remove the VPLGen designation completely for a schematic cell.

Caution

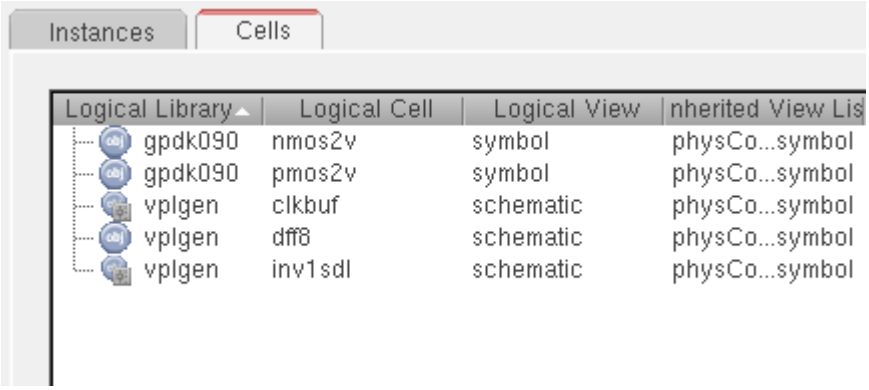
Cadence recommends that you do not use this function unless you are sure that you no longer want to use the VPLGen in question. Using this function to delete a VPLGen means that any designs that use that VPLGen will have missing layout instances. (Any core layouts associated with the VPLGen are retained in case they are being used as regular layouts in other designs.)

To do this:

1. In the *Cells* tab, select the VPLGen you want to remove.
2. Click the right mouse button and select *Create/remove VPLGen*.

© 2023 Cadence Design Systems, Inc.

The selected VPLGen is removed and the icon updated to reflect its status.



Instances of that cell will be generated as normal in the layout view.

Related Topics

Hierarchy Configuration Mode in CPH

vplGenParams

Creating a VPLGen Core Layout

After you have set up the parameter and property values you require in the schematic cell and have marked the cell as a VPLGen in the Configure Physical Hierarchy window, you can create a VPLGen core layout for that cell.

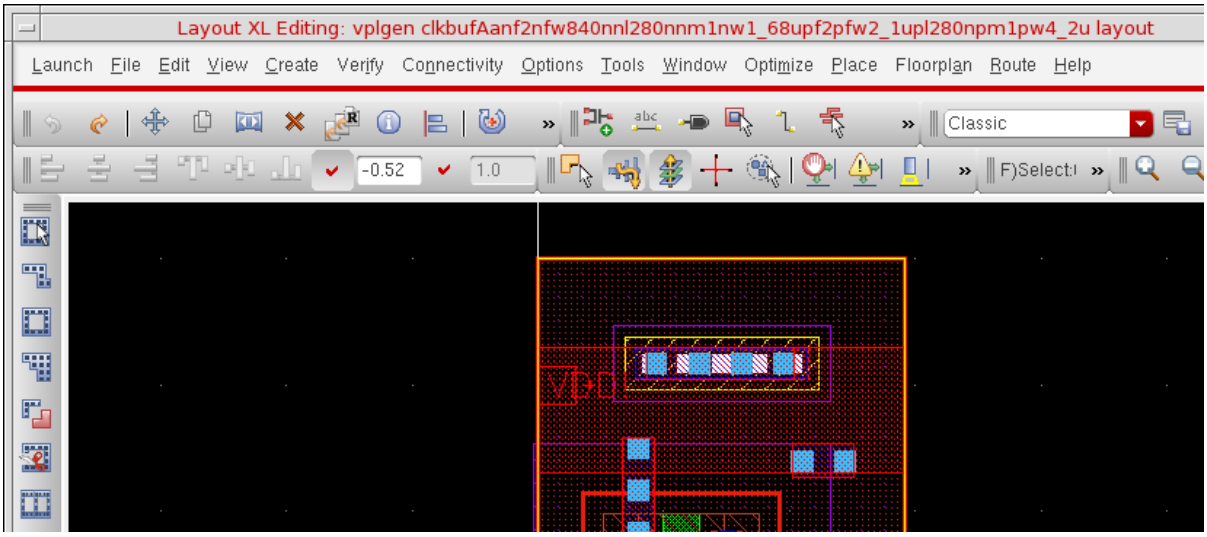
To do this:

1. Use *Connectivity – Generate – Selected From Source* to generate an instance of the VPLGen Pcell in your layout view.

   Layout XL generates a device with the default parameters defined for the VPLGen in the schematic view. If the parameters specified in the Attributes pane are different from the default parameters in the schematic, Layout XL creates two variants the first time the cell is generated in the layout, one with the default parameters and a second one with the specified set of parameters.

© 2023 Cadence Design Systems, Inc.

2. Use *Edit – Hierarchy – Descend Edit* or *Edit – Hierarchy – Edit In Place* to descend into the VPLGen Pcell instance.



The cell name for the VPLGen is derived automatically by concatenating the name of the cell and the names and values of the specified parameters.

You can use the `dbRegVPLGenCreateCellName` SKILL function to register your own SKILL function to create unique cell names.

3. Use Layout XL to lay out the cell to meet your requirements.

   You can move instances, pins, and vias and add layers, wells, high and low voltage layers, substrate ties, interactive routing, and any guard rings required. You can also add pins and terminals but only if they match the schematic.

   You can also use the Property Editor assistant to change any non-VPLGen inherited parameters as required. Take care not to change any of the inherited parameters, otherwise you will cause a mismatch between the layout instance and its schematic counterpart. If any of the inherited parameters need to be changed, change the parameter on the schematic instance and run the *Update Layout Parameters* command to generate a new core layout for the VPLGen.

4. When you have finished editing the cell, choose *File – Save* and then *Edit – Hierarchy – Return* to return to the top level.

   By default, the VPLGen core layout is saved to your design library. If required, you can create a dedicated library to store all your VPLGen variants separately from your design cells. You do this in the *Cells* tab in the Configure Physical Hierarchy window by editing the physical library field to point to the library you want.

© 2023 Cadence Design Systems, Inc.

Related Topics

dbRegVPLGenCreateCellName

Layout Parameters Updates

Specifying Logical to Physical Mappings for the Components in CPH

Setting the Default Layout

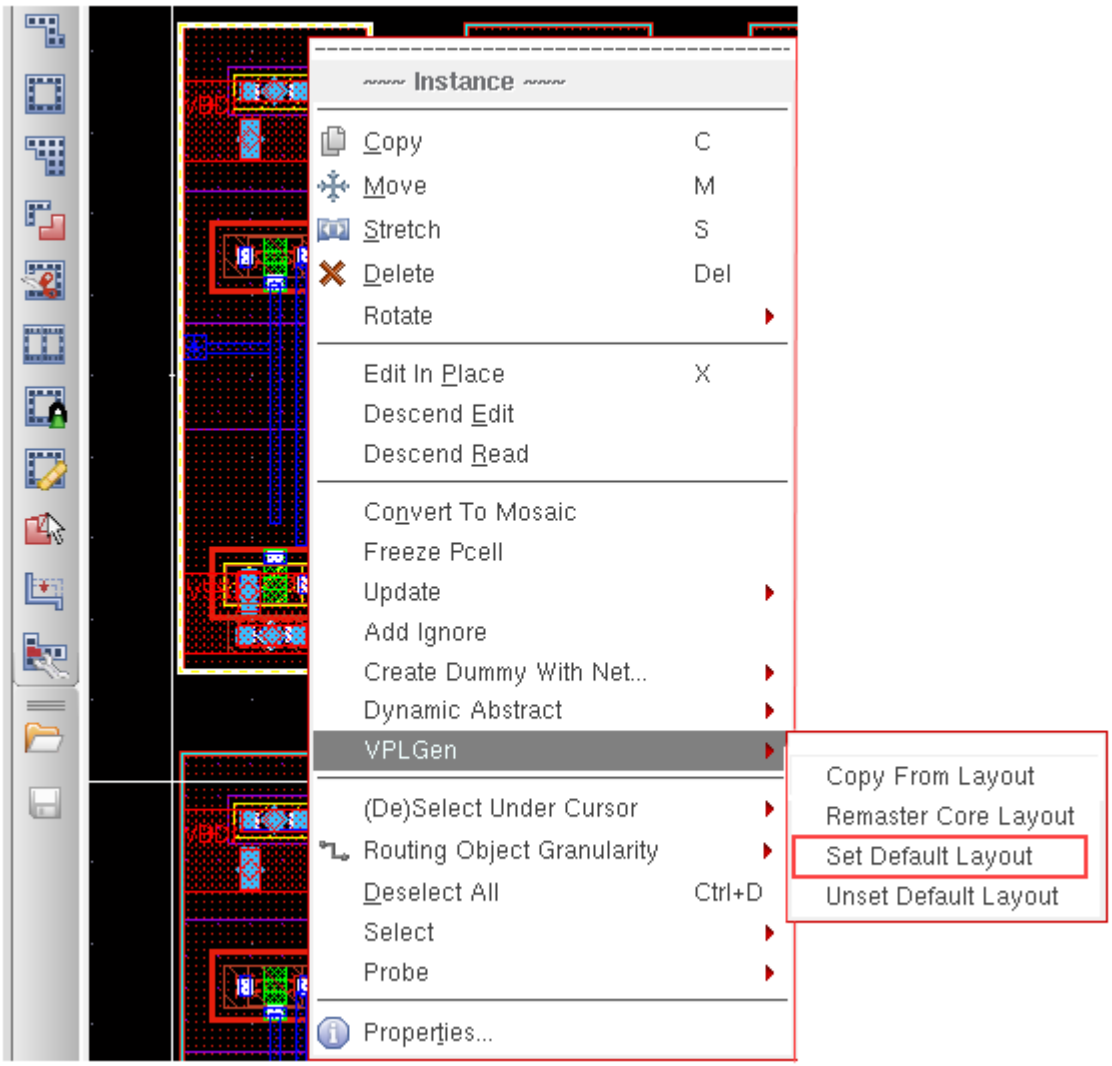
When you have finished editing the VPLGen core layout, you can set it to be the *default layout* for that schematic cell. Layout XL uses the default layout as a template the next time the cell is generated in the design, copies it into the layout, and updates the parameters values as required. This lets you re-use as much as possible from the core layout (placement, routing, guard-rings, and so on) the next time you generate that cell in the layout, making it easier to complete minor edits on new core layouts that differ only in their inherited parameter values.

To set the default layout:

1. Select the VPLGen Pcell instance in the layout window.

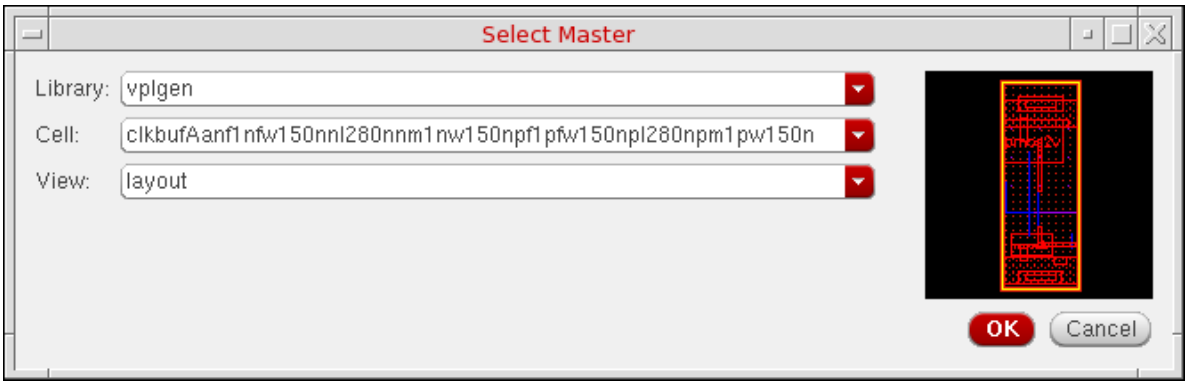
© 2023 Cadence Design Systems, Inc.

2. Click the right mouse button and select *VPLGen – Set Default Layout*.



© 2023 Cadence Design Systems, Inc.

The Select Master dialog is displayed.



If there is a default layout already set, the form is populated with the name and thumbnail of that cellview. If there is no default layout set, the form contains the name and thumbnail of the currently selected layout view to make it easy to set this one as the default.

3. Confirm the library, cell, and view names are as required and click *OK*.

   The next time you generate this VPLGen, Layout XL copies the selected master into the design and automatically updates the parameter and property values to reflect those in the schematic. You can use this as a starting point from which to edit the layout to meet your requirements.

To set a different core layout as the default layout,

➡ Follow the steps outlined above and set the cell you require in the **Select Master** dialog.

Unsetting the Default Layout

If you do not want to use the default layout for a particular VPLGen Pcell instance, you can unset it. To do this:

1. Select the VPLGen in the Navigator assistant or layout window.
2. Click the right mouse button and select *VPLGen – Unset Default Layout*.

   Subsequent VPLGen Pcell instances are generated from scratch by Layout XL.

Related Topics

The Navigator Assistant

© 2023 Cadence Design Systems, Inc.

Generating VPLGen Pcell Instances

This section describes how to use the VPLGen core layouts you have created to quickly generate and manipulate layout components in your design. You can do this in either of the two ways below:

- After you have generated the VPLGen instances in your layout, you can update the core layout by copying an existing VPLGen core layout and then editing it as required without those edits affecting other users. Because the layout is copied, the design remains self-contained and therefore more portable. However, any subsequent changes to the VPLGen core layout are not automatically reflected in the copied VPLGen.
- After you have generated the VPLGen instances in your layout, you can update the core layout by mapping (remastering) to any layout defined in one of the available libraries.

  This lets you re-use any pre-existing standard cell library you might have created for the VPLGen without having to update every single instance with a parameter different in Configure Physical Hierarchy.

  - The layout you map to need not be a VPLGen core layout. You can map to any layout in your library.
  - This is not a copy of the layout. Any edits to the new core layout will also change any other instances that use this layout as their master.

Related Topics

Remastering a VPLGen to a Different Core Layout

Copying a VPLGen from Layout

Copying a VPLGen from Layout

Use *Copy From Layout* to copy an existing core layout into the current design. You can then edit the resultant local copy to meet your specifications.

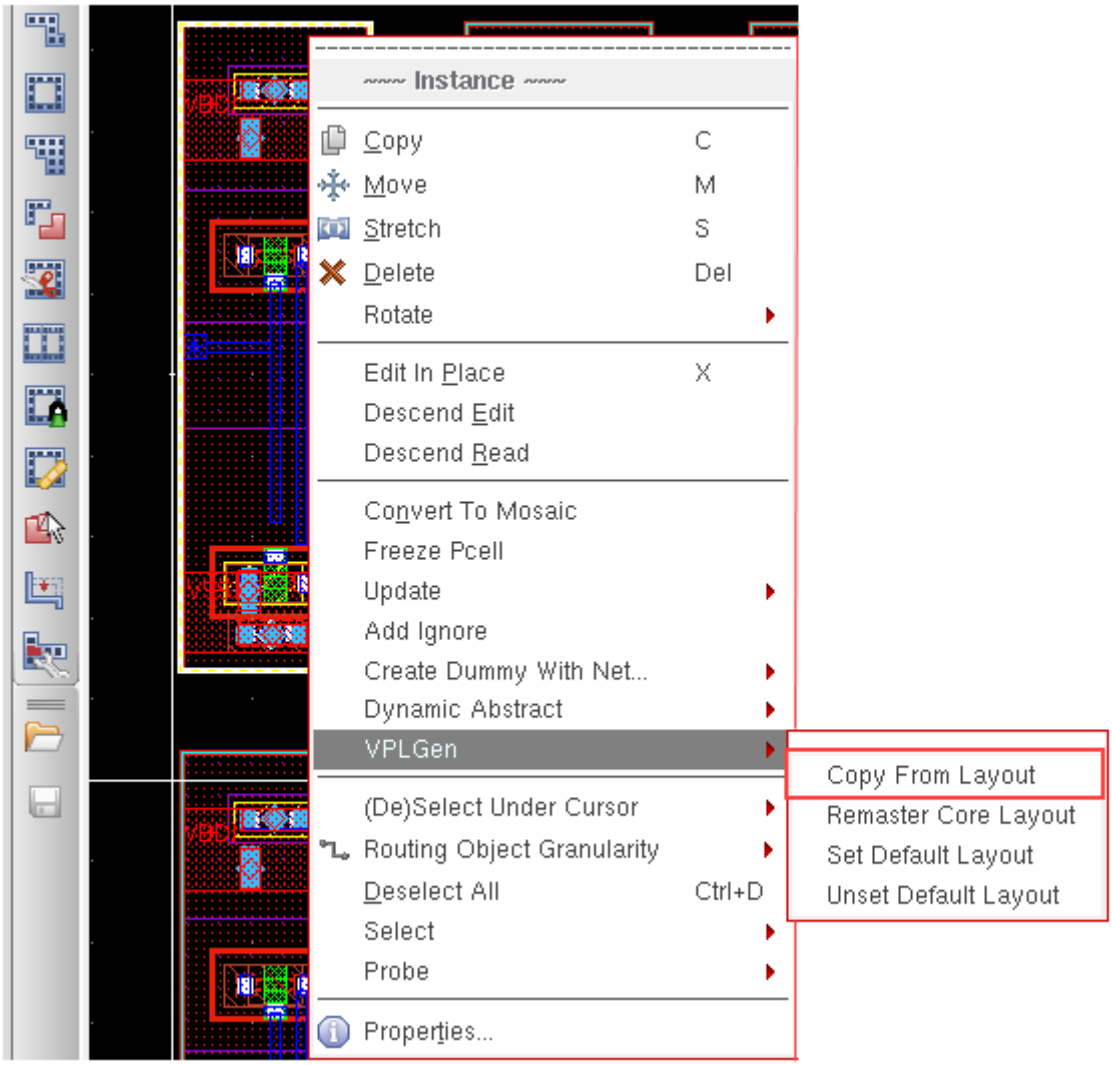
Copy From Layout requires write permission for the library in order to create the new VPLGen variant in the layout view.

To copy a VPLGen from an existing layout:

1. Use *Generate Selected From Source* to generate an instance of the VPLGen Pcell in your layout.

© 2023 Cadence Design Systems, Inc.

2. Right click the VPLGen Pcell instance and select *VPLGen – Copy From Layout*.



The Select Master dialog is displayed.

3. Select the library, cell, and view name of the VPLGen variant you want to copy into the design and click *OK*.

Layout XL copies the selected variant into the design and automatically runs *Update Components And Nets* with *Update Layout Parameters*.

You can use *Copy From Layout* for any Layout XL compliant layout in your Virtuoso session; it need not be a VPLGen core layout.

© 2023 Cadence Design Systems, Inc.

4. Use *Edit – Hierarchy – Descend Edit* or *Edit – Hierarchy – Edit In Place* to descend into the resultant instance and make any edits you require.

   When a VPLGen instance is edited, the corresponding schematic instance is automatically descended into.

5. Choose *File – Save* and *Edit – Hierarchy – Return* to save your edited layout as a new layout variant for the VPLGen and return to the top level of your design.

   If the variant exists already it is not overwritten. Once created, a variant is never deleted. to remove it, you must delete it manually from the library.

Related Topics

Generate Selected Components Form

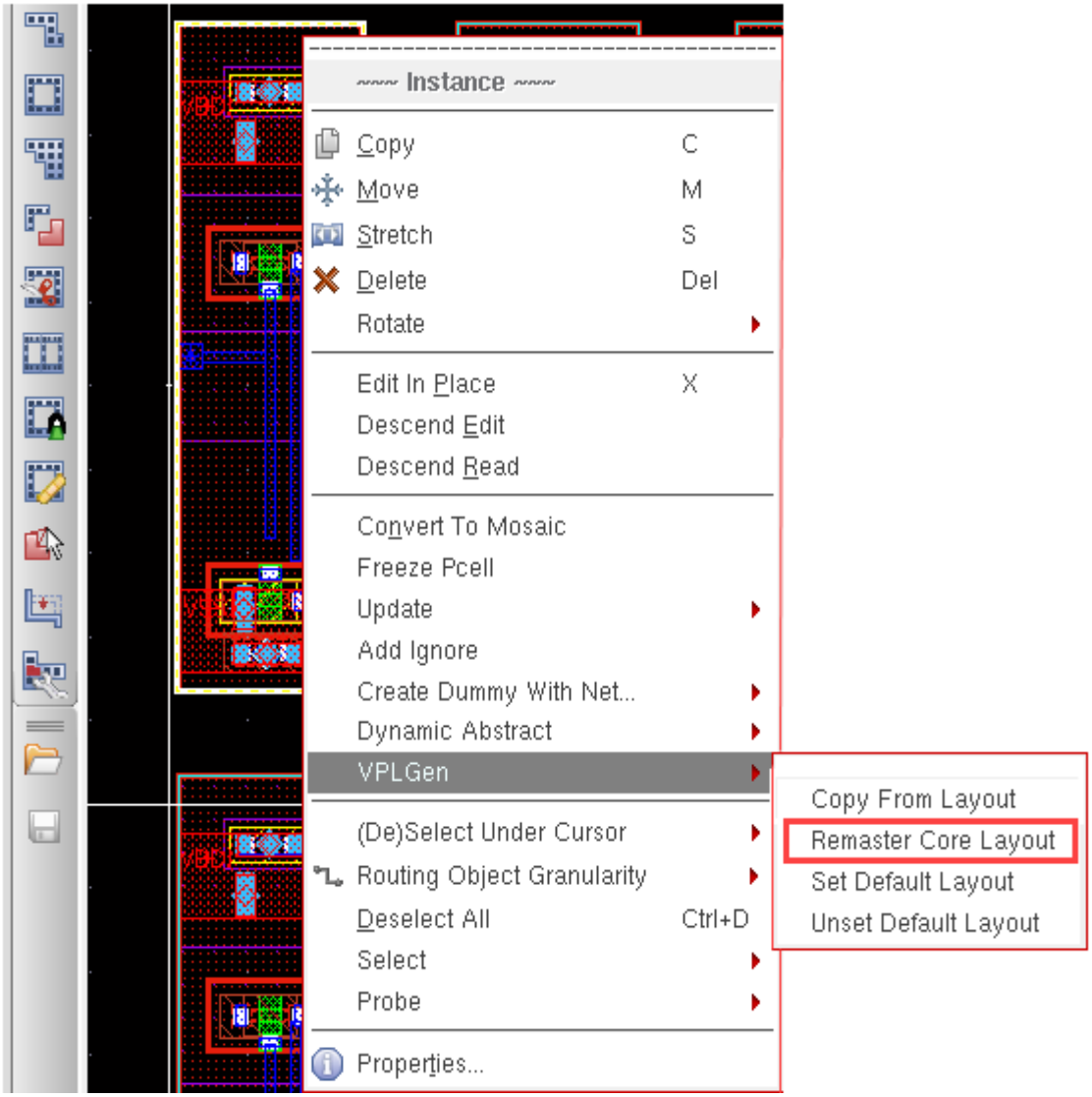
Remastering a VPLGen to a Different Core Layout

Use *Remaster Core Layout* to map a VPLGen Pcell instance to a different core layout already defined in the library. This feature lets CAD teams control what variants are provided by defining and maintaining them centrally and making them available to designers in standard, non-writable libraries. Updates to these centrally-controlled core layouts are automatically reflected in the layout designs that use them.

To remaster a VPLGen Pcell instance to a different core layout,

© 2023 Cadence Design Systems, Inc.

1. Use *Generate Selected From Source* to generate a VPLGen Pcell instance.
2. Right-click the VPLGen Pcell instance and select *VPLGen – Remaster Core Layout*.



The Select Master dialog is displayed.

3. Select the library, cell, and view name of the core layout you want to map to and click *OK*.

   The VPLGen Pcell instance is updated to reflect the parameter values set in the selected core layout. Any time that variant changes, the VPLGen Pcell instances that are mapped to it are changed as well.

© 2023 Cadence Design Systems, Inc.

Related Topics

Generate Selected Components Form

Updating VPLGen Pcell Instances

VPLGens support all the standard Layout XL update commands, meaning that changes in the schematic view can be propagated to the layout view.

For...	Use...
Changes to a parameter or property value in a schematic instance. For example, after the layout is generated, if you want to change any original parameter values, such as change the length parameter from *l = 1u* to *l = 2u*, the parameter updates must be made in the schematic view and an *Update Layout Parameters* command run to reflect the change in the layout view.	*Update Layout Parameters* to propagate the changes to the VPLGen Pcell instances in the layout view. There is no need to use *Update Components And Nets* because only parameter or property values have been updated.
New or copied instances added to the schematic view.	*Generate Selected From Source* to add the new VPLGen Pcell instance to your layout, while re-using any VPLGen default layout already defined.

Changes you make in the layout view cannot be propagated back to the schematic view.

Related Topics

Layout Parameters Updates

Generate Selected Components Form

(Non)parameterized Layout Cells Generation (VLGen)

For layout generation that does not need the schematic cells to have parameters or properties defined, use the Virtuoso Layout Generation (VLGen) functionality. The VLGen functionality lets you automatically generate the layout for schematic symbols that do not

© 2023 Cadence Design Systems, Inc.

have an existing layout available to reuse and have no CDF parameters defined. After the layout of a schematic symbol is generated, it can be used for generating further instances of the same symbol.

The VLGen functionality is recommended when:

- Additional levels of hierarchy need to be created to match the schematic.
- Write access to the schematic is not available and a single layout needs to be used to create multiple instances of the schematic symbol.
- Limited CAD support is desired and editing of schematic symbols needs to be avoided.

You can set a schematic cell for VLGen or VPLGen, not both.

The VLGen functionality supports the following modes:

- ***Manual*** lets you specify in the LAM file or in the CPH window the schematic symbols that need to be automatically generated using the VLGen functionality.

  It is recommended to define the VLGen in a LAM file and a SKILL utility is available to help set the LAM file. But, if you prefer setting the VLGen in CPH, do this:

➥ In the *Cells* table, right-click the cell that you want to set for VLGen and choose *Create/ remove VLGen*.

- ***Automatic*** lets you set for VLGen all schematic symbols that do not have their CDF parameters defined and have no existing layouts. To enable the automatic VLGen mode, set the vlGenAutoDetectCells and vlGenCreateCells environment variable to `t` before launching VLS XL.

Related Topics

Generating Parameterized Layout Cells (VPLGen)

Generating Master Variants

In the Layout XL connectivity-driven flow, by default, there is only one layout view supported for a symbol or schematic view. But, the same schematic logic topology may need to be generated using a different layout topology, requiring the creation of another layout variant. You can do this by generating a copy of the instance master and manually editing the specific occurrence of the instance to create a new variant.

If the layout variant is needed for Pcells, you can use the *Freeze Pcell* command.

© 2023 Cadence Design Systems, Inc.

If the layout variant is needed for nonparameterized cells, you can use the *Master Variant* command.

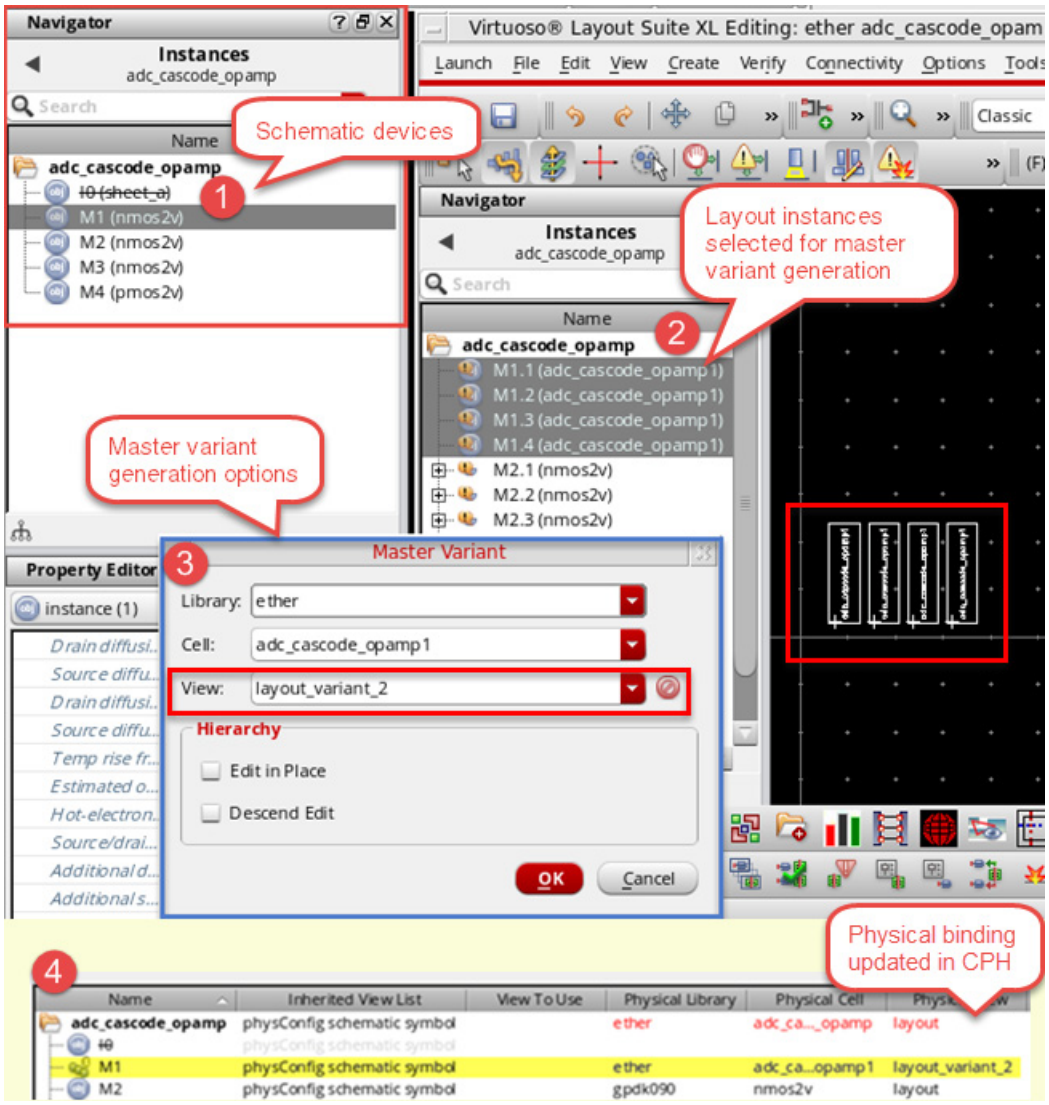
The *Master Variant* command uses the existing logic topology to create a variation or a *variant* of the existing layout, which has the schematic correspondence already in place, as displayed in the figure below. This makes master variant generation a quicker option compared to manually copying the cell, remastering it, and ensuring that the schematic and the layout instances bind, if the selected instance is not already bound. If you manually bind the remastered instance such that the schematic and layout instance masters are different, you may need to update the physical binding of the remastered instances.

When creating a master variant for a transparent instance, the physical binding is automatically updated in CPH although the instance remains unbound.

To update the physical binding on selected instances:

© 2023 Cadence Design Systems, Inc.

➡ Right-click the remastered instances and choose *Update – Physical Bindings* from the shortcut menu.



If a layout variant already exists for an instance, you can use that as the source for the master variant generation. In this case, the cellview that is selected to form the variant view becomes the new master.

For an existing view to be used as the new master, the master variant must have all the same instance terminals as the source, although the master variant can have additional instance terminals as well.

If the variant view does not already exist in a design library, a new master variant cellview is created on disk, which is a copy of the original master. By default, the first variant view that

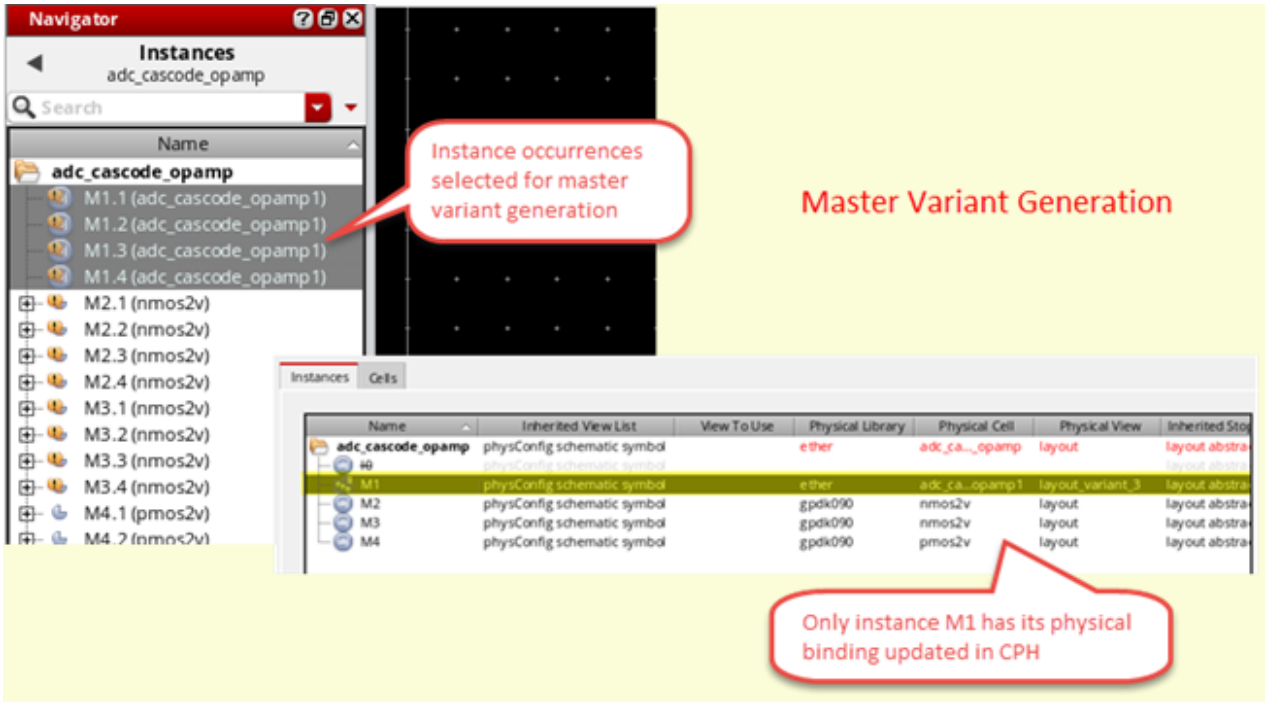
© 2023 Cadence Design Systems, Inc.

gets created for an instance is named as `layout_variant_1`. Any subsequent new variants added get incremental names `layout_variant_2`, `layout_variant_3`, and so on.

You can give the layout variant a name of your choice. You can also change the default library and cell to be used for the master variant generation to a library and cell of your choice.

After you have generated the variant, say instance `M1`, in the figure below, you can individually update that instance without impacting the other instances, `M2`, `M3`, and so on.

In addition, any updates that you make to the master variant are automatically saved in the physConfig view, and can be verified using the Configure Physical Hierarchy Window, as displayed in the figure below.

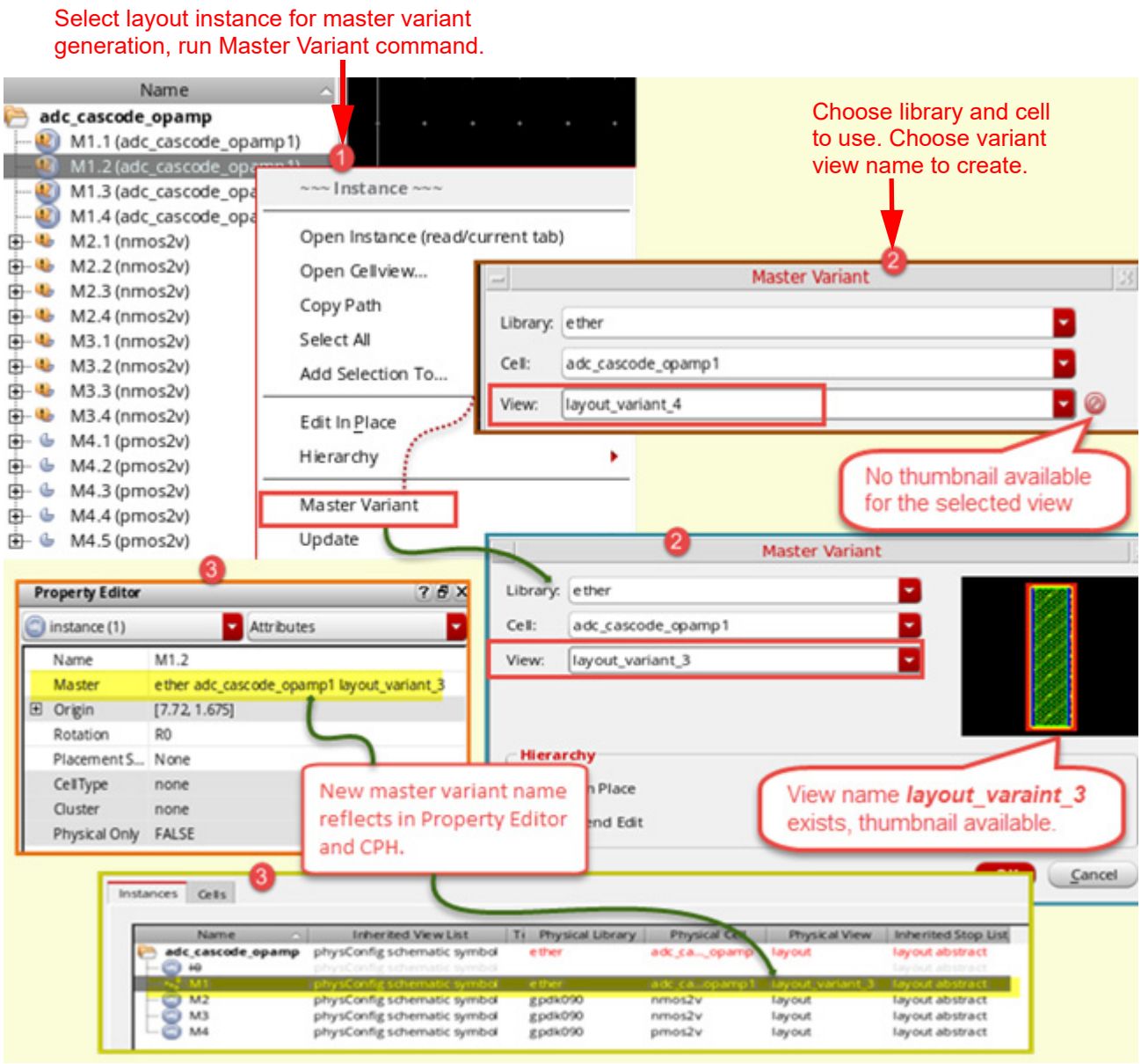


By default, any updates made to the phsyConfig view by Layout XL commands are automatically saved in CPH. But, you can choose to set the `autoMirrorFilter` to `nil` to prevent such automatic updates from being saved in CPH.

Any manual edits made to the CPH instance master are not automatically saved even when `autoMirrorFilter` is set to `t`. These updates must be manually saved in the CPH window.

© 2023 Cadence Design Systems, Inc.

The figure below illustrates the basic master variant generation flow, assuming the master view already exists.



Related Topics

Freezing and Unfreezing Pcell Instances

autoMirrorFilter

© 2023 Cadence Design Systems, Inc.

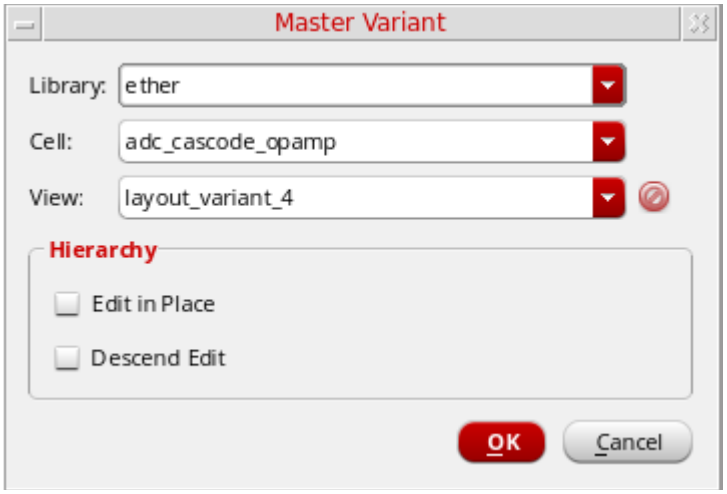
Generating Master Variant

To generate a master variant:

1. Right-click the layout instance in the Navigator assistant or canvas and choose the *Master Variant* command.

   You can select multiple instances simultaneously for variant view generation. But, all the selected instances must belong to the same master. Else, the master variant generation fails.

   The Master Variant form opens with the library, cell, and view names same as that of the source layout view.



2. (*Optional*) Choose a library in which the master variant of the selected layout instance needs to be created. Leave the default selection, if you want the variant to be created in the current selected instance master library.

   If you do not have permission to edit the selected library, a message is issued indicating that the master variant generation has failed.

3. (*Optional*) Choose a cell in which the master variant of the selected layout instance needs to be created. Leave the default selection, if you want the variant to be created in the current cell.

   (*Optional*) Choose the view name for the master variant. You can choose to retain the default variant name, `layout_variant_<number>`, or type in a new view name.

   If the view name to use already exists, you can choose it from the drop-down list. In this case, a thumbnail image of the view displays in the Master Variant form. When an existing variant view is selected, the new variant is not created on disk, but the selected layout instance is remastered.

© 2023 Cadence Design Systems, Inc.

4. Click *OK* to proceed with the master variant generation.

   The physical binding of the new master is automatically updated in CPH and the instance remains bound to the schematic instance. If the instance selected is a transparent instance, the physical binding is automatically updated in CPH, but the instance remains unbound.

   Cadence recommends that manual editing of physConfig be avoided. But, in cases where such edits are required, these edits will need to be manually saved in the Configure Physical Hierarchy window.

Related Topics

Master Variant Form

Flattening a Layout Instance

Configure Physical Hierarchy Window

Transparent Layout Hierarchy Generation

Transparent layout hierarchy is the layout hierarchy other than the top-level hierarchy, which does not match the schematic hierarchy, but the binder can still bind to the instances within it. You can think of transparent instances as instances that the binder can “see through”. This means the binder does not consider the transparent instances as leaf-level instances and can bind to the instances and nets within.

You may choose to introduce a transparent layout hierarchy in your design to expedite layout design when using design styles such as half-cells. Alternatively, you can introduce a transparent layout hierarchy in a design and reuse the layout within the same design or in another design.

Layout XL does not support automatic layout generation (*Generate All From Source)* when descended into a transparent instance master but automatic updates to rectify any connectivity, parameter, and master mismatches with the source are supported via the *Update Components And Nets* command. In addition, interactive commands such as interactive chaining and folding are supported and so are the manual edits within a descended transparent instance master. The *Check Against Source* command can be used to report any mismatches with the source and the Navigator assistant gets interactively updated to reflect the correct binding status of the edited instances.

Likewise, the Define Device Correspondence form accurately updates the schematic and the layout trees when you descend into the transparent hierarchy.

© 2023 Cadence Design Systems, Inc.

- The layout tree of the Define Device Correspondence form updates to display only the instances that are within context of the transparent instance that you have descended into.
- The schematic tree of the Define Device Correspondence form updates to display only the unbound instances and the bound instances that are within context of the transparent instance that you have descended into.

  Cadence does not advise introducing an additional hierarchy in Layout XL and recommends that you keep the schematic and the layout hierarchies in sync as much as possible. But, for design situations where an additional level of hierarchy is required, Layout XL can help by keeping the additional, introduced hierarchy XL-compliant.

When working with a transparent layout hierarchy, using the following commands in Layout XL or a higher tier can ensure XL-compliance:

- Make Cell
- Set Instance as Transparent
- Unset a Transparent Instance
- Flattening a Layout Instance

Related Topics

Staying XL-Compliant by Manipulating the Layout Hierarchy - Part 1 (Introduction)

Staying XL-Compliant by Manipulating the Layout Hierarchy - Part 2 (Using the Flatten Command)

Staying XL-Compliant by Manipulating the Layout Hierarchy - Part 3 (Using the Make Cell Command)

Make Cell

When used in Layout XL, the *Edit – Hierarchy – Make Cell* command enables you to create a new cellview carrying a transparent cell instance by using existing layout objects such as devices, nets, and pins. The made cell is saved on the disk and can be consumed within the layout from which it is generated or in some other layout.

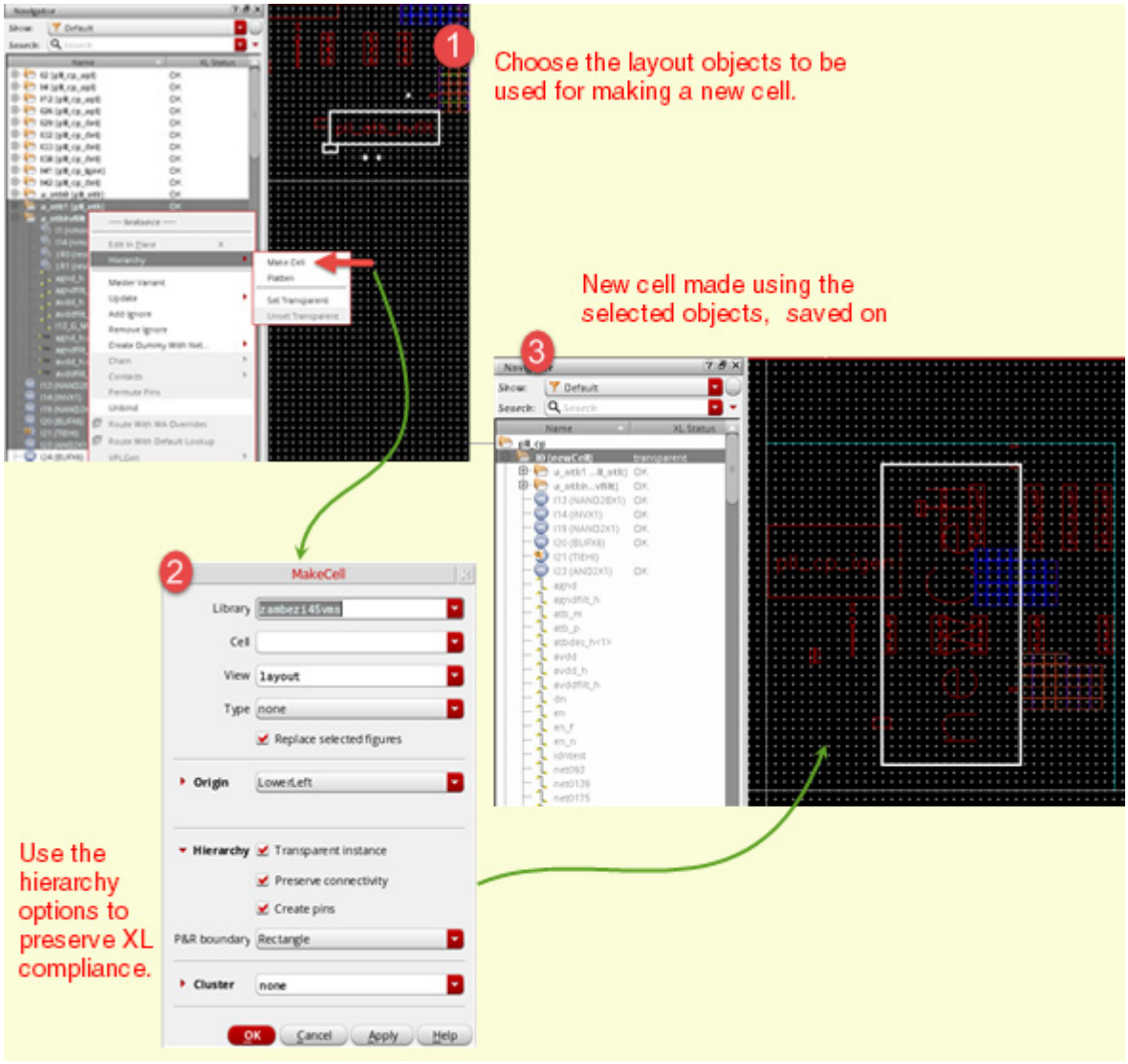
What is worth noting is that when the made cell instance is generated using Layout XL, it is XL-compliant because:

© 2023 Cadence Design Systems, Inc.

- The made cell instance created in Layout XL has logical connectivity and pins. Each made cell instance has instTerms connecting to nets, making the made cell instance connectivity-aware.
- The made cell instance created in Layout XL is, by default, transparent. This means that the binder can create one-to-one bindings to the instances inside the made cell instance.

You can benefit from creating a transparent made cell instance, if working on a half-cell design because reusing the made cell instances can expedite the layout generation for such designs.

Alternatively, you can create a transparent made cell instance using one layout and consume it in another layout.



© 2023 Cadence Design Systems, Inc.

As mentioned above, when used in Layout XL, the *Make Cell* command generates a cell that is by default:

- transparent,
- connected, and
- has its pins created.

Related Topics

Make Cell – Transparent

Make Cell – Connected

Make Cell – Pins Created

Hierarchy Manipulation. Staying XL connectivity compliant when using Flatten, Make Cell and Transparent Instances (RAK)

Working with the Binder to stay XL-compliant (RAK)

Make Cell – Transparent

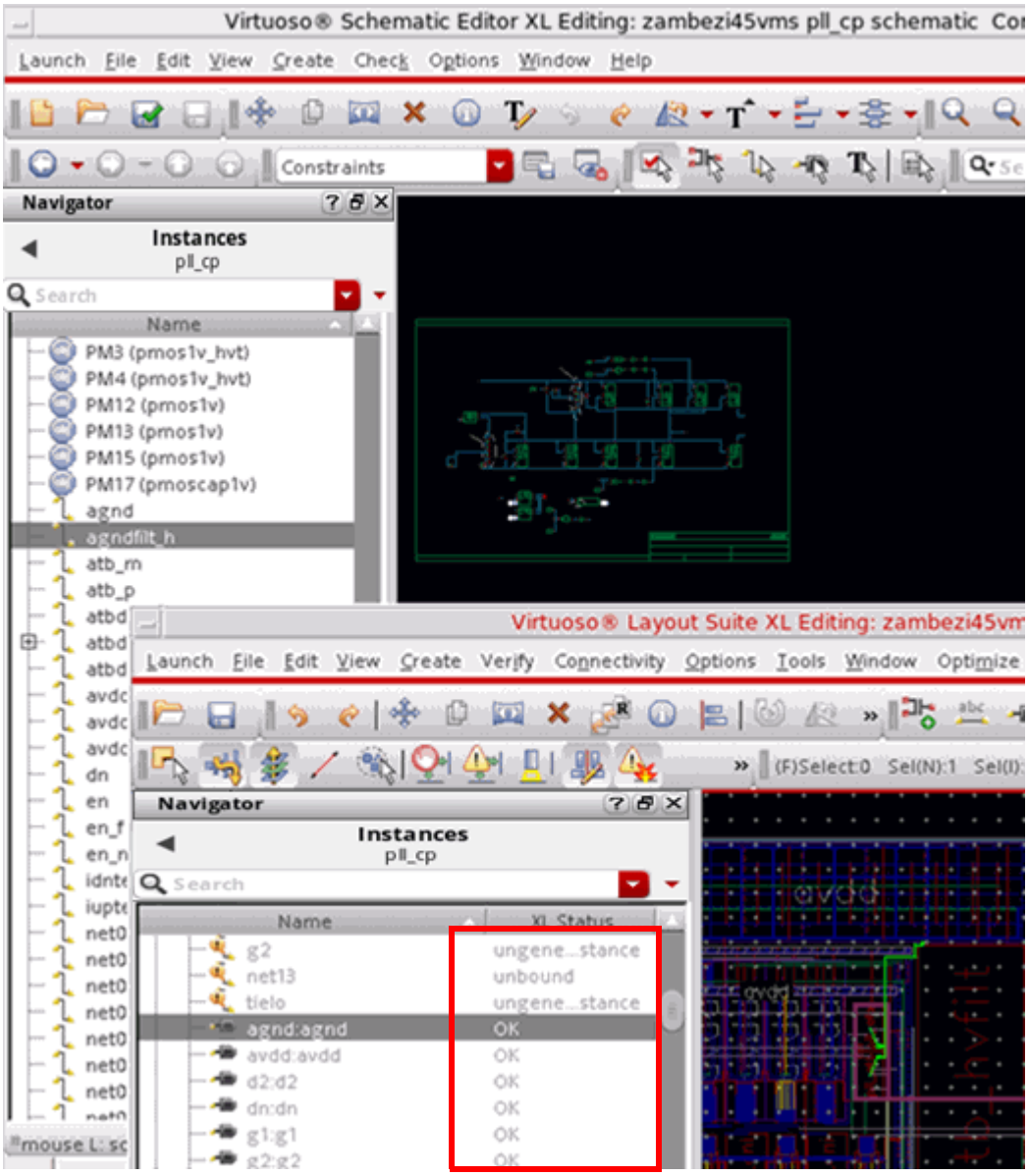
The binder can recognize the instances and nets within a made cell instance and can bind them in a one-to-one correspondence with the schematic instances.

- Even when a made cell does not have instances and nets within, the made cell is defined as ‘transparent’. For route cells without pins, this allows shape-chasing for connectivity extraction without needing an increase in the extraction depth.
- Set `makeCellTransparent` to `false`, if you do not want the made cell instance to be transparent.

  It is likely that a schematic instance in the design is bound to a layout instance that exists inside the transparent made cell instance. Layout XL supports cross-selecting such

© 2023 Cadence Design Systems, Inc.

instances that exist within a transparent instance and their corresponding schematic instances, as shown in the figure below.



Related Topics

makeCellTransparent

Make Cell – Connected

Make Cell – Pins Created

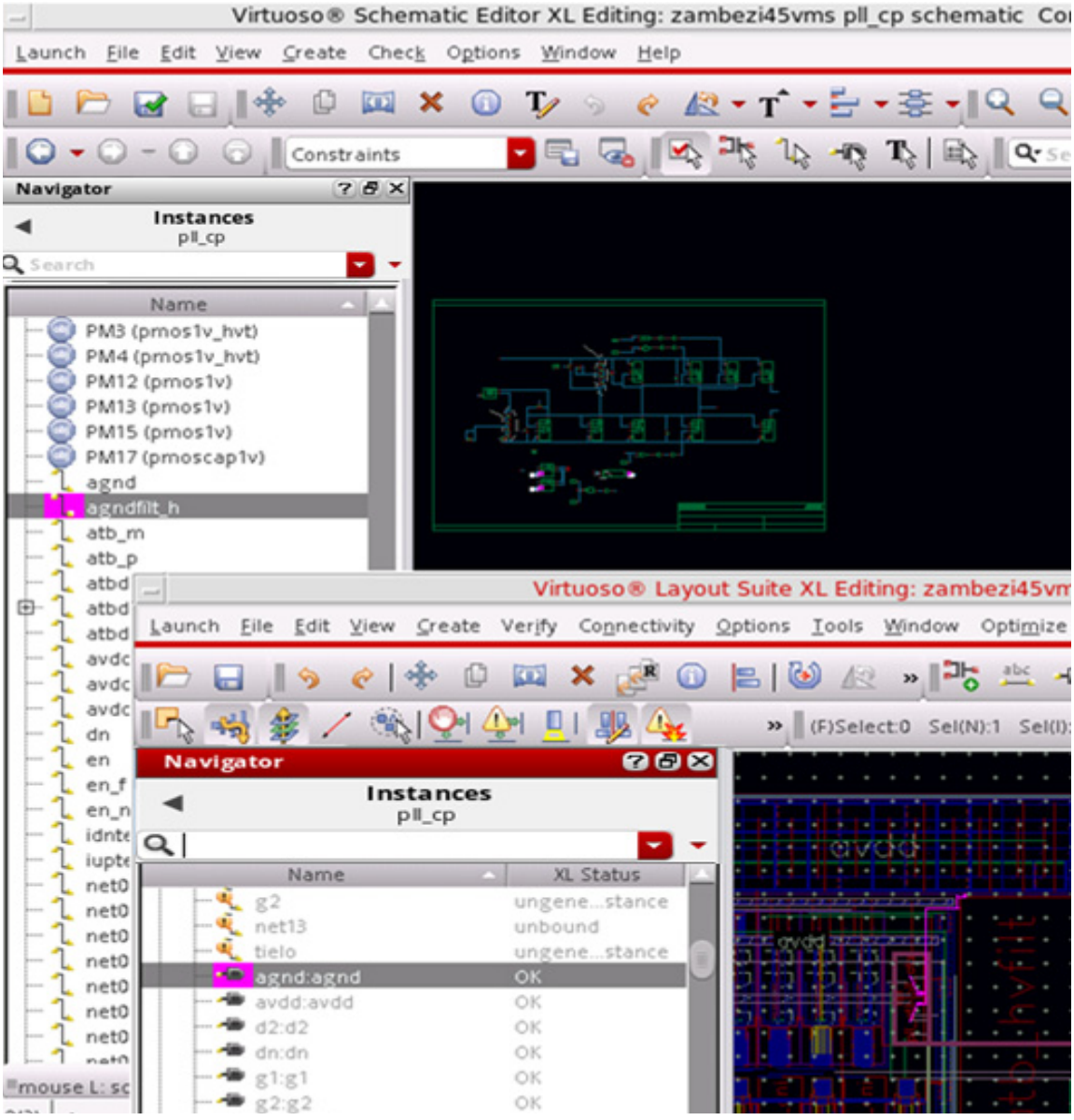
© 2023 Cadence Design Systems, Inc.

Make Cell – Connected

The made cell reflects the correct logical connectivity in terms of the instance terminals, terminals and nets, shapes, and their assignments.

Set `makeCellPreserveConn` to `false`, if you do not want connectivity of the instances inside the made cell to be preserved.

To view the connections between the transparent made cell instance and the corresponding schematic instances, you can probe the instances and nets across the schematic and layout views, as shown in the figure below.



© 2023 Cadence Design Systems, Inc.

Related Topics

makeCellPreserveConn

Make Cell – Transparent

Make Cell – Pins Created

Make Cell – Pins Created

Pins are created for nets that are external to the made cell, if the nets to do not already have a pin created.

Set `makeCellCreatePins` to `false`, if you do not want pins to be created for the made cell instance.

The transparent made cell introduces an additional level of hierarchy that does not exist in the schematic.



Cadence recommends that you keep the logical and the physical hierarchies synchronized, wherever possible. But, for design situations where an additional level of hierarchy is required, Layout XL can help you by keeping the additional, introduced hierarchy XL-compliant.

If an additional hierarchy already exists in the design and you want to flatten it, you can use the *Flatten* command.

Related Topics

Make Cell Pin Creation

makeCellCreatePins

Flattening a Layout Instance

Make Cell – Transparent

Make Cell – Connected

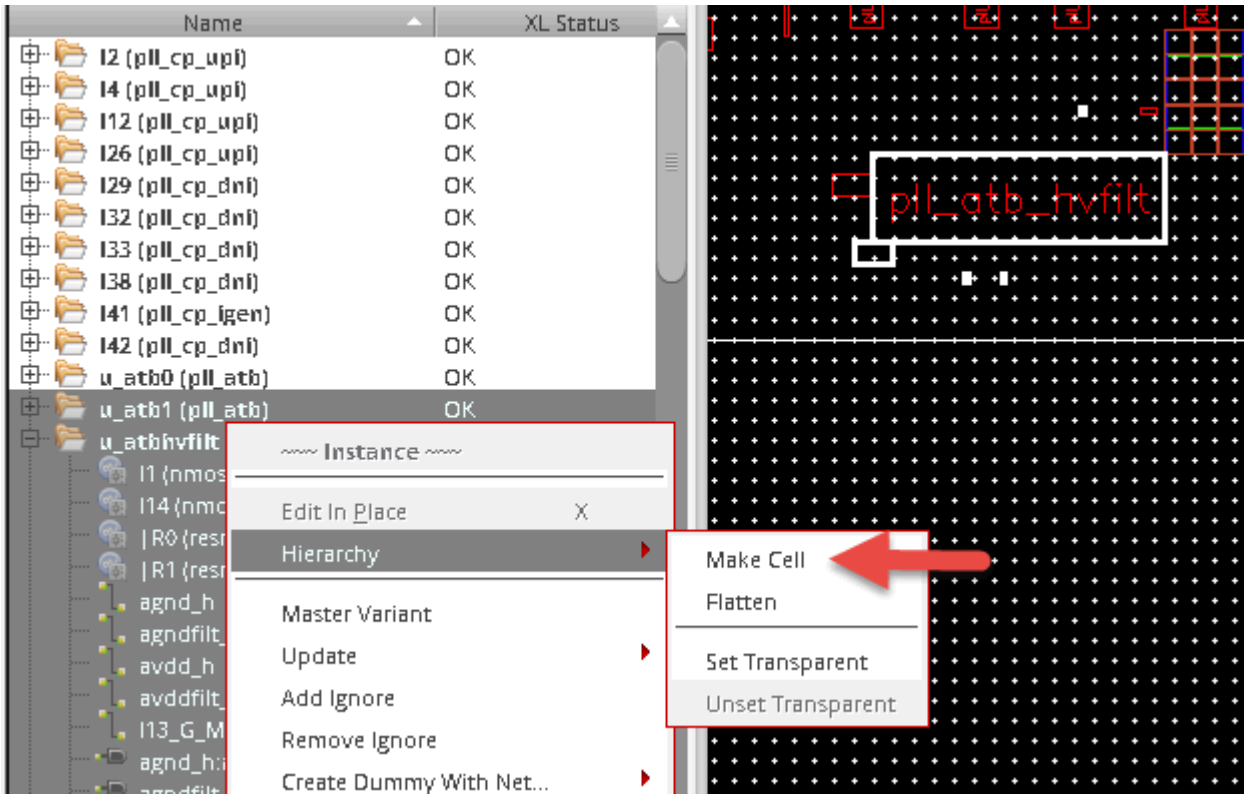
© 2023 Cadence Design Systems, Inc.

Creating a Made Cell

To create a made cell with or without objects selected:

1. In the layout canvas or the Navigator assistant, right-click and choose the *Hierarchy – Make Cell* command from the shortcut menu.

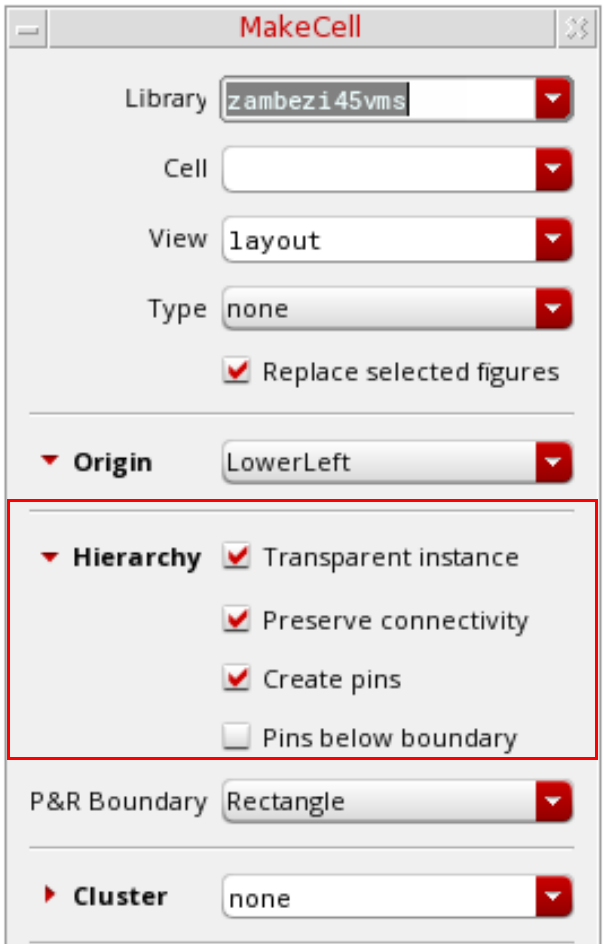
   Alternatively, you can choose the *Edit – Hierarchy – Make Cell* command.



2. The MakeCell form opens. The *Hierarchy* group box options *Transparent instance*, *Preserve connectivity*, and *Create pins* are selected by default to generate a made cell that is XL compliant.

© 2023 Cadence Design Systems, Inc.

The form displays the *Hierarchy* group box only when opened using Layout XL.



Specify a name for the new cell.

Hierarchy options are displayed only when the form is opened in Layout XL. The made cell, by default, is transparent, preserves its connectivity, and has its pins created.

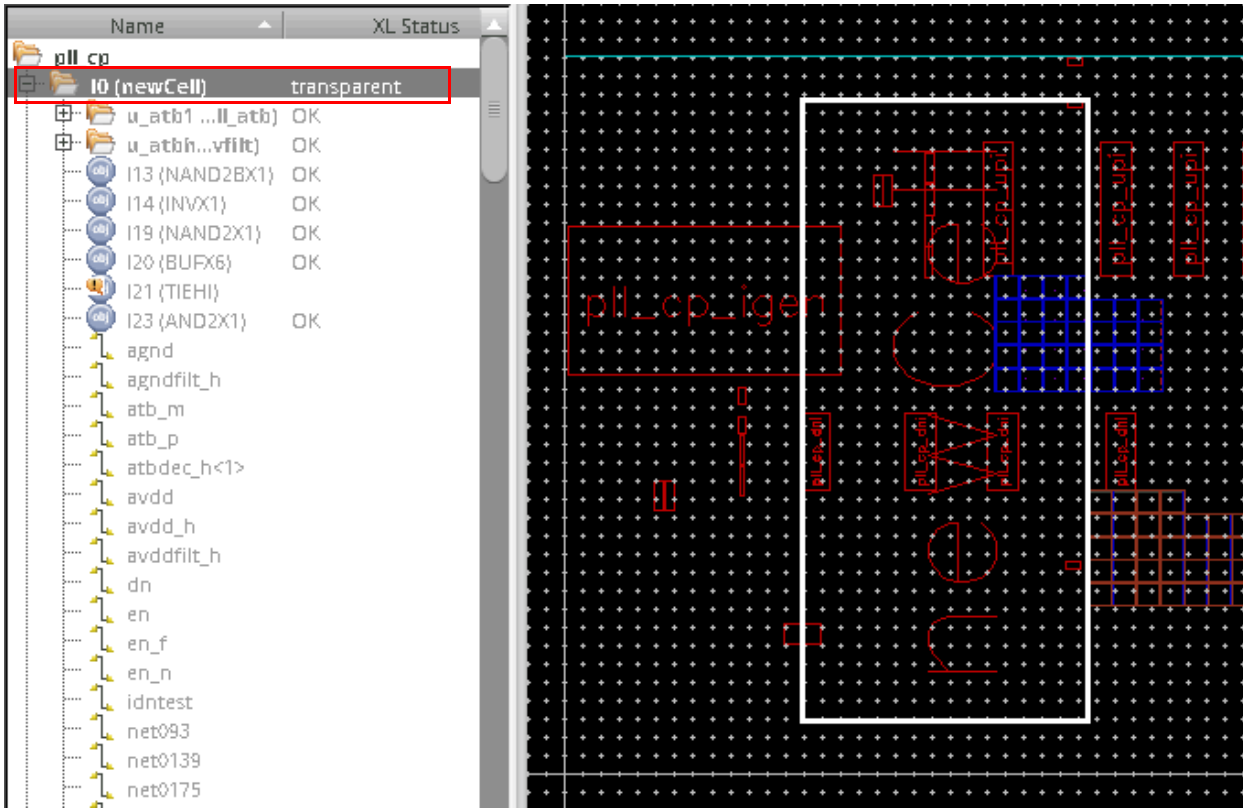
3. In the *Cell* field, type a name for the cell.
4. Optionally,
   a. Specify the *Library* and *View* name, if different from the layout from which the objects have been selected.
   b. Choose a predefined make cell type.
   c. Choose the other Make Cell options, as required.
   d. Choose to create a PR boundary for the made cell. You can also choose the PR boundary shape.

Alternatively, you can use the `makeCellPRBoundary` environment variable to specify whether a PR boundary needs to be created for the made cell, and if so, what should be the shape of the PR boundary.

© 2023 Cadence Design Systems, Inc.

5. Click *OK*.

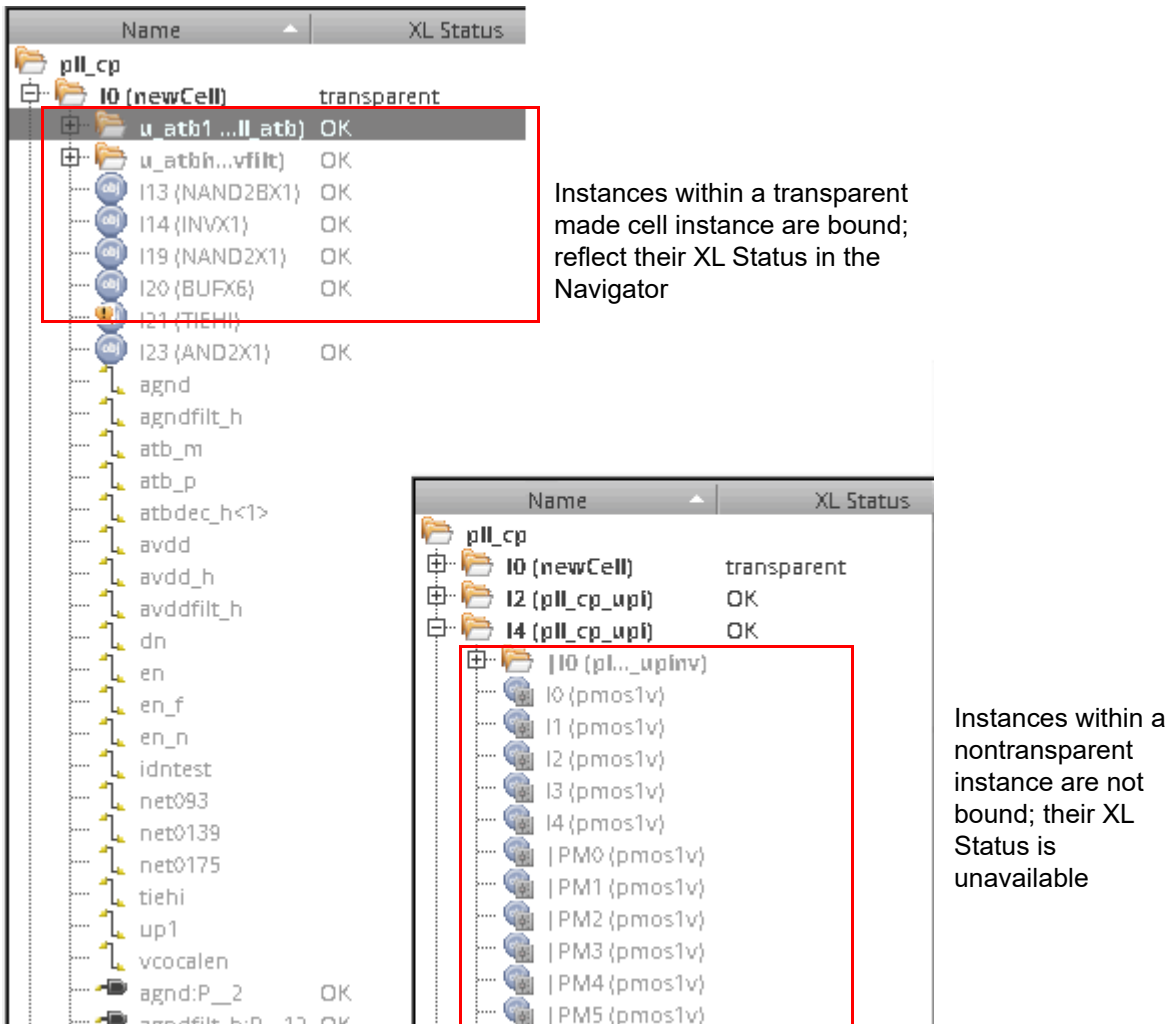
   A new cellview carrying an instance of the made cell is created, as shown in the figure below. The made cell instance reflects an *XL Status*, *transparent*.



   Notice in the figure below that the instances within the transparent made cell instance `I0` are bound and reflect their *XL Status* as *OK*. On the other hand, instances within a regular, non-transparent instance, such as `I4` do not reflect any *XL Status* because the

© 2023 Cadence Design Systems, Inc.

binder treats them as leaf-level instances. Therefore, the binder cannot bind to the instances and nets within such instances.



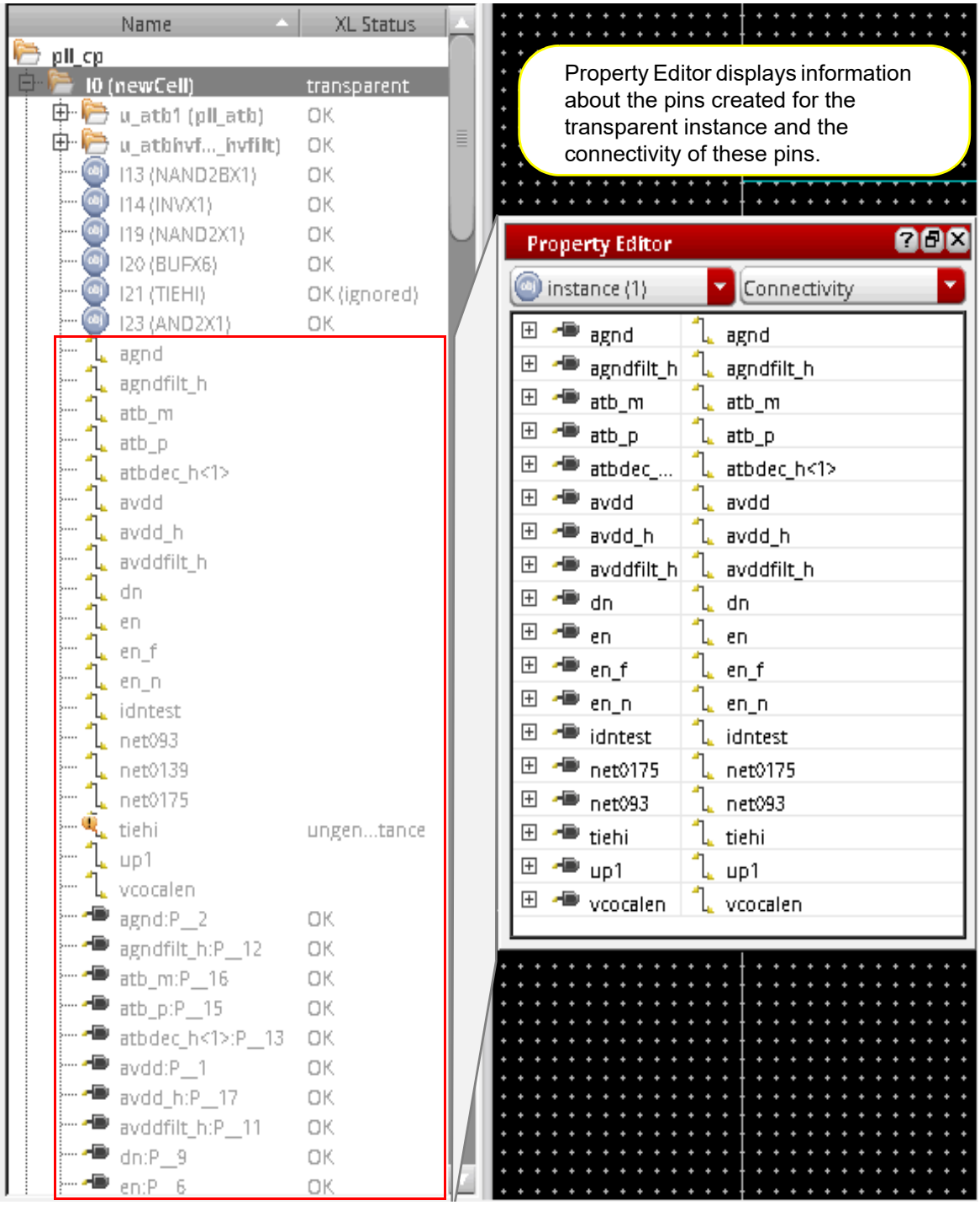
Instances within a transparent made cell instance are bound; reflect their XL Status in the Navigator

Instances within a nontransparent instance are not bound; their XL Status is unavailable

However, if an existing, hierarchical instance such as, |*4,* can be made *visible* to the binder by setting the instance as transparent, the binder can see inside the instance and

© 2023 Cadence Design Systems, Inc.

bind it to its schematic counterpart. In addition to being transparent by default, the made cell also has its pins created, as displayed in the figure below.



© 2023 Cadence Design Systems, Inc.

Having the pins created allows the made cell to be connected to the current level. Alternatively, you can choose to not have pins created when creating a make cell. For this, you must set the `makeCellCreatePins` environment variable to `false`.

By default, the make cell you create also has its connectivity preserved. But, you can choose to not preserve connectivity by setting the `makeCellPreserveConn` environment variable to `false`.

To ensure the top cellview remains XL compliant when the *Make Cell* command is run with the *Preserve connectivity* option selected, the connectivity and shape of the last pin on the terminal is retained.

Related Topics

makeCellCreatePins

makeCellPreserveConn

makeCellPRBoundary

Make Cell Form

Set Instance as Transparent

Staying XL-Compliant by Manipulating the Layout Hierarchy - Part 3 (*Using the Make Cell Command* (video)

Make Cell Pin Creation

By default, Layout XL creates pins for new made cells. However, whether a pin is created for a net depends on the following criteria:

- If the nets are external to the new made cell and they do not already have pins created, Layout XL creates new pins for the nets.
  - Strongly connected pins are created for shapes and vias that have the highest metal layer, which means multiple pinFigs can have a single pin created.
- If the masters of instances in the new made cell do not already have pins created for their terminals in the made cell, Layout XL promotes pins from the masters of these instances.
  - Shapes on the highest metal layer are promoted as pinFigs.

© 2023 Cadence Design Systems, Inc.

If the design has a Fluid Guard Ring (FGR) associated with the made cell, Layout XL promotes the pin shapes from the master if the `maskLayer` of the pin shapes is higher than or equal to that of the FGR.

For situations that a pin needs to be promoted, the layer purpose pair (LPP) is determined based on the following:

- If the `initIOPinPurposeNames` environment variable is set, the *Make Cell* command checks if the pin purpose can make a valid LPP for pin promotion and uses it, if found valid.
- If the `initIOPinPurposeNames` environment variable is not set, and a purpose other than `pin` or `drawing` is specified, a valid layer with its purpose defined as `pin` is used, if available. If a valid layer with `pin` purpose does not exist, a valid layer with purpose defined as `drawing` is used.
- If the `initIOPinPurposeNames` environment variable is not set and the purpose of the original pinFig is either `pin` or `drawing`, the purpose of the pin is not changed during promotion.

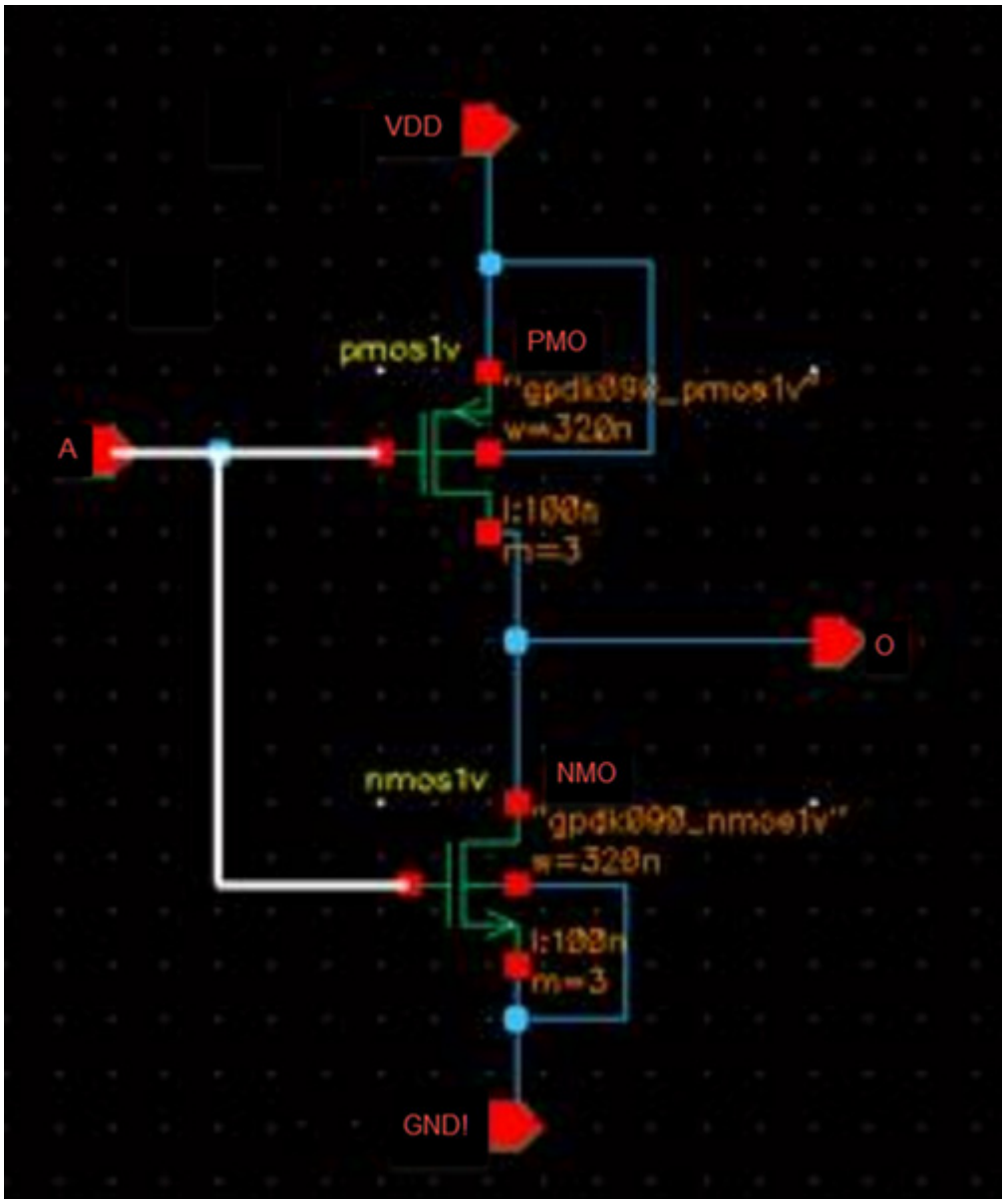
Examples

Let us now consider a few design scenarios to see how Layout XL handles pin creation for made cells that have a single external net and multiple internal nets. Let us assume that:

- `NMO` and `PMO` have `m`=3

© 2023 Cadence Design Systems, Inc.

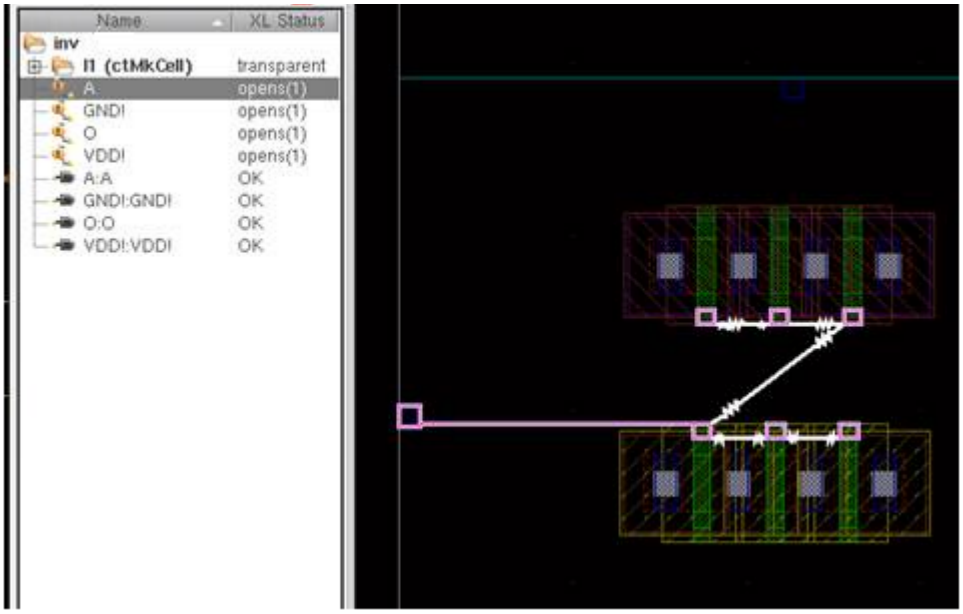
- `net A` is connected to instTerm `G`, making six points of connection



Scenario 1: Net A has no internal routing in the made cell

© 2023 Cadence Design Systems, Inc.

In this scenario, Layout XL creates pins for all the instance terminals connected to net A, making six connection points, as shown in the figure below.



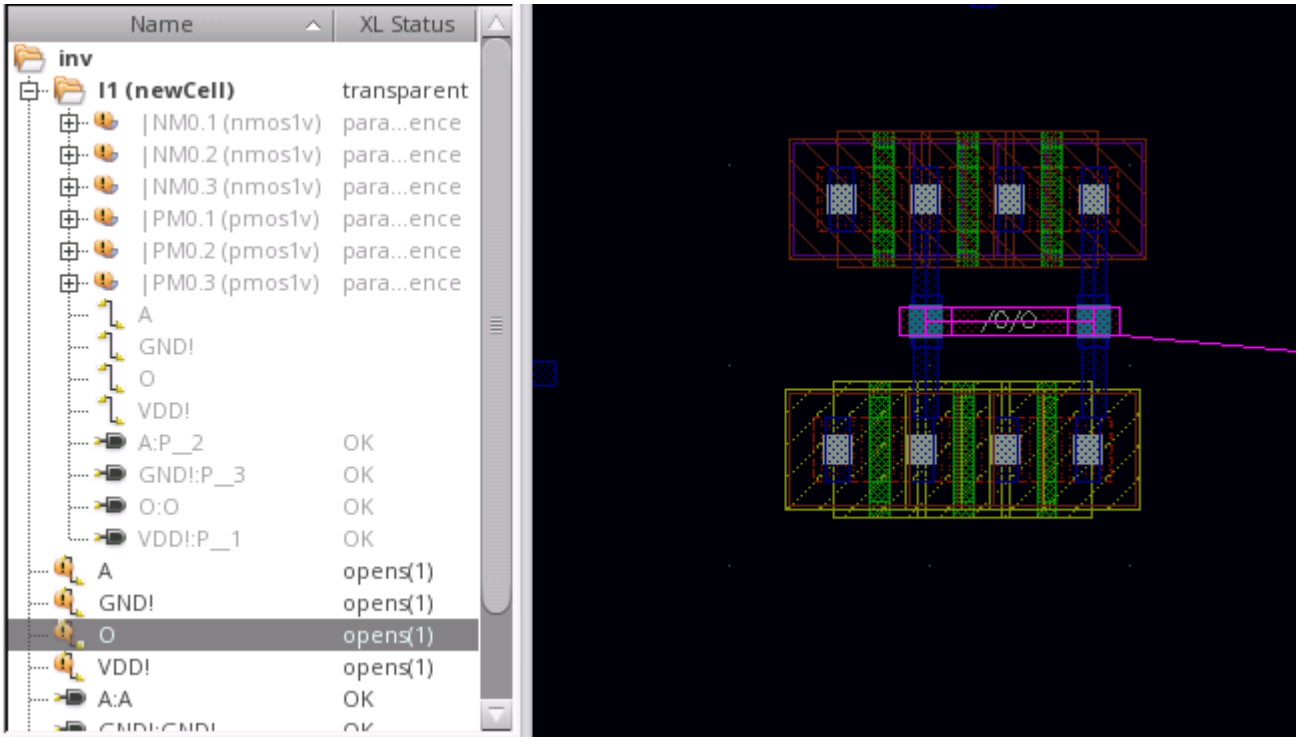
By creating pins for all the instance terminals, Layout XL gives you the flexibility to choose the instance terminal with which to connect. In addition, it gives you a clear picture of the opens from the top level, although it correctly reports only a single open. Therefore, when you route a pin `A` from the top level to one of the pins in the made cell, `A`, Layout XL marks the top level net, `net A`, as complete.

If you descend into the transparent hierarchy, the Navigator assistant displays the correct net status and selecting the net highlights the nets that need to be routed inside the made cell.

Scenario 2: Net `O` *is connected to all the four connection points inside the made cell*

© 2023 Cadence Design Systems, Inc.

In this scenario, Layout XL creates pins for all the shapes on the highest level metal to increase the routability at the next level. As shown in the figure below, `net O` selected at the top level has only one pin in the made cell, although it has multiple pinFigs.



Related Topics

Fluid Guard Ring

initIOPinPurposeNames

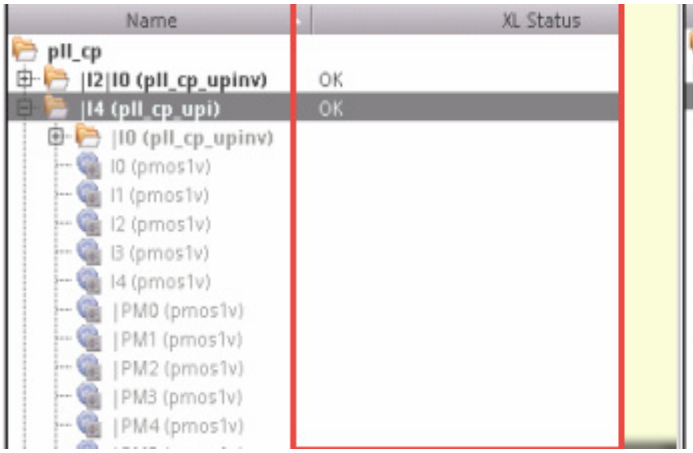
Set Instance as Transparent

Legacy designs that have additional levels of layout hierarchy compared to the schematic can be made XL-compliant if the additional layout hierarchy is made *transparent*. This enables the binder to bind to the instances and nets within the instance that was earlier considered to be a leaf-level instance.

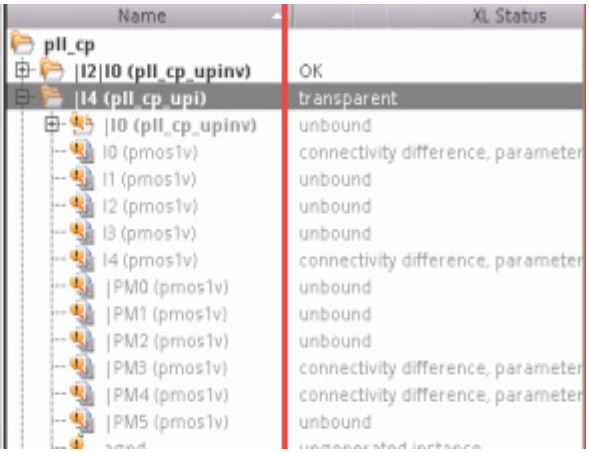
You can use the *Set Transparent* command to manually make an existing layout instance transparent.

© 2023 Cadence Design Systems, Inc.

In some cases, an existing layout instance may be automatically set as transparent during an *Update Binding* run. This can be done to improve the Layout XL compliance of the design, while preserving the layout hierarchy.



XL Status reported only for the top-level instance because the instance is not set as ‘transparent’



XL Status for the top- level instance changed to ‘transparent’

XL status also reported for the instances and nets within the transparent instance

Alternatively, if a new level of hierarchy needs to be introduced in the layout without losing XL-compliance, the *Make Cell* command is more appropriate to use.

Irrespective of the method in which a transparent instance is created, Layout XL allows these instances to be copied and replicated in the design. The transparent instances so created continue to be transparent. However, the instances and nets within the transparent instance copy might be unbound and lack XL-compliance until sufficient connectivity is added to bind them.

To identify the connectivity and binding differences between the instances and nets within a transparent instance and their schematic counterparts, you can run the *Check Against Source* command at the top level. The *Check Against Source command* reports the binding differences for instances and nets within a transparent instance in the same way as it does for regular instances and nets at the top level.

To resolve the connectivity and binding issues with the transparent instance, you can modify the transparent instance at the top level, or descend into the hierarchy by using the *Edit In Place* command. Using the *Edit In Place* command to modify the transparent instance ensures that only the selected occurrence of the transparent instance is updated, not the transparent instance master.

© 2023 Cadence Design Systems, Inc.

When you descend edit into a transparent instance master, no new schematic is opened and no new binder is launched. Layout XL remains in the context of the current schematic, binder, and the physConfig. This is unlike when descending into a nontransparent instance, which causes Layout XL to launch a new binder, a Layout XL session, and a corresponding schematic and physConfig.

When you descend edit into a transparent instance master, you cannot open the master independently in another window and launch another Layout XL session from that window. Layout XL issues a warning message indicating the operation is not supported.

In some design situations, instead of updating the instance and nets within a transparent instance, Layout XL compliance can be achieved by ignoring the additional layout hierarchy. For such designs, you can set the layout instance as nontransparent by using the *Unset Transparent* command.

Related Topics

Binding Updates

Make Cell

Creating a Transparent Instance

Unset a Transparent Instance

Creating a Transparent Instance

To create a transparent instance, which is an instance that has its hierarchy set as transparent:

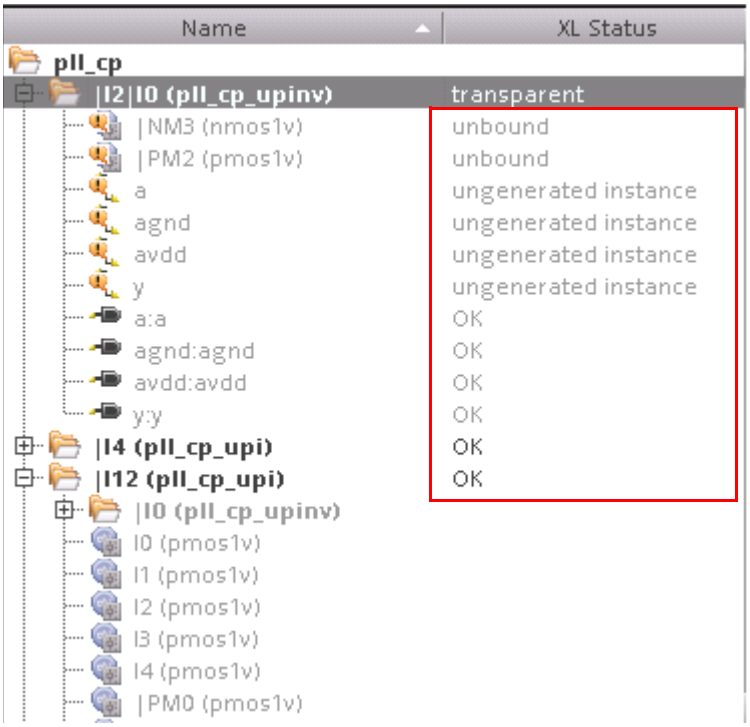
- In the Navigator assistant or canvas, right-click the instance to be set as transparent and choose the *Hierarchy – Set Transparent* command.
  - The *Set Transparent* command is context-sensitive. Only those instances that support the command show it as enabled in the shortcut menu.

© 2023 Cadence Design Systems, Inc.

- The *Set Transparent* command is not supported for mosaic instances.



The *XL Status* of the selected instance is set as `transparent`, as displayed in the figure below. This implies that the instances within the transparent instance can now be recognized by the binder. The instances and nets within the transparent instance are not grayed out like before, which indicates that these are now *visible* to the binder. In addition, the XL status of the instances and nets within the transparent instance is now reflected.



Contrary to the *Set Transparent* command, if an existing transparent instance needs to be marked as non transparent, use the *Unset Transparent* command.

Related Topics

Set Instance as Transparent

© 2023 Cadence Design Systems, Inc.

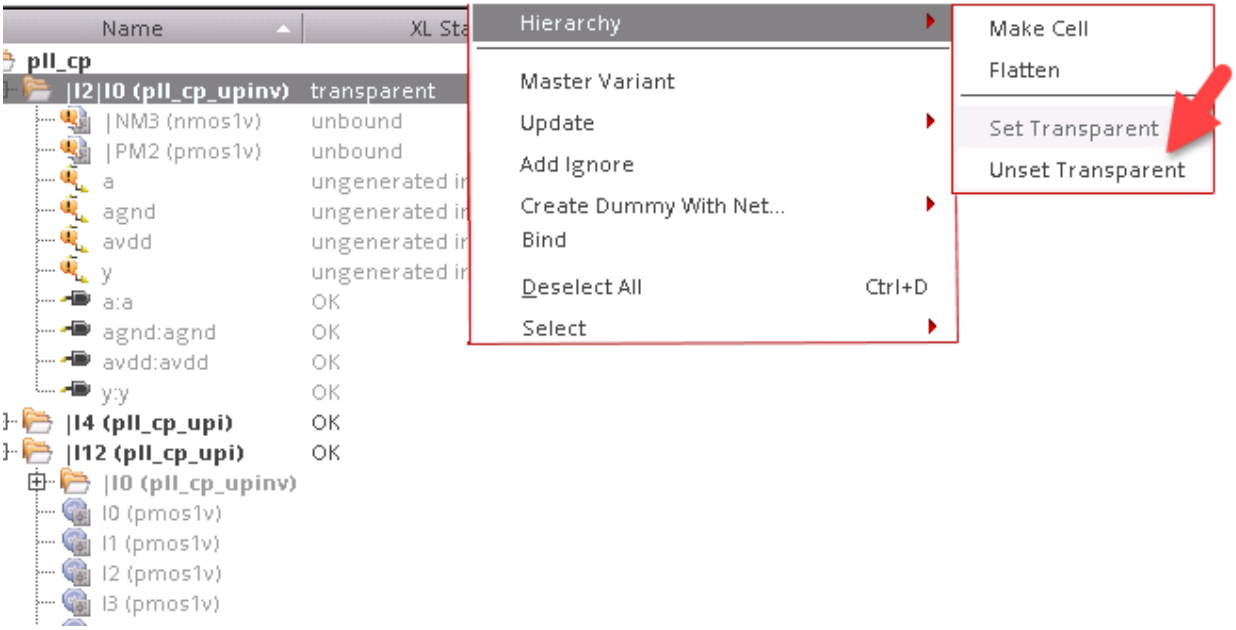
Unset a Transparent Instance

Unset a Transparent Instance

To mark a transparent instance as ‘untransparent’ so that the binder ignores the instances and nets within:

➡ In the Navigator assistant or layout canvas, right-click the transparent instance to be marked as ‘untransparent’ and choose the *Hierarchy – Unset Transparent* command.

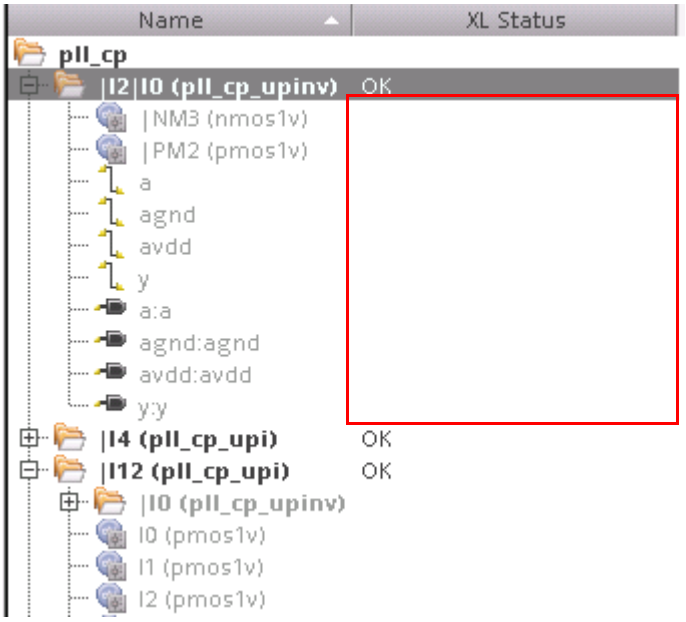
The *Unset Transparent* command is context-sensitive. Only those instances that have an XL Status set to ‘transparent’ show the *Unset Transparent* command as enabled in the shortcut menu.



The instance is now set as non transparent and the *XL Status* changes from *transparent* to reflect the current binding status, as shown in the figure below. The instances and nets within the transparent instance are no longer visible to the binder,

© 2023 Cadence Design Systems, Inc.

which means that the binding status of the instances and nets within is no longer displayed in the *XL Status* column, as displayed in the figure below.



Selecting the *Unset Transparent* command in the top-level layout removes both instance-based and occurrence-based transparency.

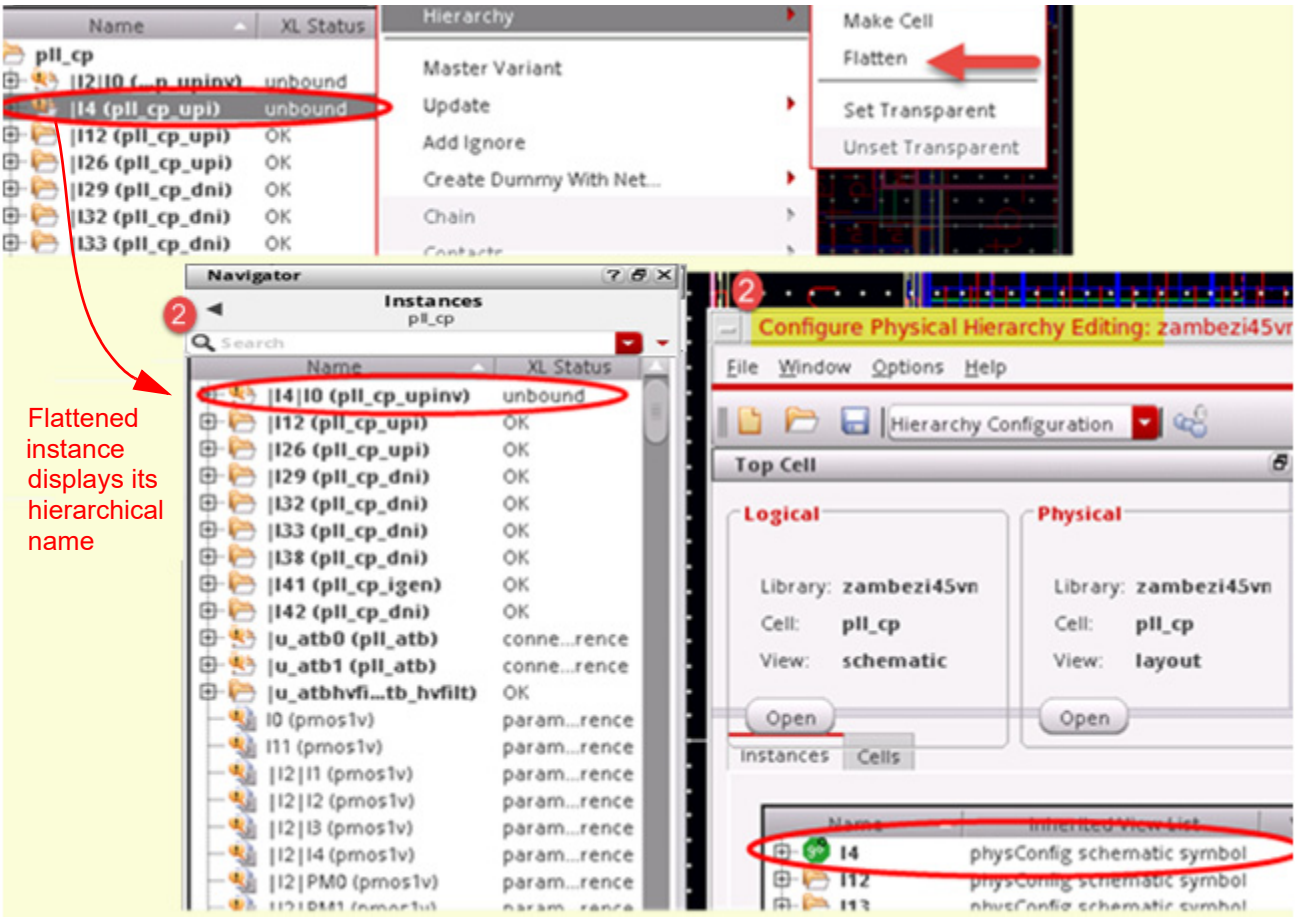
Related Topics

Set Instance as Transparent

© 2023 Cadence Design Systems, Inc.

Hierarchical Layout Instance Flattening

When used in Layout XL, the *Edit – Hierarchy – Flatten* command not only brings the contents of the flattened instance up to the top level to allow editing at the top level, but also preserves the connectivity of the flattened instance. The command also automatically sets a force descend on the flattened instance to allow binding it to the schematic instance.



By default, flattened instances have:

- *hierarchical instance names*

  The flattened instances have hierarchical names, such as `I4|I0`, where the pipe separator (`|`) separates the original name of the instance, (`I4`), and the name after the instance is flattened (`I0`).

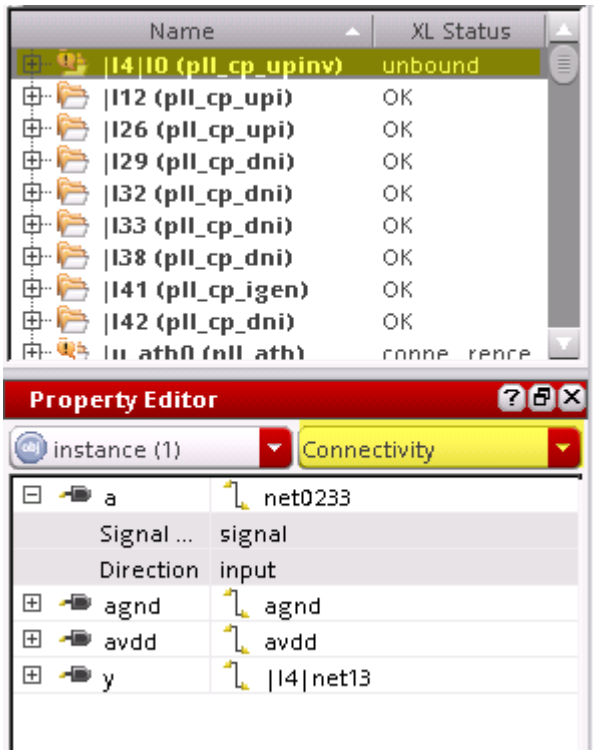
© 2023 Cadence Design Systems, Inc.

- You can choose to not display the hierarchical name for the flattened instances by setting the `flattenHierNameSep` environment variable to `false`.
- Mosaic instances do not get a pipe-separated name, when flattened. Mosaics retain their dot-separated name format, such as `M50.I51.r0c0`, after flattening.
- Unless a shape is a pinFig and the `preservePins` variable is set to `t`, labels on shapes are discarded when the *Flatten* command is run in Layout XL. You can choose to retain labels on shapes by setting the environment variable `flattenKeepLabels` to `t`.

- *connectivity preserved*

  By default, the connectivity of the flattened instance is preserved and can be verified using the Property Editor. You can also probe the selected nets. The nets that are local to the flattened instance display a hierarchical name, as shown in the figure below.

  You can choose to not have the connectivity of flattened instances preserved. For this, you must set the `flattenPreserveConn` environment variable to `false`.



- *physConfig view automatically updated*

  When a layout instance is flattened, a hierarchy mismatch is introduced. To resolve the mismatch and to maintain the correct binding and XL compliance, Layout XL automatically sets a force descend on the flattened instance in the physConfig view. This

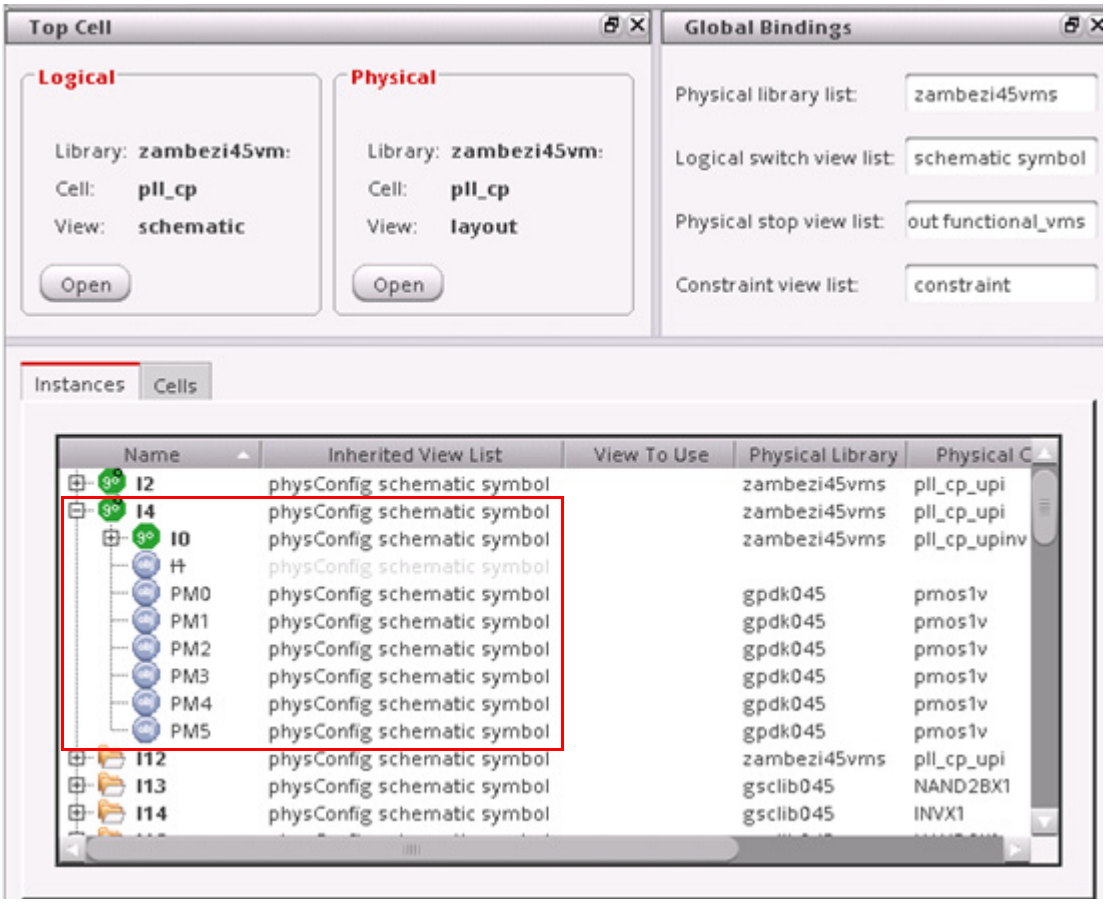
© 2023 Cadence Design Systems, Inc.

ensures that Layout XL can *look* inside the schematic instance and bind its contents to the corresponding layout instances that have been flattened and brought up to the top level.

The physConfig view is modified when Layout XL commands such as *Flatten*, *Move Hierarchy*, and *Make Transparent* are performed.

If the physConfig and layout views are not kept synchronized, it leads to incorrect bindings and a layout that is no longer XL compliant.

The changes to the physConfig view are automatically saved, unless the `autoPermuteNoAbutProp` environment variable is set to `false`. If the environment variable is set to `false`, the CPH window automatically opens and you can choose whether to save or discard the changes.



The physConfig view is synchronized with the layout if the *Undo* or *Redo* operations are performed after saving the layout.

If you have multiple instances of the same nonparameterized cell master in your layout, but you would like to update only one or few of the instances, use the *Master Variant*

© 2023 Cadence Design Systems, Inc.

command instead of *Flatten*. *Flatten* ungroups the instances at the top level, but *Master Variant* generates a different, individually editable variant of the instance, still keeping the schematic and layout correspondence in place.

Related Topics

flattenHierNameSep

flattenKeepLabels

flattenPreserveConn

autoPermuteNoAbutProp

Master Variant

Flattening a Layout Instance

To flatten an existing layout instance:

1. In the Navigator assistant or canvas, right-click the instance to be flattened and choose the *Hierarchy – Flatten* command.
   - Alternatively, you can use the *Edit – Hierarchy – Flatten* command.
   - When used in Layout XL, the *Flatten* command provides some additional features that are useful in improving the XL compliance of the design.



   The Flatten form opens.

2. Choose the appropriate flattening options and click *OK*.

© 2023 Cadence Design Systems, Inc.

Important

The Flatten form does not display any Layout XL-specific options when opened in Layout XL. This is unlike the MakeCell form, which has additional Layout XL-only options.

3. (Optional) Choose the flattening options, as required.

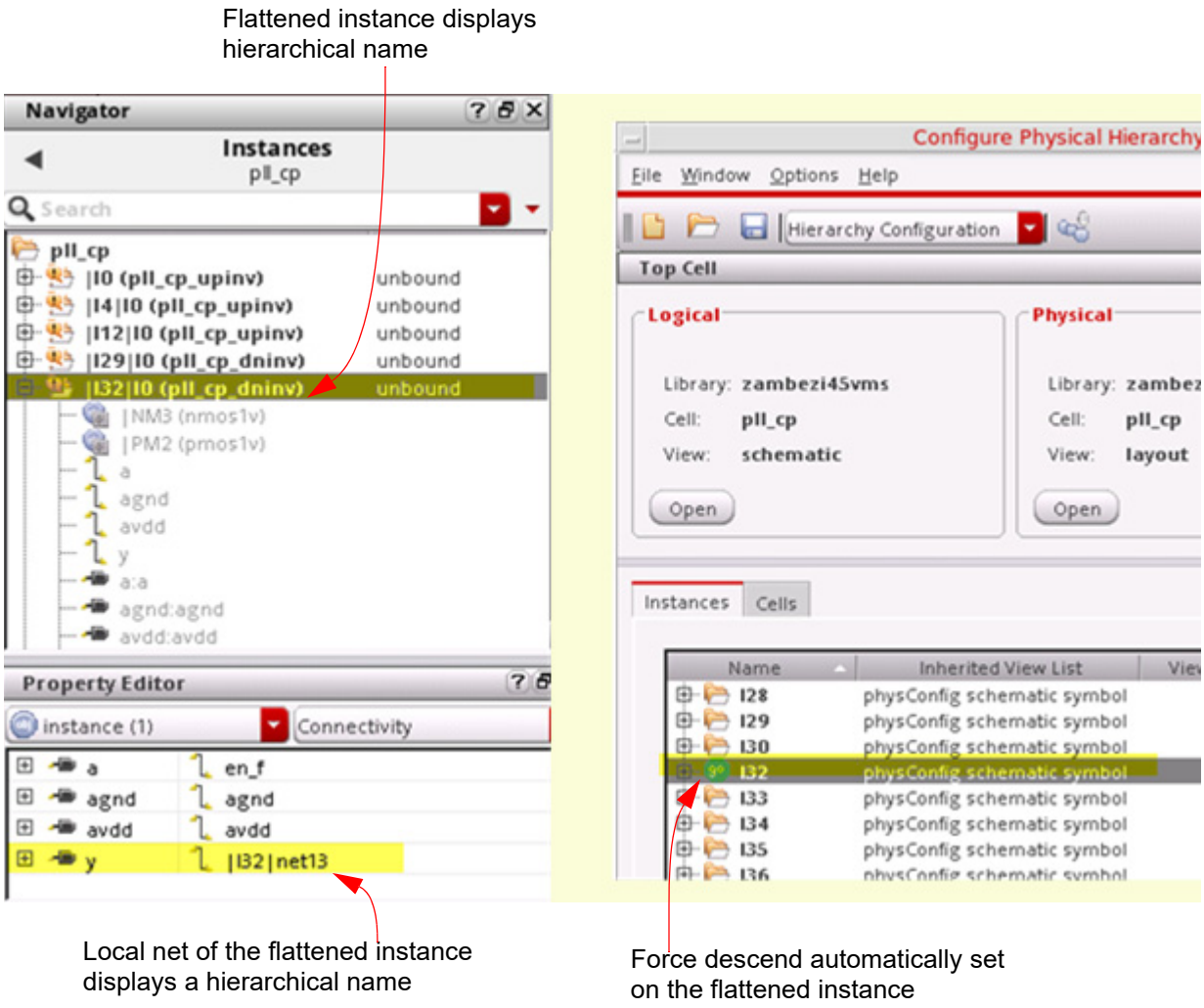
   If you enable the *Preserve Pins* option on the Flatten form, the pin labels might get flattened, if the figure is a pinFig. By default, labels are not flattened.

   For information about the various flattening options, see *Flatten Form* in the *Virtuoso Layout Suite XL: Basic Editing User Guide*.

4. Click *OK*.

© 2023 Cadence Design Systems, Inc.

The selected instance is flattened and displays a hierarchical name in the Navigator assistant. The nets local to the flattened instance also display a hierarchical name, as displayed. The CPH window displays a force descend on the flattened instance.



Related Topics

Make Cell Form

Generating Master Variants

Staying XL-Compliant by Manipulating the Layout Hierarchy - Part 2 (*Using the Flatten Command*) (video)

Working with the Binder to stay XL-compliant (RAK)

© 2023 Cadence Design Systems, Inc.

Hierarchy Manipulation. Staying XL connectivity compliant when using Flatten, Make Cell and Transparent Instances (RAK)

The Load Physical View Command

The *File – Load Physical View* command lets you import into the current cellview predefined information from another layout cellview, known as a source physical cellview. This approach lets you

- Use the full range of functionality available through the Virtuoso Layout Suite graphical user interface or the public SKILL API to create and edit the source cellview. You no longer need to manually edit an ASCII template file.
- Extend the functionality to accommodate new database objects as they are developed.
- Use any existing layout cellview as the source cellview.

There is no mechanism to resolve inconsistencies in the technology data referenced by the source cellview and the target cellview – both must use the same technology database.

Related Topics

Creating, Loading, and Modifying a Source Cellview

Loading Physical Information from Another Cellview

Specifications for Loading Cellview Information from Source to Target Cellview

Creating, Loading, and Modifying a Source Cellview

To create a source cellview:

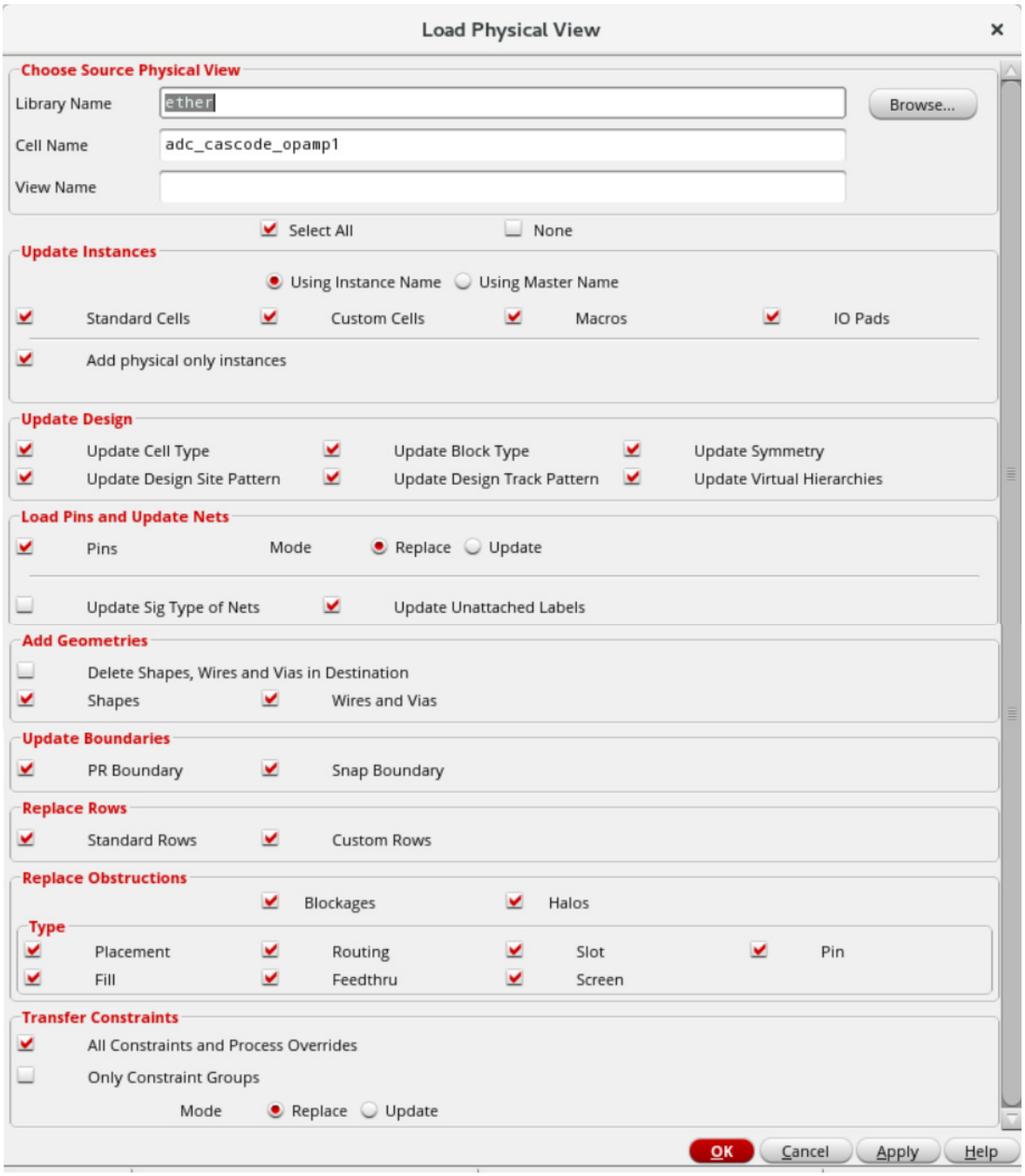
1. Use the Virtuoso Layout Suite graphical user interface or the public SKILL Function to create a cellview.
2. Save the cellview by doing one of the following.
   - Choose *File – Save* to make the current working cellview the source cellview.
   - Choose *File – Save A Copy* to save the current cellview as a source cellview with a different name.

To load information from the source cellview into a target cellview:

© 2023 Cadence Design Systems, Inc.

1. From the layout window menu bar, choose *File – Load Physical View*.

   The Load Physical View form is displayed.



2. Specify the library, cell, and view names of the source cellview in the fields provided.

© 2023 Cadence Design Systems, Inc.

3. Select a mapping file that defines instance mapping and layer mapping between the source and target cellviews.
4. Select the object information you want to copy into the target cellview by selecting relevant check boxes in each section.

   To select everything, check *Select All*. To deselect everything, check *None*.

   Choosing *Add Geometries* or *Transfer Constraints* requires four Layout GXL tokens.

   If you choose to load a particular category, **all** the objects in that category are updated. For example, if you choose to update *Standard Cells*, then all the standard cells in the source cellview are updated to the target cellview. To copy only some cells, open both cellviews and use the *Copy* and *Paste* commands instead.
5. Click *OK* to copy the selected object information from the source cellview into the target cellview.

   The target cellview is updated accordingly.

Loading Soft Block Parameters from a Source Cellview

When using the Configure Physical Hierarchy window to define soft block parameters, you can use the *Initialize Soft Block Parameters Using Physical View* command, to import soft block parameters from an existing layout cellview.

This command uses the same functionality as the *Load Physical View* command described in the previous section.

Modifying a Source Cellview

You can modify a source cellview in the same way as you would any other cellview.

- Load the cellview into the appropriate Virtuoso Layout Suite application and use the graphical user interface to edit the cellview as required and save, or
- Use the publicly available SKILL commands to edit the cellview as required.

Related Topics

Load Physical View Form

Specifications for Loading Cellview Information from Source to Target Cellview

Loading Soft Block Parameters from Another Cellview

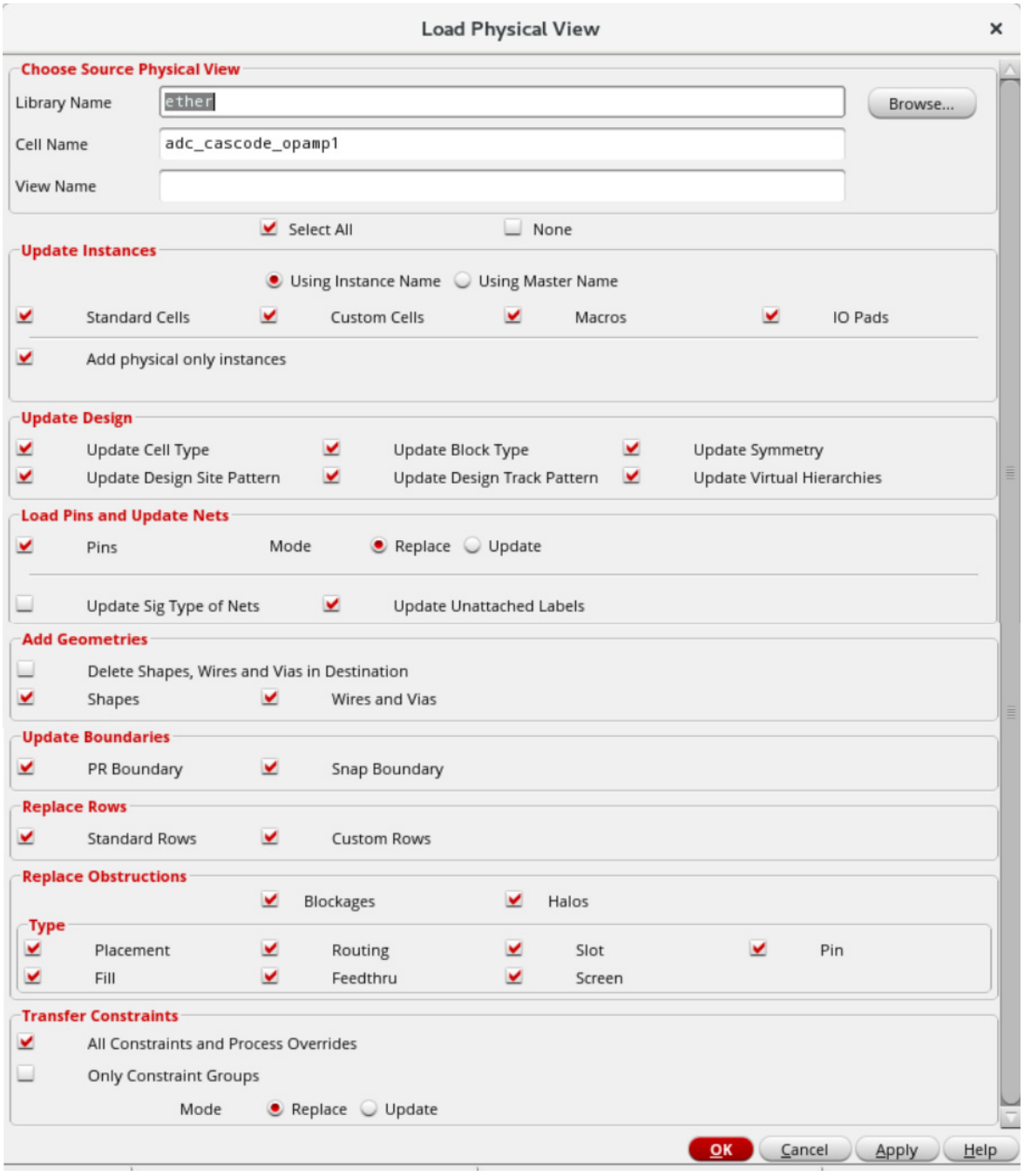
© 2023 Cadence Design Systems, Inc.

Loading Physical Information from Another Cellview

To load information from a source cellview into a target cellview:

1. From the layout window menu bar, choose *File – Load Physical View*.

   The Load Physical View form is displayed.



© 2023 Cadence Design Systems, Inc.

2. Specify the library, cell, and view names of the source cellview in the fields provided.
3. Select a mapping file that defines instance mapping and layer mapping between the source and target cellviews.
4. Select the object information you want to copy into the target cellview by selecting relevant check boxes in each section.

   To select everything, check *Select All*. To deselect everything, check *None*.

   Add Geometries or *Transfer Constraints* require four Layout GXL tokens.

   If you choose to load a particular category, all the objects in that category are updated. For example, if you choose to update *Standard Cells*, then all the standard cells in the source cellview are updated to the target cellview. To copy only some cells, open both cellviews and use the *Copy* and *Paste* commands instead.
5. Click *OK* to copy the selected object information from the source cellview into the target cellview.

   The target cellview is updated accordingly.

Related Topics

Specifications for Loading Cellview Information from Source to Target Cellview

Specifications for Loading Cellview Information from Source to Target Cellview

This topic explains how information is loaded from the source into the target cellview for each of the categories listed on the Load Physical View form. It lists the conditions under which certain information cannot be loaded and explains the dependencies between objects.

- Update Instances
- Map Instances, Layers, and LPPs between Source and Target Cellviews
- Update Design
- Load Pins and Update Nets
- Add Geometries
- Filter Pins and Geometries
- Update Boundaries

© 2023 Cadence Design Systems, Inc.

- Replace Rows
- Replace Obstructions
- Transfer Constraints

Update Instances

To preserve the connectivity of the current design, only those instance attributes that do not affect either the connectivity or the constraints are loaded from the source into the target.

Source Cellview	Target Cellview	Action
Instance `I1` present	Instance `I1` not present	Instance is not updated.
Instance `I1` present	Instance `I1` present	Update the location, orientation, and placement status of the target instance, unless its placement status is either `locked` or `firm`. Attributes such as instance master, block type, cell type and number of bits are not changed because these can affect connectivity and constraints.

Add Physical Only Instances

Physical-only instances are instances that exist in the physical domain and do not have corresponding instances in the logical domain, for example, filler cells.

The *Add physical only instances* check box lets you add physical-only instances from the source cellview to the target cellview.

The `oaBlockDomainVisibility` attribute determines whether a block is physical-only or not. You can also use the `ins -> physOnly (t/nil)` SKILL function to identify if the instance is physical or not.

© 2023 Cadence Design Systems, Inc.

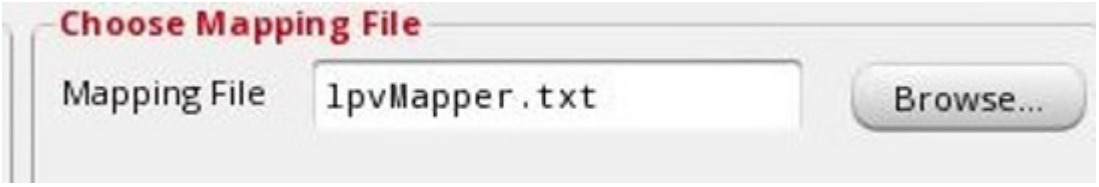
The *Add physical only instances* check box adds physical-only instances in the target cellview if they do not already exist. If the physical-only instances exist already in the target cellview, the existing instances are updated.

	Source Cellview	Target Cellview
Before	`I1 (LIB:AND)` `I2 (LIB:FILLER)` (physOnly) `I3 (LIB:FILLER1)` (physOnly)	`I1 (LIB:AND)` `I2 (LIB:FILLER)` (physOnly)
After	`I1 (LIB:AND)` `I2 (LIB:FILLER)` (physOnly) `I3 (LIB:FILLER1)` (physOnly)	`I1 (LIB:AND)` `I2 (LIB:FILLER)` (physOnly) ***Updated*** `I3 (LIB:FILLER1)` (physOnly) ***Added***

In the above example, I1 and I2 instances are updated in the target cellview because they existed in the source cellview. The I3 instance is added in the target cellview because it did not exist in the target cellview.

Map Instances, Layers, and LPPs between Source and Target Cellviews

In Layout EXL or higher tier, the *Choose Mapping File* section of the Load Physical View form, specify a file that defines the mapping between instances, pin layers, and LPPs in the source and target cellviews.



A mapping file is a text file that defines mapping information. A sample mapping file is displayed below. Here, a colon (:) separates the source and target cellviews. Lines starting with a hash (`#`) are ignored, except lines starting with `#Instance` and `#Layer`, which define instance mapping or pin layer mapping information, respectively. You can map multiple layers to the source by using a comma (,) and space as a separator as shown in the following example.

```
#Source : design:top_level:layout_xy
#Target : design:top_level:layout
#Instance(Supports one to one mapping only)
```


© 2023 Cadence Design Systems, Inc.

```
# - - - - - - - - - -
#Source     Target
|I1   :     |I2
|I3   :     |I5
|I17  :     |I16
|I9   :     |I17
#Layer Mapping(Supports one to one and many to one mapping)
# -- - - - - -- - - - -- -
#Source        Target
Metal1         : Metal8
Metal1, Metal3 : Metal5
```

A mapping file supports the following:

- Instance mapping: A mapping file supports one-to-one mapping only. In the following example, instance `I1` in the source is mapped to instance `I2` in the target. Instance `I2` updates the location, orientation, and placement status of the target instances as per the instance `I1`.

  ```
  #Instance
  #Source   Target
  I1       : I2
  ```

- Layer mapping: A mapping file supports one-to-one and many-to-one mapping for all types of shapes, wires, and routing blockages on the selected nets and metal layers. It supports the mapping of multiple layers to a single layer through the comma symbol.

  ```
  #Layer Mapping
  #Source Target
  Metal11        : Metal7
  Metal12, Metal10 : Metal8
  ```

  In the above example, the entry `Metal11:Metal7` means that `Metal11` in the source is mapped to `Metal7` in the target. The entry `Metal10, Metal12: Metal8` means that `Metal10` and `Metal12` in the source are mapped to `Metal8` in the target. All the pins of `Metal8` are created at the location of `Metal10` after pin layer mapping.

  For the pin layers, many layers can be mapped to a single layer as many-to-one mapping is supported. This kind of many-to-one mapping is not supported for instances.

- Layer purpose pair mapping: A mapping file supports one-to-one and many-to-one mapping. You can map the layer purpose pairs between source and target cellviews as shown below.

  ```
  #Layer Purpose Pair(Supports one to one and many to one mapping)
  # -- - - - - -- - - - -- -
  #Source Target
  ```

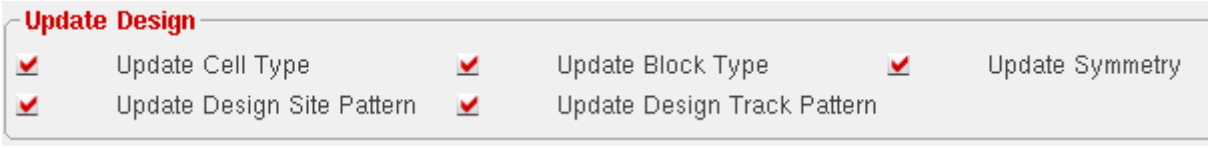

© 2023 Cadence Design Systems, Inc.

```
Metal1 : M1
Metal1 pin, Metal2 drawing : M2 pin
```

In the above example, the entry `Metal1:M1` means that `Metal1` in the source is mapped to `M1` layer purpose in the target. The entry `Metal1 pin, Metal2 drawing : M2 pin` means that `Metal1 pin` and `Metal2 drawing` in the source are mapped to `M2` layer purpose in the target.

Update Design

In the *Update Design* group box, choose the parameters that you want to update in the target cellview.



- **Update Cell Type**

  To update the cell type in the current design based on the source cellview, select the *Update Cell Type* check box. Use the Set Cell Type form to verify the changes.

- **Update Block Type**

  To update the block type in the current design based on the source cellview, select the *Update Block Type* check box. Use the Edit Properties form to verify changes.

- **Update Symmetry**

  To update the symmetry of the current cellview based on the source cellview, select the *Update Symmetry* check box.

- **Update Design Site Pattern**

  To update the `sitePattern` of the current design based on the source cellview, select the *Update Design Site Pattern* check box. A `sitePattern` can have one or more site definitions (`siteDefs`). `siteDefs` are technology library objects, which are the basic elements from which placement rows are constructed.

  Typically, macros are associated with multiple `sitedefs`.

  A `sitePattern` includes the following information:

  - **siteDefs:** Names of the `siteDefs` that are associated with the current cellview
  - **Origin:** Position of the `sitePattern`

© 2023 Cadence Design Systems, Inc.

- **Rotation:** Possible rotation values that can be associated with the `sitePattern`

Use the Create Row form to verify the changes.

- **Update Track Pattern**

  To update the track patterns in the current design based on the target cellview, select the *Update Track Pattern* check box. Track patterns may be either user-defined or defined in the technology file. Only the user-defined track patterns are updated when the *Update Track Pattern* check box is selected.

  Use the Edit Track Pattern form to verify the changes.

Load Pins and Update Nets

The *Load Pins and Update Nets* option provides the following two modes for updating pins.

- Replace Mode

  If a terminal corresponding to the source terminal exists in the target cellview, the existing terminal pins in the target cellview are deleted and the terminal pins from the source cellview are copied to the target cellview. If a terminal corresponding to the source terminal does not exist in the target cellview, the extra terminals are not created in the target cellview. The table summarizes the behavior.

Source Cellview	Target Cellview	Action
`T1` terminal `P1` pin `F1` pin figure exist	`T1` terminal does not exist	The extra terminal (`T1`) is **not** created in the target cellview to keep the schematic and logical views in sync
`T1` terminal `P1` pin `F1` pin figure exist	`T1` terminal `P2` pin `F3` and `F4` pin figures exist	The `P2` pin, `F3`, and `F4` pin figures are deleted from the target cellview. The `P1` pin and the `F1` pin figure are copied from the source cellview to the target cellview.

  The placement status of the target pins is not considered during replacement.

- Update Mode

  If the names of the terminal, pin, and pin figure in the `terminal:pin:pin figure` structure of the target cellview matches the corresponding names in the source cellview, the layer-purpose pair and the bounding box of the target pin figure is updated (as long as the placement status is not `locked` or `firm`).

© 2023 Cadence Design Systems, Inc.

If only terminal and pin names match, the extra pin figures are created in the target cellview (see example below). If only terminal names match, the extra pins and pin figures are created in the target cellview. The existing pins and pin figures are not deleted in the target cellview.

For example, consider the pin structure in the source and target cellview in the following table.

	Pin Structure in Source Cellview	**Pin Structure in Target Cellview**

Before	`T1`
	`  P1`
	`  F1, F2, F3`
	`  P2`
	`  F1`
	`Tx`
	`  Px`
	`  Fx`
		`T1`
		`  P1`
		`  F1, F2,`

After		`T1`
		`  P1`
		`  F1, F2, F3`
		`  P2`
		`  F1`

The `F3` pin figure in pin `P1`, the `P2` pin, and the `F1` pin figure in `P2` pin (highlighted above) are added in the target cellview. In addition, the `F1` and `F2` pin figures in `P1` pin are updated and no new terminal corresponding to `Tx` is created.

If any pin in the source terminal has the must-connect-all-pins mode set, the Load Physical View command copies the property along with the pin from the source terminal to the target terminal.

Add Geometries

The **Add Geometries** functionality requires 4 Layout GXL tokens.

Geometries are considered to be shapes (rectangles, polygons, paths, circles, ellipses, and donuts) and wires (pathSegs and vias). Because of the difficulty in establishing that shapes are at exactly the same location in the source and target views, this control copies geometries

© 2023 Cadence Design Systems, Inc.

from the source to the target. This can result in the creation of overlapping shapes in the target layout.

Source Cellview	Target Cellview	Action
Shape or wire present	A shape or wire is present at the same location	Shape or wire is copied, which might lead to the creation of overlapping shapes.
Shape or wire present	No shape or wire present at the same location	Shape or wire is copied

The table below summarizes how net assignments are handled when a shape is copied from the source to the target.

Source Cellview	Target Cellview	Action
Shape is assigned to `netA` and has `lxStickyNet=t`	`netA` present	Shape is copied and assigned to `netA`. `lxStickyNet` property is added to the shape and set to `t`.
Shape is assigned to `netA` and has `lxStickyNet=t`	`netA` not present	Only the shape is copied. `lxStickyNet` property and net assignment are both ignored.
Shape has no `lxStickyNet` property		Only the shape is copied.
Shape has no net assignment but has `lxStickyNet=t`		Only the shape is copied. `lxStickyNet` property is ignored.

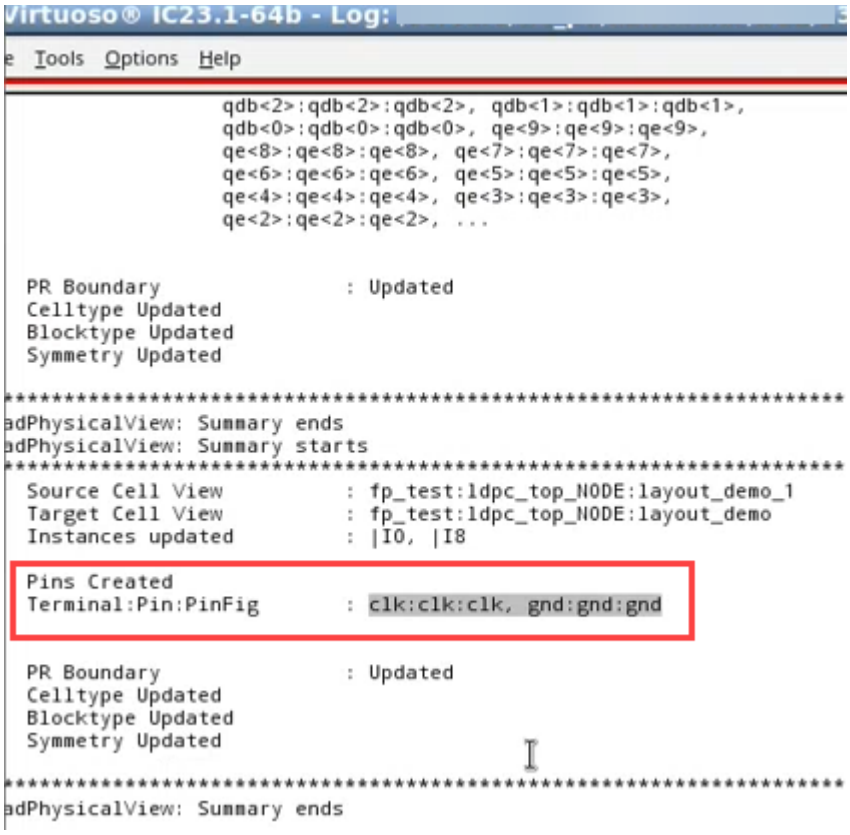
© 2023 Cadence Design Systems, Inc.

Filter Pins and Geometries

This option allows you to selectively transfer pins, routes, wires, and shapes on the specified metal layers.

Add Pins and Geometries for Selected Nets

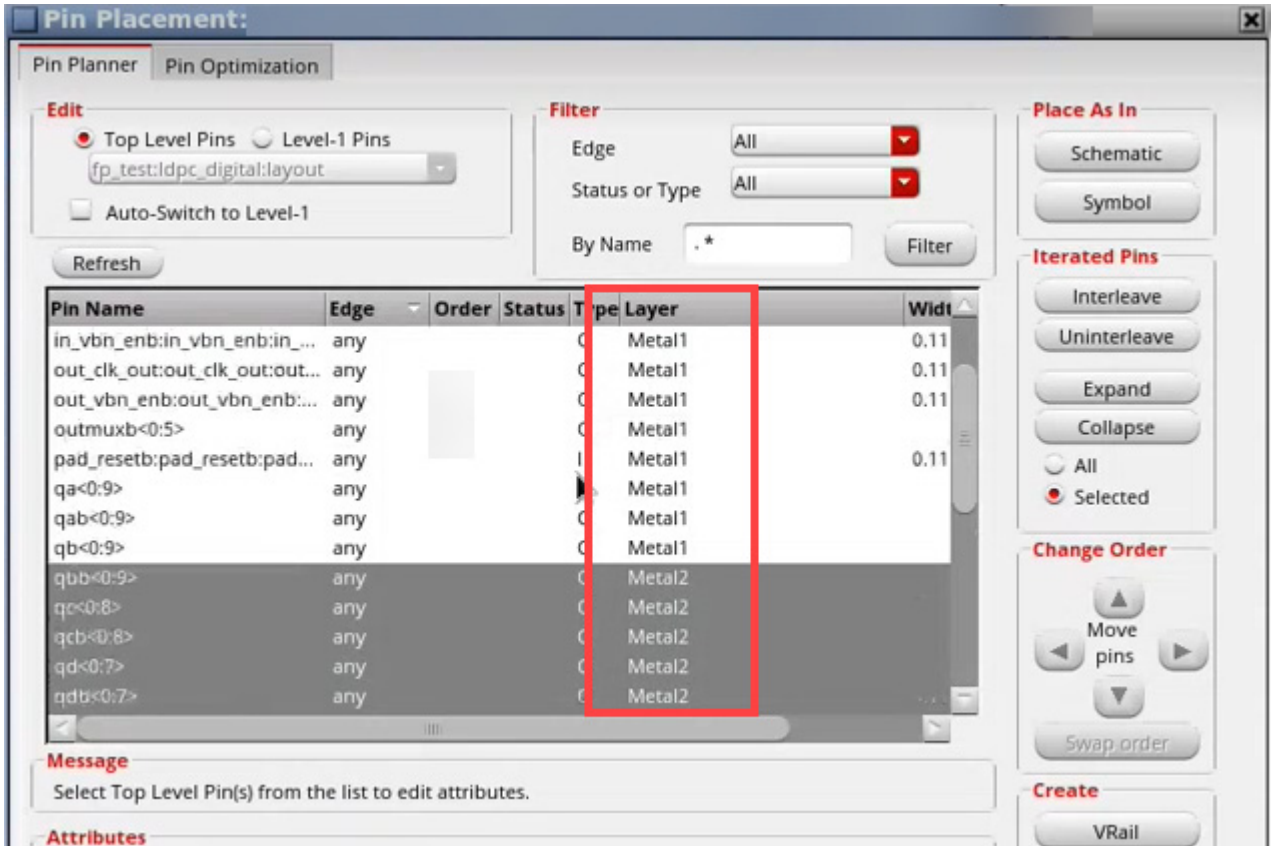
Consider a layout design where you add `clk` and `gnd` pins on the selected nets in the target cellview. When you select *Add Pins and Geometries for Selected Nets* and run the *Load Physical View* command, only the `clk` and `gnd` pins are created in the target cellview.



© 2023 Cadence Design Systems, Inc.

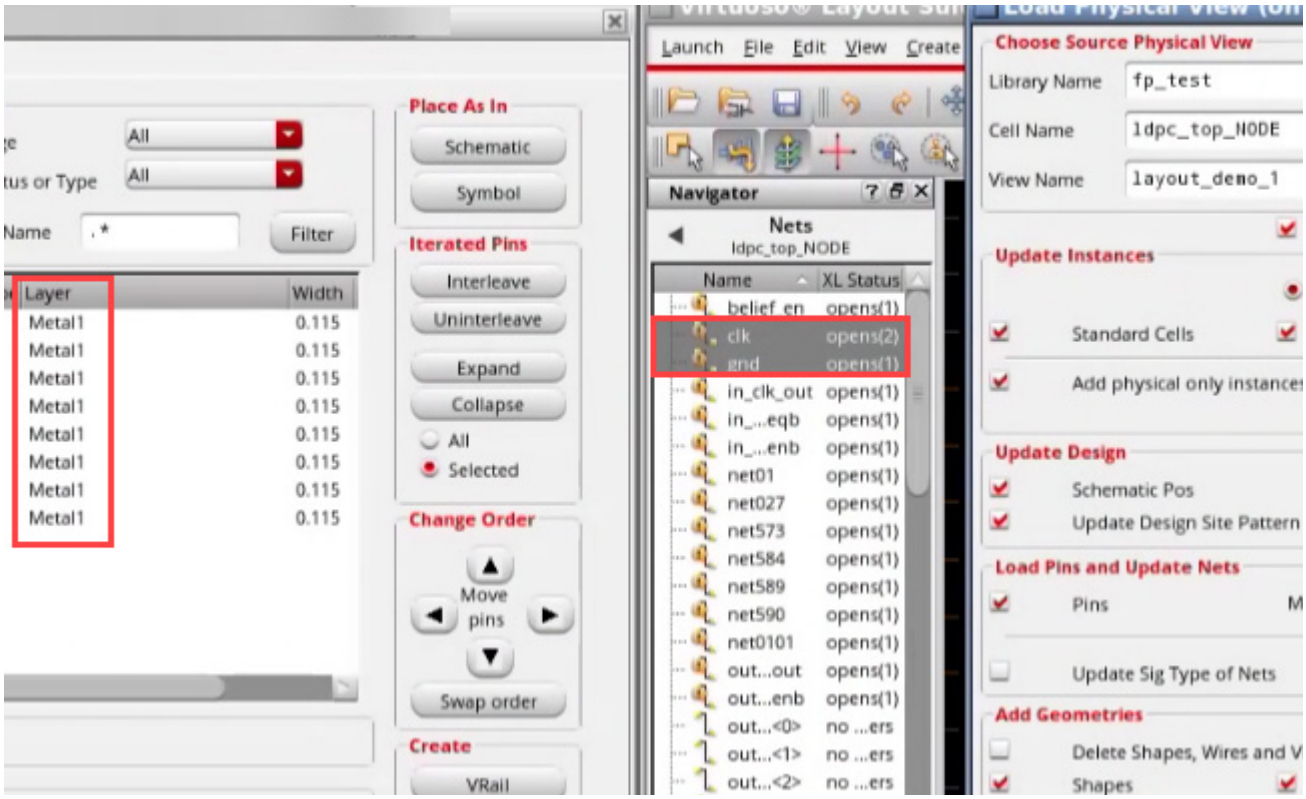
Add Pins and Geometries for Specified Layers

Consider a layout design where the source view contains pins from `Metal1` to `Metal2` metal layers as shown in the image below.



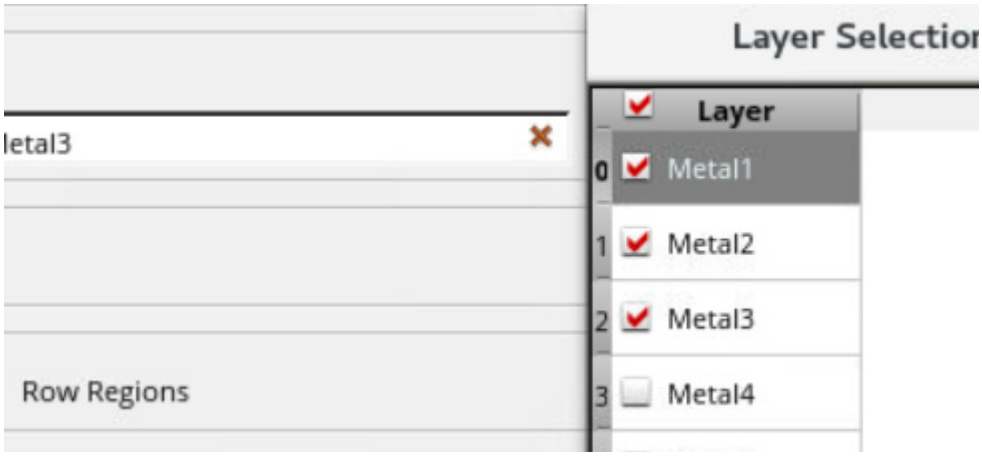
© 2023 Cadence Design Systems, Inc.

When you select `Metal1` in *Layer Selection* table and run the *Load Physical View* command, only those pins which are on `Metal1` layer are transferred to the target view.



Define

The *Define* button opens the Layer Selection table, which allows you to select the metal or poly layers.



© 2023 Cadence Design Systems, Inc.

Update Boundaries

This information applies to both the place and route boundary and the snap boundary.

Source Cellview	Target Cellview	Action
Boundary present	Boundary not present	Copy the boundary from the source to the target cellview.
Boundary present	Boundary present	Update the points of the target boundary, along with other attributes such as `IOBox` and `CoreBox`.

Replace Rows

The standard and custom rows in the target cellview are deleted and replaced by the rows defined in the source cellview.

In Layout EXL or higher tier, row regions in the target cellview are replaced with the row regions defined in the source cellview.

Replace Obstructions

An obstruction is implemented either as a blockage or a halo of one of the following types.

- Placement (area blockage or halo)
- Routing (layer blockage or halo)
- Slot (layer blockage or halo)
- Pin (layer blockage or halo)
- Fill (layer blockage or halo)
- Feedthru (layer blockage or halo)
- Screen (layer blockage or halo)

All the obstructions in the target cellview are deleted and replaced by the obstructions defined in the source cellview with the following conditions.

- A blockage can be owned by an instance or by a cluster. If the owner is present in the target cellview, it is updated on copying the blockage from source to target.

© 2023 Cadence Design Systems, Inc.

- A halo can be owned by an instance or by the place and route boundary. Before copying a halo, the software checks that the owner is present in the target cellview. If it is not, the halo is not copied.

  Wiring and *Via* obstructions are to be removed from DFII, therefore they are not handled by this functionality.

- Any existing obstructions in the target cellview automatically get deleted before any new obstructions get loaded into the target cellview.

Transfer Constraints

The **Transfer Constraints** functionality requires 4 Layout GXL tokens.

Use the options in the **Transfer Constraints** section to transfer relationship constraints, such as the alignment constraint, process rule overrides, and constraint groups, from the source maskLayout view to the target maskLayout view. This section contains the **All Constraints and Process Overrides** and **Only Constraint Groups** check boxes, which are mutually exclusive. Therefore, selecting one check box automatically deselects the other.

If none of the check boxes is selected, no constraints are transferred. Also, the *Replace* and *Update* buttons for *Mode* are disabled.



All Constraints and Process Overrides

Select the **All Constraints and Process Overrides** check box to transfer all constraints, process rule overrides, and constraint groups from the source cellview to the target cellview. The constraints can be transferred in one of the following modes:

- **Replace**: Deletes all constraints, process rule overrides, and floating constraint groups from the target cellview; and copies those from the source cellview to the target cellview.

  Floating constraint groups are the constraint groups that are either not assigned to an owner or do not have a built-in constraint group.

- **Update**: Copies constraints, process rule overrides, and constraint groups from the source cellview to the target cellview.

© 2023 Cadence Design Systems, Inc.

- The mapping criteria for constraint and process rule override updates is found by matching members in the source and target cellviews. A given source constraint is mapped to a target constraint if the number of matching members is highest. Load Physical View updates the members and parameters of the mapped target constraint from those in the source constraint.
- The mapping criteria for constraint groups are the constraint group names. In case a match is found, the contents of the constraint group or the process rule overrides are replaced with that in the source.

The default constraint group contents are transferred in the *Replace* and *Update* modes.

Only Constraint Groups

Select the **Only Constraint Groups** check box to transfer only constraint groups. Choose one of the following modes:

- **Replace**: Deletes all floating constraint groups from the target cellview, and copies those from the source cellview to the target cellview.
- **Update**: Copies new floating constraint groups from the source cellview to the target cellview. For constraint groups with matching names in the source and target cellviews, the content of the target constraint group is replaced with that of the source constraint group.

The default constraint group contents are transferred in the *Replace* and *Update* modes.

Related Topics

Edit Soft Blocks Form

Preparing Data for Floorplanning

Creating Track Patterns

Working With Pin Connectivity Models

Layout Support for Row Regions

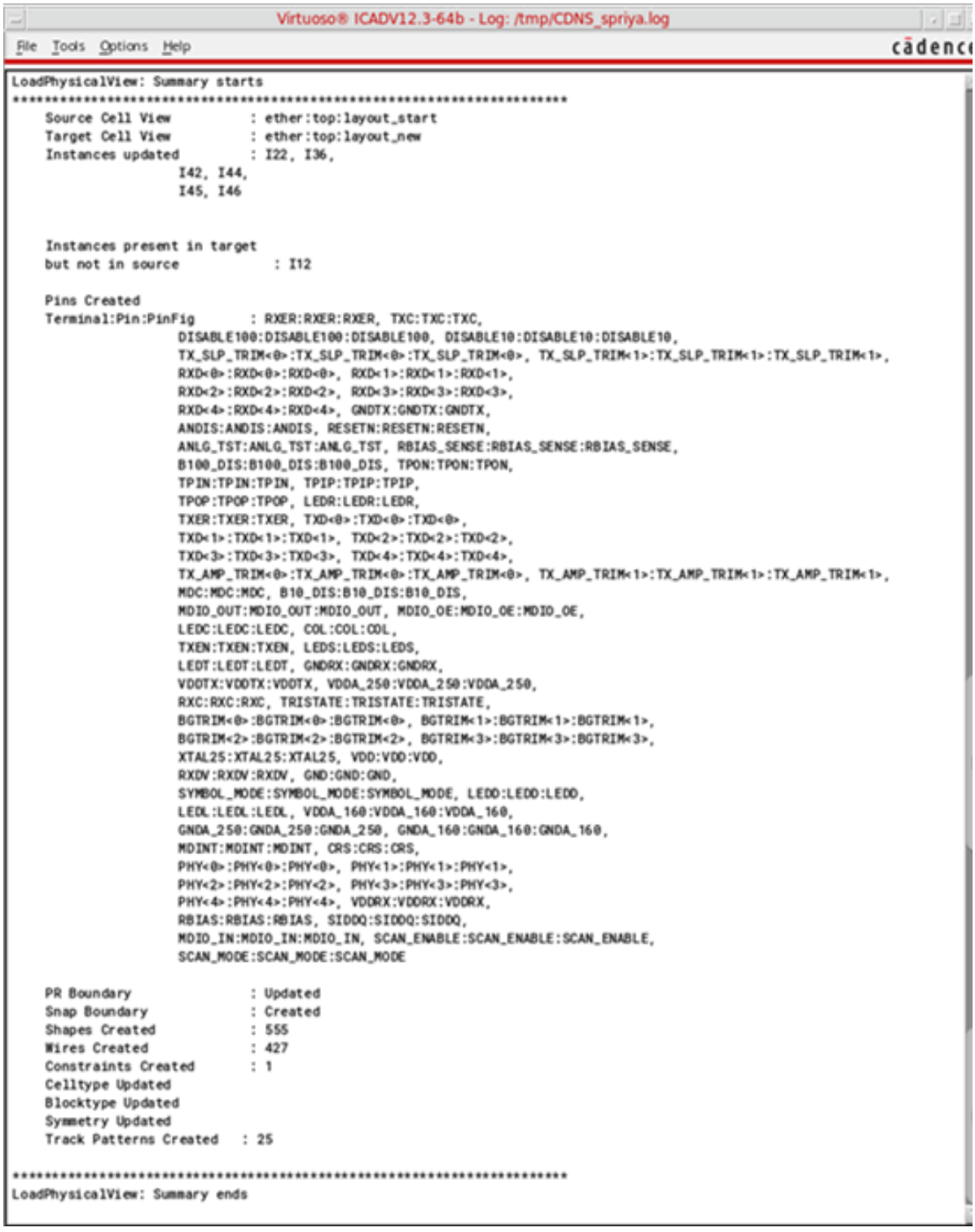
© 2023 Cadence Design Systems, Inc.

Load Physical View Summary Report

After running the *Load Physical View* command, a summary report is displayed in the CIW, which provides details about the information copied from the source cellview to the target cellview.

© 2023 Cadence Design Systems, Inc.

.




```
LoadPhysicalView: Summary starts
**********************************************************************
    Source Cell View          : ether:top:layout_start
    Target Cell View          : ether:top:layout_new
    Instances updated         : I22, I36,
                      I42, I44,
                      I45, I46


    Instances present in target
    but not in source            : I12

    Pins Created
    Terminal:Pin:PinFig       : RXER:RXER:RXER, TXC:TXC:TXC,
                      DISABLE100:DISABLE100:DISABLE100, DISABLE10:DISABLE10:DISABLE10,
                      TX_SLP_TRIM<0>:TX_SLP_TRIM<0>:TX_SLP_TRIM<0>, TX_SLP_TRIM<1>:TX_SLP_TRIM<1>:TX_SLP_TRIM<1>,
                      RXD<0>:RXD<0>:RXD<0>, RXD<1>:RXD<1>:RXD<1>,
                      RXD<2>:RXD<2>:RXD<2>, RXD<3>:RXD<3>:RXD<3>,
                      RXD<4>:RXD<4>:RXD<4>, GNDTX:GNDTX:GNDTX,
                      ANDIS:ANDIS:ANDIS, RESETN:RESETN:RESETN,
                      ANLG_TST:ANLG_TST:ANLG_TST, RBIAS_SENSE:RBIAS_SENSE:RBIAS_SENSE,
                      B100_DIS:B100_DIS:B100_DIS, TPON:TPON:TPON,
                      TPIN:TPIN:TPIN, TPIP:TPIP:TPIP,
                      TPOP:TPOP:TPOP, LEDR:LEDR:LEDR,
                      TXER:TXER:TXER, TXD<0>:TXD<0>:TXD<0>,
                      TXD<1>:TXD<1>:TXD<1>, TXD<2>:TXD<2>:TXD<2>,
                      TXD<3>:TXD<3>:TXD<3>, TXD<4>:TXD<4>:TXD<4>,
                      TX_AMP_TRIM<0>:TX_AMP_TRIM<0>:TX_AMP_TRIM<0>, TX_AMP_TRIM<1>:TX_AMP_TRIM<1>:TX_AMP_TRIM<1>,
                      MDC:MDC:MDC, B10_DIS:B10_DIS:B10_DIS,
                      MDIO_OUT:MDIO_OUT:MDIO_OUT, MDIO_OE:MDIO_OE:MDIO_OE,
                      LEDC:LEDC:LEDC, COL:COL:COL,
                      TXEN:TXEN:TXEN, LEDS:LEDS:LEDS,
                      LEDT:LEDT:LEDT, GNDRX:GNDRX:GNDRX,
                      VDDTX:VDDTX:VDDTX, VDDA_250:VDDA_250:VDDA_250,
                      RXC:RXC:RXC, TRISTATE:TRISTATE:TRISTATE,
                      BGTRIM<0>:BGTRIM<0>:BGTRIM<0>, BGTRIM<1>:BGTRIM<1>:BGTRIM<1>,
                      BGTRIM<2>:BGTRIM<2>:BGTRIM<2>, BGTRIM<3>:BGTRIM<3>:BGTRIM<3>,
                      XTAL25:XTAL25:XTAL25, VDD:VDD:VDD,
                      RXDV:RXDV:RXDV, GND:GND:GND,
                      SYMBOL_MODE:SYMBOL_MODE:SYMBOL_MODE, LEDD:LEDD:LEDD,
                      LEDL:LEDL:LEDL, VDDA_160:VDDA_160:VDDA_160,
                      GNDA_250:GNDA_250:GNDA_250, GNDA_160:GNDA_160:GNDA_160,
                      MDINT:MDINT:MDINT, CRS:CRS:CRS,
                      PHY<0>:PHY<0>:PHY<0>, PHY<1>:PHY<1>:PHY<1>,
                      PHY<2>:PHY<2>:PHY<2>, PHY<3>:PHY<3>:PHY<3>,
                      PHY<4>:PHY<4>:PHY<4>, VDDRX:VDDRX:VDDRX,
                      RBIAS:RBIAS:RBIAS, SIDDQ:SIDDQ:SIDDQ,
                      MDIO_IN:MDIO_IN:MDIO_IN, SCAN_ENABLE:SCAN_ENABLE:SCAN_ENABLE,
                      SCAN_MODE:SCAN_MODE:SCAN_MODE

    PR Boundary               : Updated
    Snap Boundary             : Created
    Shapes Created            : 555
    Wires Created             : 427
    Constraints Created       : 1
    Celltype Updated
    Blocktype Updated
    Symmetry Updated
    Track Patterns Created    : 25

**********************************************************************
LoadPhysicalView: Summary ends
```


© 2023 Cadence Design Systems, Inc.

This report includes the following information:

- Names of:
  - Source cellview
  - Target cellview
- Names of instances that were:
  - Updated in the target cellview.
  - Missing in the target cellview, but present in the source cellview.
  - Present in the target cellview, but missing in the source cellview.
- Names of physical-only instances that were:
  - Updated
  - Added
  - Present in the target cellview, but not in the source cellview
- Names of re-mastered Pcell instances
- Names of pins that were:
  - Updated
  - Created
- Whether the following attributes were created or updated:
  - PR boundary
  - Snap boundary
- Number of the following attributes created or updated:
  - Halos
  - Blockages
  - Rows
  - Row regions
  - Placement areas
  - Constraints
- Number of the following attributes created:

© 2023 Cadence Design Systems, Inc.

- Shapes
- Wires
- FigGroups
- Routes
- Count
- Site patterns
- Track patterns

- Whether the following attributes were updated:
  - Cell type
  - Block type
  - Symmetry
  - Width Spacing snap patterns

If the number of instances to be updated in the target cellview exceeds 100, an abridged report can be displayed.

Related Topics

The Load Physical View Command

Pseudoparallel Connections

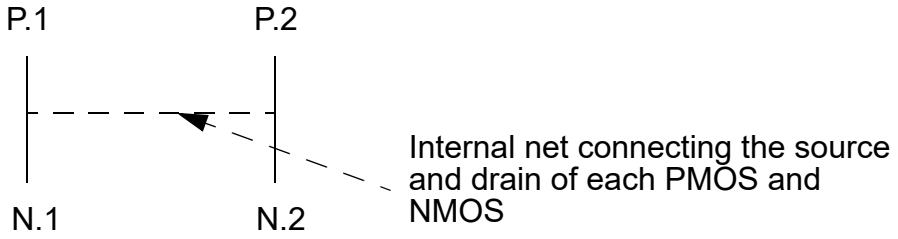
A pseudoparallel connection is defined as a group of instance terminals on a net that are at the same voltage, due to which they do not need a physical connection.

It is similar to a must-connect; the only difference being that it is implemented not using pins but using instance terminals that belong to a specified net. The main advantage of recognizing that a net can be made pseudoparallel is that it saves space because less routing is required to connect it.

© 2023 Cadence Design Systems, Inc.

What is a Pseudoparallel Connection

Consider a simple inverter comprising an NMOS and PMOS pair. If you set an mfactor of 2 on the inverter, when it is flattened in the layout, the connectivity is as represented below.



The broken line represents an internal net connecting the source and drain of each PMOS and NMOS. This net has four instance terminals; however, the router does not need to make a physical connection from the `P.1`/`N.1` pair to the `P.2`/`N.2` pair in order for the circuit to function correctly. Because it is mfactored, the nets connecting to all the other terminals of `P.1` and `P.2` are exactly the same, as are the nets connecting to all the other terminals of `N.1` and `N.2`. Therefore, the voltage on the node between `P.1` and `N.1` is the same as that on the node between `P.2` and `N.2`, removing the need to connect them up with a wire.

To represent this situation in the database, the instance terminals are partitioned into subnets. Each subnet on a pseudoparallel net contains two instance terminals which must be connected together, but no connection is necessary between the instance terminals on different subnets.

In the example above, there would be two subnets. The instance terminals from `P.1`/`N.1` would be contained in one of them, representing the fact that they must be connected in order for the circuit to work. The instance terminals from `P.2`/`N.2` would be in the other, again representing the fact that they must be connected in order for the circuit to work. But no connection is required between the two. The circuit would work equally well without the connecting wire, as shown below.



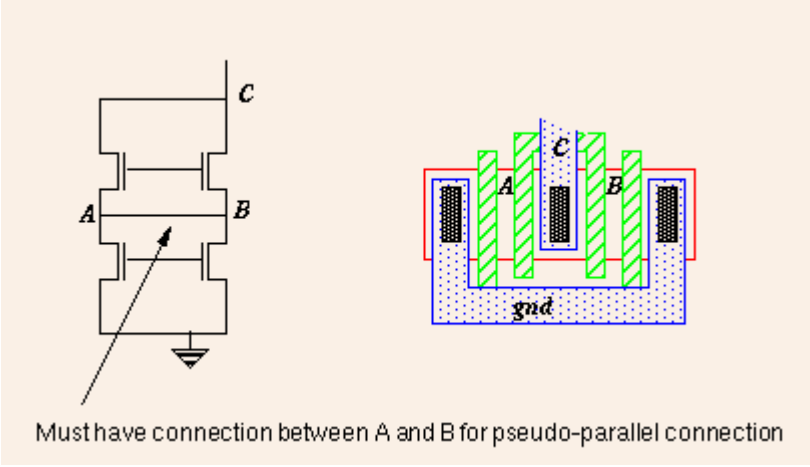
The model extends depending on the `mfactor` employed. In the example above, if the mfactor is `x`, then for the “internal” net connecting all the sources and drains together there are a total of `x` subnets each containing two instance terminals.

© 2023 Cadence Design Systems, Inc.

Layout XL does not consider the mfactor in the source when determining whether a net is pseudoparallel or not. Instead it considers the net, the instances attached to it, and all the other connections to those instances. In general, any time there is a pair of devices in series in the schematic, each with the same `mfactor` value, then the net implementing the series connection can be made pseudoparallel in the layout.

Examples of Pseudoparallel Connections

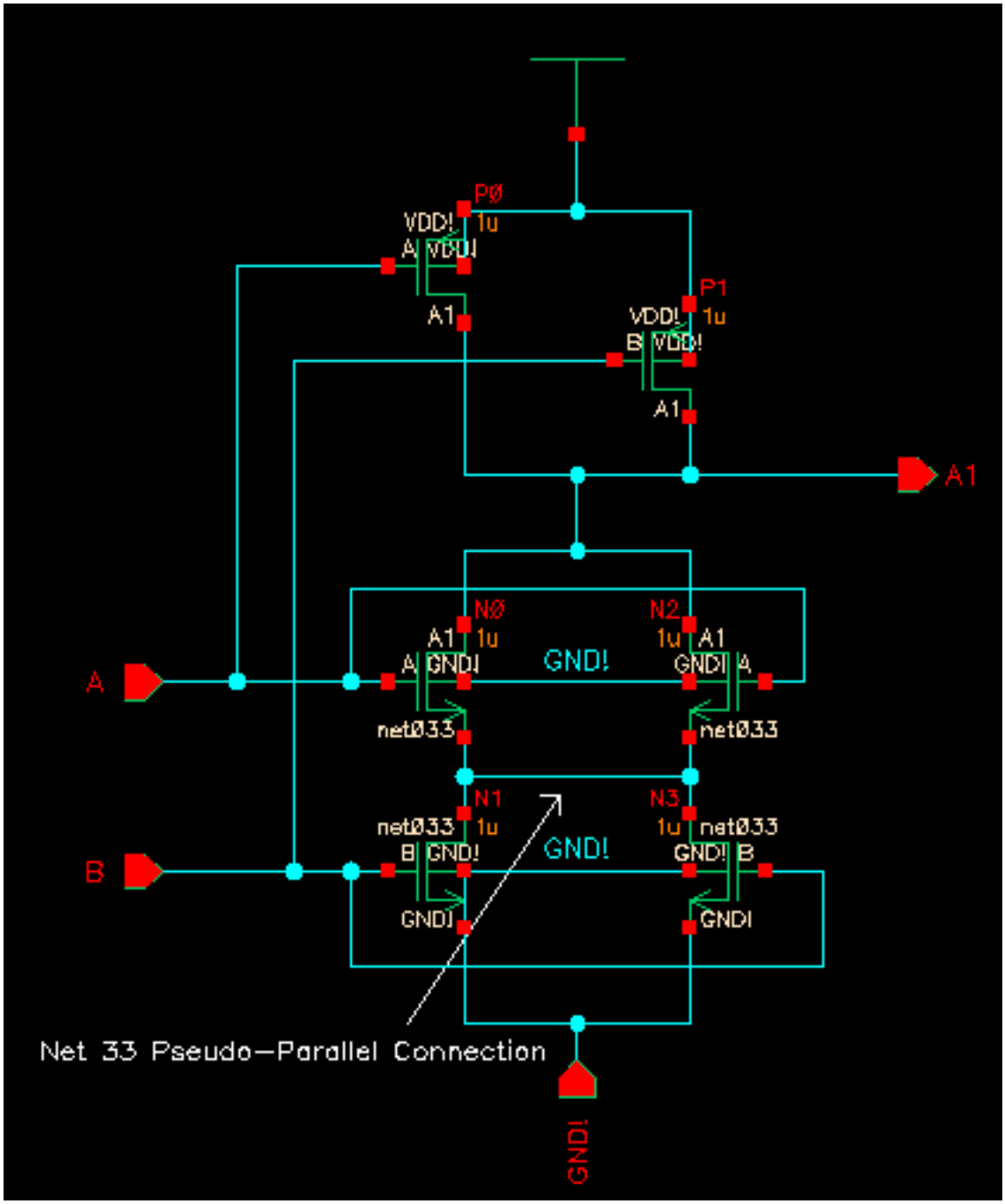
Pseudoparallel connections are useful in situations like the one shown below.



In this example, nodes A and B can be considered equipotential, i.e., there is no current flowing through them. You can save area in your design if this connection is skipped, even though the connectivity reference requires all instance terminals on the same net to be physically connected. With pseudoparallel connections on A and B, no physical connection is required between the two instance terminals even though they are on the same net.

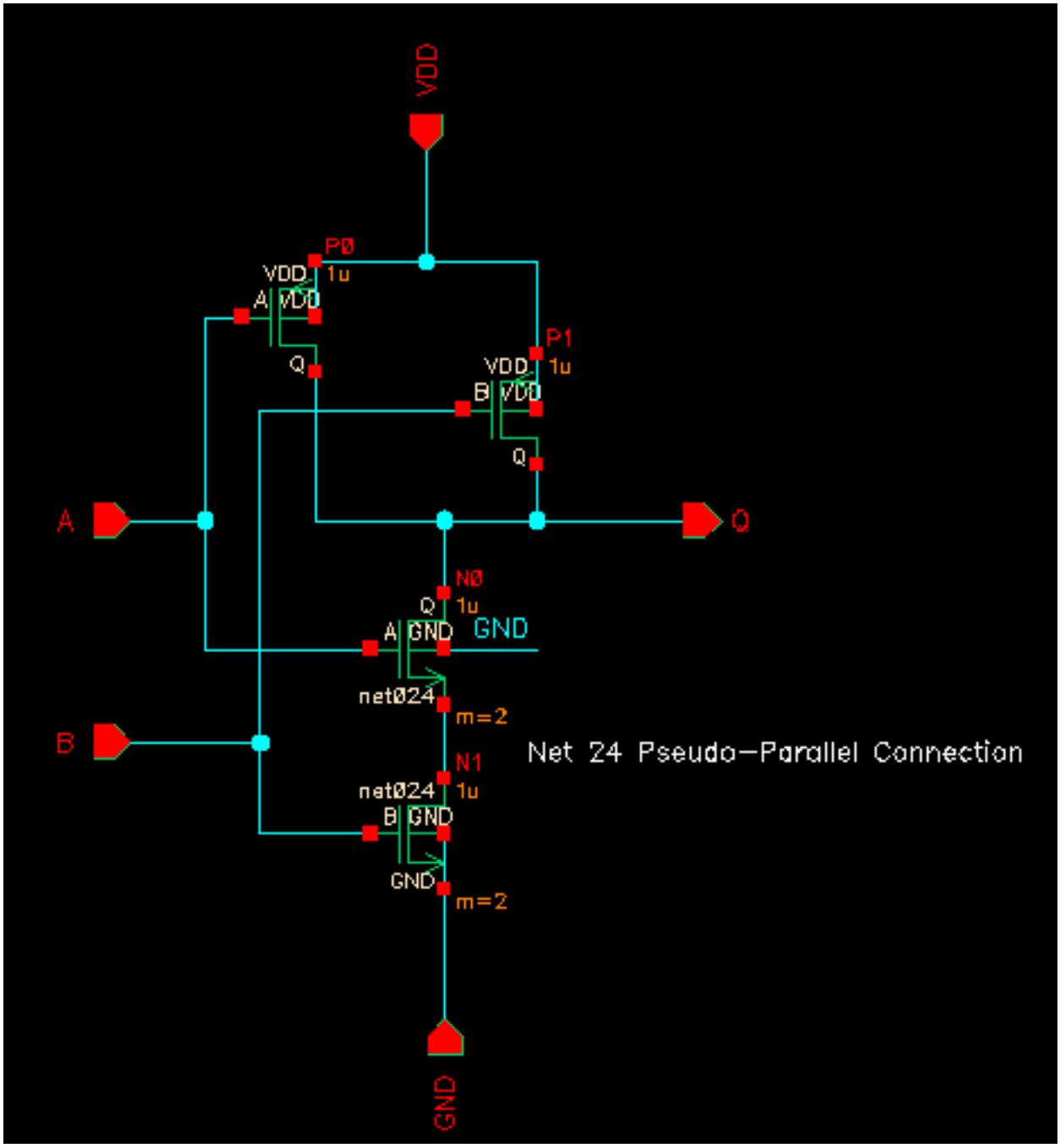
© 2023 Cadence Design Systems, Inc.

The picture below shows an explicit pseudoparallel connection of two or more groups of parallel devices in series.



© 2023 Cadence Design Systems, Inc.

The picture below shows an implicit pseudoparallel connection of two or more groups of mfactored devices in series.



Layout XL automatically finds possible pseudoparallel connections and treats them so when you generate the layout using the *Generate All From Source* command.

© 2023 Cadence Design Systems, Inc.

If the layout exists already, you can create pseudoparallel connections automatically by running the *Update Components And Nets* command.

Related Topics

Defining a Pseudoparallel Connection.

Defining a Pseudoparallel Connection

There are two ways in which you can define a pseudoparallel connection.

- Using the Pseudoparallel Connect Command
- Using the setPPConn Environment Variable

Using the Pseudoparallel Connect Command

To define a pseudoparallel connection for a set of instance terminals on a net:

1. From the layout window menu bar, choose *Connectivity – Pins – Pseudo Parallel Connect*.

   A message in the layout window status banner prompts you to click to select a net.

2. Click the net containing the instance terminals you want to connect.

   Flight lines appear between the pins on the specified net that can be pseudoparallel connected.

   If you were showing incomplete nets, those flight lines are removed automatically.

   A message in the layout window status banner prompts you to select an instance terminal.

3. Click one or more pins of the selected net to select the instance terminals you want to connect in a pseudoparallel connection.

   The pseudoparallel connection is defined and the flight lines updated to show the connection.

4. When you are done, press `Esc` to cancel the command.

Instead of selecting nets by clicking in the layout canvas, you can use the Define Pseudo Parallel Connected Net form.

© 2023 Cadence Design Systems, Inc.

To open the form:

1. Press `F3` while the *Pseudo Parallel Connect* command is active.

   The Define Pseudo Parallel Connected Net form is displayed. The form lists all the nets with no I/O pins.

   In the form, type the name of a net in the *Net* field or select the net names from the list.

   You can work on more than one net at a time by choosing multiple nets in the form or dragging your cursor in the design window to select an area when the command line prompts you to select a net.

   You must define pseudoparallel connectivity within a net. You cannot define pseudoparallel connectivity between nets.

2. When you are done, press `Esc` to cancel the command.

   You can change between the *Must Connect*, *Strongly Connected*, *Weakly Connected*, and *Pseudo Parallel Connect* commands using the right mouse button.

Using the setPPConn Environment Variable

Setting the `setPPConn` environment variable to `t` automatically identifies and defines nodes that qualify as pseudoparallel connections.

To use the `setPPConn` environment variable,

➤ Add the following line to the `.cdsenv` file.

```
layoutXL setPPConn boolean t
```

Related Topics

setPPConn

Pseudoparallel Connections

Resetting a Pseudoparallel Connection

To reset a pseudoparallel connection,

1. Delete one of the instances in the pseudoparallel connection.
2. Set the `setPPConn` environment variable to `nil`.

© 2023 Cadence Design Systems, Inc.

3. Run either *Update Components And Nets* or *Generate All From Source* to regenerate the deleted instance.

Related Topics

Pseudoparallel Connections

Generating a Layout without Chaining and Folding on Pseudoparallel Nets

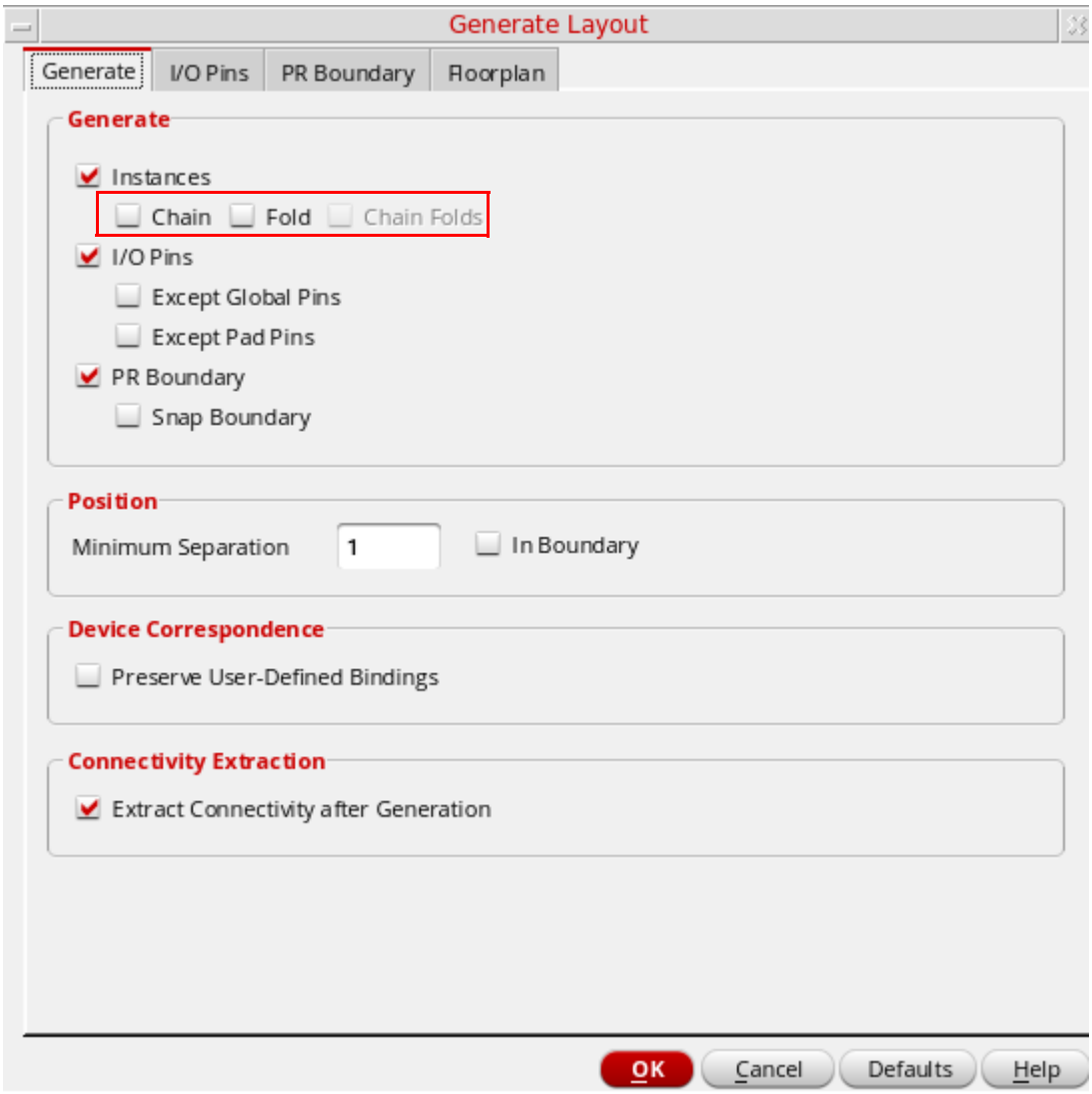
Pseudoparallel nets are shown as must-connects when the `setPPConn` environmental variable is set to `t` and the layout is generated without chaining and folding.

To do this,

1. From the layout window menu bar, choose *Connectivity – Generate – All From Source*.
2. On the Generate tab of the Generate Layout form, turn off the *Chain* and *Fold* options.

© 2023 Cadence Design Systems, Inc.

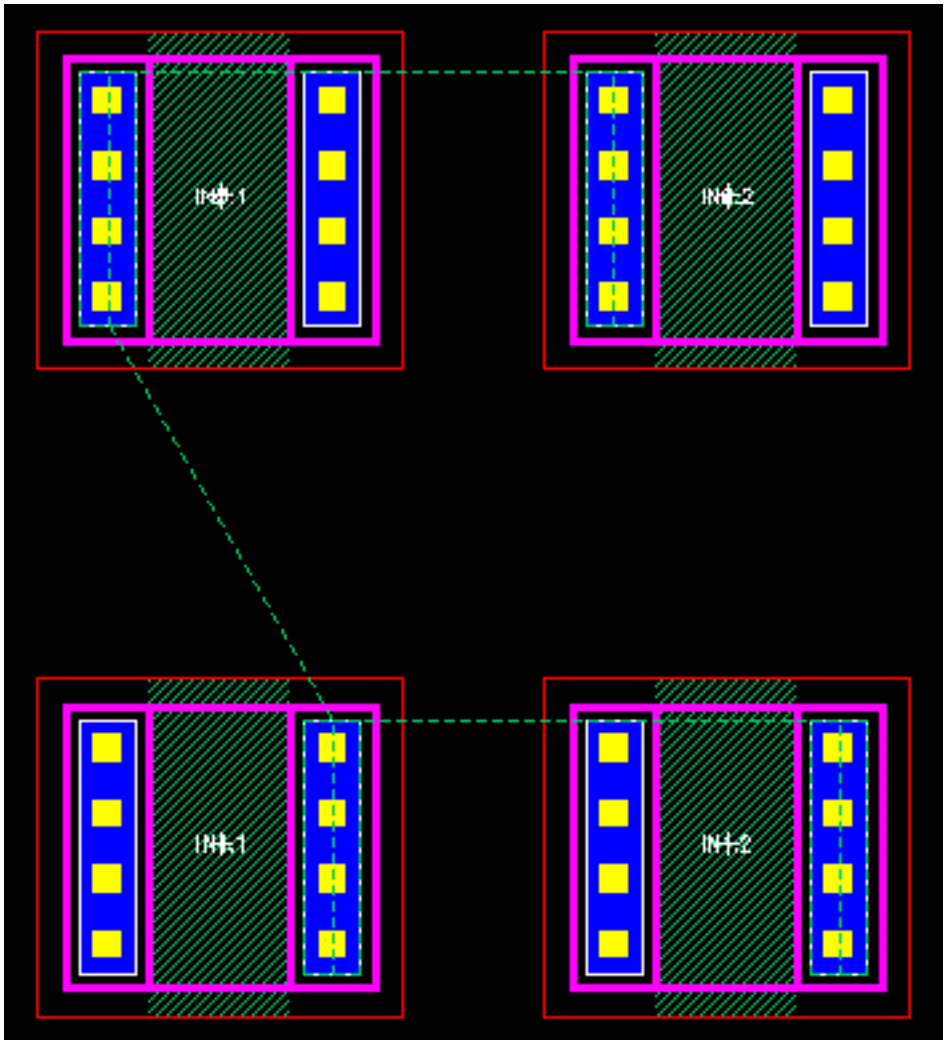
The *Chain Folds* option is automatically deselected when the *Fold* check box is deselected.



3. Set the rest of the options for your layout and click *OK*.

© 2023 Cadence Design Systems, Inc.

4. Open the Annotation Browser assistant, go to the *Connectivity* tab and select the pseudoparallel net.



Caution

Abutting instances with pseudoparallel nets is order-dependent. If you manually abut instances with pseudoparallel nets they will abut correctly. If you then abut instances that do not have pseudoparallel nets they will not abut and the pseudoparallel nets that were previously abutted will no longer be pseudoparallel nets.

Related Topics

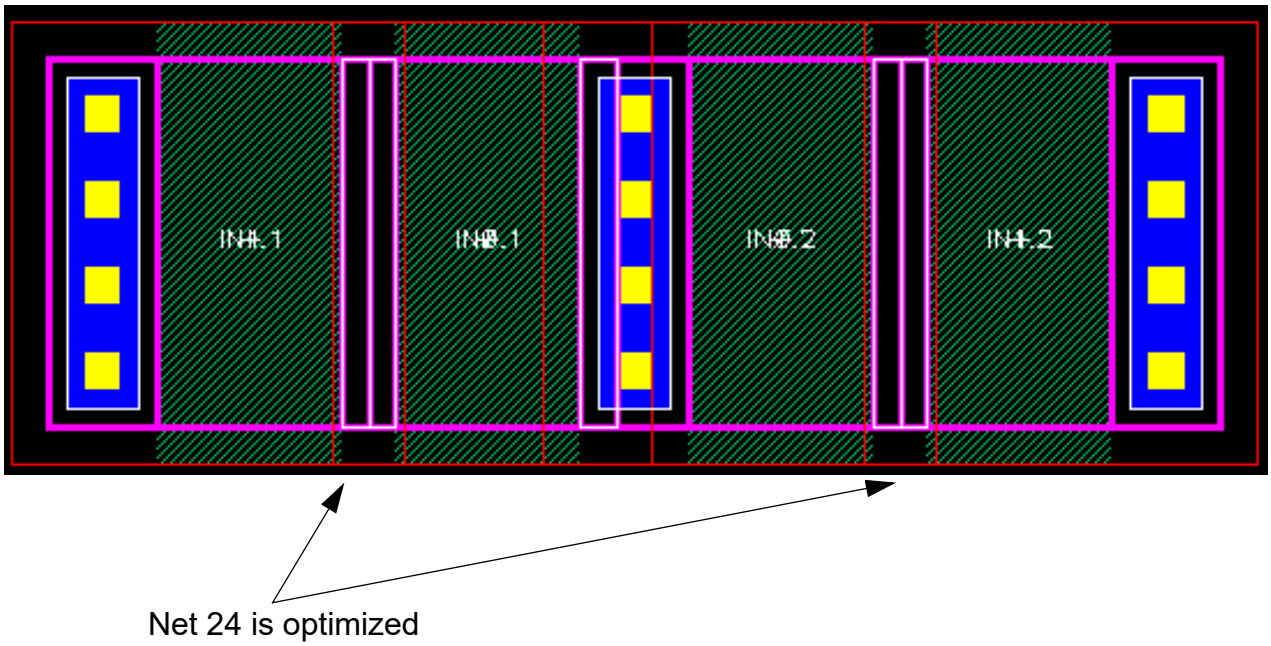
Generate Layout Form

© 2023 Cadence Design Systems, Inc.

Generating a Layout with Chaining and Folding on Pseudoparallel Nets

When the `lxAllowPseudoParallelNets` environment variable is set to `t`, layout generation optimizes pseudoparallel nets for implicit pseudoparallel connections.

This also works with the *Generate Selected From Source* command.



To do this:

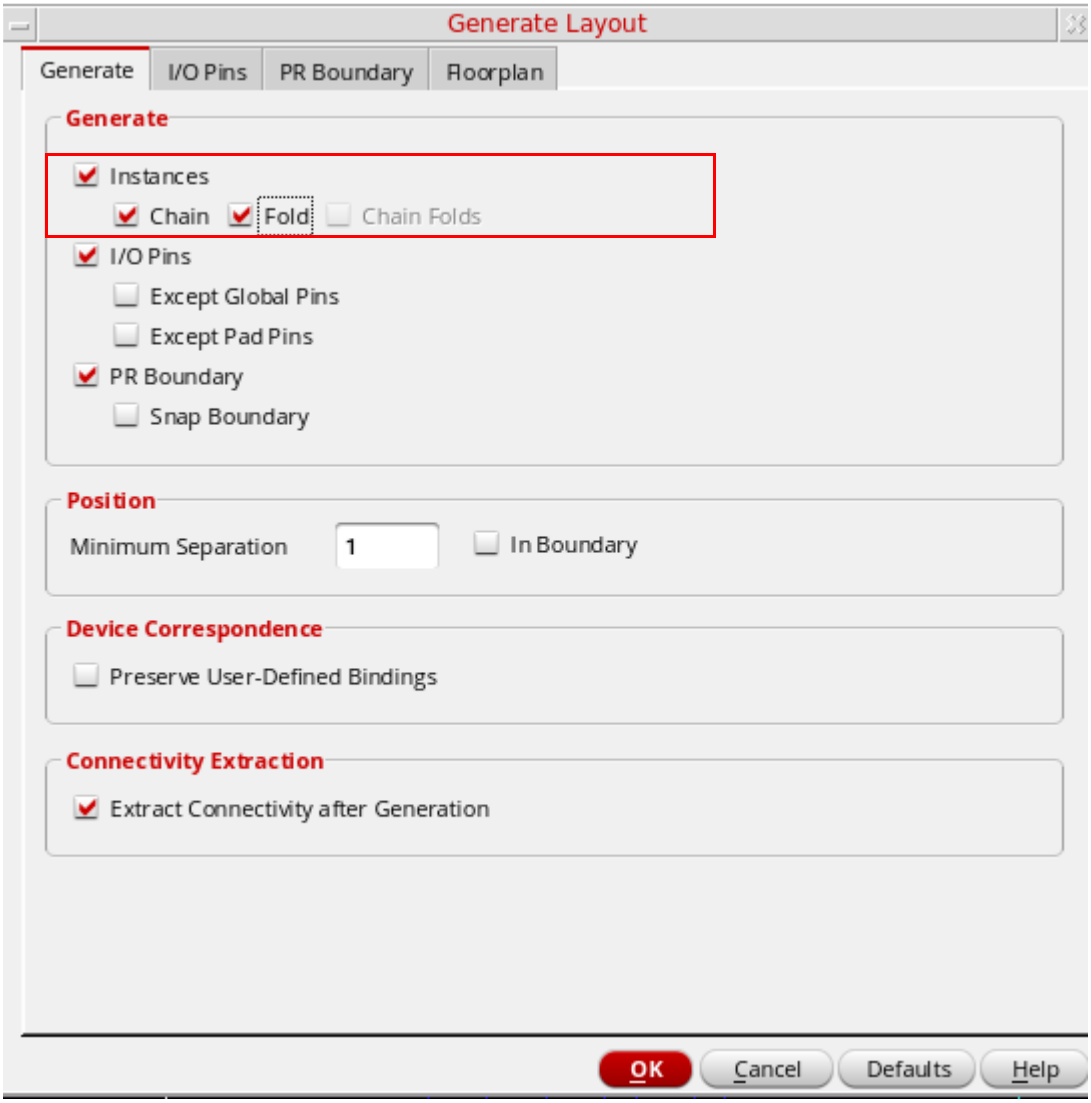
1. Add the following line to your `.cdsenv` file.

   ```
   layoutXL lxAllowPseudoParallelNets boolean t
   ```

2. From the layout window menu bar, choose *Connectivity – Generate – All From Source*.

© 2023 Cadence Design Systems, Inc.

3. On the Generate tab of the Generate Layout form, select the *Chain* and *Fold* options.



4. Set the rest of the options for your layout and click *OK*.
5. To verify that the pseudoparallel nets are complete, open the Annotation Browser assistant and go to the *Connectivity* tab.

   The pseudoparallel nets in question are no longer listed in the *Incomplete Nets* category in the browser.

Related Topics

Generate Layout Form

© 2023 Cadence Design Systems, Inc.

Selected Components with Pseudoparallel Nets Generation

You can set pseudoparallel nets when using the *Generate Selected From Source* command if

- The *Group As In Schematic* option is turned on.
- The nets are internal to the instances created in the temporary cellview.
- The `setPPConn` environment variable is set to `t`.
- All the rules for pseudoparallel nets are met.

Related Topics

setPPConn

Pseudoparallel Connections

© 2023 Cadence Design Systems, Inc.

6

Layout Editing

This topic summarizes how to edit the placement of components in the layout and how to add additional components.

- Constraint-Aware Editing - This mode ensures that the Layout XL manual editing commands respect physical constraints that may be set on your design components.
- Moving Objects - Different ways to move an object to another location in the current or another cellview.
- Adding an Instance in Layout - Different ways to the layout an instance that is not present in the schematic.
- Adding a Pin - Steps to add pins to layout.
- Assigning Instance Pins to a Net - Steps to connect a new instance to an existing net in a design that has a connectivity source.
- Transistor Folding - Interactively divide the different types of transistors into two or more layout instances with terminals all connected in parallel to the same nets.
- Pin Permutation - Exchange the connectivity or net connections of the pins of a component.
- Incomplete Nets - Show incomplete electrical connections between the component terminals, pins, and shapes associated with each net in your design.
- Editing In Place - Use the Edit In Place command to edit a placed instance in your design and maintain the connectivity.
- Soft Block Editing - Modify the soft blocks in your design.
- Smart Display - A layout editing environment that can be used to generate a symbolic representation of the layout by using one of the two modes, *Abstracted* or *Schematic*.
- Net Name Display - Add net name labels in the layout canvas for shapes that have associated labels. Having the net name labels display in the layout canvas makes layout editing simpler, especially for large designs.

© 2023 Cadence Design Systems, Inc.

- Composite Waveguides (Virtuoso Photonics Option) - If you are using the Virtuoso Photonics Solution, Layout EXL or higher enables you to use optical connections called waveguides in your design. When several such optical connections or waveguides are used in a design, the resultant waveguide is called a composite waveguide.

Related Topics

Using Connectivity

Photonics in Virtuoso Schematic Editor, ADE, and Layout XL (Virtuoso Photonics Option)

© 2023 Cadence Design Systems, Inc.

Constraint-Aware Editing

Constraint-aware editing mode ensures that the Layout XL manual editing commands respect physical constraints that may be set on your design components.

Constraint-aware editing also checks constraints and displays violation markers at the appropriate location in the design window, if necessary. These checks are not run automatically but need to be explicitly initiated.

For example, if you run a **Check Constraints** batch check command or edit an object involved in a constraint, then Constraint-Aware Editing checks will be run on the appropriate objects in the layout. However, if you toggle the Constraint-Aware Editing mode, the constraint checks will not be automatically run nor will any violation markers be created or deleted.

As design edits are made, Constraint-Aware Editing continuously checks and readjusts the markers associated with the editing action. Constraint-Aware Editing mode is ON by default.

This feature allows layouts to be generated interactively while constraints are respected and the overall topology of the design is maintained, ensuring the designer's original intent is honored throughout the development of the physical implementation.

Constraint-Aware Editing supports the following design object constraint members:

- Instances, pins, and groups
- Wires and vias, but only for the `locked` constraint
- Place and route boundary, but only for pin placement constraints when using the *Stretch* command

The following interactive commands are supported by Constraint-Aware Editing:

- *Edit – Copy*
- *Edit – Move*
- *Edit – Stretch*
- *Edit – Rotate*
- *Edit – Quick Align* and *Edit – Advanced – Align*

© 2023 Cadence Design Systems, Inc.

These commands honor the following constraints:

Alignment	Defines how the constrained objects are aligned, either relative to each other or to some other reference object.
Fixed	Prevents the constrained objects from being edited by an automatic tool, but allows them to be edited manually.
Locked	Prevents the constrained objects from being edited in any way.
Matched Parameters	Defines that the specified parameters for the constrained objects must have values in ratios specified by the ratio parameter. An empty Match Subset implies that all device parameters must be in the correct ratio. If a layout XL environment variable named `matchedToIgnore` exists then the parameter names specified by that space-separated string will not be checked.
Orientation	Defines allowed orientations for the constrained objects.
Matched Orientation	Defines that the constrained objects must have the same orientation.
Symmetry	Defines symmetries for the constrained objects about some axis.
Distance	Defines a range of allowed distances between constrained objects with respect to a specified reference point (i.e. origin, left side, right side).
Cluster	Prevents non-cluster objects from encroaching cluster space of constrained objects.
Cluster Boundary	Defines a space for a cluster. Depending upon whether Cluster Boundary is set as exclusive or not and whether flexibleFlag parameter is true or not the constraint will prevent the movement of the members outside the cluster boundary or will adjust the cluster boundary as per the movement. For the interactive CAE move to happen, the flexibleFlag is ignored.

© 2023 Cadence Design Systems, Inc.

Constraint-Aware Editing Considerations

- The constraint checker cannot recognize certain types of user-defined Pcell guard ring instances and as a result reports these as violating cluster constraints. You can specify that these Pcell guard ring instances are ignored by the checker by setting the `caeIgnoreInCluster` property either on a specific guard ring instance or its supermaster (to ignore all instances of that supermaster).

  CAE is able to recognize and ignore regular fluid guard rings based on their attributes. The `caeIgnoreInCluster` property is required only for legacy guard rings implemented using custom Pcells.

- If you use design-rule-driven editing in the Enforce mode in conjunction with constraint-aware editing, design rules are enforced not only for the object you are editing, but for all its constraint partners as well.

- Constraint-aware editing does not edit or ghost constraint partners in pseudo-hierarchy levels. For example, if you have a symmetry constraint with one member inside a group and another member outside the group, then an edit to one member will not cause an additional edit to the other, even if the constraint becomes violated. In this situation, the following warning message is displayed:

  "*Device will not be edited because it is on a different hierarchy level than its constraint partner. Ungroup this device to prevent constraints from being broken*."

- To facilitate editing of members in pseudo-hierarchy levels, constraint-aware editing supports promoting the Symmetry and Matched Orientation constraints from the pseudo-hierarchy members to their pseudo-hierarchy containers.

  However, for the constraints to be promoted, each constraint should have instances as members, and each instance member should belong to a group. After a constraint is promoted from its member to the member's container, the containers behave in a constraint-aware manner, while the constraint itself is not modified.

- If all the constraint members belong to a single figGroup, constraint-aware editing enforces these constraints by supporting the transparent group mode. However, if the constraint members belong to different figGroups, the transparent group mode cannot be supported and none of the constraints are enforced.

- When rotating a quad-symmetric device configuration using the axes of symmetry, you must select both the axes—vertical and horizontal—and then rotate them. If you rotate only one of the symmetry axes, the quad symmetry of the device is destroyed, resulting in constraint violations.

- Constraint-Aware Editing supports constraints at soft-block level. If you apply constraints, such as Alignment, Symmetry, and Distance, to the pins in a softblock and

© 2023 Cadence Design Systems, Inc.

then edit those pins at the level above the soft-block level, the pins will be edited in accordance with the constraints applied at the soft-block level.

Related Topics

Default Constraint Types

caeIgnoreInCluster Property

Design Rule Driven Editing

Editing Group Contents by Using Transparent Group Mode

Enabling and Disabling Constraint-Aware Editing Mode

Constraint-aware editing is a session-wide setting. It applies to all existing and new windows opened in the current Virtuoso session. To toggle constraint-aware editing, do one of the following.

1. With your cursor in the layout window, press `F10`.
2. Press the  *CAE* button on the *Options* toolbar.
3. Click the right mouse button and choose *Constraint-aware Editing* while using one of the supported commands: *Create Wire*, *Create Point to Point*, *Copy*, *Move*, *Stretch*, and *Rotate*.
4. Change the setting on the *General* of the Connectivity form:
   a. From the layout window menu bar, choose *Options – Connectivity*.
   b. On the *General* tab, check or uncheck the box labeled *Constraint-aware editing*.
5. Click *OK* to accept the changes.

Related Topics

Connectivity Form

Bindkeys in Virtuoso Layout Suite

© 2023 Cadence Design Systems, Inc.

Moving Objects

There are three ways to move devices, pins, or shapes in the layout canvas.

- Use the *Move* command to manually drag the object to its new location.
- Use the Move options form to specify more precisely the conditions of the move operation.
- Use the *Place As In Schematic* command to move all the generated components inside the design boundary. If your design carries any unbound instances or pins, these will be placed below the design boundary.

Related Topics

Moving Objects Manually

Objects Moving

Moving Generated Components inside the PR Boundary (Placing As In Schematic)

Changing the Appearance of Draglines

Changing the Appearance of Draglines

When using the commands listed below, Layout XL displays draglines which connect the pins on the object to the nearest pins on another object on the net. When you move the object closer to a different object on the same net, the lines reconnect to the new object.

- *Generate Selected From Source*
- *Generate Clones*
- *Move*
- *Stretch*

If you need to know more about the potential connections for the instance you are working on, you can change both the number and the appearance of the draglines using the options under *Draglines* in the *General* tab of the Connectivity form.

You can display draglines for all the connections to the instance you are working on, hide the draglines for global nets and for power and ground nets, and have each dragline shown in a different color.

© 2023 Cadence Design Systems, Inc.

Displaying many draglines during interactive commands can impair the performance of Layout XL. To mitigate these effects, use the `maxDragFig` environment variable to limit the number of figures that can be in a dragset.

- To set `maxDragFig` interactively so that it takes effect in the current session, type the following in the CIW, where *`win`* is the window to which the limit applies.

  ```
  win~>maxDragFig = 1000
  ```

- To set a default value for `maxDragFig`, which will be used in all future Layout XL sessions, put the following line in your `.cdsenv` file.

  ```
  graphic maxDragFig int 1000 nil
  ```

Related Topics

Connectivity Form

Swapping Components

The *Swap Components* command lets you swap the positions of two components in the layout window.

Layout XL does not move any connections along with the components. If swapping a pair of components causes a short, markers show you where the short is. If swapping components causes an open connection, flight lines show the open connection.

The *Swap Components* command is not constraint aware.

To swap components:

1. From the layout window menu bar, choose *Edit – Advanced – Swap Components*.

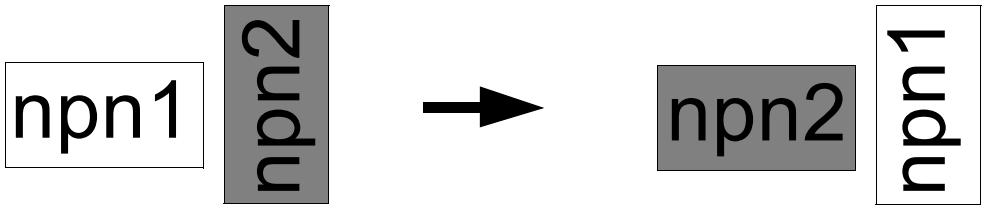
   The Swap Components form is displayed below.



© 2023 Cadence Design Systems, Inc.

2. Click the first component in the layout canvas.
3. Click the second component in the layout canvas.

   Layout XL switches the locations of the components you selected, but retains the orientation of the component originally associated with the position.



To undo the swap:

➤ Choose *Edit – Undo* from the layout window menu bar.

Related Topics

highlightPotentialSwap

highlightPotentialSwapColor

Swap Components Form

Locking Devices

The Layout XL device locking capability sets a temporary locked constraint on a device in the layout. A locked device cannot be moved during interactive or automatic placement.

You can view the locked constraint for the selected devices in the *Placement Status* field of the Property Editor and Constraint Manager assistants.

Locking a Device Using the Property Editor

To lock a device using the Property Editor assistant,

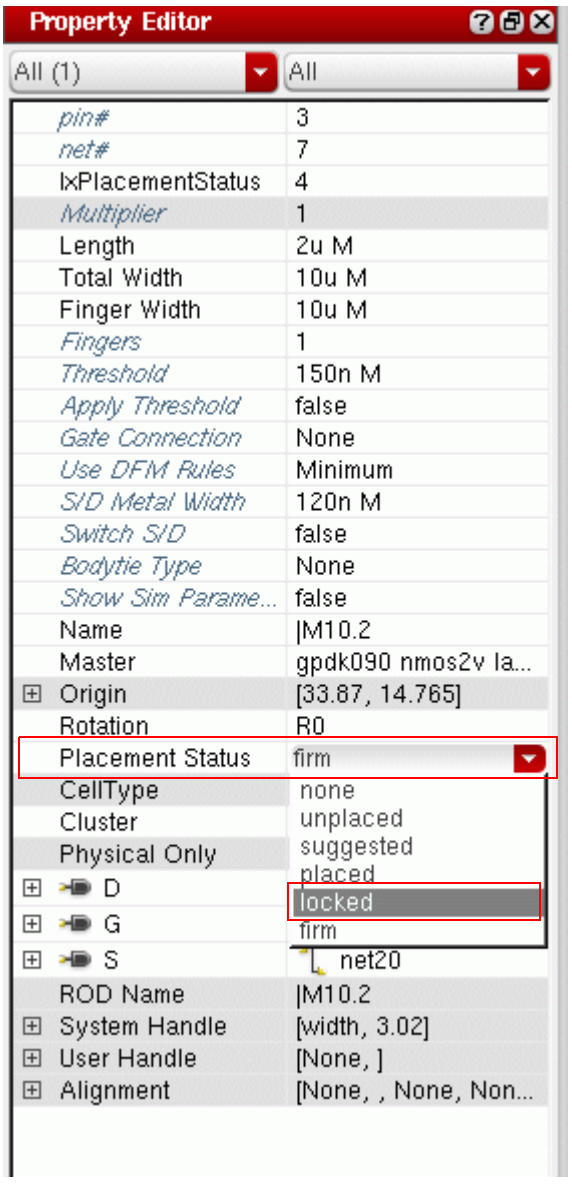
1. From the layout window menu bar, choose *Window – Assistants – Property Editor*.

   The Property Editor assistant is displayed.
2. In the canvas, select the device you want to lock.

© 2023 Cadence Design Systems, Inc.

The property information for the selected device is displayed in the Property Editor assistant.

3. In the Property Editor assistant, select the *Placement Status* field and choose *locked* from the drop-down list in the right hand column.



A locked constraint is placed on the selected device. The Constraint Manager assistant is updated accordingly.

© 2023 Cadence Design Systems, Inc.

Locking a Device Using the Constraint Manager

To lock a device using the Constraint Manager assistant,

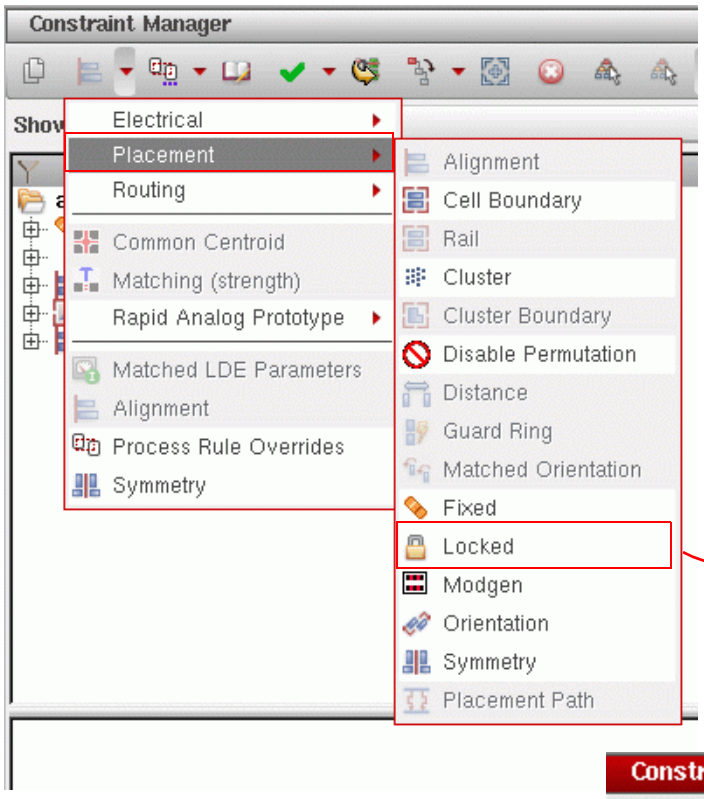
1. From the layout window menu bar, choose *Window – Assistants – Constraint Manager*.

   The Constraint Manager assistant is displayed.

2. In the canvas, select the device you want to lock.

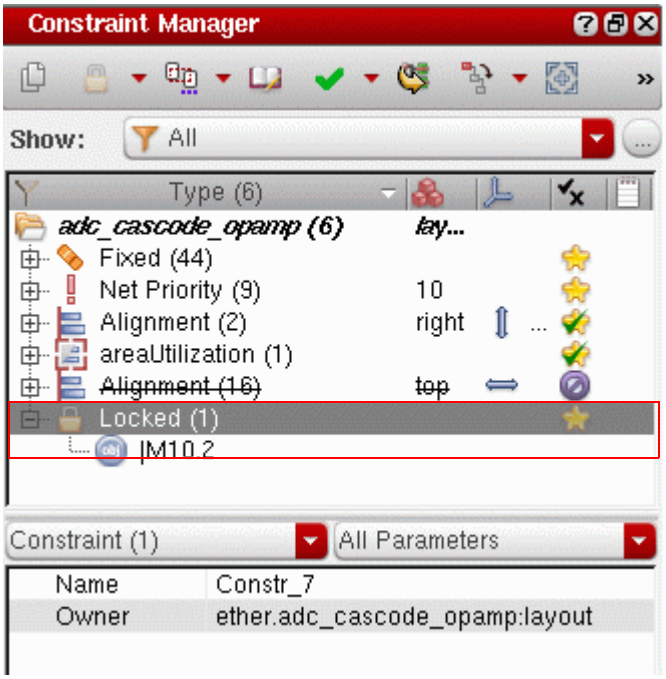
© 2023 Cadence Design Systems, Inc.

3. Choose *Placement – Locked* from the constraint pull-down in the Constraint Manager assistant toolbar.



Locked constraint selected

Locked constraint applied



A locked constraint is placed on the selected device and the Constraint Manager assistant is updated accordingly.

© 2023 Cadence Design Systems, Inc.

Related Topics

Constraint Manager Assistant

Property Editor Assistant

Locking a Device Using the Property Editor

Locking a Device Using the Constraint Manager

Unlocking a Device

Unlocking a Device

To remove a lock on a device:

1. From the layout window menu bar, choose *Window – Assistants – Constraint Manager*.

   The Constraint Manager assistant is displayed.

2. Choose the constraint you want to delete and click the *Delete* icon in the Constraint Manager toolbar.

   The constraint is removed from the device.

You can also remove the locked constraint by changing the *Placement Status* in the Property Editor assistant.

To lock a device using the Property Editor assistant:

1. From the layout window menu bar, choose *Window – Assistants – Property Editor*.

   The Property Editor assistant is displayed.

2. In the canvas, select the device you want to unlock.

   The property information for the selected device is displayed in the Property Editor assistant.

3. In the Property Editor assistant, select the *Placement Status* field and change the value from *locked* to *none*.

   The locked constraint is removed from the selected device. The Constraint Manager assistant is updated accordingly.

© 2023 Cadence Design Systems, Inc.

Related Topics

Constraint Manager Assistant

Property Editor Assistant

Adding an Instance in Layout

There are two ways to add to the layout an instance that is not present in the schematic.

- Add the instance to the layout using the *Create Instance* command.

  The incremental binder engine creates the appropriate bindings automatically, or you can use the *Define Device Correspondence* command to create user-defined bindings as before.

  When adding physical-only instances, add the `ignore` property either at creation time or afterwards using the Property Editor assistant.

- Add the instance to the schematic, connect it to a net, and then use *Update Components And Nets* to generate the new instance in the layout view. You need write permission for the schematic to do this.

To add an instance to the layout using *Update Components And Nets*:

1. In the schematic window, use the Add Instance form to add an instance to the schematic.

   See Adding Instances in Schematic

2. To add the new instance to the layout view, choose *Connectivity – Update – Components And Nets* from the layout window menu bar.

3. Click *Yes* to re-extract the schematic.

   Layout XL extracts the schematic and adds the new instance to the layout.

   The incremental binder engine automatically determines the binding for the newly added instance. To change the binding after creation, use the Define Device Correspondence form.

If the instance does not need to be connected to a net, add it directly to the layout and add the `ignore` property to it.

Related Topics

Add Instance form

© 2023 Cadence Design Systems, Inc.

ignore Property

Adding Instances in Schematic

Define Device Correspondence Form

Adding a Pin

Pins that occur in both the schematic and the layout must have identical names. If you add pins that are not in the schematic to the layout, Layout XL maintains their connectivity in the layout.

To add a pin to the layout,

1. In the Palette Assistant, choose the layer on which you want to create the pin.
2. From the layout window menu bar, choose *Create – Pin*.

   The Create Pin form displays.
3. In the *Terminal Names* field, type the terminal name for the pin.

   The pin name must match the name of an existing net in the schematic. You can create more than one pin with the same name.

   To create a pin for a net that is not on the schematic (for example, a feedthrough), give the pin a new name.
4. In the *I/O Type* section, choose the appropriate I/O type.
5. In the layout canvas, click to place one corner of the pin; then click to place the opposite corner of the pin.

   The pin appears in the layout canvas. If the Annotation Browser assistant is showing incomplete nets, flight lines are drawn to show the pin connected to the net.

Do not place pins where you do not want to make a connection. For example, do not create a pin on polysilicon that covers the gate area of a FET.

You can also add pins defined with shapes. You can use any layer-purpose pair for pin shapes. Pins made as large as possible make routing easier.

Related Topics

Creating Pins

© 2023 Cadence Design Systems, Inc.

Create Pin Form

Assigning Instance Pins to a Net

If you place an instance in a layout that does not have a connectivity source, the software does not connect the instances to any net unless you specifically assign the instance pins to a net using the *Assign Nets* command.

You can also use *Assign Nets* to connect a new instance to an existing net in a design that has a connectivity source.

To assign the pins of an instance to a net:

1. Place new instances in the layout using the *Create – Instance* command.
2. Use the Annotation Browser assistant to display the incomplete nets in the canvas.
3. Choose *Connectivity – Nets – Assign*.

   The layout window and the CIW prompt you to select a pin.
4. Click a pin in one of the new instances you have placed.

   You can select multiple pins to be attached to the same net.

   The pin you select is highlighted. The layout window and the CIW prompt you to select a net.
5. Click a pin that is already connected to the net you want to add the new pin to.

   The flight lines representing that net are extended to show that the new pin has been added to the net. The CIW lists the name of the pin and the name of the net it was connected to.

   The layout window and the CIW prompt you to select another pin to add to a net.
6. Press `F3` to display a list of all the nets in the layout.

© 2023 Cadence Design Systems, Inc.

The Assign Nets form is displayed.



7. To add a pin you selected in the layout to a net listed in this form, click the name of the net in the form and then click *Apply*.

   The flight lines representing that net are extended to show that the pin is added to the net.

8. Press `Esc` to exit the command.

Related Topics

Assign Nets Form

Incomplete Nets

Viewing Extended Selection for Instances, Pins, and Nets in Navigator

For an instance that you have selected, you can choose to extend the selection to any associated pins, nets, or instances, such as folded, mfactored, sfactored, `lxCombination` devices, vector pins or bus pins, bundled nets or bus bundles.

If the selected instance is an ungenerated instance, the selection can extend to any associated ungenerated or generated instances. Likewise, if an ungenerated instance is associated with a selected instance, the ungenerated instance is added to the selected set.

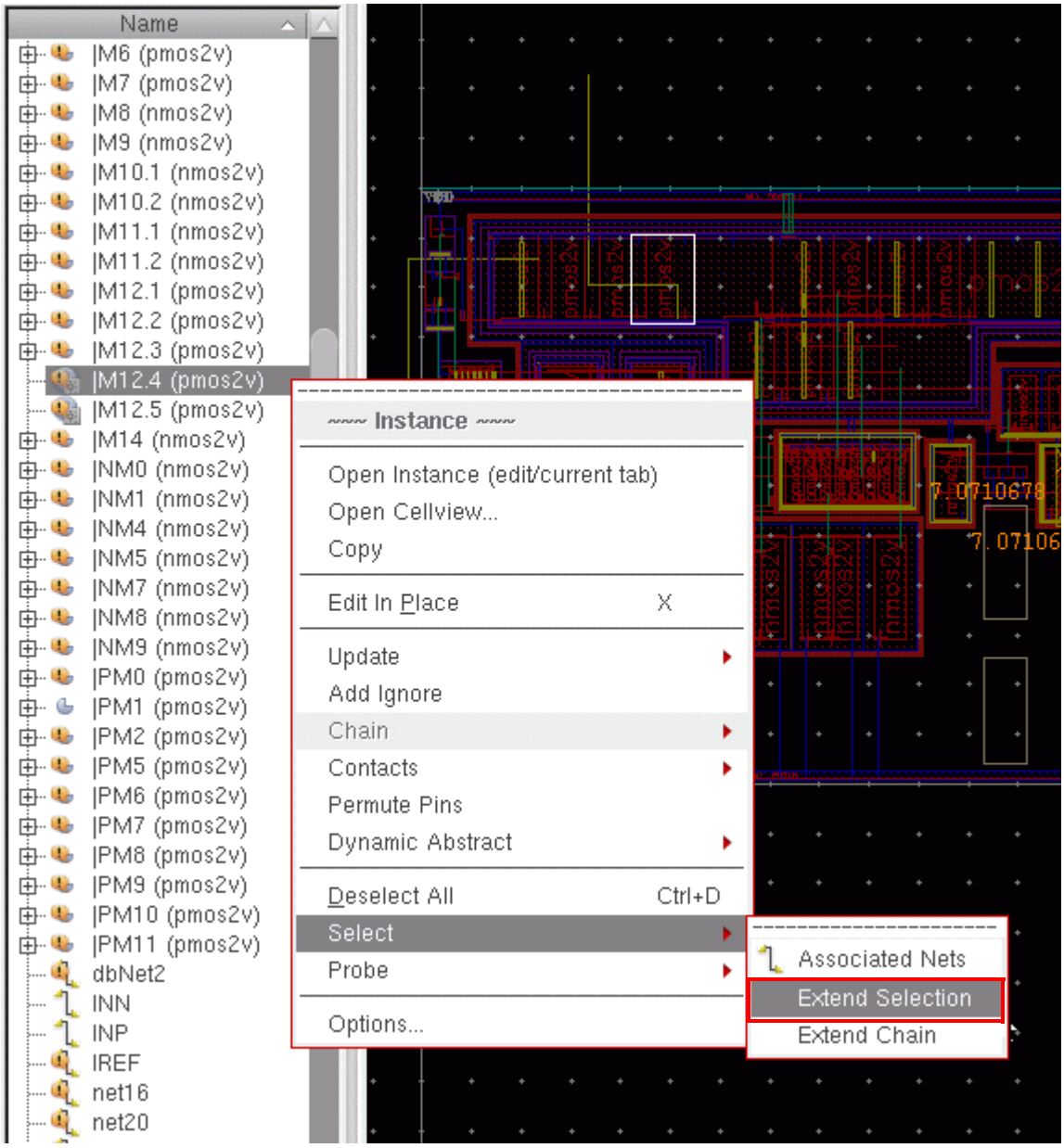
The entire selected set is highlighted in the Navigator assistant and cross-selected in the layout canvas.

© 2023 Cadence Design Systems, Inc.

To extend the instance selection to display the entire associated set:

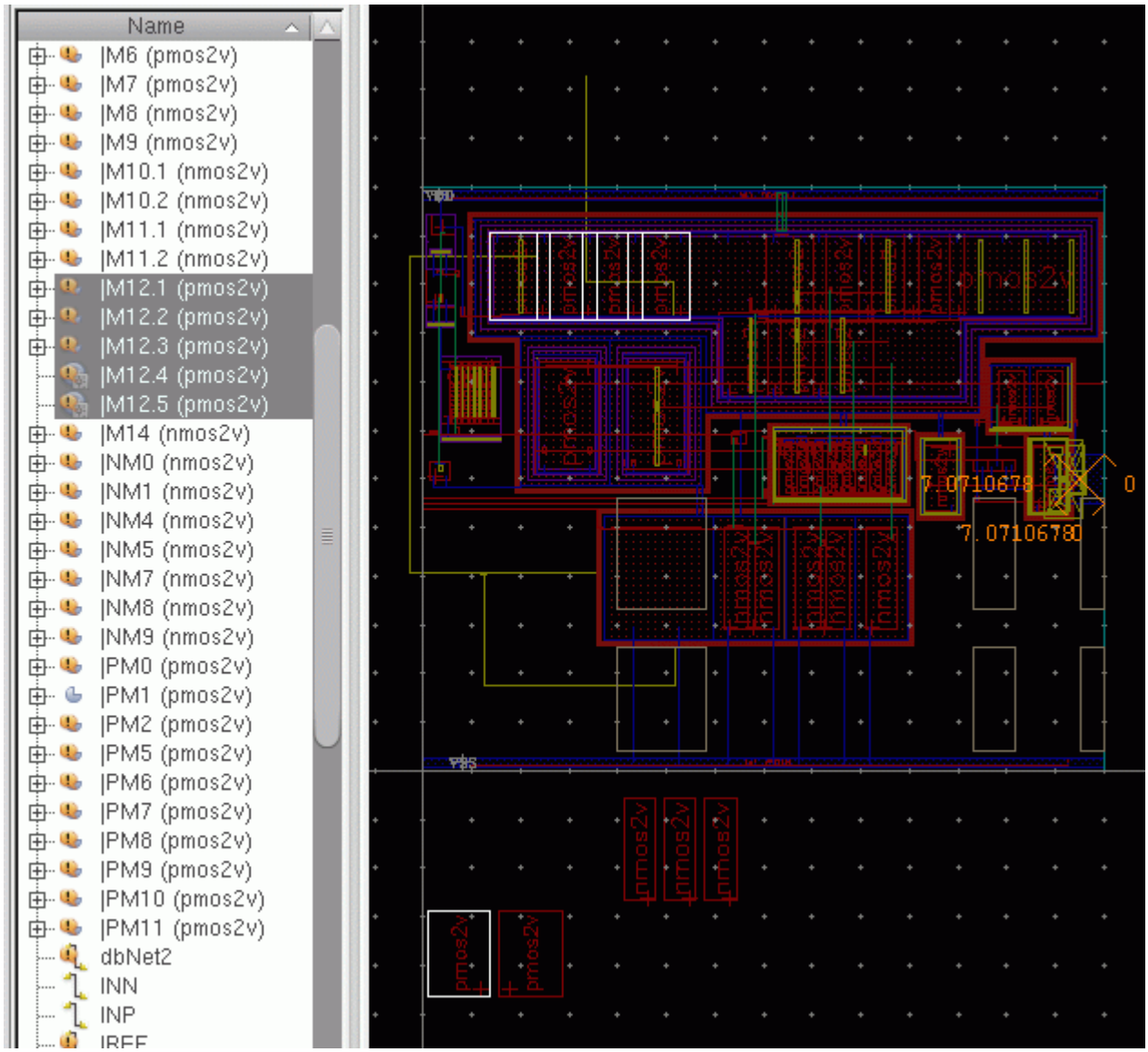
➡ Right-click an instance in the Navigator assistant (or in the layout canvas) and choose *Select – Extend Selection*, as shown in the figure below.

Alternatively, select an instance in the Navigator assistant (or in the layout canvas) and call the `lxSelectedExtendSelection` SKILL function.



© 2023 Cadence Design Systems, Inc.

All the instances, pins, or nets that are associated with the selected instance appear selected in the Navigator assistant as well as in the layout canvas, as shown below.



For selected instances that do not have an associated instance, pin, or a net; the *Extend Selection* command does not extend the selection to any new objects. Likewise, the `lxSelectedExtendSelection` SKILL function returns a `nil`, indicating the extend operation failed as no new instances could be added to the selected set.

Now, consider a layout design that contains the following objects:

`Mfactored set = M1.1, M1.2, M1.3`

© 2023 Cadence Design Systems, Inc.

Bus_Pin<0:2> = p<0>, p<1>, p<2>

Bus_Pin<3:4> = p<3>, p<4>

Bundle Net = net1, net2, net3

Let us now consider some scenarios to see how the *Extend Selection* command behaves when the selected instance is an object or a set of objects from the example above.

Scenario 1: Select *M1.1* and *p<0>* and run *Extend Selection*

Because the mfactored instance *M1.1* is associated with instances *M1.2* and *M1.3*, the selection is extended to both these instances. Also, the selected bus pin, *p<0>* is associated with *p<1>* and *p<2>*. Therefore, the selection set is extended to include: *M1.1, M1.2, M1.3, p<0>, p<1>, p<2>*.

If the schematic equivalent of the selected instance, such as bundle net or pin, is not available, the selection cannot be extended to these objects as the binder, which provides the base names required for the selection, is unavailable in the absence of the schematic view.

Scenario 2: Select *p<0>* and run *Extend Selection*

Here, depending on the availability of the binder, the selected set can vary as follows:

Without the binder: When the binder is unavailable, information about the schematic mapping of the pin is unavailable. Therefore, *Extend Selection* looks for all the pins in the layout with the same base name as the selected pin. In this case, the resultant selected set then becomes: *p<0>, p<1>, p<2>, p<3>, p<4>*

With the binder: When the binder is available, the schematic mapping of the selected pin is available. Therefore, *Extend Selection* selects only those pins that actually map with the selected pin. In this case, the resultant selection set then becomes: *p<0>, p<1>, p<2>*

Scenario 3: Select *net1* and run *Extend Selection*

Here, depending on the availability of the binder, the selected set can vary as follows:

Without the binder: *net1*

With the binder: *net1, net2, net3*

Related Topics

lxCombination Property

© 2023 Cadence Design Systems, Inc.

lxSelectedExtendSelection

Transistor Folding

Transistor folding lets you interactively divide the following into two or more layout instances with terminals all connected in parallel to the same nets:

- an individual transistor or a chain of transistors
- an already folded transistor or a chain of folded transistors
- an individual FinFET device or a chain of FinFET devices
- a multi-fingered device or transistor.

Folding lets you change the aspect ratio of the devices in your design while retaining the original location and orientation of the device that was folded. The net on the left of the device (or on the bottom for vertically-oriented devices) is the same before and after folding.

The transistors to be folded need not be bound to a schematic instance but they must be assigned to a component type, which stores the width attributes set on the cell.

If the devices to be folded are FinFETs, the devices must have their *Number of fins* parameter or the *Width parameter* set to enable folding.

You set this information using the *Edit Component Types* command. For more information, see Component Types Mode.

The individual device folds are named *instName.integer* in the layout. For example, if you have an instance named `I0` and you divide it into three folds, the folds are named as follows.

Instance Name	Folded Instance Names

`I0`	`I0.1`
	`I0.2`
	`I0.3`

You must also have the mfactorSplit environment variable set to `t` in a setup file for the design, otherwise, the *Generate Folded Devices* command does not work.

© 2023 Cadence Design Systems, Inc.

Related Topics

Unit Identifiers in Layout XL

Folding a Multi-fingered Device to Create Split Fingers

Rounding Parameter Values

Generate Folded Devices Form

retainSelectionSet

Interactive Transistor Folding

Interactive folding might overwrite the default folding threshold value using the specifications you supply. For example, if you divide the transistor into too few folds, each fold might end up longer than the folding threshold value you specified for the component type. Automatic folding always respects the folding threshold value.

If you enter a folding threshold value that would yield a device with more than 999 folds (for example, a device width of 2 and folding threshold of1µ, which can result in a device folded into 2M folds), the CIW prints a warning and does not fold the device. You can still fold such a large device manually.

Interactive folding supports a maximum number of 500 folds to be generated for a device. Specifying a value greater than 500 issues a warning message in the CIW and no folding is performed.

Interactive folding rounds the width values using the rounding values specified in CPH to provide the accurate Total Width and individual width values prior to creating devices in the layout. For example, assume you have a device |P0 with `w = 3u` and CPH `Rounding = (w 0.5u round)`. If you fold `|P0` into 7 folds with equal widths, each width in the form will be displayed as 428.571429n. Now, if `Rounding` is applied when the devices are placed in the layout, the width of each folded device is set to 500n, making the form widths inconsistent with the device widths in the layout.

Instead, if the interactive folding uses the CPH `Rounding` value to round the width values at the time of the initial calculation, the form widths achieved are consistent with the layout widths. This provides a more realistic and reliable fold width information before the actual creation.

Interactive folding always considers the parameter that sets the device width in the source, even if you have specified that this parameter is to be ignored in the `paramsToIgnoreForCheck` environment variables.

© 2023 Cadence Design Systems, Inc.

To keep folded devices from becoming off-grid, set the *Rounding* option in the Configure Physical Hierarchy window.

When folding FinFET devices, folding can be driven by the *Number of fins* parameter or the *Width parameter* set on the devices that are selected for folding. When folding based on *Number of fins*, the Generate Folded Devices form displays the folding values in fins instead of width.

Interactive folding can retain folded devices as selected even after the devices are placed in the layout canvas. This behavior can be controlled using the `retainSelectionSet` environment variable.

Related Topics

Unit Identifiers in Layout XL

Folding a Multi-fingered Device to Create Split Fingers

Rounding Parameter Values

Generate Folded Devices Form

retainSelectionSet

Folding a Transistor or FinFET Device

To fold a MOS transistor or a FinFET device:

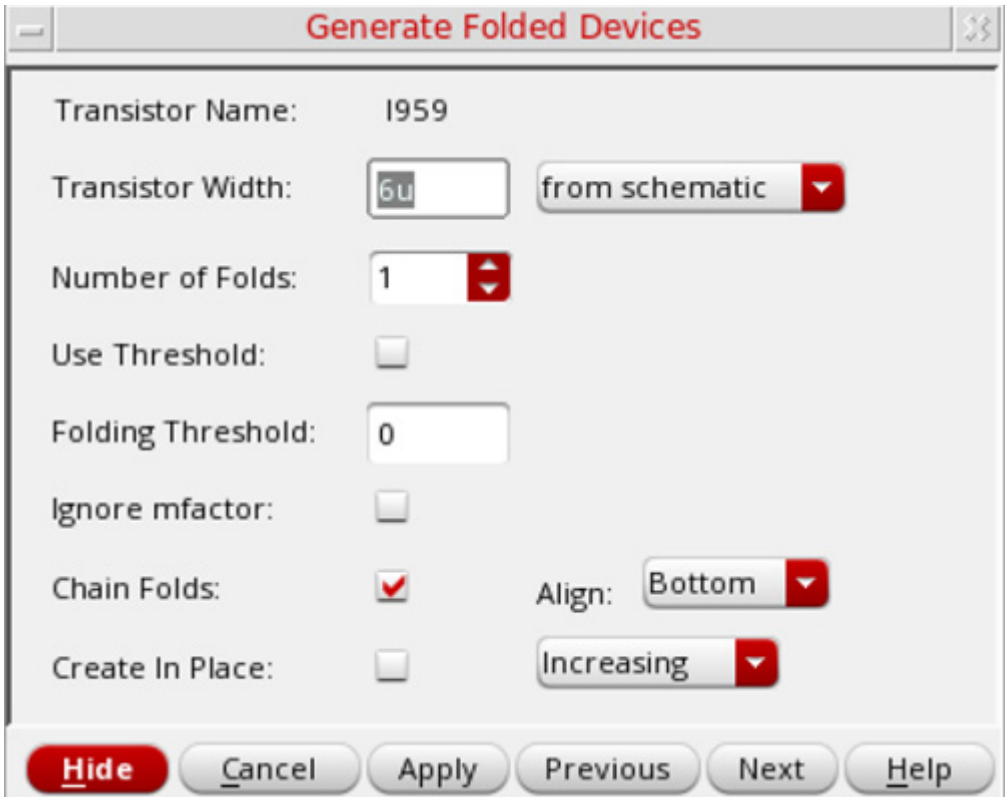
1. In the schematic or layout canvas, select the device you want to fold.

   When selecting a device that has already been folded, make sure you select the whole device. To do this, select it from the schematic window, or select all its folds in the layout window.

2. Choose *Connectivity – Generate – Folded Devices*.

© 2023 Cadence Design Systems, Inc.

The Generate Folded Devices form is displayed.



The name of the selected transistor or FinFET device is shown in the *Transistor Name* field. (If you did not select any transistors before choosing the command, no transistor-specific information is shown in the form until you select at least one transistor.)

The width of the selected transistor is shown in the *Transistor Width* field, which also indicates whether the value was retrieved from the schematic view or the layout view.

The *Transistor Width* field is displayed when the devices selected for folding are MOS transistors that have the *Width parameter* set.

Important

If FinFET devices are driven for folding by *Number of fins*, the **Generate Folded Devices** form displays *Transistor Fins.* Also indicating whether the value is taken from the schematic or the layout view.

3. Type the *Number of Folds* into which you want to divide the selected device.

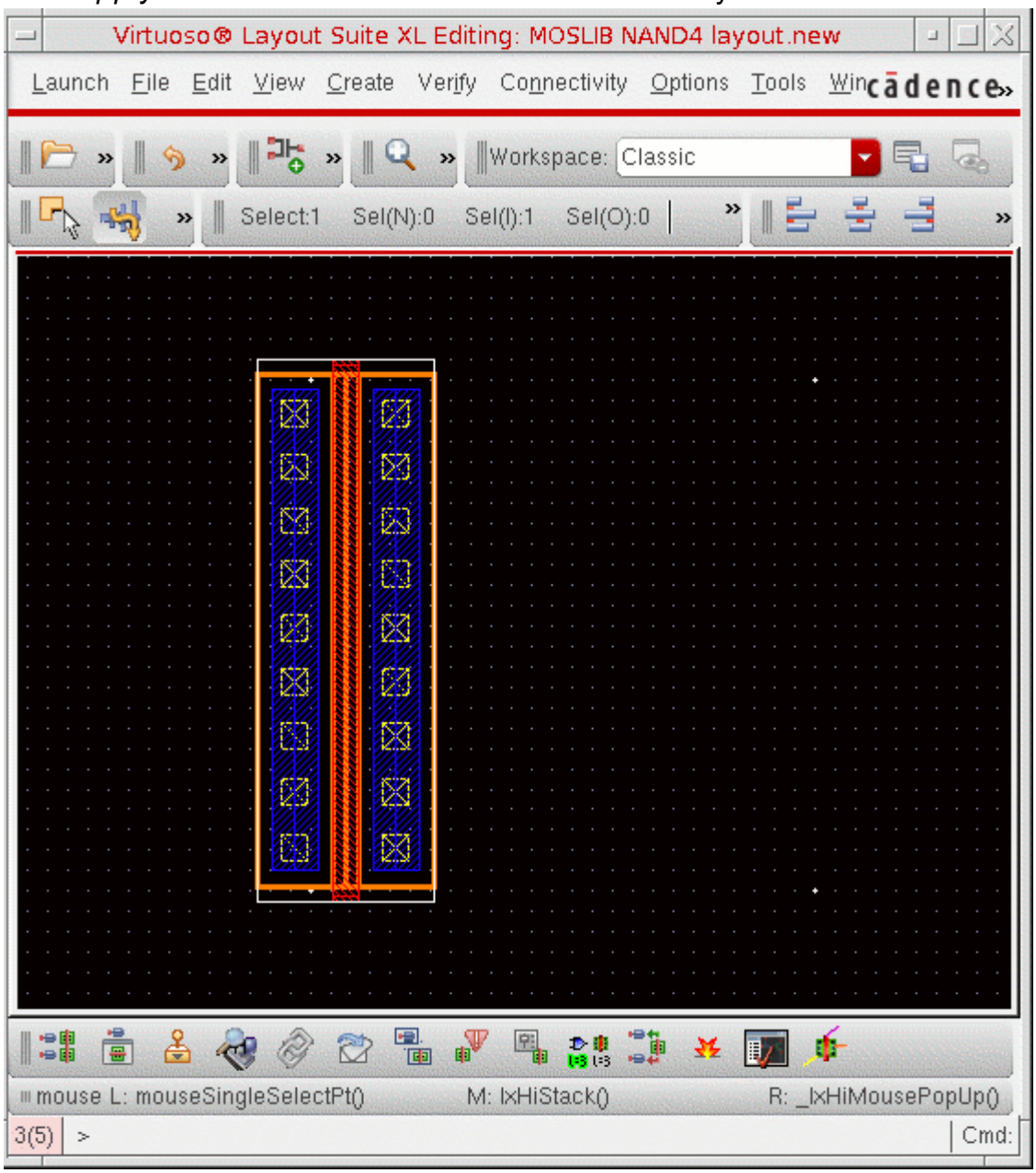
   Alternatively, use the bindkey *Shift + Scroll Up/Down* operation to increase or decrease the number of folds.

© 2023 Cadence Design Systems, Inc.

The maximum number of folds supported for a device is 500. Use the `foldsLimit` environment variable to control the maximum number of folds to be generated.

This option is not available when folding a multi-fingered device.

4. Click *Apply* in the form and move the cursor into the layout canvas.

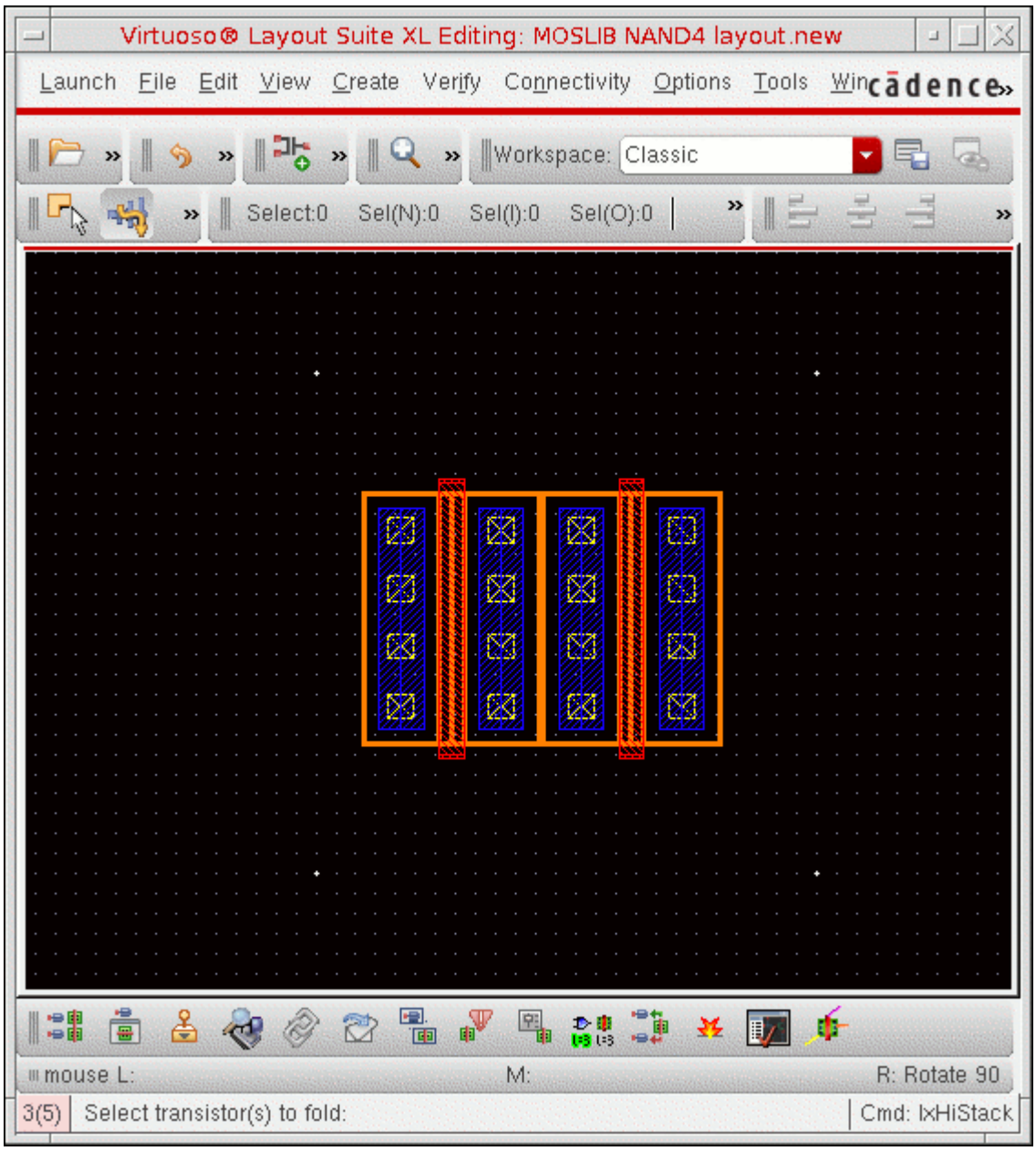


The outline of the folded device follows the cursor in the canvas.

© 2023 Cadence Design Systems, Inc.

If you have the `lxBindKeys.il` loaded, you can click the middle mouse button to rotate the device by R90, or `Shift`-click the middle mouse button to flip the device alternately about its X and Y axes.

5. Click in the layout where you want to place the folded device.



The folded device is placed in the canvas. The original device orientation is preserved. The net on the left of the device (or on the bottom for vertically-oriented devices) is the same before and after folding.

© 2023 Cadence Design Systems, Inc.

6. Select the device again in the canvas.

   The form updates to show the individual folds and the widths specified for them.

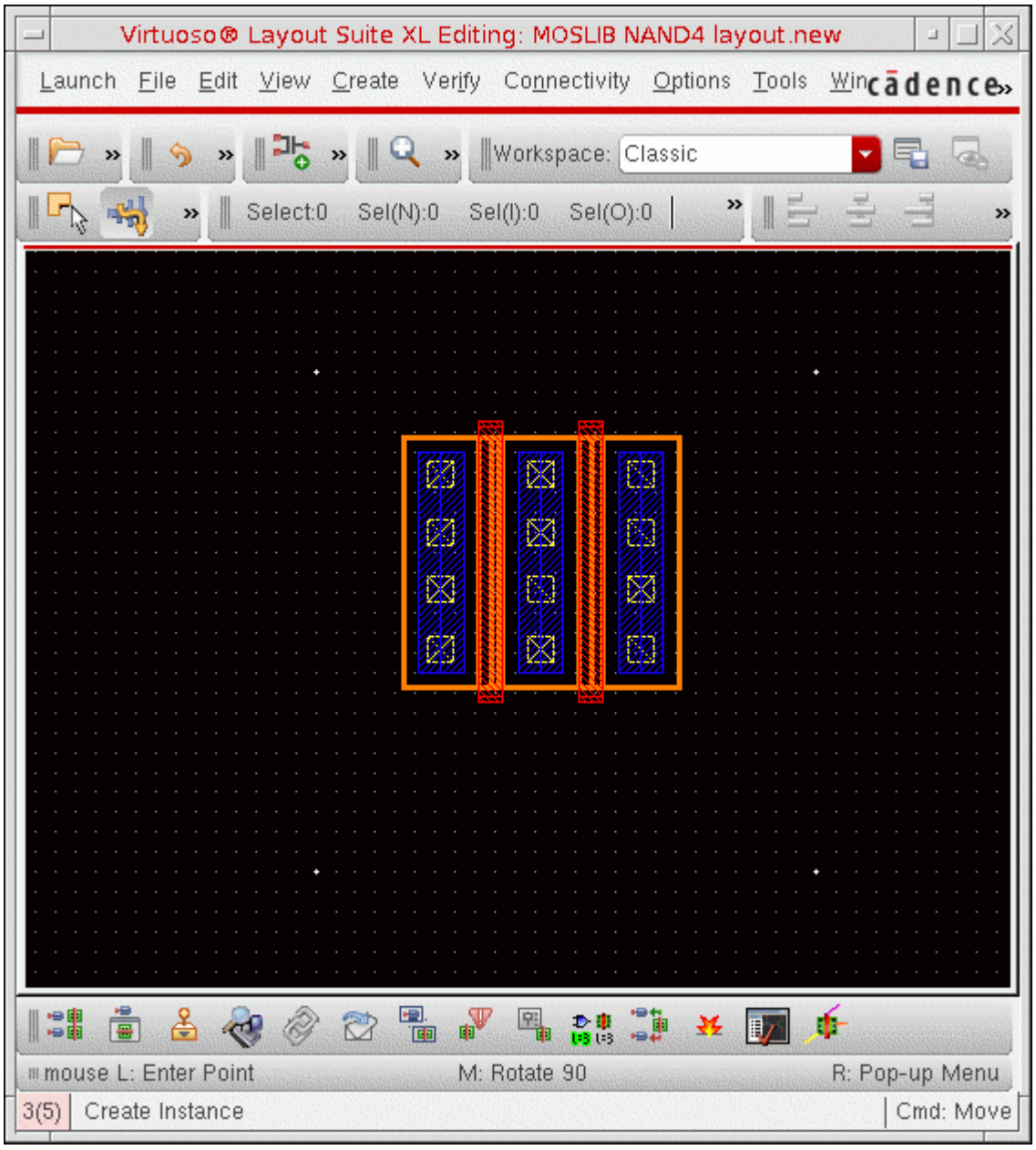
7. Select *Use Threshold* to fold based on the folding threshold value for the selected component type.

8. Type the *Folding Threshold* value to specify the maximum device width beyond which the selected devices will be automatically folded.

9. Select *Chain Folds* to automatically chain the newly created folds.

© 2023 Cadence Design Systems, Inc.

10. Click *Apply* in the form, move the cursor into the layout canvas, and click where you want to place the folded device.



The new folds are now chained together.

11. Select the device again in the canvas and change the widths of the individual folds in the form.

This option is not available when folding a multi-fingered device.

© 2023 Cadence Design Systems, Inc.

12. Click the *Distribute* button to distribute the device width across the remaining folds, or type individual values into each *Width* field.

The system issues a warning if the total width you specify exceeds the schematic width or the effective width.

If you set a width with no unit, the unit is automatically updated with the default value of the CDF width parameter. For example, if the default CDF value is `3u` and you enter a width of `6`, the width is set to `6u`. If the default CDF value is `3` and you enter a width of `6`, the width remains set to `6`.

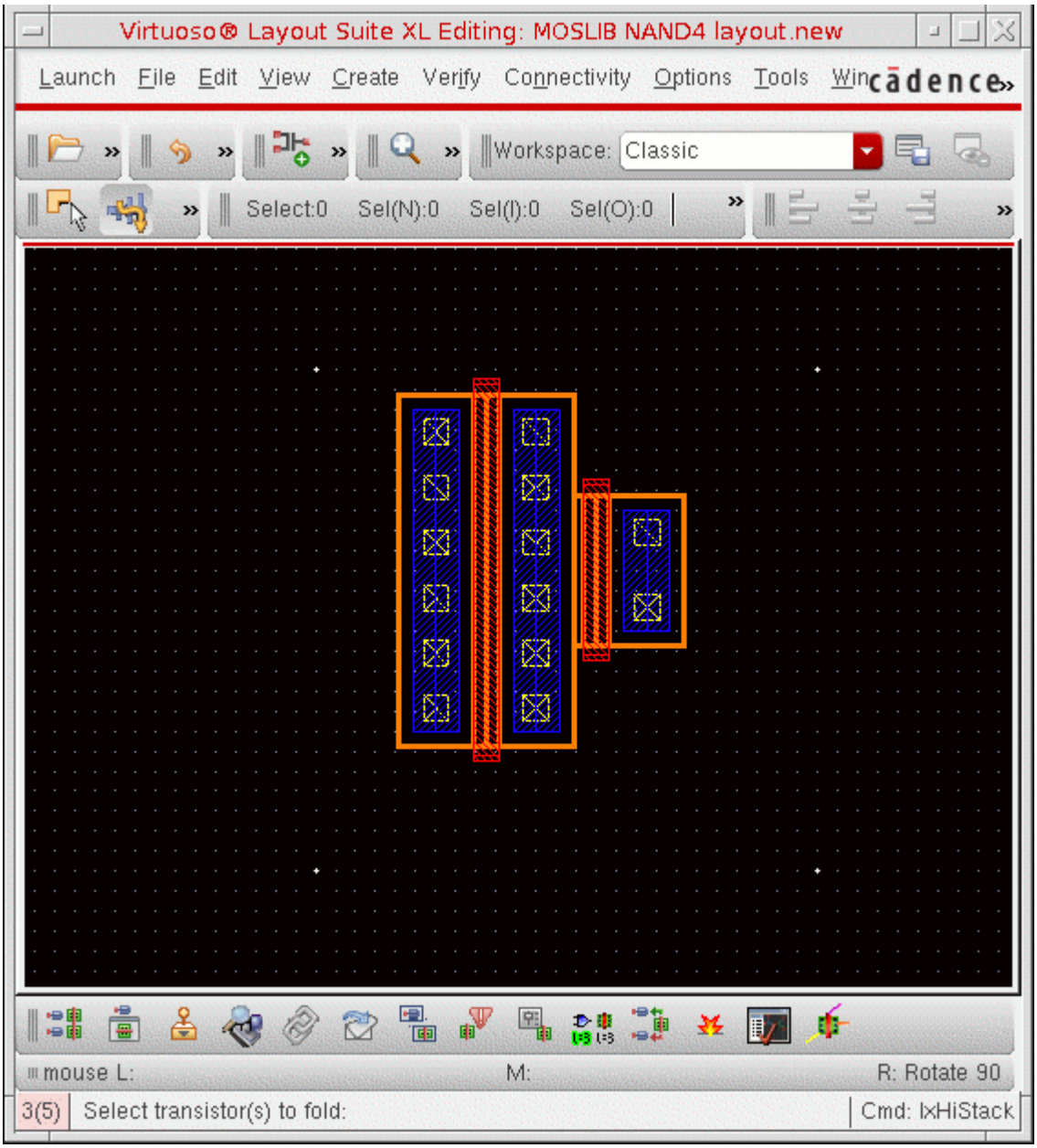
If the device selected for folding is a FinFET device driven for folding by the *Number of fins* value, clicking *Distribute* divides the total fins across all the folds such that each fold gets a whole number of fins. *Total Fins* is the sum of the fins in each fold of a FinFET device.

The *Fins* field accepts only integer values for fins. If you type in a non-integer value, Layout XL automatically reverts to the last integer value.

If a FinFET device has two fins and the *Number of folds* is specified as *2*, clicking *Distribute* splits the fins between the two folds such that one fold has two fins and the other fold has one fin. When folding based on the number of fins, the distribution is always such that each fold gets a whole number of fins.

© 2023 Cadence Design Systems, Inc.

13. Click *Apply* in the form, and then click in the canvas to place the new folded device.



To unfold a device:

➡ In the Navigator assistant or layout canvas, right-click the folded device and choose *Unfold* from the shortcut menu.

© 2023 Cadence Design Systems, Inc.

The device is unfolded and reduced to a single fold of the same width as the schematic instance or to the total layout width if the `unfoldUseLayoutWidths` environment variable is set to t.

Related Topics

Generate Folded Devices Form

foldsLimit

Folding a Multi-fingered Device to Create Split Fingers

Unit Identifiers in Layout XL

unfoldUseLayoutWidths

Split-Fingered Device Folding

Layout XL recognizes the fingers of a split fingered device and enables generating the device as folded in the layout. This means, if a schematic device has the *Split fingered devices* option selected, Layout XL recognizes each finger as a fold and allows generating the device as a folded device.

For example, if a schematic device has the *Split fingered devices* option set to `true` and the schematic instance has the following parameters set:

Fingers = 3

Total Width = 900n

Finger Width = 300n

The schematic device is generated in the layout as three instances with one finger each. The width of each generated device is 300n and the finger width is 300n. Any modifications made to the *Number of Folds* value in the Generate Folded Devices form accordingly modifies the generated device.

For example, if the number of folds value is changed from `3` to `5`, `5` instances instead of `3` are generated in the layout, with a finger each. Likewise, the *Total Width* and the *Finger Width* also get modified accordingly to `180n` each.

If a multi-fingered device requires to be split after the device has been generated in layout, you can fold the multi-fingered device in order to create a separate device for each finger.

© 2023 Cadence Design Systems, Inc.

Related Topics

Connectivity Form

Splitting Fingered Devices

Generate Folded Devices Form

Folding a Transistor or FinFET Device

Folding a Multi-fingered Device to Create Split Fingers

Folding a Multi-fingered Device to Create Split Fingers

The *Split fingered devices* option controls whether each finger of a schematic device is placed in the layout as a separate device. If a multi-fingered device requires to be split after the device has been generated in layout, you can fold the multi-fingered device in order to create a separate device for each finger.

To fold a multi-fingered device or transistor:

1. In the schematic or layout canvas, select the multi-fingered device you want to fold.
2. Choose *Connectivity – Generate – Folded Devices*.

   The Generate Folded Devices form is displayed.

   The number of existing folds is displayed in the *Number of Folds*. Folding creates a new device for each fold by setting the *Fingers* parameter to `1`. By default, the width of each new device relies on the fingers callback of the PDK to set the correct width. By setting the environment variable `foldFingerSplitUseSchWidth` to `t`, you can specify that the width be calculated by dividing the schematic instance by the number of fingers and applying the calculated number to the width of each new device created.
3. Click *Apply* in the form and move the cursor into the layout canvas. The outline of the folded device follows the cursor in the canvas.
4. Click in the layout where you want to place the folded device.

   The folded device is placed in the canvas. The original device orientation is preserved. The net on the left of the device (or on the bottom for vertically-oriented devices) is the same before and after folding.

   The *Split Fingered Devices* option in CPH is automatically set for that device. This means that if you delete and regenerate the layout view, the CPH settings are retained

© 2023 Cadence Design Systems, Inc.

and each finger of the multi-fingered device will automatically be placed in the layout as a separate device.

Related Topics

foldFingerSplitUseSchWidth

Connectivity Form

Splitting Fingered Devices

Generate Folded Devices Form

Folding a Transistor or FinFET Device

Split-Fingered Device Folding

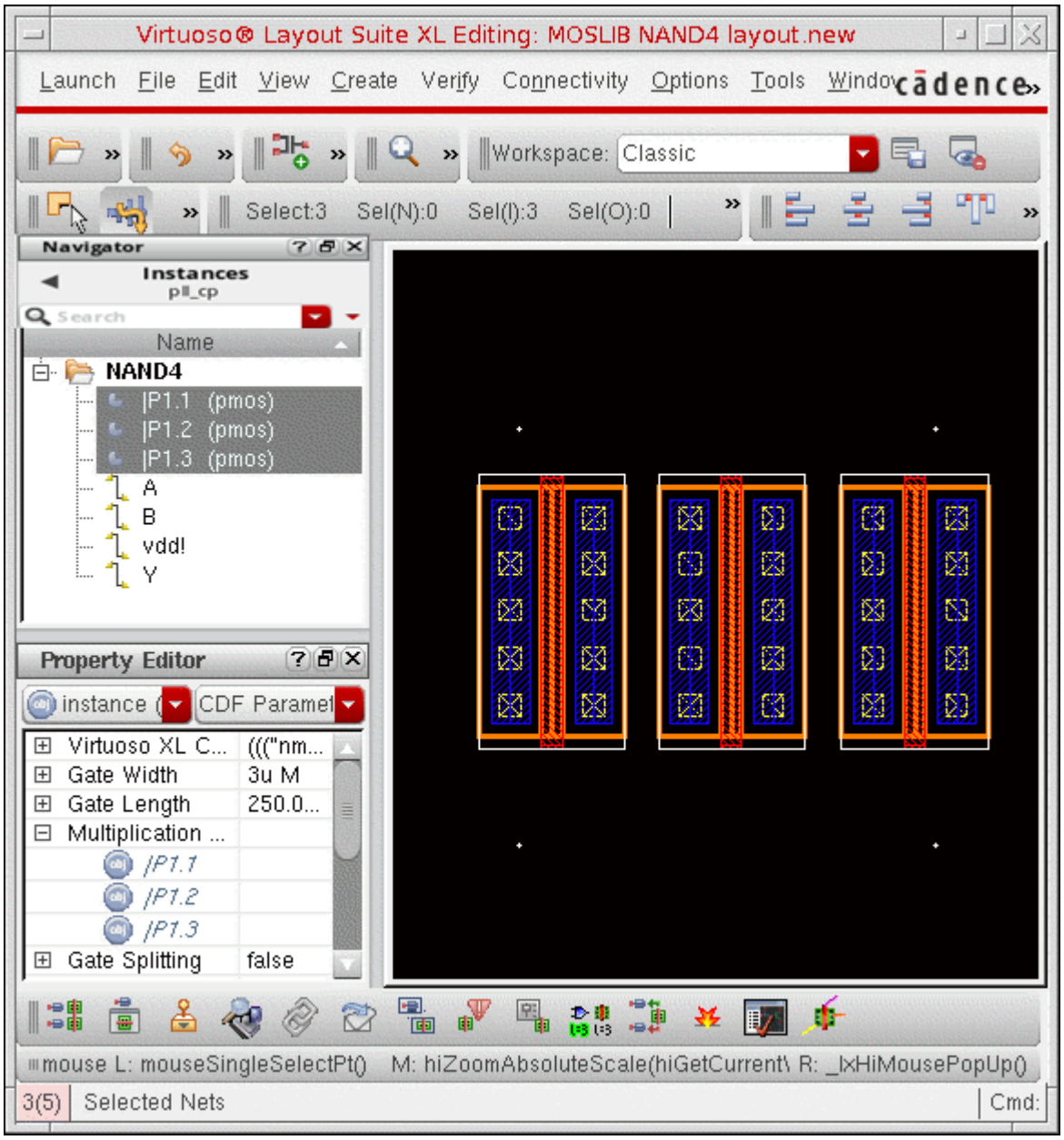
Folding a Multi-fingered Device to Create Split Fingers

Ignoring Schematic Mfactor during Folding

By default, interactive folding is constrained by the multiplication factor property set for the instance in the schematic view. You can set the *Ignore mfactor* option in the Generate Folded Devices form to gain more flexibility when folding your device.

© 2023 Cadence Design Systems, Inc.

Consider a schematic instance `P1` with a multiplication factor of 3 and each part with a gate width of 3u. The device is generated in the layout view as shown below.



You want to regenerate this device as two folds, each 4.5u wide. To do this:

1. Select any of the mfactor parts in the layout window.
2. Choose *Connectivity – Generate – Folded Devices*.

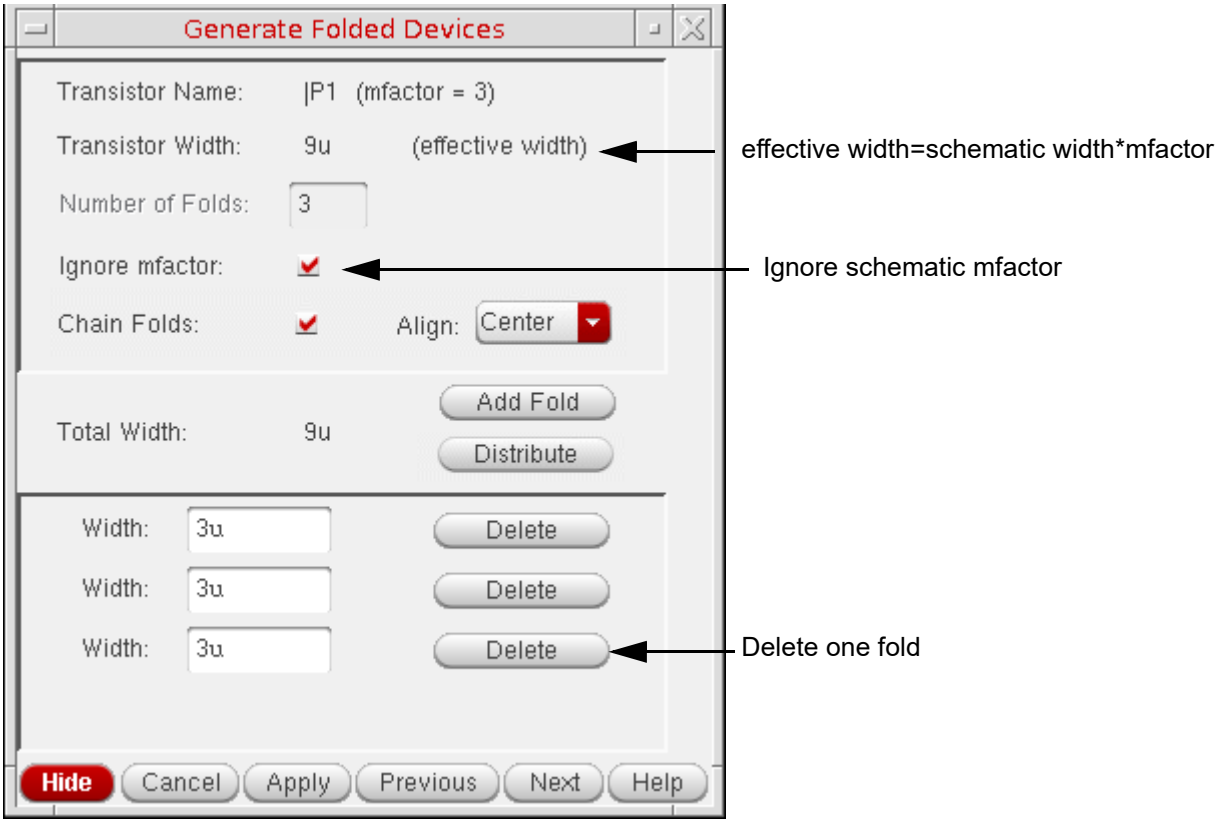
   The Generate Folded Devices form is displayed.

© 2023 Cadence Design Systems, Inc.

3. Select the *Ignore mfactor* option.

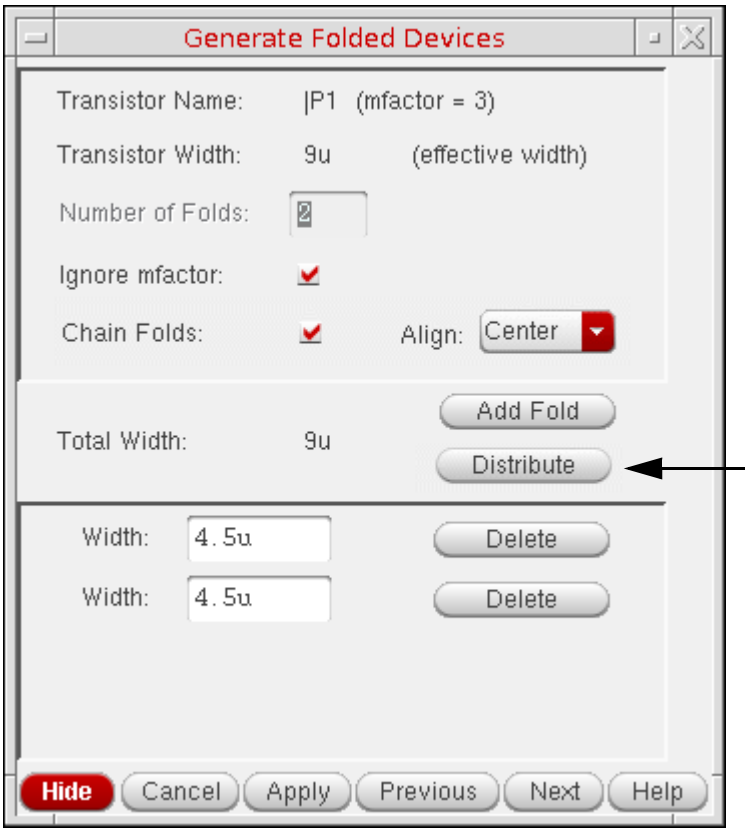
   The *Transistor Width* is set to the effective width in the schematic; that is, the product of mfactor and schematic gate width.

4. Delete one of the listed folds.



© 2023 Cadence Design Systems, Inc.

5. Click the *Distribute* button to divide the transistor width evenly across the remaining folds, resulting in two folds of *4.5u*.

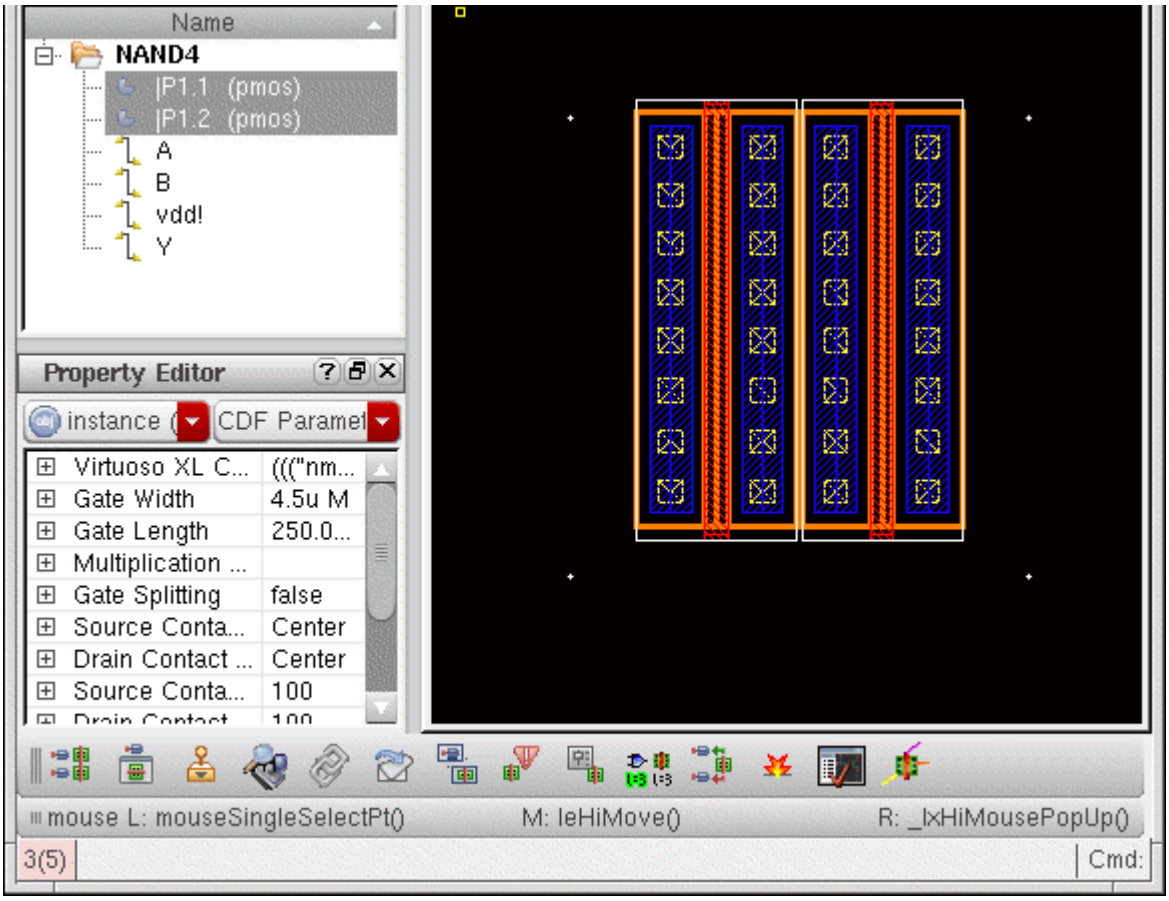


You can also type values into each *Width* field. The system issues a warning if the total width you specify exceeds the schematic width or the effective width.

6. Click *Apply* in the form, and then click in the canvas to place the new folded device.

© 2023 Cadence Design Systems, Inc.

The device is generated as two folds, each 4.5u wide.



Check Against Source will report that the device is implemented in the layout using fewer instances than are specified in the schematic.

Related Topics

Generate Folded Devices Form

Folding a Transistor or FinFET Device

Split-Fingered Device Folding

Folding a Multi-fingered Device to Create Split Fingers

© 2023 Cadence Design Systems, Inc.

Unit Identifiers in Layout XL

When specifying widths, use the following identifiers to specify different units.

Identifier	Denotes	Factor
`y`	yocto	10^{-24}
`z`	zepto	10^{-21}
`a`	atto	10^{-18}
`f`	femto	10^{-15}
`p`	pico	10^{-12}
`n`	nano	10^{-9}
`u` (the default)	micro	10^{-6}
`m`	milli	10^{-3}
`k`	kilo	10^{3}
`K`	kilo	10^{3}
`M`	mega	10^{6}
`G`	giga	10^{9}
`T`	tera	10^{12}
`P`	peta	10^{15}
`E`	exa	10^{18}
`Z`	zetta	10^{21}
`Y`	yotta	10^{24}

Related Topics

Transistor Folding

Pin Permutation

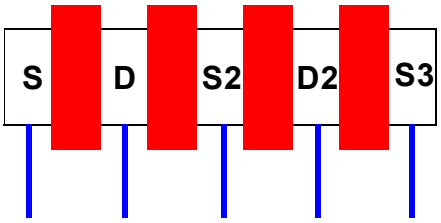
Layout XL lets you exchange the connectivity or net connections of the pins of a component. This operation is called pin permutation. The pins to be permuted must belong to different

© 2023 Cadence Design Systems, Inc.

nets and must first be defined as permutable terminals. If one of the pins to be permuted is on an instance contained in any type of group (including a synchronous clone), the other pin must be on an instance in the same group.

When a device has must-connected terminals connected to different nets at the current level, these nets, by definition, must be connected further up the hierarchy. For the purposes of pin permutation, such nets are considered equivalent.

In the device below, nets terminals `S`, `S2`, and `S3` are defined in a must-connect relationship. Terminals `D` and `D2` are in a separate must-connect relationship. All the terminals are connected to different nets at the current level.

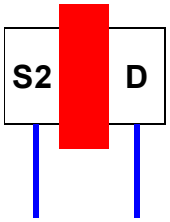


Device model

The following permuteRule is specified for the device.

```
p(S2 D)
```

Any net connectivity changes made to `S2` will also be applied to all its other must-connected terminals, for example; `S`, `S3`. The same applies for `D` and `D2`. The permute model is as illustrated below.



Permute model

Related Topics

Permuting Pins Manually

Permuting Pins Automatically

Disabling Permutation for an Instance

© 2023 Cadence Design Systems, Inc.

Permuting Pins Manually

You can permute only those pins that were not previously connected to a path, polygon, rectangle, or component pin.

To manually permute pins within a component:

1. From the layout window menu bar, choose *Connectivity – Pins – Permute*.

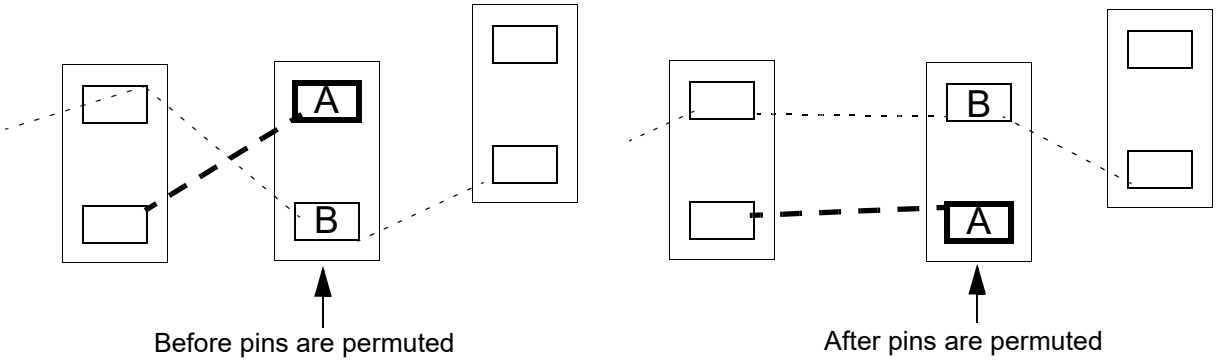
   Layout XL prompts you to select a pin.

2. Click the first pin in the layout canvas.

   Layout XL prompts you to select a second pin.

3. Click the second pin in the layout canvas.

   Layout XL switches the net connections of the pins you selected. If flight lines are displayed, they change to show the new nets assigned to the pins.



   The *Permute Pins* command is modal, meaning that you can continue to select sets of pins to permute without re-choosing the command every time.

4. To undo the permutation, choose *Edit – Undo* from the layout window menu bar.

   The pins return to their original net connection in the layout.

5. To cancel the command, press `Esc`.

   To permute the source and drain for an instance with a simple permute rule for two terminals only, you can right-click the instance and choose *Permute Pins* from the context menu.

© 2023 Cadence Design Systems, Inc.

Permuting Chained Instance Pins

To permute all the source and drain pins on selected chained instances:

➡ Right-click the instance chain and choose *Permute Pins* from the shortcut menu.

The source and drain pins on all the instances in the selected set are permuted, leaving any existing abutments undisturbed.

Related Topics

Chaining Devices Interactively

Permuting Pins Automatically

Permuting Pins Automatically

When automatic pin permutation is switched on, Layout XL automatically permutes pins in a component if doing so corrects a short caused by:

- Interactive routing (using automatic abutment).
- Moving or stretching a shape or using any operation that changes connectivity.

If a short involves stopped layers, automatic pin permutation may not resolve the shorts because Layout XL does not support stopped layers for automatic pin permutation. To resolve such shorts, perform manual pin permutation.

Important

If you are experiencing slow bus creation, Cadence recommends that you temporarily switch off automatic pin permutation until routing is complete.

To switch on automatic pin permutation:

1. From the layout window menu bar, choose *Options – Connectivity*.

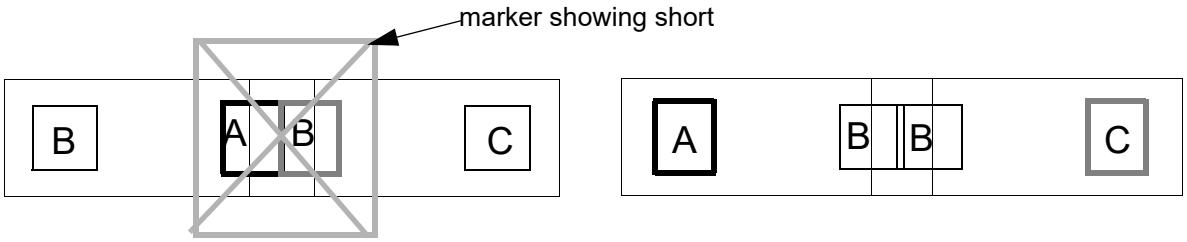
   The Connectivity form is displayed.
2. In the *Generation* tab, select *Permute pins*.
3. Click *OK*.

© 2023 Cadence Design Systems, Inc.

Example 1

Layout XL automatically permutes the pins of a single component when doing so removes a short.

In the diagram below, two pins belonging to different nets (A and B) create a short when they are touching each other. If these pins are permutable and if reversing the pins of one component connects the two pins that are on the same net (B), Layout XL permutes them automatically.



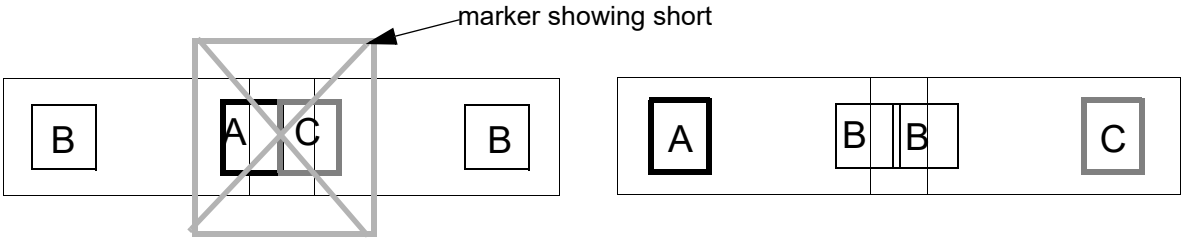
Before pins are permuted

After pins are permuted

Example 2

Layout XL also automatically permutes the pins of two components simultaneously when doing so removes a short.

In the diagram below, two pins belonging to different nets (A and C) create a short when they are touching each other. If these pins are permutable, and reversing the pins of both components connects the two pins that are on the same net (B), Layout XL permutes them.



Before pins are permuted

After pins are permuted

© 2023 Cadence Design Systems, Inc.

Checking Permutation Information

To see information about permuted pins:

1. From the layout window menu bar, choose *Connectivity – Pins – Permute*.
2. Choose the pins you want to permute.
3. Press `F3`.

   The Permutation Information form appears telling you what was done. If the pins you select do not permute, the info window tells you why.



Cadence recommends not setting the Object Filter option on the XL Probe form to 'Pins' when probing pins for permuted instance. Instead set the option to 'Nets to' view the correct information.

Updating Permuted Pins

Permuted pins are not preserved by the *Update Components And Nets* command. If any component in the design needs to be updated, the command resets the permutation status of all the cells to that originally stored in the schematic.

Related Topics

Connectivity Form

Permutation Information Form

Chaining Devices Interactively

Permuting Pins Manually

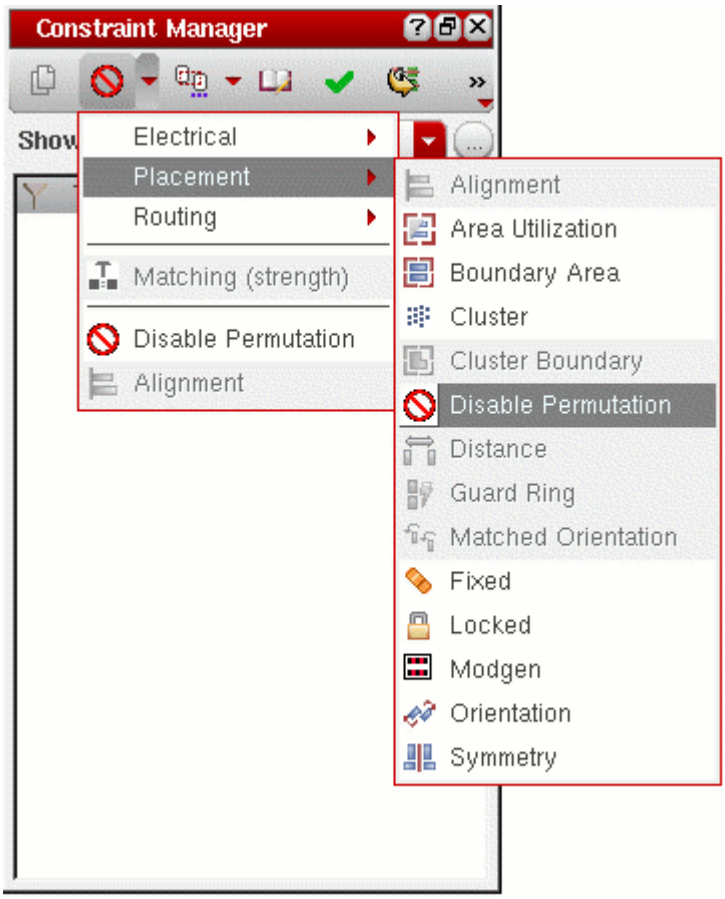
Disabling Permutation for an Instance

You can use the `disablePermutation` constraint to prevent permutation on a selected instance without having write access to the schematic design. The `disablePermutation` constraint overrides any permuteRule set for the instance in question and applies throughout the flow.

To create a `disablePermutation` constraint for an instance,

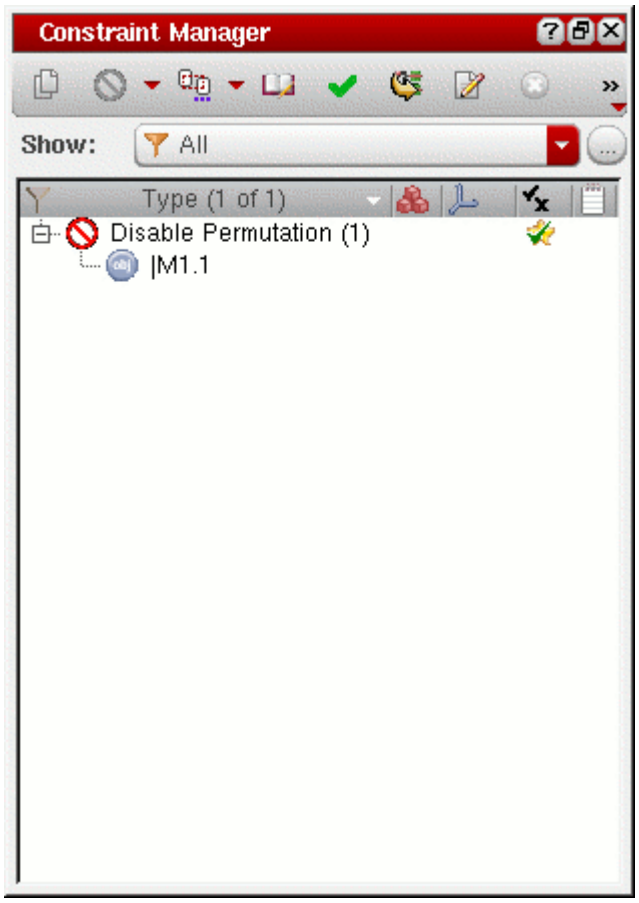
© 2023 Cadence Design Systems, Inc.

1. Select the instance in question in the layout view.
2. In the Constraint Manager assistant, use the pull-down to choose *Disable Permutation*.



© 2023 Cadence Design Systems, Inc.

The Constraint Manager is updated to show the new constraint on the selected instance.



Related Topics

disablePermutation

The Constraint Manager Assistant

Incomplete Nets

Layout XL can show you incomplete electrical connections between the component terminals, pins, and shapes associated with each net in your design. These incomplete connections include unrouted instance terminals either with no pins or with pins that are not extractable, and instance terminals with dangling must-join instance pins.

For display purposes the system defines an incomplete net as a net which has at least one *open marker* attached to it in the design database. Each open marker can be drawn as a

© 2023 Cadence Design Systems, Inc.

flight line in the design canvas. Each incomplete net can therefore be represented by one or more flight lines in the design canvas.

Layout XL draws flight lines showing incomplete electrical connections between the devices on each net. The CIW reports how many incomplete nets there are.

If you draw a path between two components that completes the connection, the flight lines disappear. If the path does not complete the connection, the flight lines remain.

To see incomplete net flight lines in the design canvas you must:

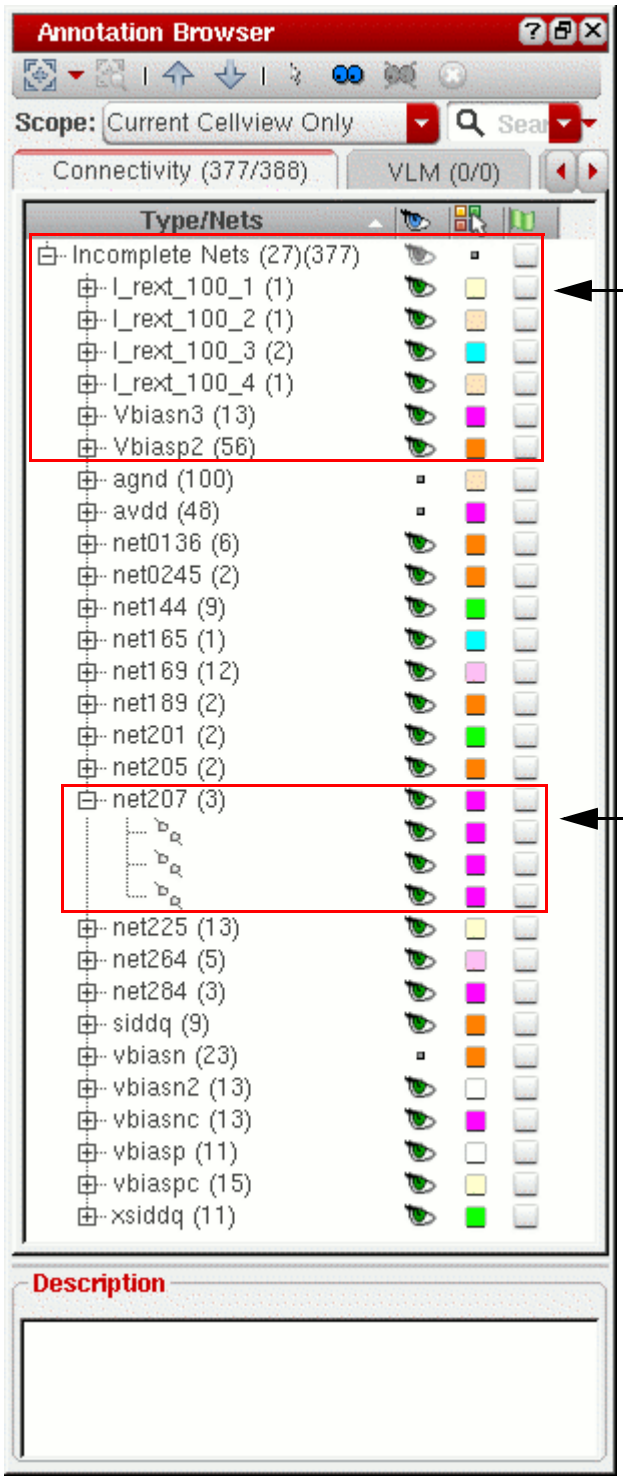
1. Extract the connectivity of your design to generate open markers.

   The connectivity extractor runs automatically during the *Generate All From Source*, *Generate Selected From Source*, and *Update Components And Nets* commands, and when you use the automatic placement and routing tools. You can also extract your design manually using the *Extract Layout* menu command or toolbar button.

2. Initialize the Annotation Browser assistant.

© 2023 Cadence Design Systems, Inc.

You use the Annotation Browser to view the markers in your design and specify how they are displayed in the canvas.



Incomplete nets are listed in the *Incomplete Nets* category in the *Connectivity* tab.

The first number in parentheses represents the number of incomplete nets; the second the total number of markers.

Expand an entry to see the open markers attached to the net. You can specify whether a marker is highlighted or not and which color is used to draw it in the canvas.

© 2023 Cadence Design Systems, Inc.

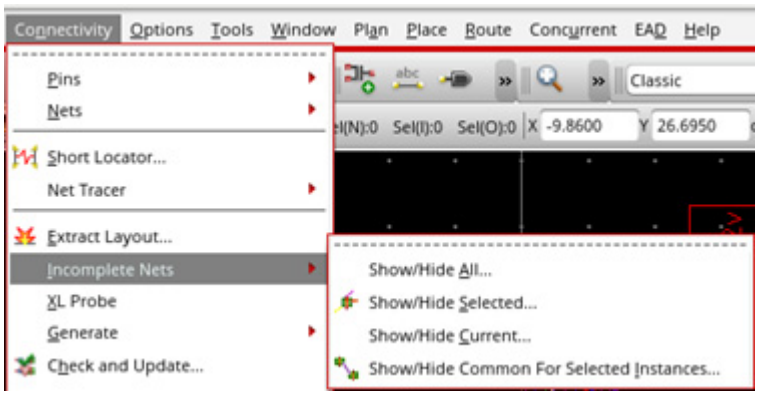
Important

Although it must be initialized, the Annotation Browser assistant itself need not be displayed on the desktop in order to show and hide incomplete nets. For example, you can close it and show and hide incomplete nets in postselection mode directly from the Navigator assistant. To do this, though, you must switch off the *Remove highlighting when browser is closed* option in the Annotation Browser Options form, otherwise there can be no visible incomplete nets to show and hide.

If you select one or more incomplete net markers in the Annotation Browser assistant, you can click the right mouse button and choose *Select Nets* to cross-select the nets in the layout canvas and Navigator assistant. You can then invoke the router on the selected incomplete nets.

When you edit an instance in place, any flight lines highlighted at the higher level of hierarchy remain highlighted while you edit in place.

Choose *Connectivity – Incomplete Nets* in the Layout XL toolbar to view related commands.



Related Topics

Connectivity Extraction Use Model.

Annotation Browser Assistant

Incomplete Nets Commands

© 2023 Cadence Design Systems, Inc.

Incomplete Nets Commands

Command	Description
Show/Hide All	Show and hide all the incomplete nets in your design. You can also access this command using: ■ The `Shift+Ctrl+h` bindkey on your keyboard. ■ The lxToggleShowAllIncompleteNets SKILL function. If you run the *Show/Hide All* command with the Annotation Browser closed, the command fails. You must open the Annotation Browser before you run the command. To automatically display all the incomplete nets in the design when the layout is launched in Layout XL, set the environment variable showIncompleteNets to `t`.
Show/Hide Selected	Show and hide the incomplete nets associated with selected objects in your design. You can also access this command using ■ The *Show/Hide Selected Incomplete Nets* button in the Layout XL toolbar. (This is the method used in this document.) ■ The `Shift+h` bindkey on your keyboard. ■ The lxShowHideIncompleteNets SKILL function. When you start the command, the Annotation Browser is automatically initialized if required. The command does not support blockage, boundary, or row objects. These objects are silently ignored unless a preselected set contains only unsupported objects, in which case you see a message informing you of this fact. The command behaves differently depending on whether you launch it with objects already preselected or postselected. See Showing Incomplete Nets in Preselection Mode and Showing Incomplete Nets in Postselection Mode Environment variable: showIncompleteNets

© 2023 Cadence Design Systems, Inc.

Show/Hide Current	Change the visibility of the currently visible incomplete nets in the design canvas. You can also access this command using: ■ The `h` bindkey on your keyboard. ■ The lxToggleShowIncompleteNets SKILL function. See Showing and Hiding Current Incomplete Nets
Show/Hide Common For Selected Instances	Display incomplete nets that are common between selected instances. ■ *Show/Hide Common For Selected Instances* button in the Layout XL toolbar. ■ The `h` bindkey on your keyboard. ■ The lxShowCommonIncompleteNetsForSelectedInsts SKILL function. See Showing Common Incomplete Nets for Selected Instances

Related Topics

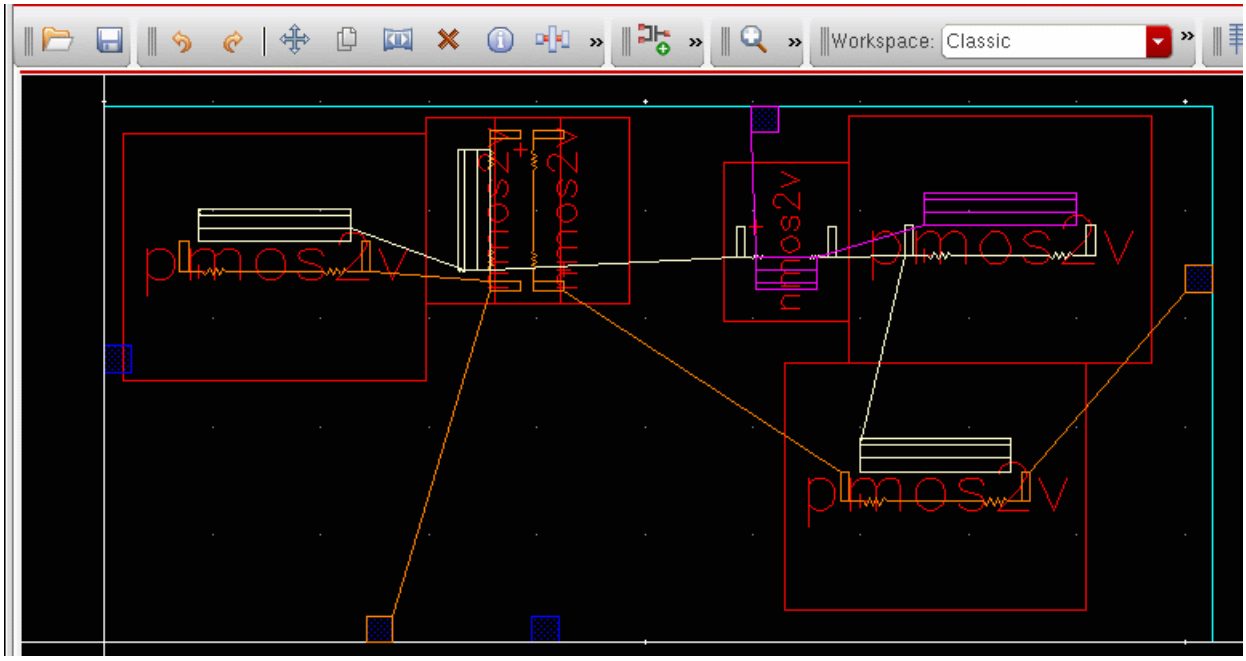
Incomplete Nets

© 2023 Cadence Design Systems, Inc.

Showing and Hiding Current Incomplete Nets

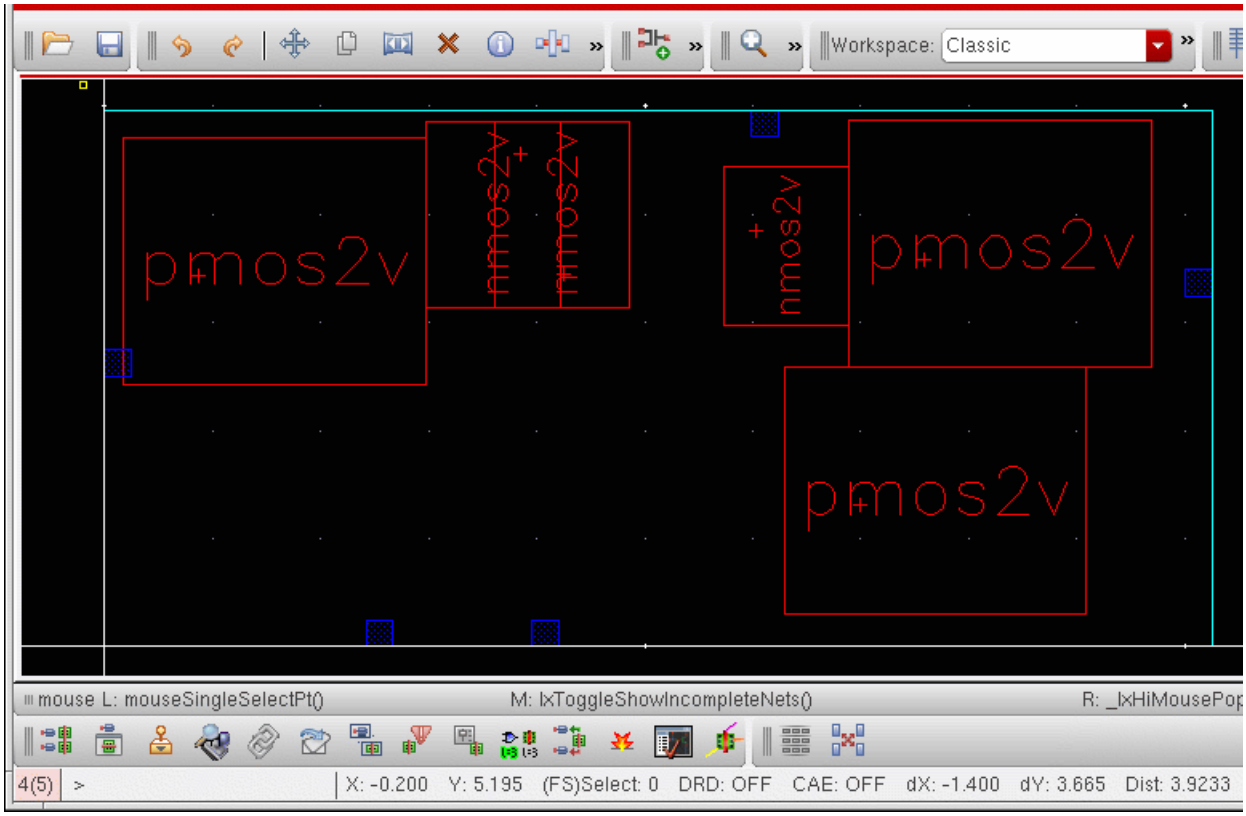
To show or hide current incomplete nets:

1. Place your cursor in a layout window showing **at least one incomplete net**.



© 2023 Cadence Design Systems, Inc.

2. Choose *Connectivity – Incomplete Nets – Show/Hide Current* or press `h` on your keyboard.



The function hides all the incomplete nets that were visible or partially visible in the canvas. The list of nets that were hidden is stored in the system.

3. To make the stored nets visible, choose *Connectivity – Incomplete Nets – Show/Hide Current* (or press `h`) again.

If you run the function for the first time on a design with no incomplete nets on display, it does nothing because there is no stored list of nets to display.

The function does nothing unless the Annotation Browser has been initialized. The assistant itself need not be displayed on the desktop. If you do close the browser, make sure you switch off the *Remove highlighting when browser is closed* option in the Annotation Browser Options form, otherwise there can be no visible incomplete nets to hide.

When an incomplete net is only partially visible (some of the open markers on the net are visible, others are not) the function considers the incomplete net as fully visible. The first call to the function hides the open markers that are currently visible; the second shows all the open markers associated with the incomplete net.

© 2023 Cadence Design Systems, Inc.

If you hide all the visible open markers in the canvas, then use the Annotation Browser controls to make an open marker visible, calling *Show/Hide Current Incomplete Nets* again does not restore the visibility of the previously hidden markers. The function **always** hides any visible open markers.

If you delete a net, any open markers associated with that net are removed from the stored list and cannot be restored when you run the command again.

Related Topics

The Navigator Assistant

Incomplete Nets Commands

Showing Common Incomplete Nets for Selected Instances

The *Show Common For Selected Instances* command enables you to display only those incomplete nets that are common between the selected instances.

For example, if a design has three instances, |1, |2, and |3, such that:

- I1 is connected to Net A and Net C
- I2 is connected to Net B
- |3 is connected to Net B and Net C

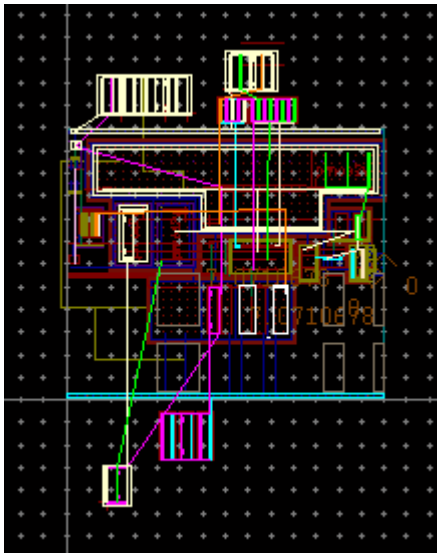
Then, with instances |2 and |3 selected, only Net B is displayed.

To see the result of running the *Show Common Incomplete Nets* command:

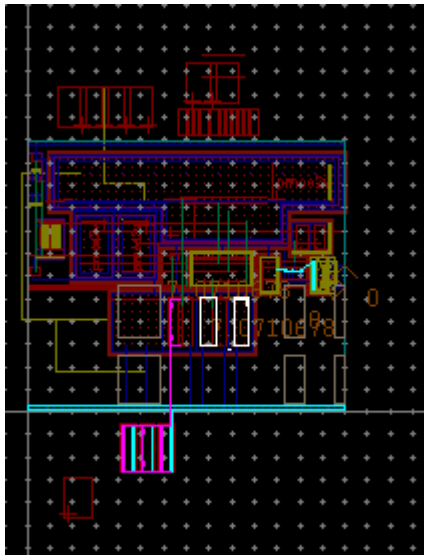
© 2023 Cadence Design Systems, Inc.

1. Choose *Connectivity – Incomplete Nets – Show/Hide All Incomplete Nets* command to display all the incomplete nets in the layout canvas.

Layout canvas displaying all incomplete nets



Layout canvas displaying only common incomplete nets between two selected instances



The *Show Common Incomplete Nets* command supports both preselection and postselection modes.

- In preselection mode, the command toggles between displaying and hiding the open markers for the nets that are common to the selected instances.
- In postselection mode, you can use an enter-function to select or deselect the instances for which you want the open markers on the corresponding nets to be displayed. In post-select mode, the open markers stay visible after the enter-function is canceled.

Related Topics

The Navigator Assistant

Showing Incomplete Nets in Preselection Mode

Showing Incomplete Nets in Postselection Mode

Incomplete Nets Commands

© 2023 Cadence Design Systems, Inc.

Showing Incomplete Nets in Preselection Mode

To show and hide incomplete nets for preselected objects,

1. In the layout canvas, select the objects for which you want to show or hide incomplete nets.

   The system gets all the incomplete nets associated with the selected objects. The system considers a partially-selected object to be fully selected.

2. In the Layout XL toolbar, click the *Show/Hide Selected Incomplete Nets* button.

   What happens next depends on the current visibility of the open markers associated with the selected incomplete nets.

   - If all the open markers are visible in the design canvas, the function hides them all.
   - If at least one open marker on one of the incomplete nets is hidden, the function makes them all visible.

3. Click the *Show/Hide Selected Incomplete Nets* button again to toggle the visibility of the incomplete nets.

Related Topics

Behavior by Object Type in Preselection Mode of Incomplete Nets

Showing Incomplete Nets in Postselection Mode

Incomplete Nets Commands

© 2023 Cadence Design Systems, Inc.

Behavior by Object Type in Preselection Mode of Incomplete Nets

The table below summarizes the behavior of the *Show/Hide Selected* incomplete nets command for different preselected objects.

Object Type	Behavior
Net	If the net is complete, the function does nothing; otherwise it toggles the visibility of the incomplete net as follows. ■ If all the open markers on the incomplete net are visible, the function hides them all. ■ If at least one open marker on the net is hidden, the function makes all the open markers on the net visible.
Shape or Via	If the shape or via has no net associated with it or if the associated net is complete, the function does nothing; otherwise it toggles the visibility of the incomplete net as follows. ■ If all the open markers on the incomplete net are visible, the function hides them all. ■ If at least one open marker on the net is hidden, the function makes all the open markers on the net visible.
Instance	If the instance has no net connected to it, the function does nothing; otherwise it gets all the incomplete nets associated with the instance and toggles the visibility of these nets as follows. ■ If all the open markers on all the incomplete nets are visible, the function hides them all. ■ If at least one open marker on one net is hidden, the function makes all the open markers on all the nets visible.

© 2023 Cadence Design Systems, Inc.

Object Type	Behavior
Group	The function recursively gets all the incomplete nets associated with all the objects inside the group and toggles the visibility of these nets as follows. ■ If all the open markers on all the incomplete nets are visible, the function hides them all. ■ If at least one open marker on one net is hidden, the function makes all the open markers on all the nets visible. An object inside a group is considered selectable even if it is set as not visible and not selectable in the Palette assistant.
Marker	A marker object cannot have any nets associated with it. The function gets all the incomplete nets attached to all the objects associated with the marker and then toggles the visibility of these nets as follows. ■ If all the open markers on all the incomplete nets are visible, the function hides them all. ■ If at least one open marker on one net is hidden, the function makes all the open markers on all the nets visible.

Related Topics

Behavior by Object Type in Preselection Mode of Incomplete Nets

Showing Incomplete Nets in Postselection Mode

Incomplete Nets Commands

Showing Incomplete Nets in Postselection Mode

You can also start the *Show/Hide Selected Incomplete Nets* command with nothing preselected in the canvas. When you then click an object, the function shows the incomplete nets associated with that object. To do this:

1. With nothing selected in the layout canvas, click the *Show/Hide Selected Incomplete Nets* button in the Layout XL toolbar.



© 2023 Cadence Design Systems, Inc.

You are prompted to click in the canvas to show the incomplete nets for the selected object.

2. Click an object in the canvas.

   The function shows all the open markers for the incomplete nets associated with the object you selected.

   In this mode, if you can see an object in the canvas, you can show the incomplete nets for that object regardless of whether it is set to be selectable in the Palette assistant.

3. Use `Shift`-click to add another object to the selected set.

   The function adds the incomplete nets associated with the object you just selected to the existing set of incomplete nets on display.

4. Use `Ctrl`-click to remove an object from the selected set.

   The function hides the incomplete nets associated with the object you clicked on.

   You can achieve the same effect by clicking a second time immediately after you have clicked to show the nets in step 2 above.

5. Press `h` on your keyboard to hide all the incomplete nets currently displayed in the design canvas.

   You can restore the visibility of those incomplete nets by pressing `h` again.

6. To end the *Show/Hide Selected Incomplete Nets* function, do one of the following.

   - Press the `Esc` key on your keyboard
   - Press `Ctrl-c` on your keyboard
   - Run another non-nestable function

   When you leave the function, the incomplete nets are deselected but remain highlighted in the canvas.

Showing Incomplete Nets for Overlapping Objects

When you click an area of the canvas where several objects overlap, the function gets the list of objects under the cursor that have incomplete nets associated with them, hides any visible incomplete nets that are directly under the cursor, and shows the incomplete nets for the first object in the list. When you click again, the function hides the incomplete nets for the first object and shows the incomplete nets for the second object, and so on. To see all the incomplete nets associated with a group of overlapping objects, press `Shift` and click the left mouse button repeatedly until all the objects under the cursor are selected.

© 2023 Cadence Design Systems, Inc.

Important

The function hides only those incomplete nets that are directly under the cursor when you click. This lets you show incomplete nets in a different area of the design without hiding the incomplete nets currently displayed elsewhere.

Showing Incomplete Nets Using Area Selection

You can use area selection to select the objects for which you want to see incomplete nets.

Only objects that are fully enclosed by the selected area are considered. For example, if an area selection encloses an instance pin but not the instance itself, the function shows (or hides) only those incomplete nets associated with the corresponding instance terminal and excludes any other incomplete nets associated with the instance.

The same applies for objects inside a group. If the selected area fully encloses an object inside a group but not the group itself, the function shows (or hides) only the incomplete nets associated with the fully-enclosed object.

Showing Incomplete Nets in the Navigator and Search Assistants

The *Show/Hide Selected Incomplete Nets* command can be called from both, Navigator and the Search assistants. When you select an object in either assistant, you see the incomplete nets associated with the selected object. When you deselect the object, the incomplete nets are hidden again. To hide all the incomplete nets, click an empty area in the Navigator assistant.

If you select one or more incomplete net markers in the Annotation Browser assistant, you can click the right mouse button and choose *Select Nets* to select the nets in the layout canvas and Navigator assistant. You can then invoke the router on the selected nets.

Related Topics

Showing and Hiding Current Incomplete Nets

Behavior by Object Type in Postselection Mode of Incomplete Nets

Showing Incomplete Nets in Preselection Mode

Incomplete Nets Commands

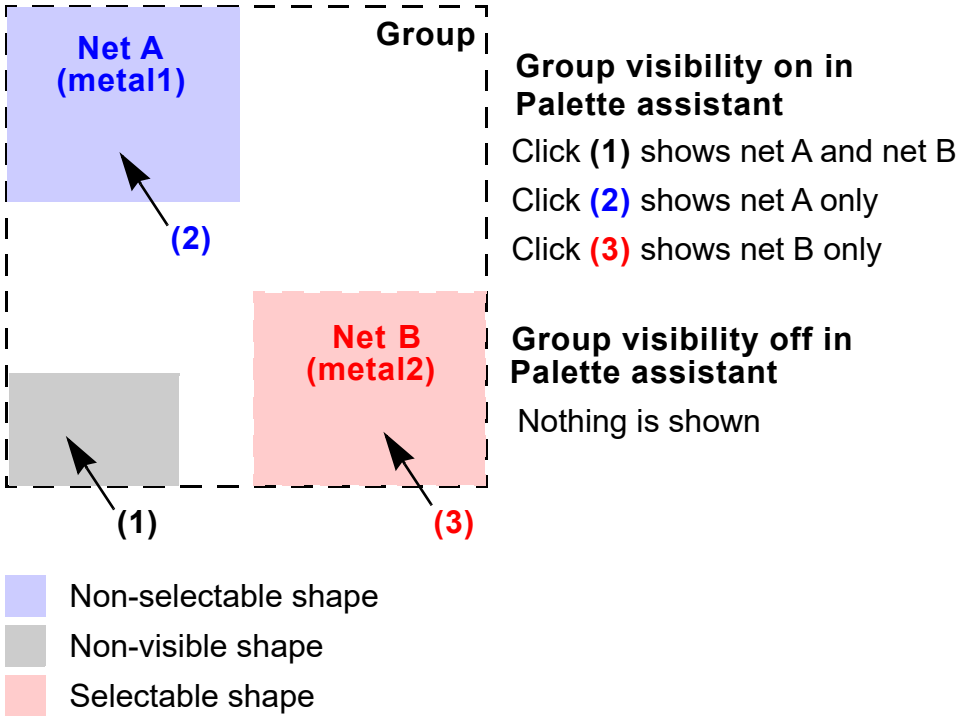
© 2023 Cadence Design Systems, Inc.

Behavior by Object Type in Postselection Mode of Incomplete Nets

The table below summarizes the behavior of the *Show/Hide Selected Incomplete Nets* command for different postselected objects.

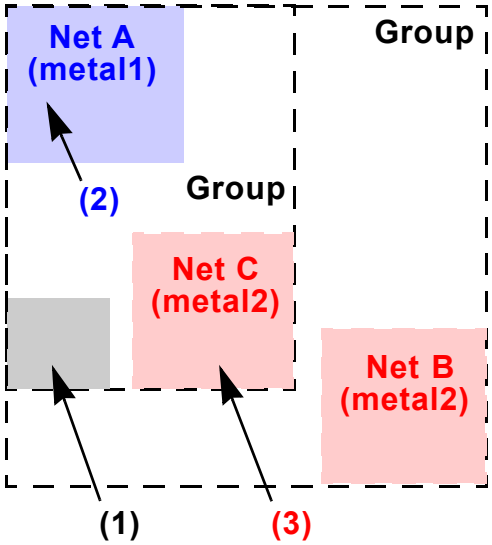
Object Type	Behavior
Net	If the net is complete, the function does nothing; otherwise it toggles the visibility of the incomplete net as follows. ■ If all the open markers on the incomplete net are visible, the function hides them all. ■ If at least one open marker on the net is hidden, the function makes all the open markers on the net visible.
Shape or Via	If the shape or via has no net associated with it or if the associated net is complete, the function does nothing; otherwise it toggles the visibility of the incomplete net as follows. ■ If all the open markers on the incomplete net are visible, the function hides them all. ■ If at least one open marker on the net is hidden, the function makes all the open markers on the net visible.
Instance	If you click a visible instance pin (with the *Display Stop Level* set to a value greater than 0 or the *Instance Pins* button turned on in the Display Options form), the function works on the top net connected to the corresponding instance terminal. If there is no visible instance pin, the function works on all the top nets associated with the instance. When the function grabs a net associated with an instance pin, if the net is complete, the function does nothing. It does not look at the other nets associated with the instance.

© 2023 Cadence Design Systems, Inc.

Object Type	Behavior
Group	When the group visibility is turned on in the Palette assistant, if you click a group at a location where there is no object, the function shows (or hides) all the incomplete nets attached to all the objects inside the group **(1)**. When you click a specific object inside a group, the function shows (or hides) only the nets that are attached to the selected object and **(2)** and **(3)**. This lets you work on a specific object contained in a group without descending into the group to select it. 

© 2023 Cadence Design Systems, Inc.

Object Type	Behavior
Hierarchical Group	The function grabs objects regardless of the level they are at in the group hierarchy. All the objects visible in a group are considered to be at the top level.



Group visibility on in Palette assistant

Click **(1)** shows nets A, B and C

Click **(2)** shows net A only

Click **(3)** shows net C only

Group visibility off in Palette assistant

Nothing is shown

Non-selectable shape

Non-visible shape

Selectable shape

Marker	A marker object cannot have any nets associated with it. The function gets all the incomplete nets attached to all the objects associated with the marker and then toggles the visibility of these nets as follows. ■ If all the open markers on all the incomplete nets are visible, the function hides them all. ■ If at least one open marker on one net is hidden, the function makes all the open markers on all the nets visible.

Related Topics

Behavior by Object Type in Postselection Mode of Incomplete Nets

Incomplete Nets Commands

© 2023 Cadence Design Systems, Inc.

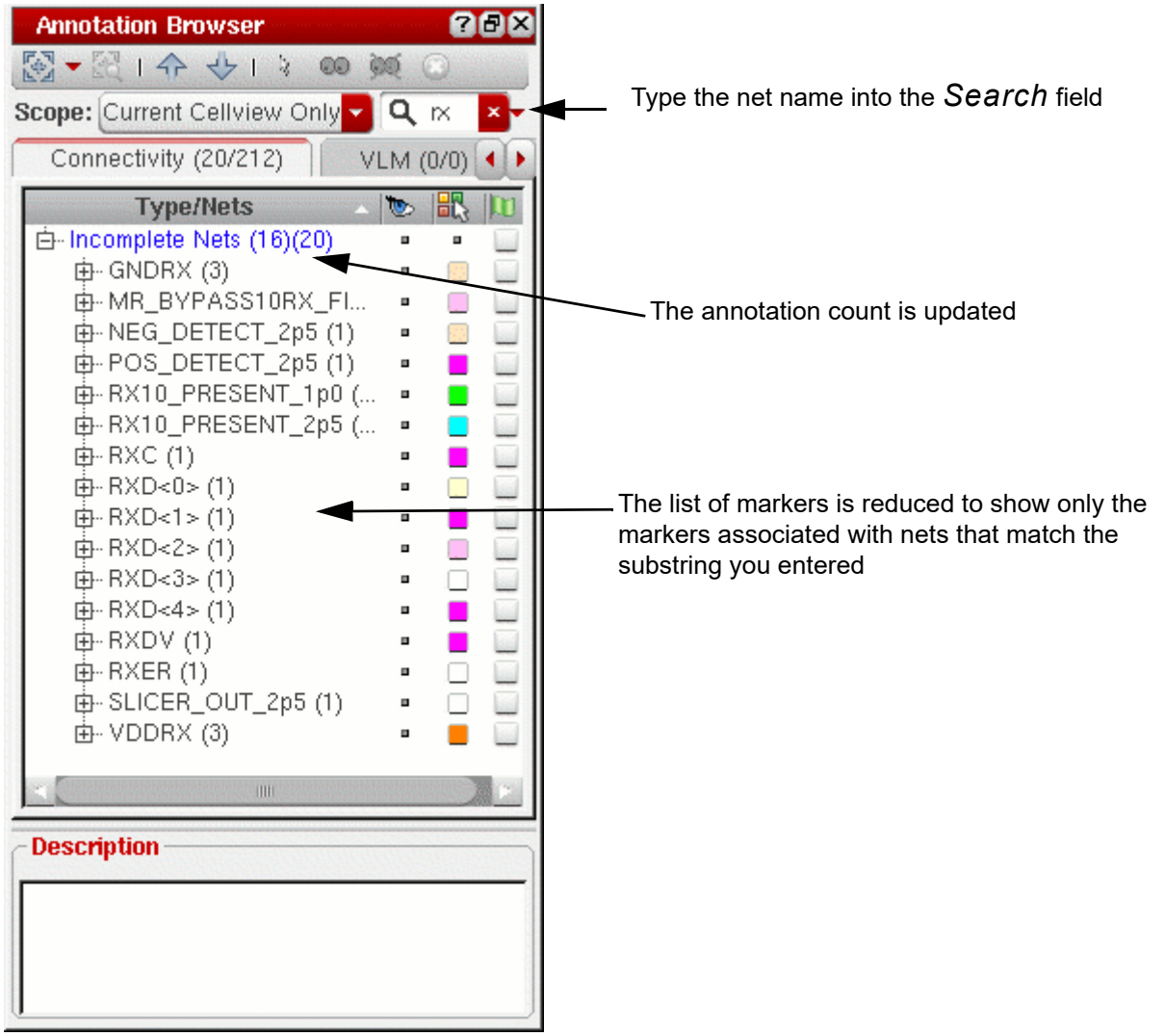
Searching for Specific Incomplete Nets

You can use the Annotation Browser *Search* function to locate all the incomplete net markers associated with a particular instance or net in the design. When you type in the name of a net, the browser updates automatically to show only those incomplete nets that feature that name either in the labels in the browser pane or in the messages shown in the *Description* field.

To search for all the incomplete net markers associated with a particular net,

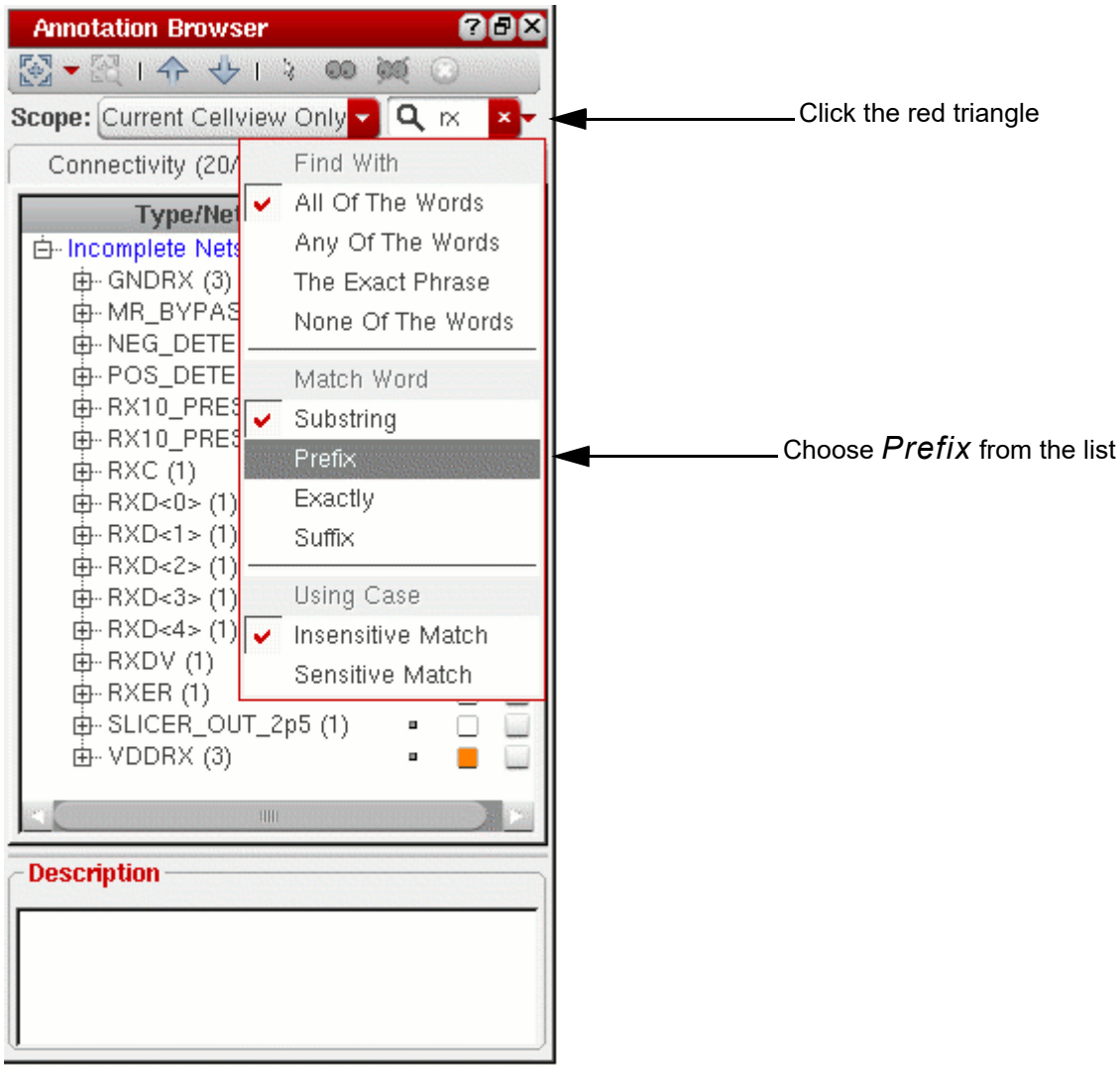
1. Type a string into the *Search* field; for example, `rx`.

   By default the search function uses the text you enter as a case-insensitive substring. See Annotation Browser Assistant.



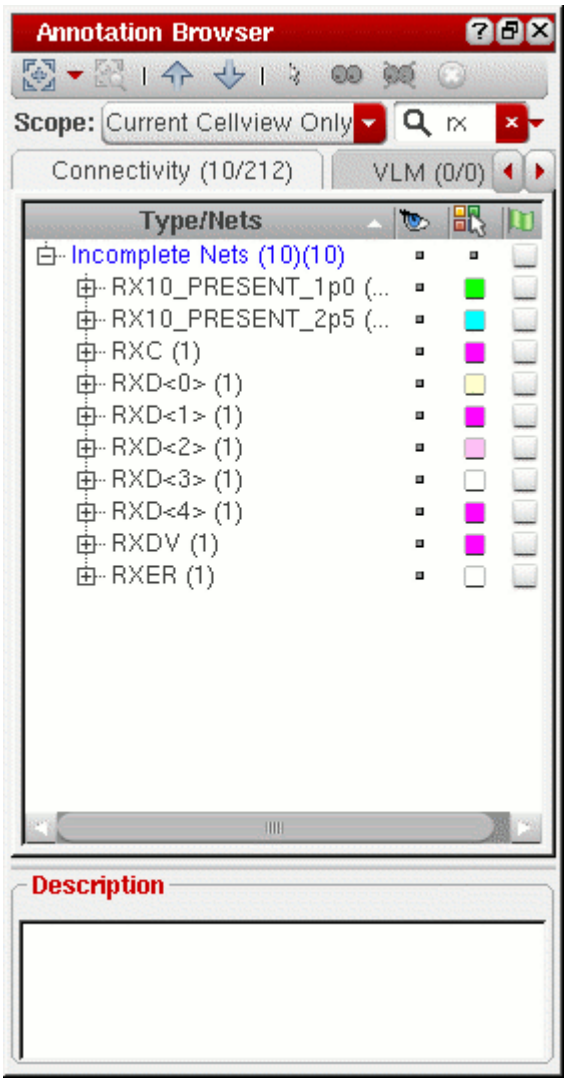
© 2023 Cadence Design Systems, Inc.

2. To search for a prefix, click the small red triangle next to the *Search* field and choose *Prefix* from the drop-down list.



© 2023 Cadence Design Systems, Inc.

The list of markers and the annotation count are both updated accordingly.



3. To search a for suffix, click the small red triangle and choose *Suffix*.

   You can also filter the incomplete net markers shown in the browser based on the set of currently selected nets in the layout canvas or Navigator assistant.

Related Topics

Setting an Incomplete Net Filter

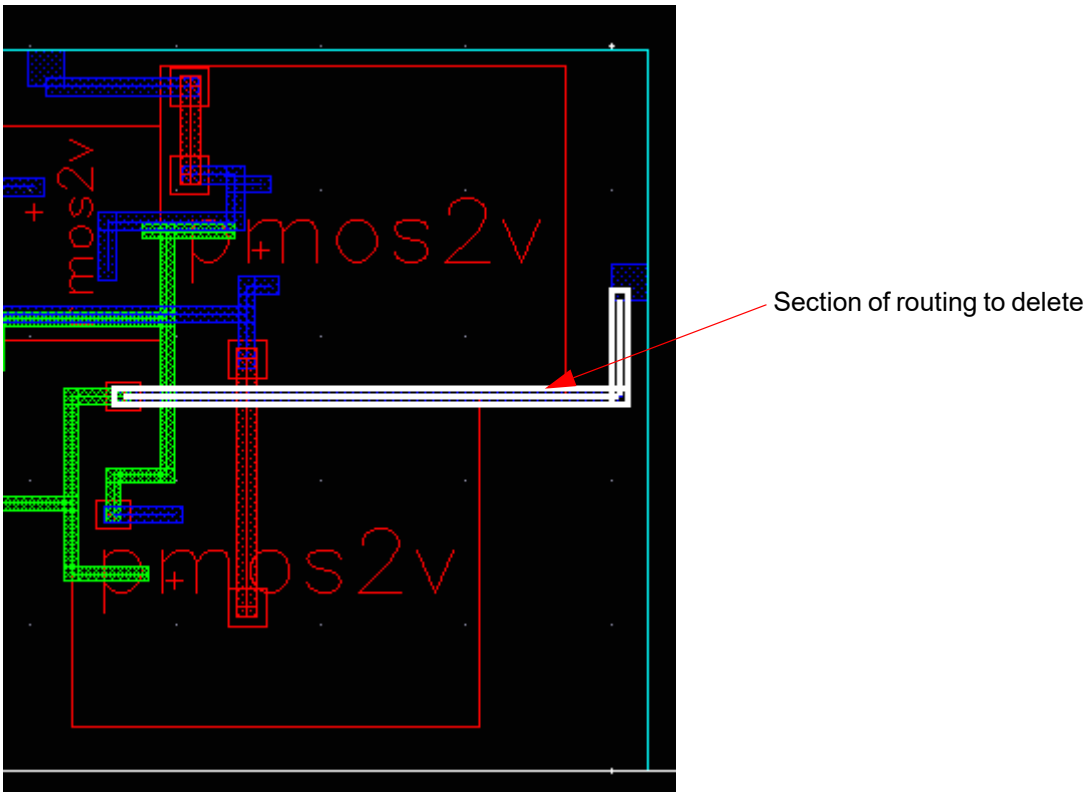
Incomplete Nets Commands

© 2023 Cadence Design Systems, Inc.

Showing Newly Created Incomplete Nets Automatically

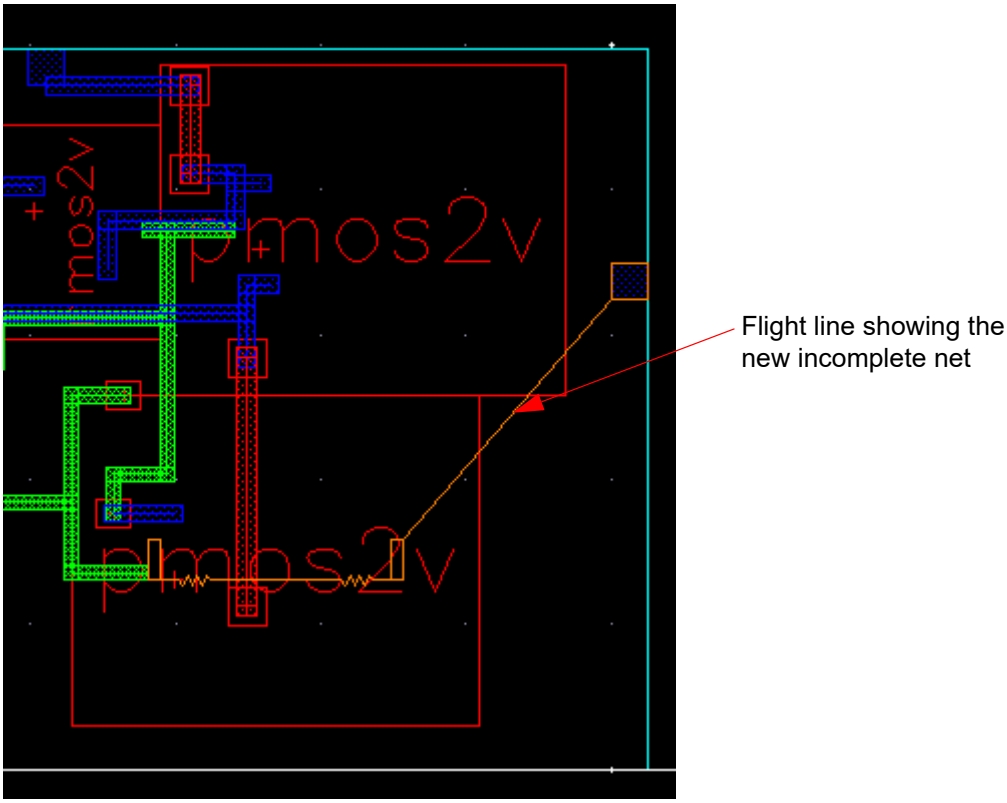
Layout XL can automatically show flight lines representing new incomplete nets you create during interactive editing. To do this,

1. In a layout window, delete a piece of routing material on a complete net.



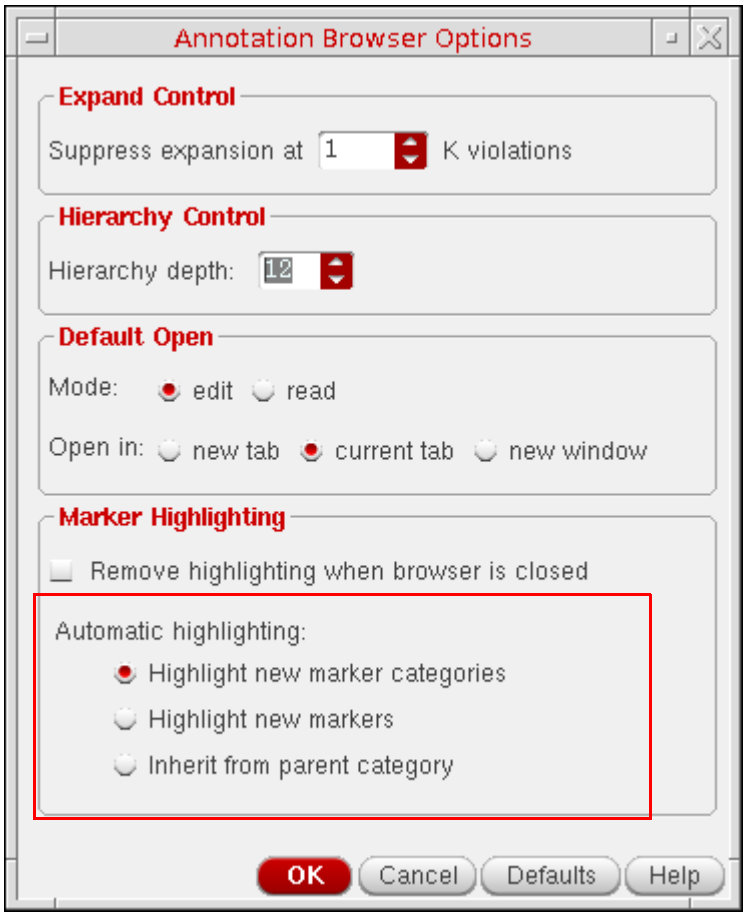
© 2023 Cadence Design Systems, Inc.

The system automatically shows a flight line representing the new incomplete net you created.



© 2023 Cadence Design Systems, Inc.

You control this behavior using the *Automatic Highlighting* options in the Annotation Browser Options form.



By default, the system highlights markers only when a new marker category is created, for example a new incomplete net.

To automatically highlight a new open marker created on an existing incomplete net, set automatic highlighting to *Highlight new markers*.

To give a newly-created marker the same visibility as its parent category, set automatic highlighting to *Inherit from parent category*.

Related Topics

The Navigator Assistant

Incomplete Nets Commands

© 2023 Cadence Design Systems, Inc.

Assigning Colors to Incomplete Nets

Layout XL uses the colors you assigned to the `y0` through `y9` entry layers when displaying flight lines in the design window. If there are more than ten incomplete nets in the design, the colors are recycled.

To assign a color other than the default to a flight line,

1. In the Annotation Browser assistant, click the Set Highlight Color column next to the entry for the net whose color you want to change.
2. Do one of the following.
   - Choose the color you want for that flight line from the predefined list.
   - Choose cycle to automatically select the next color in the predefined list.

   The flight line changes to the color you specified.

To assign colors to all the flight lines under a particular node,

1. In the Annotation Browser assistant, click the *Set Highlight Color* column for the parent node of the markers you want to change.
2. Do one of the following.
   - To show all the markers contained in the node in the same color, choose the color you want from the drop-down list.
   - To show each of the markers contained in the node in a different color, choose *cycle*. The flight lines are assigned colors automatically based on the predefined list.

   The flight lines change to the color you specified.

Related Topics

Annotation Browser Assistant

Zooming In on Incomplete Nets

To zoom in on the flight lines showing selected incomplete nets,

1. In the Annotation Browser assistant, select the nets you are interested in.

© 2023 Cadence Design Systems, Inc.

2. Click the *Zoom To Selected* button in the Annotation Browser toolbar.



The layout window display zooms to the flight lines for the selected nets. The level of detail you see depends on how many nets are selected and their positions in the design.

To automatically zoom in on the flight line showing the currently selected incomplete net,

➤ Click the Auto Zoom button in the Annotation Browser assistant toolbar.



The display automatically zooms and pans to the marker representing the currently selected entry in the Annotation Browser. When the selection changes, the display updates accordingly.

Related Topics

Annotation Browser Assistant

Editing In Place

Use the Edit In Place command to edit a placed instance in your design and maintain the connectivity. You can also use the *Edit – Hierarchy – Descend* command to descend into an instance.

When you choose *Edit In Place* (or *Descend*), by default, Layout XL opens the connectivity reference for the cell into which you are descending. To prevent this, switch off the *Open connectivity reference* option on the General of the Connectivity form. When switched off, the layout cellview is still opened in Layout XL mode, but with no connectivity reference.



When you descend into a Pcell it opens in Layout XL in a special mode with the extractor and binder disabled. Care is to be taken to ensure that any edits performed do not cause the Pcell evaluation to fail outside of the Virtuoso Layout Suite environment. For more details, see Safety Rules for Creating SKILL Pcells***.***

To edit a placed instance in your design,

© 2023 Cadence Design Systems, Inc.

1. Choose *Edit – Hierarchy – Edit In Place*.
2. Click the instance whose cell you want to edit.

   The banner at the top of the window changes to show that you are now editing the master cell for that instance. Notice the instance is highlighted along the border.

   Instances with placement status `unplaced` are not visible in the layout canvas, and so do not contribute to the floorplanning estimations. When one or more such instances are found in a design, a dialog box indicating their presence is displayed, prompting whether these instances should be made visible. If you click *Yes*, the placement status of these instances is changed to `none`, making them visible in the layout canvas. If you click *No*, the dialog box is closed without making any changes.
3. When you have finished editing, choose *Edit – Hierarchy – Return* to stop editing in place.

   Layout XL returns you to the next level of the hierarchy.

   To specify the level of hierarchy to return to, choose *Edit – Hierarchy – Return To Level*.

Related Topics

Edit In Place

Safety Rules for Creating SKILL Pcells

Connectivity Form

© 2023 Cadence Design Systems, Inc.

Soft Block Editing

The *Edit Soft Blocks* command lets you modify the soft blocks in your design. Use this command if you need to modify the attributes of individual soft blocks to take account of placement and connectivity requirements in your design.

Important

This command is recommended if you want to make changes that do not require the physical hierarchy to be regenerated. If you need to make more radical changes (by, for example, generating new soft blocks or flattening others that you no longer require), use the *Launch – Configure Physical Hierarchy* command in Soft Block mode.

Using the *Edit Soft Blocks* command, you can modify

- The type of soft block that will be created.
- The height, width, area, or shape of the place and route boundary.
- The length, width, layer, and signal type of soft pins.
- Routing and placement obstructions inside the block.

A number of Virtuoso Layout Suite commands have been enhanced to support Level-1 Editing mode, which is the ability to edit objects inside a soft block without using the *Descend* or *Edit In Place* commands. This section further describes how to enable Level-1 Editing mode and how to use the enhanced *Move*, *Stretch*, *Reshape*, and *Chop* commands.

Related Topics

Configure Physical Hierarchy Window

Configuring the Physical Hierarchy video

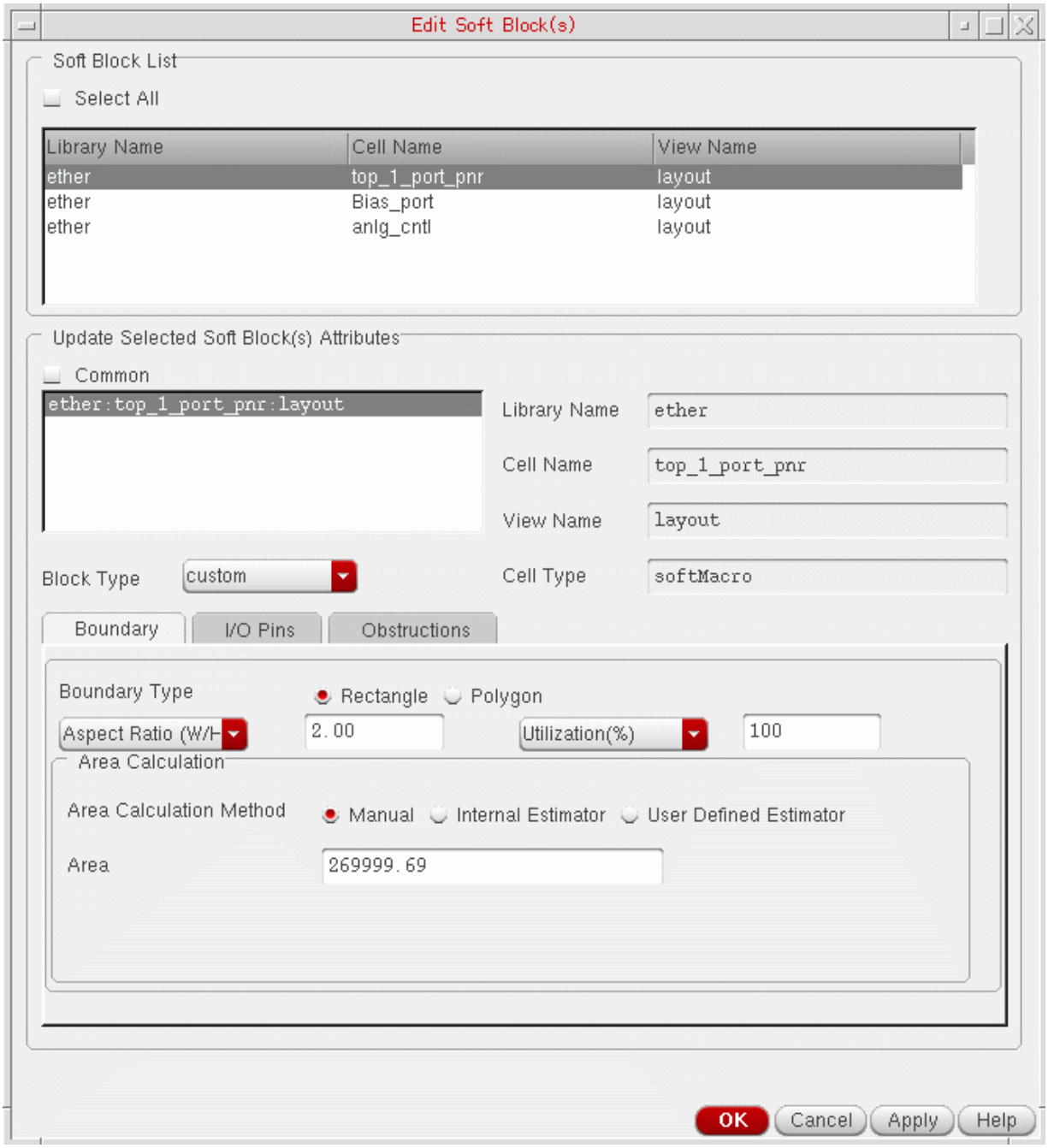
Changing the Block Type

To change the type of a soft block:

1. From the layout window menu bar, choose *Edit – Soft Blocks*.

© 2023 Cadence Design Systems, Inc.

The Edit Soft Blocks form is displayed.



2. In the *Soft Block List*, select the blocks whose type you want to change.

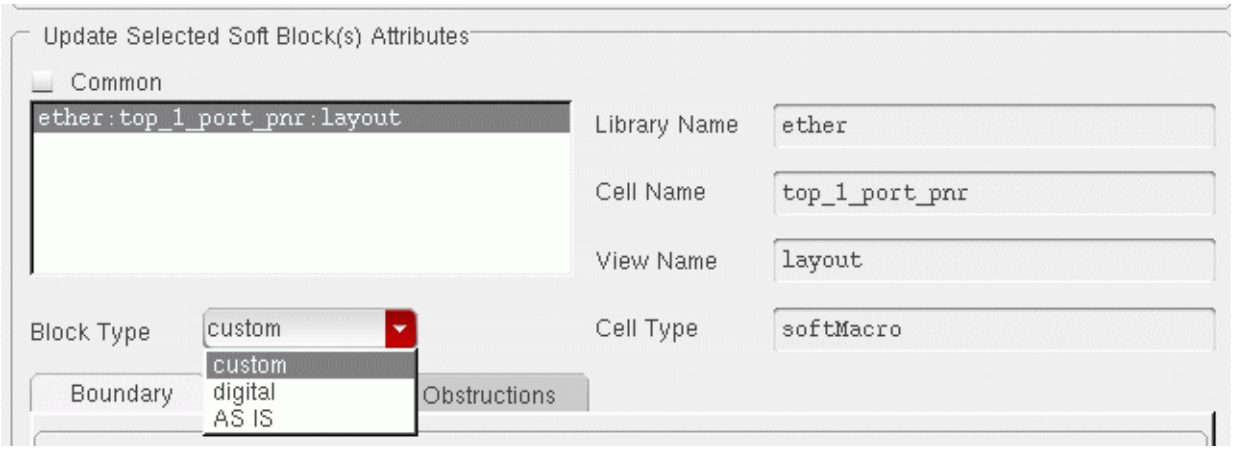
   The selected blocks are made available in the *Update Selected Soft Block Attributes* list.

3. Select the blocks whose type you want to change.

© 2023 Cadence Design Systems, Inc.

If you want to change the type for all the listed blocks, check the *Common* box.

4. Choose the *Block Type* you need from the cyclic field.



5. Click *OK* or *Apply* to accept the change.

Related Topics

Edit Soft Blocks Form

Soft Block Mode

Configure Physical Hierarchy Window

Configuring the Physical Hierarchy video

Changing the Boundary of Soft Blocks

You can change any fixed or non-fixed rectangular boundary or any polygonal boundary by adjusting the parameters that are defined in the Edit Soft Blocks form.

This topic covers:

- Changing a Rectangular Boundary in Fixed Mode
- Changing a Rectangular Boundary of a Soft Block in Non-Fixed Mode
- Changing a Polygonal Boundary

© 2023 Cadence Design Systems, Inc.

Changing a Rectangular Boundary in Fixed Mode

Fixed mode refers to a boundary defined by specifying a combination of width, height, and aspect ratio for the block in question. The boundary is considered fixed because it is defined by two fixed variables, and is not derived by estimating the area of the block.

To change a rectangular boundary in fixed mode,

1. From the layout window menu bar, choose *Edit – Soft Blocks*.

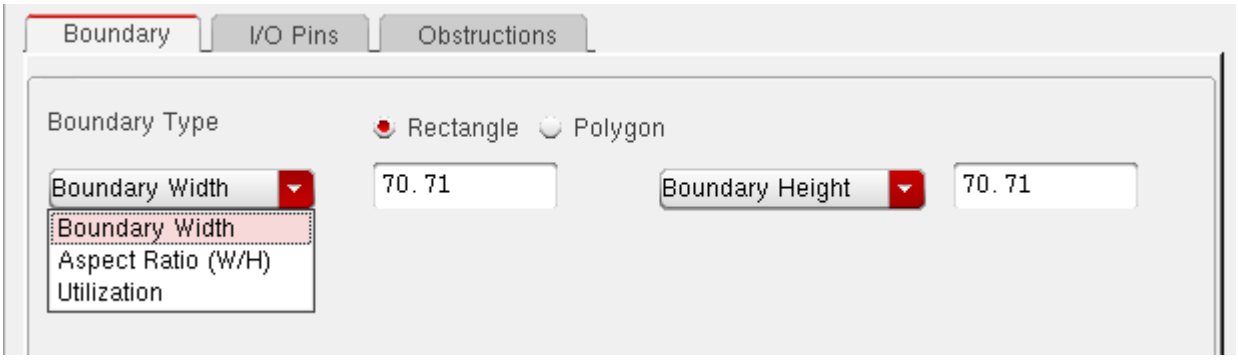
   The Edit Soft Blocks form is displayed.

2. Select the blocks whose boundary you want to change.
3. In the *Boundary* tab, check the *Rectangle* radio button.
4. Choose one of the following.

   - *Aspect Ratio (W/H)* in the first cyclic field and *Height* in the second. In this case, the boundary width is calculated from the boundary height and aspect ratio.
   - *Aspect Ratio (W/H)* in the first cyclic field and *Width* in the second. In this case, the boundary height is calculated from the boundary width and the aspect ratio.
   - *Width* in the first cyclic field and *Height* in the second. In this case, these values you type in are used to create the boundary.

   When you choose any of the above combinations, the form changes to hide the other fields.

   The picture below show a rectangular boundary specified using the width and height options.



© 2023 Cadence Design Systems, Inc.

Changing a Rectangular Boundary of a Soft Block in Non-Fixed Mode

Non-fixed mode refers to a boundary defined by estimating the area required by the block in question. The boundary is non-fixed because it is derived from only one fixed parameter (such as the height or width), an area utilization factor, and an area estimation function.

To change a rectangular boundary in non-fixed mode,

1. From the layout window menu bar, choose *Edit – Soft Blocks*.

   The Edit Soft Blocks form is displayed.

2. Select the blocks whose boundary you want to change.
3. In the *Boundary* tab, check the *Rectangle* radio button.
4. Use the two cyclic lists to specify the utilization factor and one fixed parameter for the block. The table below lists the valid combinations in non-fixed mode.

Utilization	*Height*
Utilization	*Width*
Utilization	*Aspect ratio (W/H)*

5. Choose an *Area Calculation Method*.

   The area calculation methods available depend on the cell type of the soft blocks.

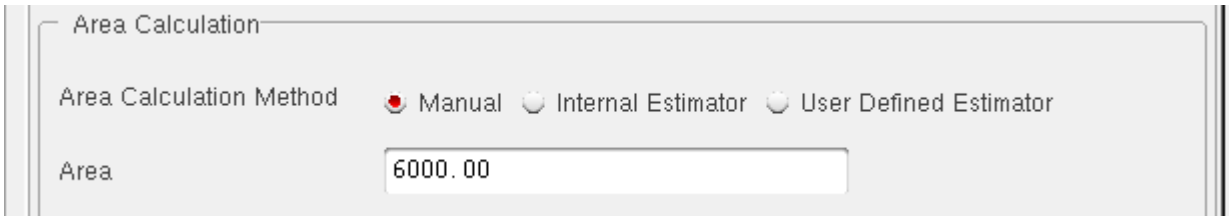
   - If the block is of type *softMacro*, you can use the *Manual*, *Internal Estimator*, or *User Defined Estimator* methods.
   - If the block is of type *blockBlackBox*, you can use the *Manual*, *Area Per Gate*, or *User Defined Estimator* methods.
     - Manual Area Estimation
     - Internal Estimators
     - User Defined Estimators
     - Area Per Gate Estimation

6. When the area has been estimated, click *OK* or *Apply* to accept the changes to the boundary.

© 2023 Cadence Design Systems, Inc.

Manual Area Estimation

Use *Manual* area estimation if you know the block area. Type the area into the *Area* field.



The boundary of the soft block modified based on the value specified in the second cyclic field.

For example, consider a block where the *Utilization* is specified as 1 and the *Boundary Height* is 63 units. If you increase the area from 1000 to 2000 units, the width of the place and route boundary increases because the value of height is fixed at 63.

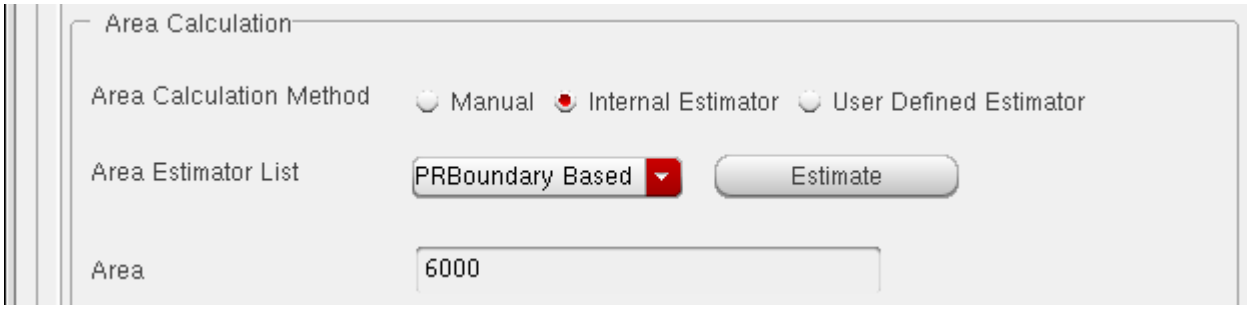
Similarly, if the value of the *Boundary Width* field is specified, the height of the block would be modified to arrive at the specified area.

Internal Estimators

Use *Internal Estimator* to have the system calculate the area of a soft block of type *softMacro* based on

- The sum of all the place and route boundaries of the objects inside the soft block
- The sum of all the bounding boxes of the components contained in the soft block.

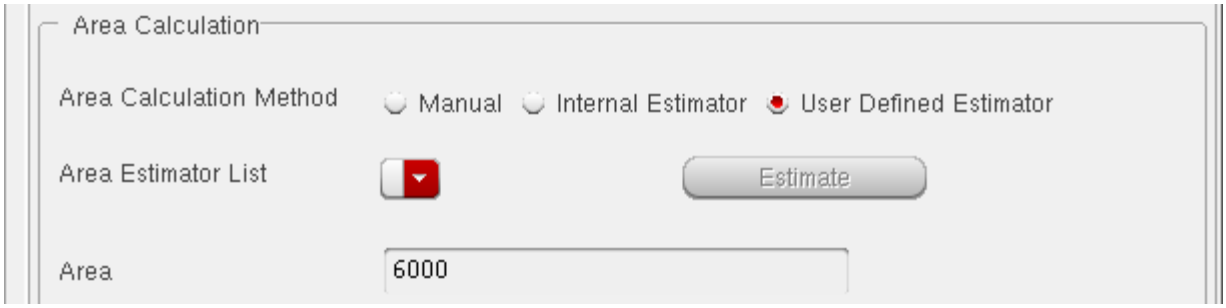
Choose either *PRBoundary Based* or *BBox Based* from the cyclic field and click the *Estimate* button to calculate the area.



© 2023 Cadence Design Systems, Inc.

User Defined Estimators

User Defined Estimator lets you specify your own area calculation function using the framework described in Creating and Registering a User-Defined Area Estimation Function. (The picture below shows no user-defined function registered.)

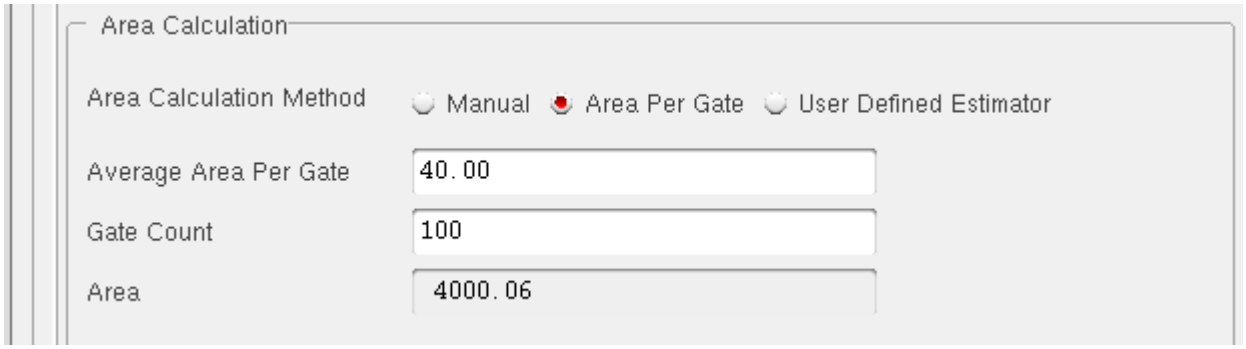


You can view all the registered area estimators in the *Area Estimator List. Select the* estimators you want to use and click the *Estimate* button to calculate the area.

Area Per Gate Estimation

Use *Area Per Gate* to calculate the area for a soft block of type *blockBlackBox*. You enter the area per gate and the gate count and the system calculates the area based on the following formula.

```
Area = Area Per Gate * Gate Count
```



The *Area* field is grayed out in this mode.

Changing a Polygonal Boundary

You can create or change a polygonal place and route boundary by specifying the points of each of the vertices of the polygon.

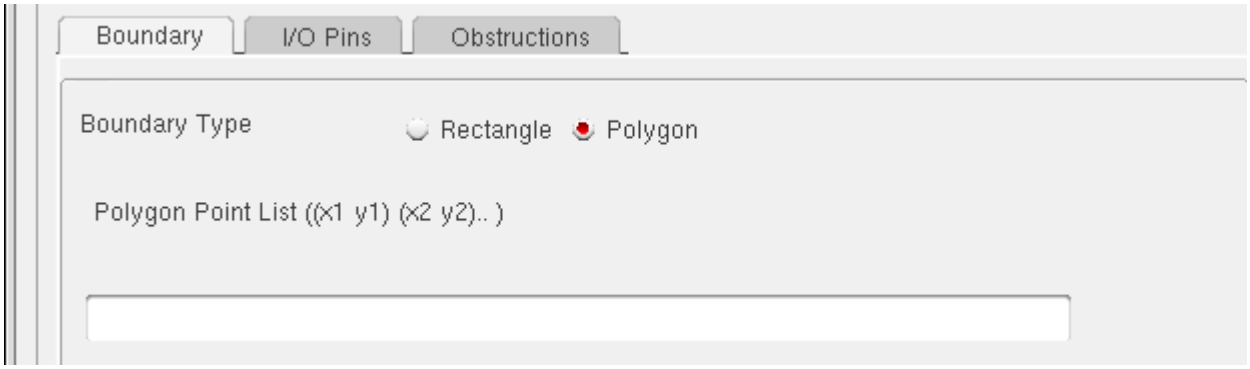
© 2023 Cadence Design Systems, Inc.

To change a polygonal boundary,

1. From the layout window menu bar, choose *Edit – Soft Blocks*.

   The Edit Soft Blocks form is displayed.
2. Select the blocks whose boundary you want to change.
3. In the *Boundary* tab, check the *Rectangle* radio button.

   The form changes as shown in the figure below.



4. Specify the points for each of the vertices of the polygon in the *Polygonal Point List* text box.
5. Click *OK* or *Apply* to change the boundary.

Related Topics

Edit Soft Blocks Form

Soft Block Mode

Configure Physical Hierarchy Window

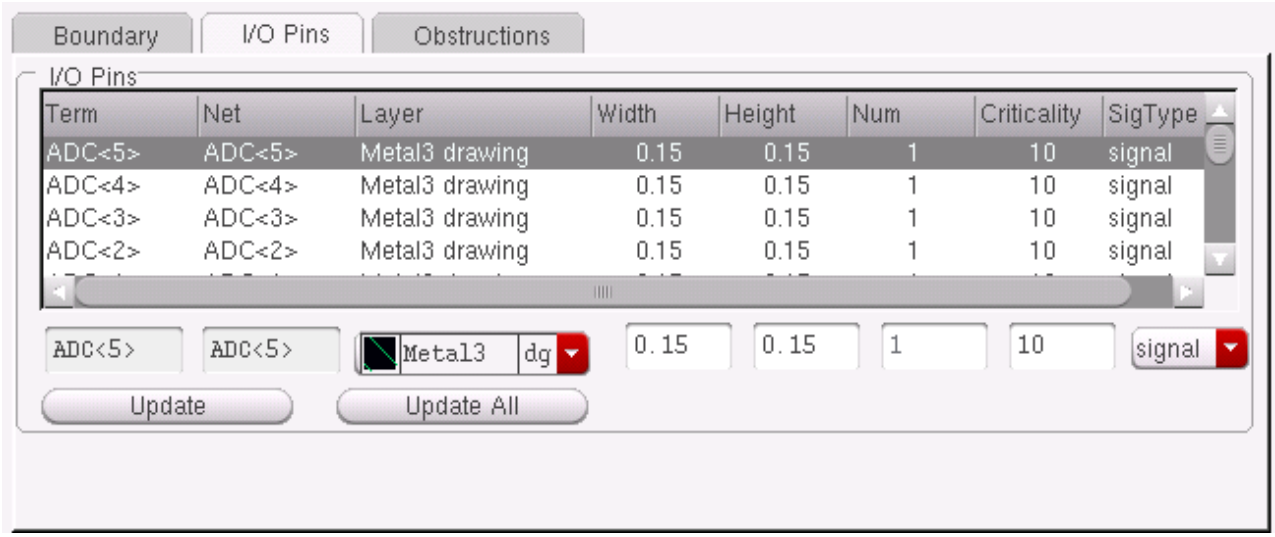
Configuring the Physical Hierarchy video

Changing I/O Pin Attributes

The *I/O Pins* tab lets you update the layer, width, height, number, criticality, and signal type of existing pins. This table is initialized with *Term* and *Net Names* from the symbol view of the selected soft block.

© 2023 Cadence Design Systems, Inc.

When you click the *I/O Pins* tab, the form changes to display the *I/O Pins* table as shown below.



You can select one or more rows to modify in the table. If you select multiple rows, the fields below the table list the respective values only if they are same for the selected rows. Otherwise, the fields display *AS IS*.

To add or change I/O pin parameters,

1. From the layout window menu bar, choose *Edit – Soft Blocks*.

   The Edit Soft Blocks form is displayed.

2. Select the blocks whose pins you want to change.
3. Select the entry in the table that you want to update.
4. Specify the new value in the field below the column that you want to update.

   For example, if you want to update the layer purpose for a net to *metal1 drawing*, click the cyclic list below the *Layer* column and select *metal1 dg*.

5. Do one of the following.

   - Click *Update* to update the selected entries in the table.
   - Click *Update All* to update all the entries in the table.

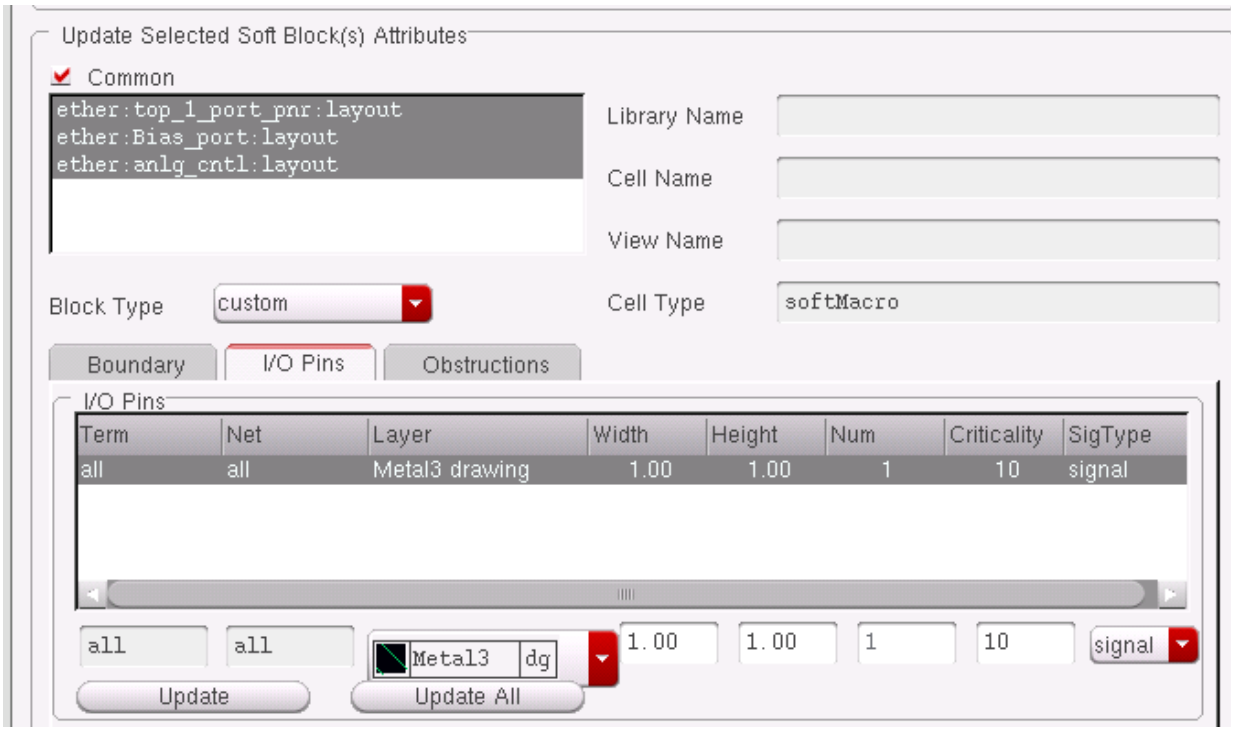
You can also update multiple soft blocks in the design simultaneously.

To update multiple soft blocks,

© 2023 Cadence Design Systems, Inc.

1. Select multiple rows in the *Soft Block List* or select the *Common* button in the *Update Selected Soft Block(s) Attributes* section.

   The *I/O Pins* tab changes as shown below.



2. Change the parameters as required and click *Update*.

   All the pins in all the selected blocks are updated with the new values.

Related Topics

Edit Soft Blocks Form

Soft Block Mode

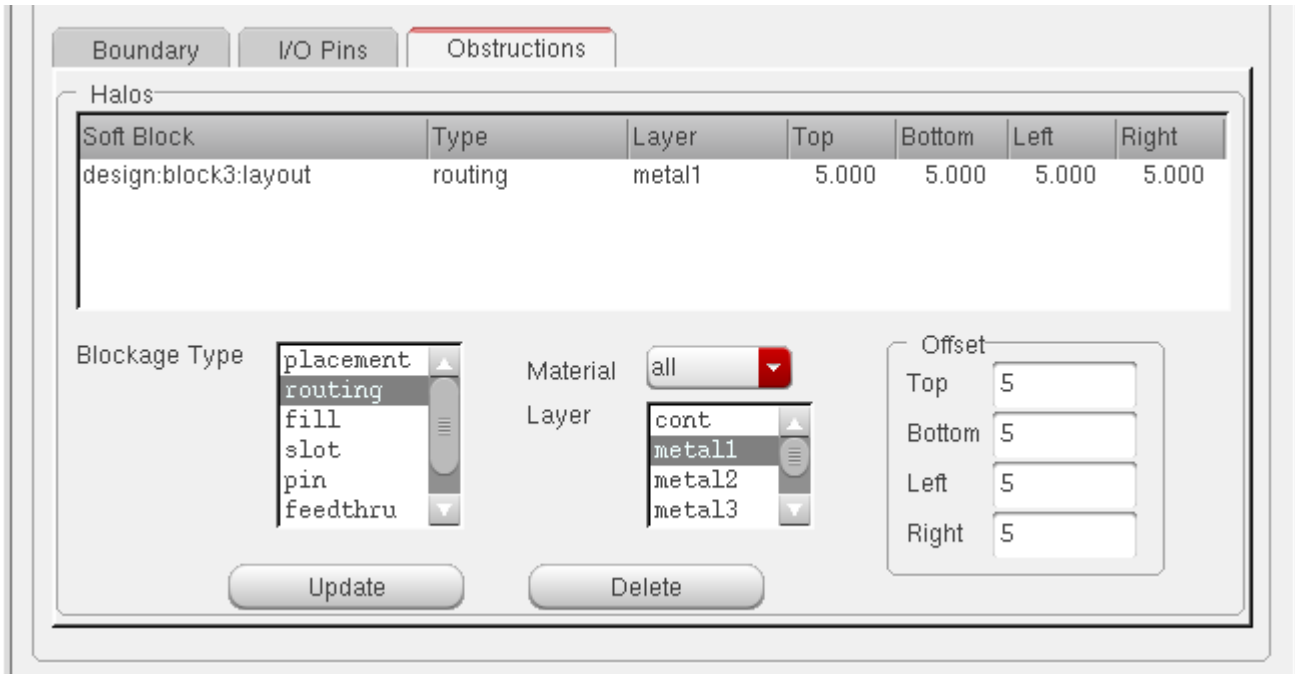
Configure Physical Hierarchy Window

Configuring the Physical Hierarchy video

© 2023 Cadence Design Systems, Inc.

Changing Obstructions

The *Obstructions* tab lets you add or change routing and placement obstructions implemented using layer and placement halos.



- To define a *routing*, *fill*, *slot*, *pin*, *feedthru*, or *screen* obstruction, you need to set the *Blockage Type*, the *Layer* and *Offsets*. There is also a *Material* filter which you can use to select layers based on the function defined in the technology file.

  The list of extractable layers is retrieved from the constraint group set for the current layout cellview. If there is no constraint group set for the current layout cellview, the software retrieves the layers from the constraint group specified by the `setupConstraintGroup` environment variable. If there are no layers defined in either location, it issues a message and disables soft block definition.

- To define a *placement* obstruction, you need only set the *Offsets*. You can define only one placement obstruction per block.

This topic covers:

- Adding an Obstruction
- Updating an Existing Obstruction
- Updating an Existing Obstruction
- Deleting an Obstruction

© 2023 Cadence Design Systems, Inc.

- Adding Obstructions on Multiple Blocks
- Moving a Place and Route Boundary at Level 1

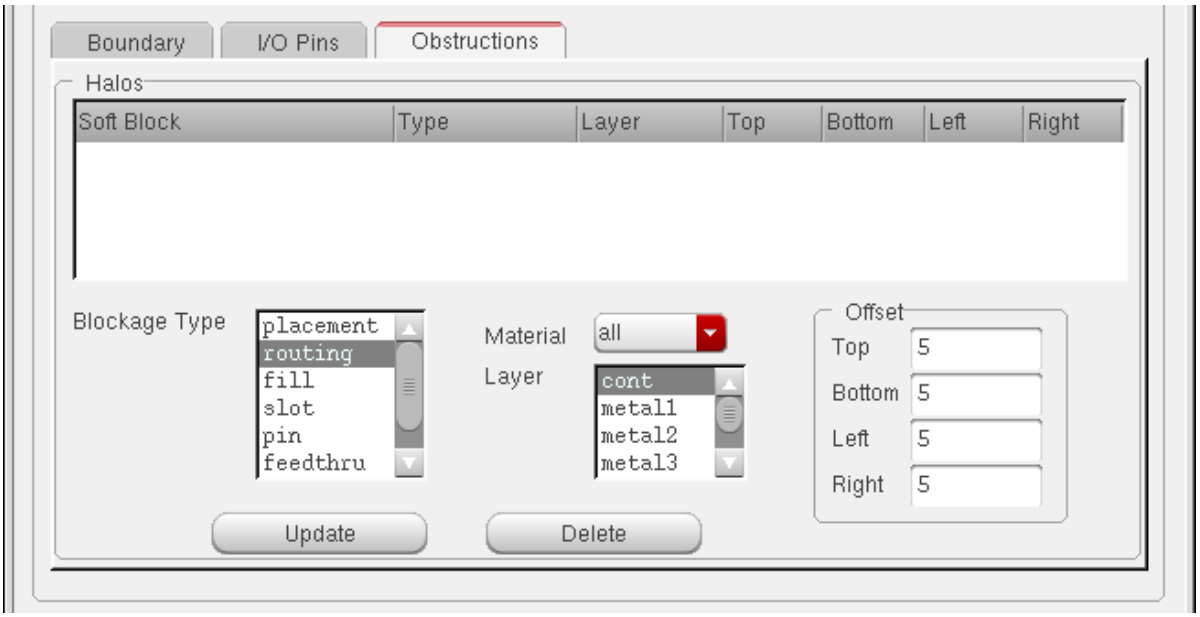
Adding an Obstruction

To add an obstruction,

1. From the layout window menu bar, choose *Edit – Soft Blocks*.

   The Edit Soft Blocks form is displayed.
2. Select the block in which you want to create the halo.
3. Select the type of halo from the *Blockage Type* list.

   The form changes as shown below.

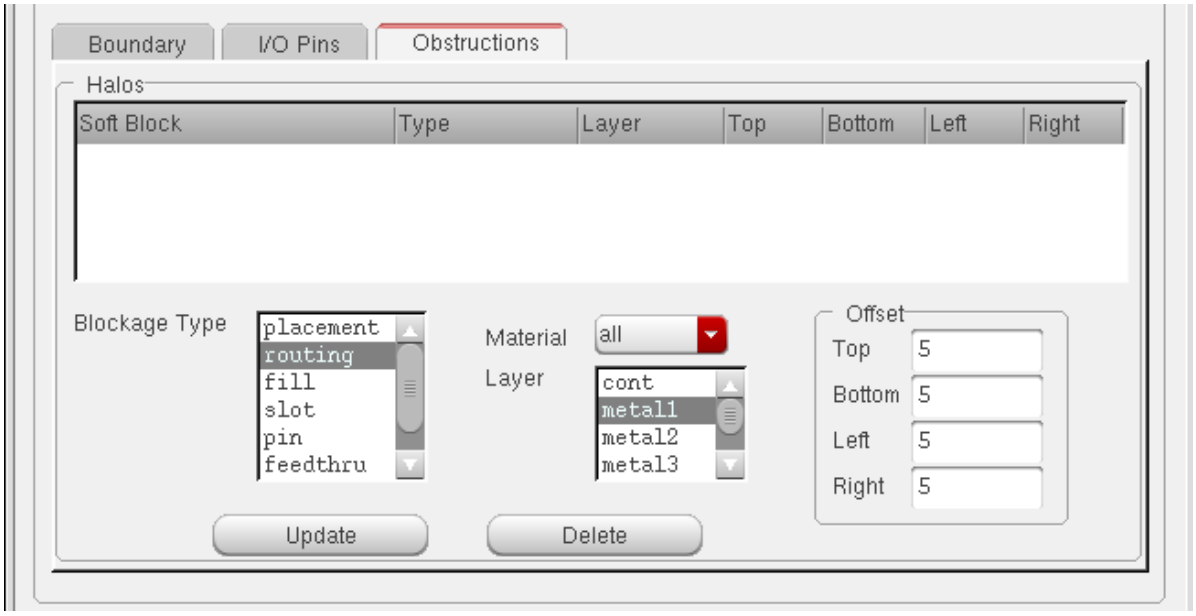


4. Select the *Layer* on which the obstruction is to be created.

   If required, you can use the *Material* cyclic list to filter the layers in the *Layer* field.

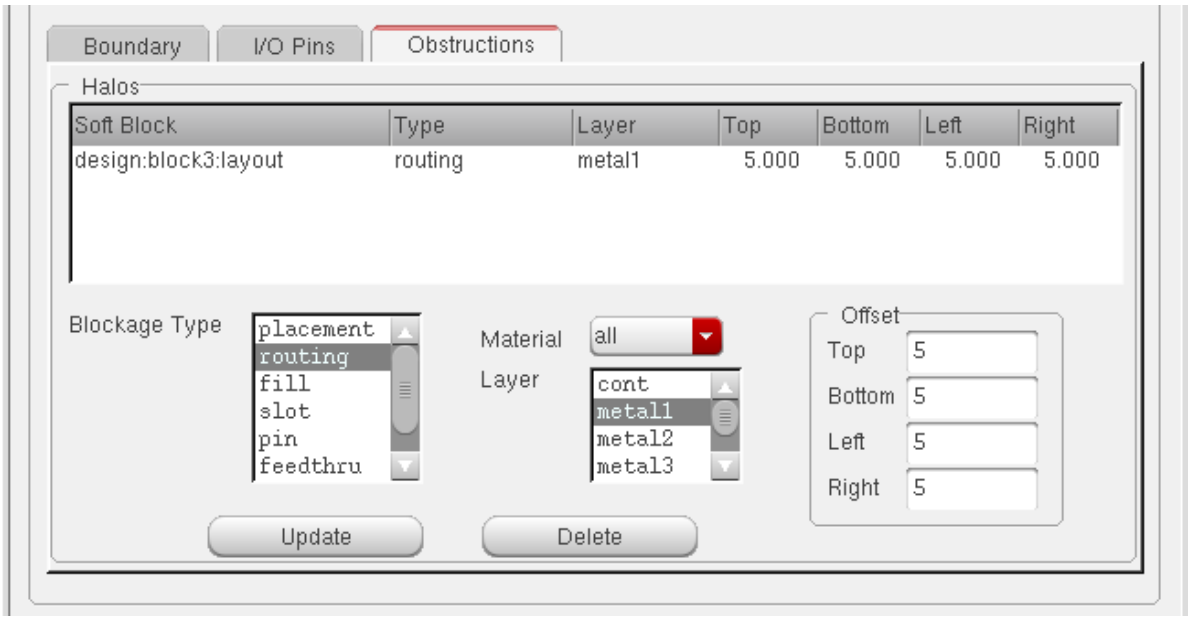
© 2023 Cadence Design Systems, Inc.

5. Specify the offsets in the *Top*, *Left*, *Right*, and *Bottom* fields.



6. Click *Update*.

   The halo is displayed in the *Halos* table.



If you create a *placement* halo, you do not need to select the *Layer* field.

© 2023 Cadence Design Systems, Inc.

Updating an Existing Obstruction

To update an existing halo,

1. From the layout window menu bar, choose *Edit – Soft Blocks*.

   The Edit Soft Blocks form is displayed.
2. Select the block which contains the halo to be updated.
3. Select the halo you want to update in the *Halos* table.
4. Modify any or all of the following.
   - Halo type in the *Blockage Type* field
   - Layer in the *Layer* field
   - The top, left, right, or bottom offsets
5. Click *Update*.
6. The updated halo is listed in the *Halos* table.

Deleting an Obstruction

To delete a halo,

1. From the layout window menu bar, choose *Edit – Soft Blocks*.

   The Edit Soft Blocks form is displayed.
2. Select the block which contains the halo to be updated.
3. Select the halo you want to delete in the *Halos* table.
4. Click *Delete*.

   The halo is deleted and removed from the *Halos* table.

Adding Obstructions on Multiple Blocks

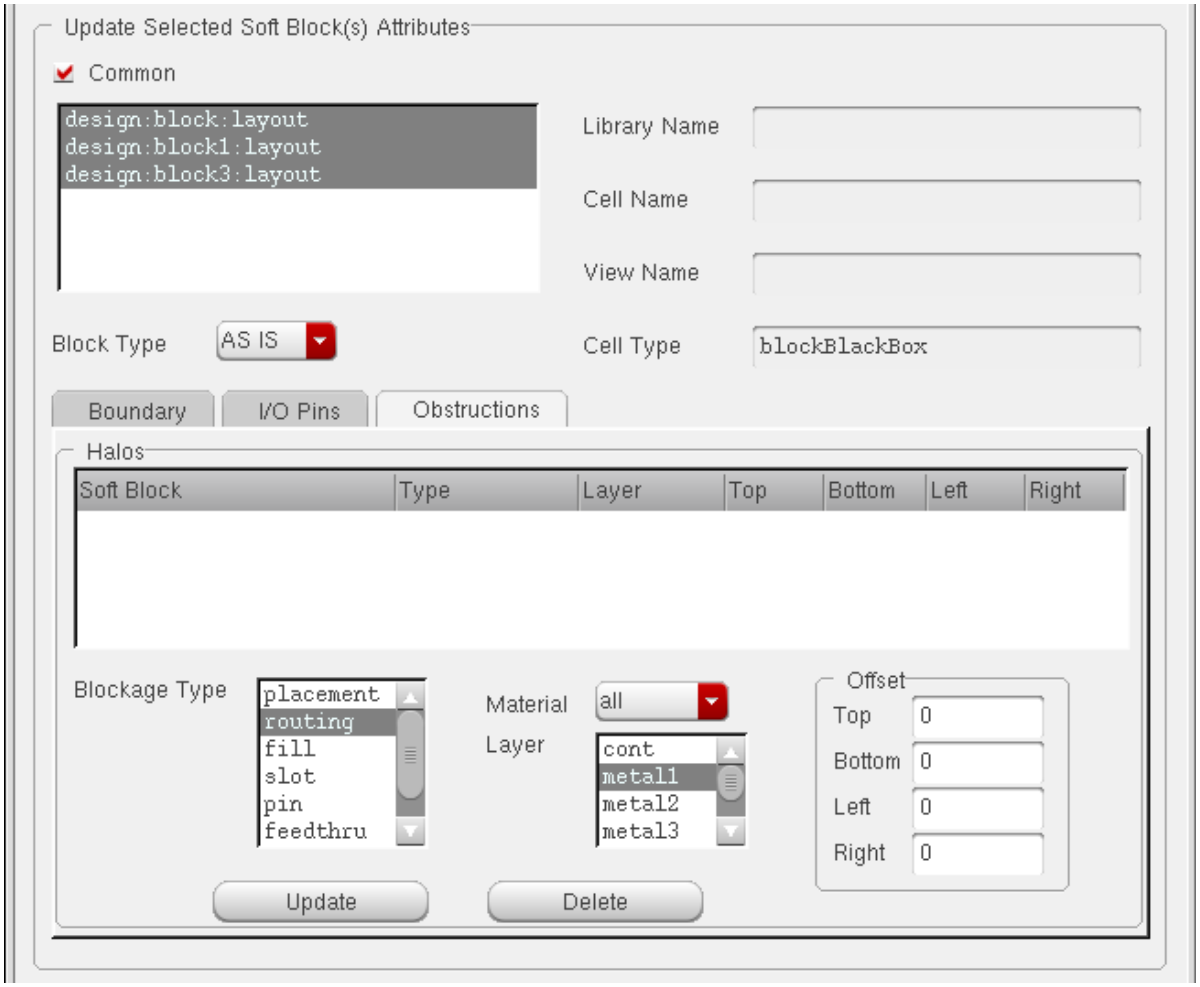
You can select multiple blocks and add commonly defined halos to each of the blocks.

To add a halo to multiple selected blocks,

1. Select multiple rows in the *Soft Block List* or select the *Common* button in the *Update Selected Soft Block(s) Attributes* section of CPH window.

© 2023 Cadence Design Systems, Inc.

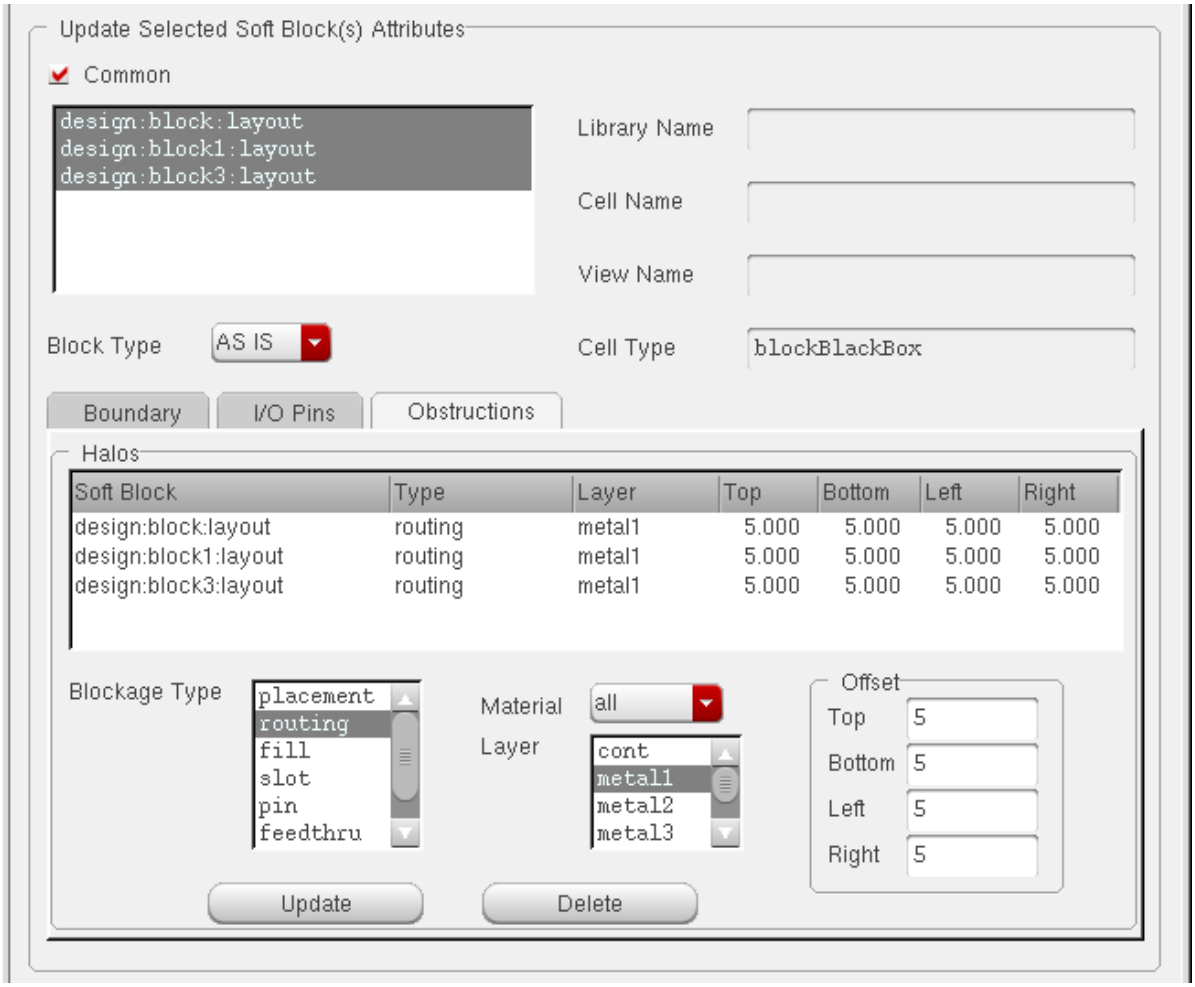
The form changes as shown below.



2. Modify any or all of the following.
   - ❑ Halo type in the *Blockage Type* field
   - ❑ Layer in the *Layer* field
   - ❑ The top, left, right, or bottom offsets
3. Click *Update*.

© 2023 Cadence Design Systems, Inc.

The specified halo is added to each of the selected blocks.



Related Topics

Edit Soft Blocks Form

Soft Block Mode

Enabling Level-1 Editing Mode

Level-1 Editing mode is the ability to edit objects inside a soft block without using the *Descend* or *Edit In Place* commands.

© 2023 Cadence Design Systems, Inc.

Important

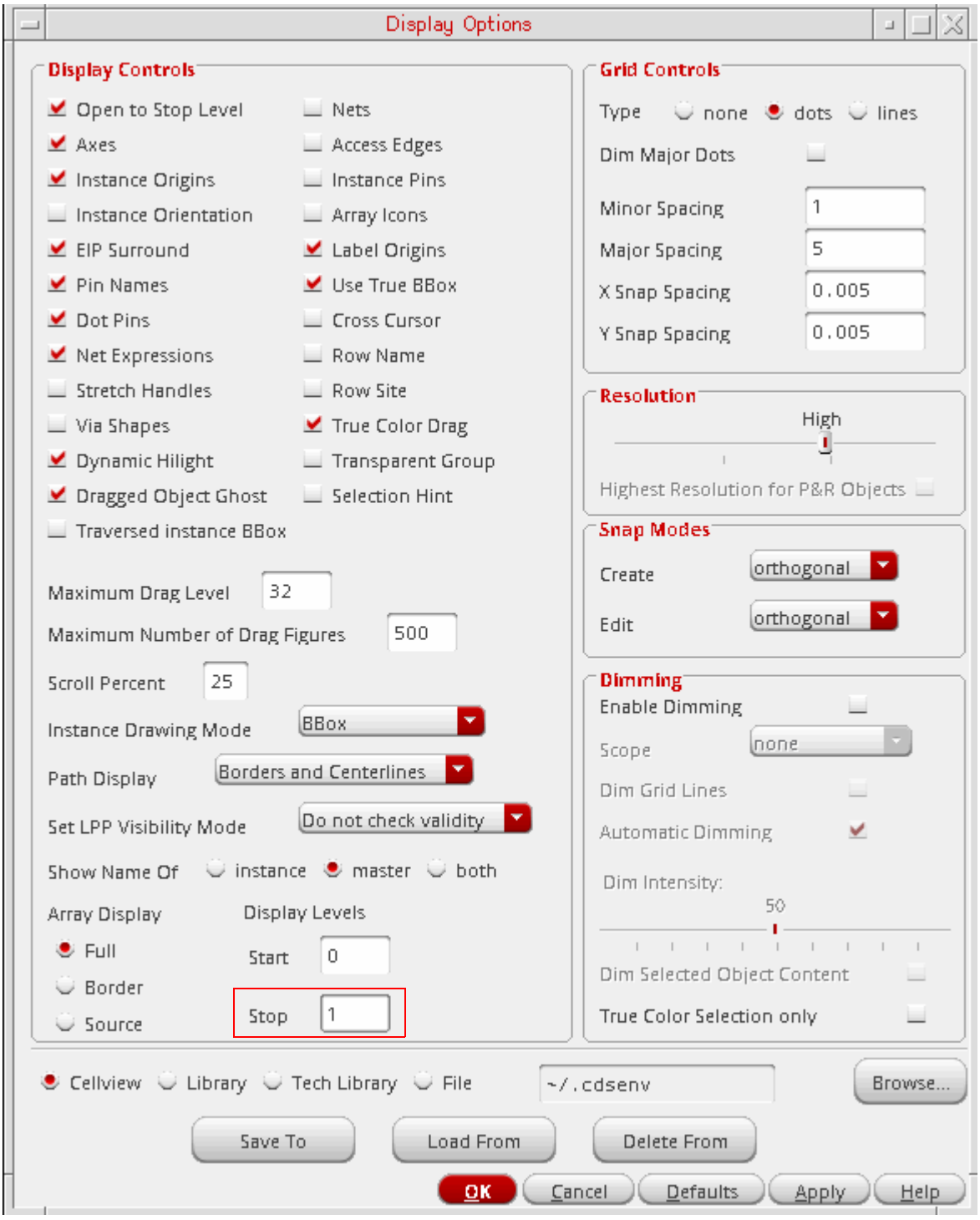
While in level-1 editing mode, you can edit the objects placed at level-1 only if the top-level is selectable.

To enable Level-1 Editing mode,

1. From the layout window menu bar, choose *Options – Display*.

© 2023 Cadence Design Systems, Inc.

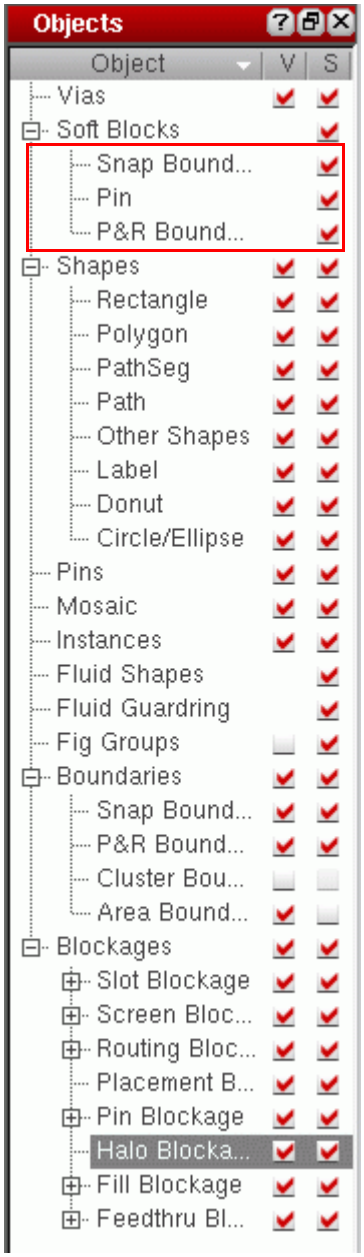
The Display Options form appears.



2. In the *Display Controls* group box, under *Display Levels*, set the *Stop* value to 1.

© 2023 Cadence Design Systems, Inc.

3. In the Objects panel, check the *V* and *S* check box to make *Pins*, *P&R Boundary,* and *Snap Boundary* visible and selectable. Check the *S check box* to make *Soft Block Pins*, *P&R* boundary, and *Snap Boundary* selectable.



© 2023 Cadence Design Systems, Inc.

Important

If an instance bounding box fully overlaps the place and route boundary, you need to click the instance twice in order to select the place and route boundary. The first click selects the instance, and the second click selects the Level-1 place and route boundary.

Once in Level-1 Editing mode, you can do the following.

- Moving an Object in Level-1 Editing Mode
- Stretching an Object in Level-1 Editing Mode
- Reshaping an Object in Level-1 Editing Mode
- Chopping an Object in Level-1 Editing Mode
- Finding and Replacing Nets
- Discarding Edits
- Saving Hierarchical Cellviews under a Different Name

Related Topics

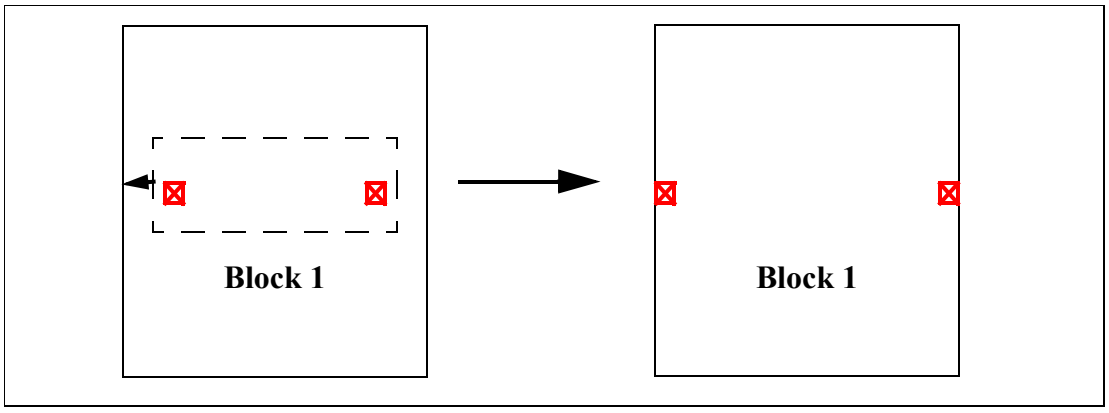
Display Options form

Moving an Object in Level-1 Editing Mode

Use the *Move* command in Level-1 Editing mode to move place and route boundaries, snap boundaries, and shape pins without entering Edit In Place mode. You can select the objects to be moved either before or after you start the *Move* command.

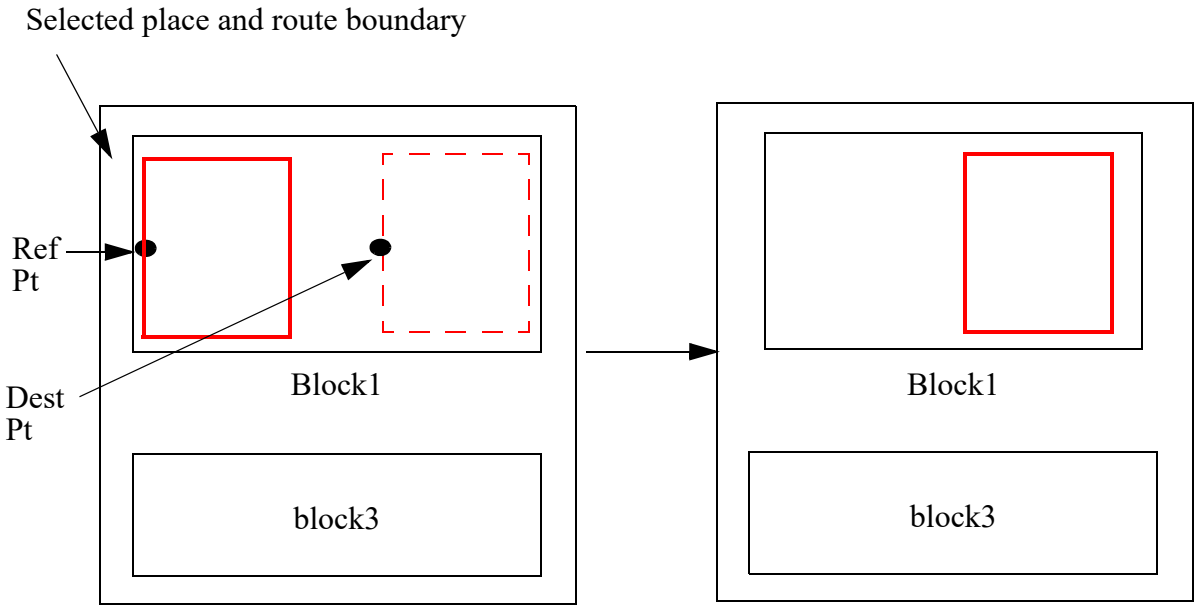
© 2023 Cadence Design Systems, Inc.

The picture below illustrates what happens when you move a shape pin.



In both cases, the pin is snapped to either the modified boundary edge or the closest boundary edge. A pin that is assigned a status of `firm` or `locked` is not moved. If a grid is not initialized, moved pins are not snapped.

The picture below illustrates what happens when you move a place and route boundary or a snap boundary.



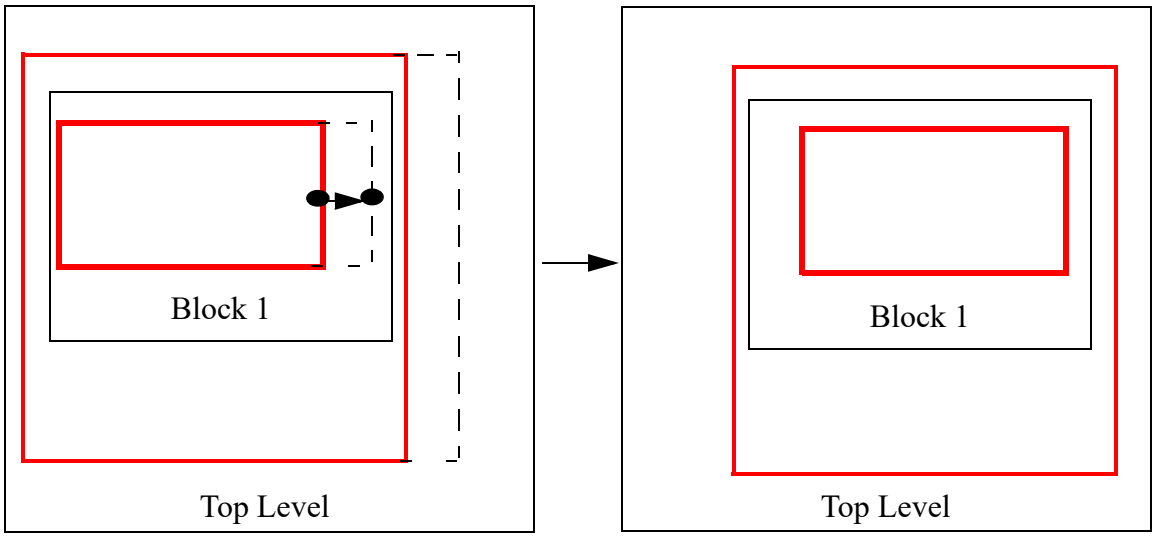
© 2023 Cadence Design Systems, Inc.

The place and route boundary is snapped according to the block type.

Block Type	Snap Behavior
Analog	The place and route boundary is snapped to the manufacturing grid.
Digital	The place and route boundary is snapped to the placement grid.

You can move multiple place and route boundaries or shape pins in a single operation. For example, you can select the corners or edges of place and route boundaries belonging to multiple blocks, as well as the top level, and apply the move operation to the entire selected set.

The picture below shows what happens when you move two place and route boundaries (`Block 1` and `Top Level`) or snap boundaries at the same time.



Moving a Place and Route Boundary at Level 1

To move a place and route boundary at level 1,

1. With Level-1 Editing mode enabled, choose *Edit – Move* from the layout window menu bar.

   A message in the status banner at the bottom of the layout window prompts you to select the figure to be moved.

© 2023 Cadence Design Systems, Inc.

2. Select the place and route boundary you want to move.

   You are prompted to point to the reference point for the move.

3. Point to the reference point.

   You are prompted to point to the new location for the move.

4. Point to the new location.

   The place and route boundary is moved from the reference point to the new location.

Related Topics

Moving Objects Manually

Objects Moving

Stretching an Object in Level-1 Editing Mode

Use the *Stretch* command in Level-1 Editing mode to stretch place and route boundaries, snap boundaries, and other pins without entering Edit In Place mode. You can stretch multiple pins, including the place and route boundary and snap boundary both at level 1 and at the top level, in a single operation.

Stretched objects are snapped to a grid depending on the block type.

Block Type	Snap Behavior
Analog	The place and route boundary and pin edges are snapped to the manufacturing grid.
Digital	The place and route boundary is snapped to the placement grid and pin centers are snapped to the routing grid defined for the metal layer on which the pins are defined. If the pins are not on a metal layer, they are snapped to the manufacturing grid.

A pin that is assigned a status of `firm` or `locked` is not moved. If a grid is not initialized, moved pins are not snapped.

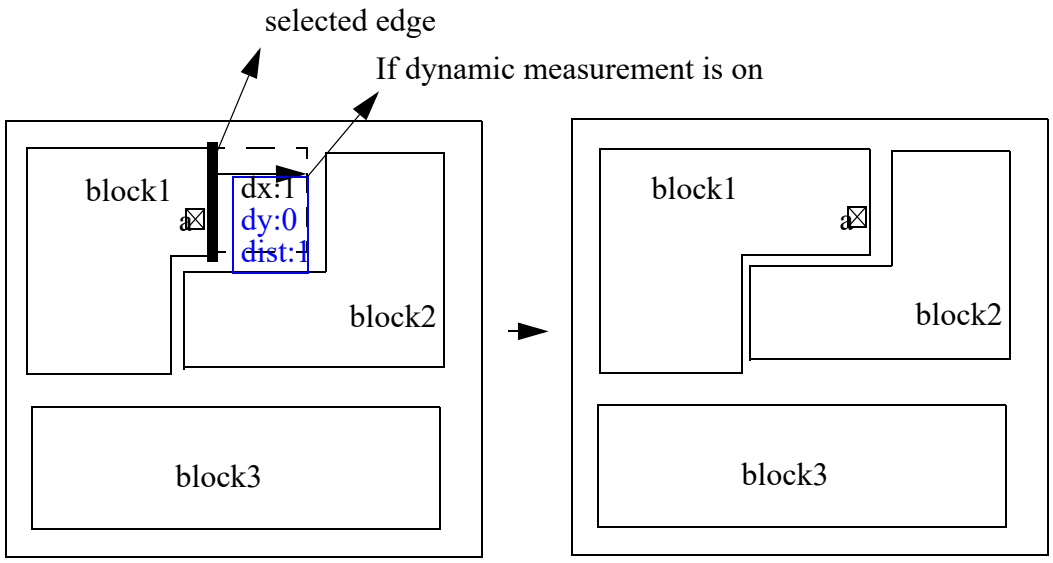
© 2023 Cadence Design Systems, Inc.

If you have Measurement Display mode switched on, the measurements (`dx`, `dy` and `dist`) of the object are shown to be changing dynamically as it is being stretched.

Important

You need to be in partial selection mode in order to select an edge or corner to stretch. Press `F4` to toggle between partial selection and full selection modes.

The picture below illustrates the effect of using the *Stretch* command on a level-1 place and route boundary or snap boundary. You can select the boundary to be stretched either before or after you have started the command.



Related Topics

Swap Components Form

Object Editing in Layout XL

Stretching a Single Edge in Level-1 Editing Mode

To stretch a single partial edge:

1. With Level-1 Editing mode enabled, choose *Edit – Stretch* from the layout window menu bar.

© 2023 Cadence Design Systems, Inc.

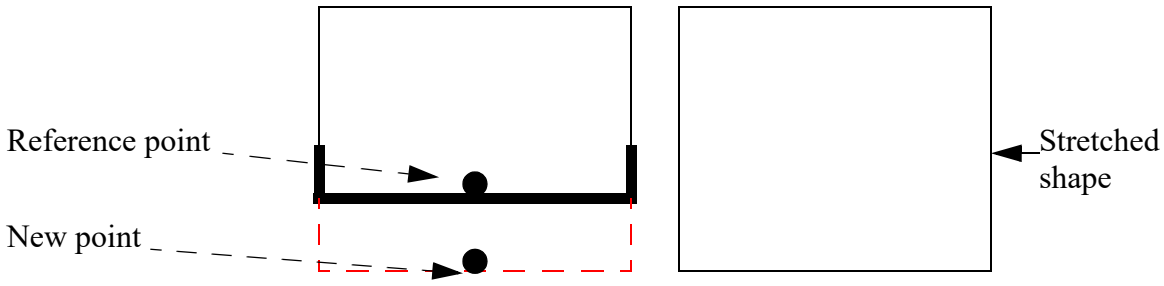
A message in the status banner at the bottom of the layout window prompts you to select the figure to be stretched.

2. Make sure that you are in partial selection mode and select the edges or corners you want to stretch.

   The selected edge or corner becomes the reference point for the stretch.

3. Point to the new location for the edge.

   The edge is stretched.



Stretching Multiple Edges

To select the edges or corners of multiple instances in partial selection mode,

1. With Level-1 Editing mode enabled, choose *Edit – Stretch* from the layout window menu bar.

   A message in the status banner at the bottom of the layout window prompts you to select the figure to be stretched.

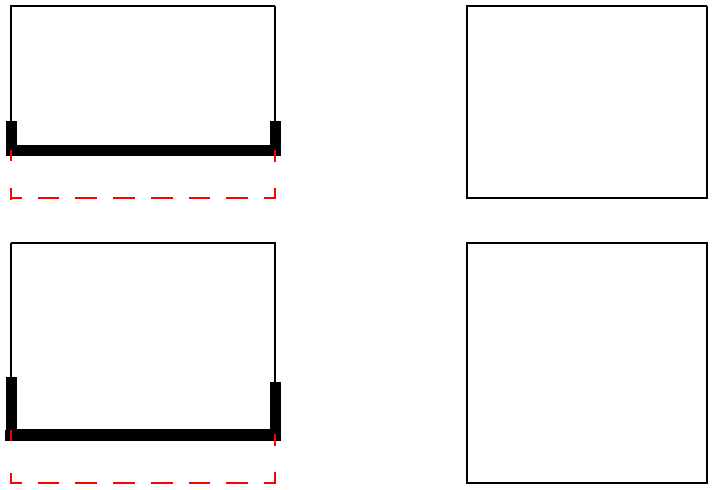
2. Make sure that you are in partial selection mode and use `Shift`-click or area selection to select the edges or corners you want to stretch.

   The selected edges become the reference point for the stretch.

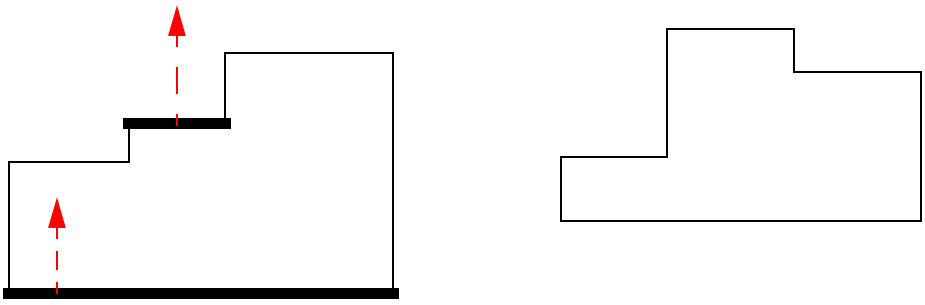
3. Point to the new location for the edges.

© 2023 Cadence Design Systems, Inc.

The edges are stretched.



4. You can also stretch multiple partial edges of same soft block.



Related Topics

Enabling Level-1 Editing Mode

Maintaining the Area of a Soft Block during Stretch

The *Use Constant Area Stretch* option on the Stretch form keeps the area of soft blocks and top-design PR boundary constant while their boundaries are stretched.

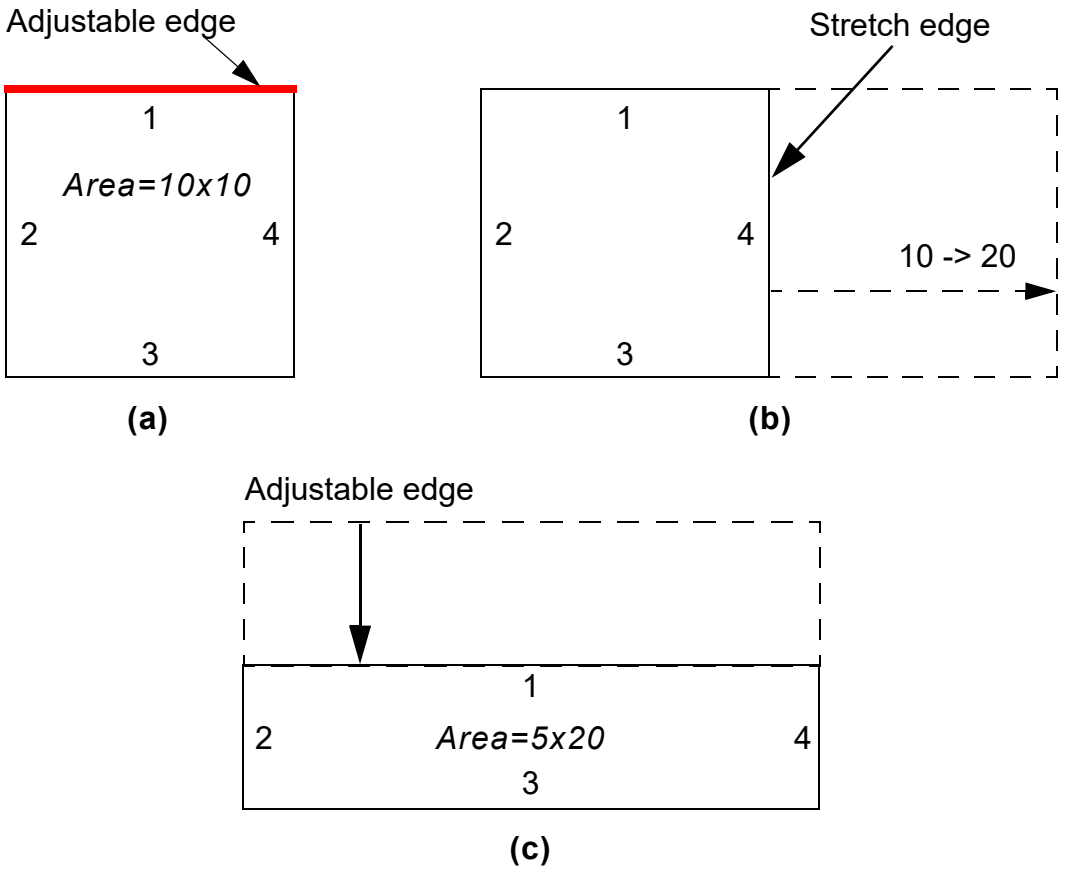
You can specify the edges that can be adjusted during the stretch operation.

The only difference between top-design PR boundary stretch and soft block stretch is that in case of soft block stretch, the soft block origin moves during the stretch, whereas in case of

© 2023 Cadence Design Systems, Inc.

PR boundary stretch, the origin of the PR boundary does not move. In addition, when the PR boundary being stretched is at the top level, the instances inside the PR boundary do not move.

Consider the following example of constant area stretch of a soft block.

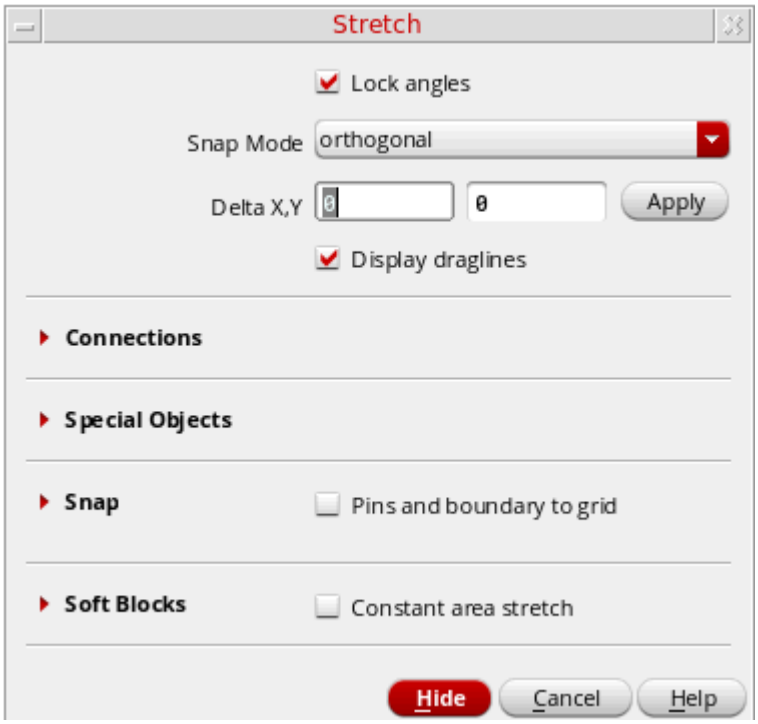


The block in the picture above has an area of 100 square units. If you specify that edge 1 is an adjustable edge [figure (a)] and then increase the width of the block from 10 to 20 [figure (b)], then the Stretch command automatically adjusts the free edge from 10 down to 5 in order to maintain a constant area of 100 square units [figure (c)].

© 2023 Cadence Design Systems, Inc.

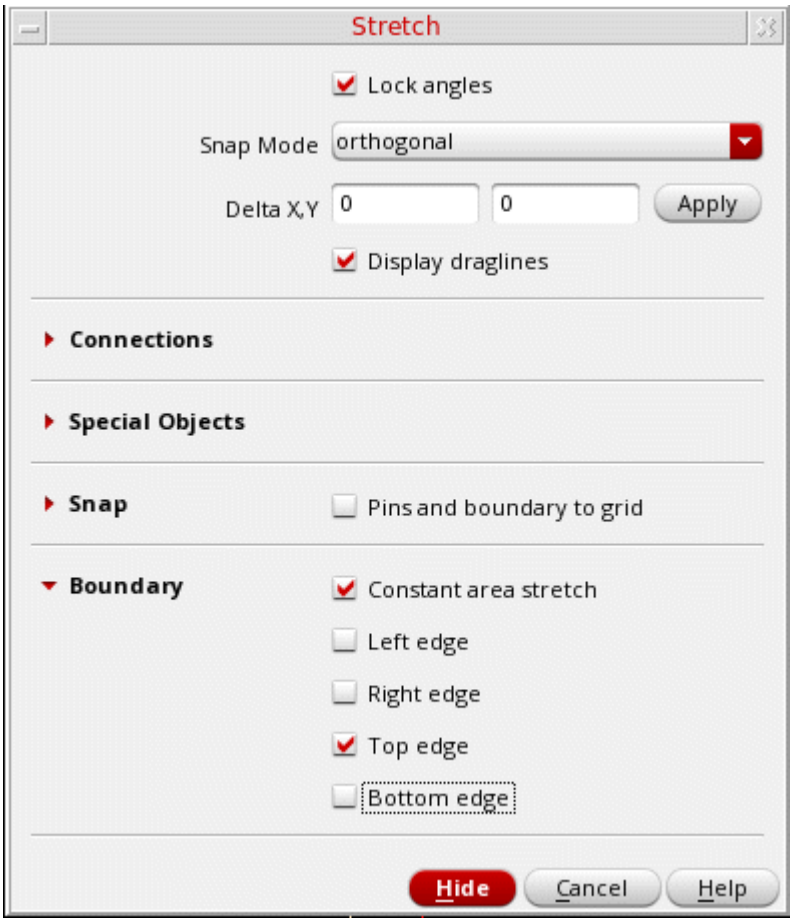
To do this:

1. From the layout window menu bar, choose *Edit* - *Stretch* and press `F3` to display the Stretch form.



© 2023 Cadence Design Systems, Inc.

2. Select the *Constant area stretch* option. The form updates to list the adjustable edges.



3. Select *Top edge* to make it adjustable.
4. Select the soft block or top-design PR boundary that you want to change and perform the stretch.

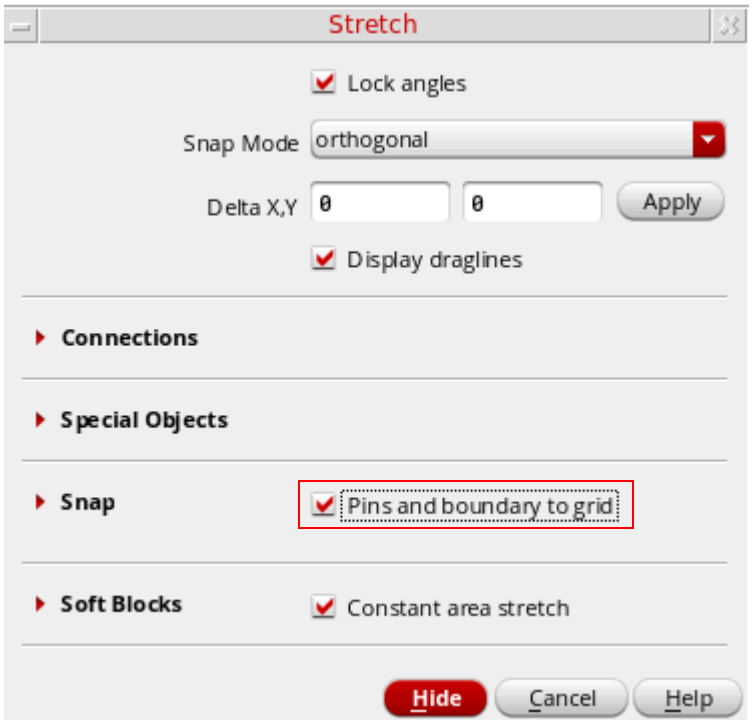
Changing the Angle of a Corner

To change the angle of a corner by stretching,

1. With Level-1 Editing mode enabled, choose *Edit – Stretch* from the layout window menu bar.

© 2023 Cadence Design Systems, Inc.

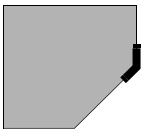
If the Stretch form does not appear automatically, press `F3` to display it.



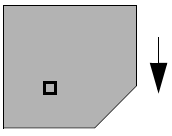
2. Turn off *Lock Angles* and turn on *Snap – Pins and boundary to grid* to snap the place and route boundary or pin to a grid.

   See the Swap Components Form

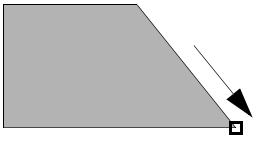
3. Click the corner you want to stretch and then move your mouse to stretch the shape as required.



The selected corner



With *Lock Angles* on, you cannot change the angle of the corner.



With *Lock Angles* off, you can stretch the corner in any direction.

Related Topics

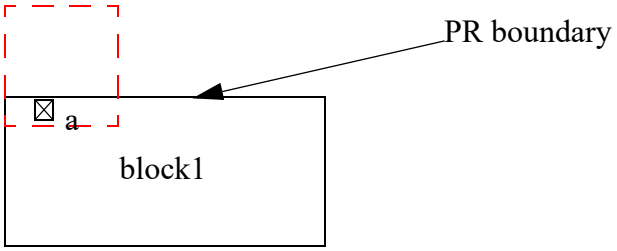
Enabling Level-1 Editing Mode

© 2023 Cadence Design Systems, Inc.

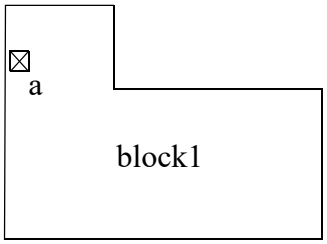
Reshaping an Object in Level-1 Editing Mode

Use the *Reshape* command in Level-1 Editing mode to reshape a place and route boundary without entering Edit In Place mode. To do this, you have to

1. Select the place and route boundary to be reshaped.
2. Draw a rectangle in such a way that it intersects at least one of the edges of the boundary or is contained within the place and route boundary.
3. Reshape the boundary with respect to this rectangle.



a) Initial block shape and pin placement



b) Block after *Reshape*

You can select the shapes to be reshaped either before of after you launch the *Reshape* command.

Stretched objects are snapped to a grid depending on the block type.

Block Type	Snap Behavior
Analog	The place and route boundary and pin edges are snapped to the manufacturing grid.

© 2023 Cadence Design Systems, Inc.

Block Type	Snap Behavior
Digital	The place and route boundary is snapped to the placement grid and pin centers are snapped to the routing grid defined for the metal layer on which the pins are defined. If the pins are not on a metal layer, they are snapped to the manufacturing grid.

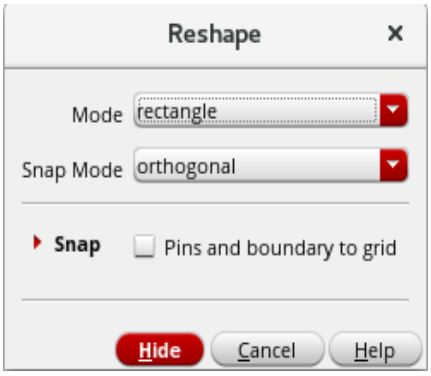
A pin that is assigned a status of `firm` or `locked` is not moved. If a grid is not initialized, moved pins are not snapped.

Reshaping a Place and Route Boundary at Level 1

To reshape a block place and route boundary,

1. With Level-1 Editing mode enabled, choose *Edit – Advanced – Reshape* from the layout window menu bar.

   If the Reshape form does not appear automatically, press `F3` to display it.



2. Select the polygon or path that you want to reshape.
3. To add or remove a rectangle shape, select *rectangle* as the *Mode*. To add a polygon to a shape or to reshape a path, select *line*.
4. Select a suitable Snap Mode to specify the shape of line segments.
5. In the *Snap* section, select *Pins and boundary to grid* to snap the stretched object to a grid depending on the block type.
6. Click *Hide*.
7. Select the place and route boundary to be reshaped.

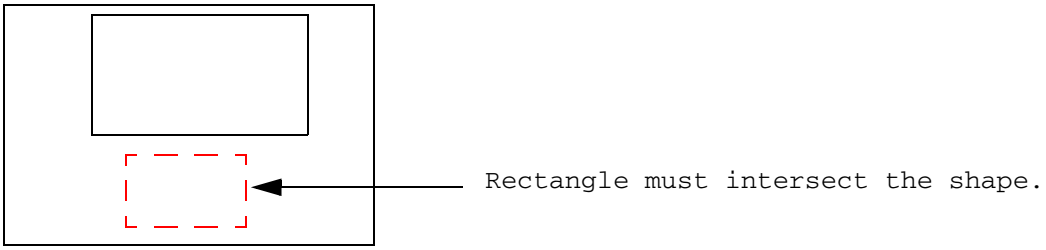
© 2023 Cadence Design Systems, Inc.

You are prompted to point to the first corner of the reshape rectangle.

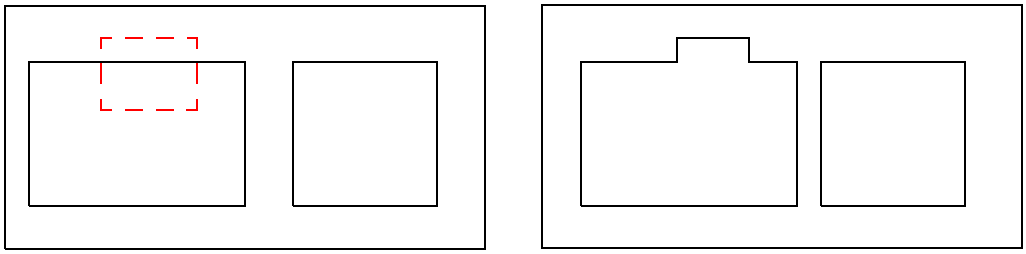
8. Point to the first corner

   You are prompted to point to the diagonally opposite corner of the reshape rectangle.

   If the boundary of the reshape rectangle does not intersect the boundary of the block to be reshaped, the system issues a message.



9. If you select multiple instances for reshaping, you need to reshape them one by one by using the procedure described above.



Related Topics

Save Short Locator View Form

Reshape Form

Editing Objects

Enabling Level-1 Editing Mode

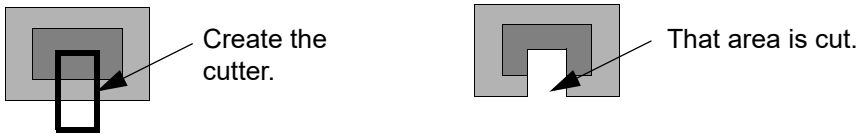
Chopping an Object in Level-1 Editing Mode

Use the *Chop* command in Level-1 Editing mode to cut away part of a place and route boundary without entering Edit In Place mode. To do this,

1. Select the boundary to be chopped.

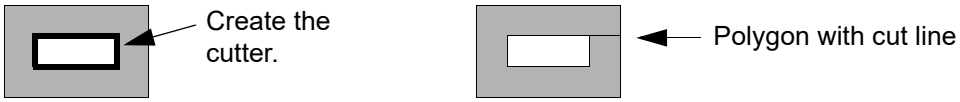
© 2023 Cadence Design Systems, Inc.

2. Draw a rectangle in such a way that it intersects at least one of the edges of the boundary.
3. Chop the boundary with respect to this rectangle



Tip

If you create a polygonal cutter, you must double-click to complete the polygon cutter and the chop. If you cut a hole in an object, it is redrawn as a polygon with a cut line.



You can select the shapes to be chopped either before or after you launch the *Chop* command. You cannot chop a place and route boundary into multiple pieces. You cannot chop pins at all.

Chopped objects are snapped to a grid depending on the block type.

Block Type	Snap Behavior
Analog	The place and route boundary and pin edges are snapped to the manufacturing grid.
Digital	The place and route boundary is snapped to the placement grid and pin centers are snapped to the routing grid defined for the metal layer on which the pins are defined. If the pins are not on a metal layer, they are snapped to the manufacturing grid.

During the snapping operation, if one pin (X) overlaps with another pin (Y) that is on the same layer and already snapped, the snap operation aborts with an error message:

```
*WARNING* Pin (Y) overlaps an already placed Pin (X) at location 2.4:294.6.
```


© 2023 Cadence Design Systems, Inc.

If there are many such pins overlapping with the already snapped pin, a list of these pins is provided in a similar error message.

A pin that is assigned a status of `firm` or `locked` is not moved. If a grid is not initialized, moved pins are not snapped.

To cut away part of an object,

1. From the layout window menu bar, choose *Edit – Basic – Chop*.

   If the Chop form does not appear automatically, press `F3` to display it.
2. Set *Chop Shape* to *rectangle*, *line*, or *polygon*. If you select line or polygon, set the *Snap Mode*.
3. You can split an object by turning off the *Remove chop* option in the Chop form and using a *rectangle* or *polygon* chop shape. This saves both the area you cut and the original objects.
4. In the *Snap* section, select *Pins and boundary to grid* to snap the stretched object to a grid depending on the block type. If it is an analog block, the place and route boundary and pin edges are snapped to the manufacturing grid. If it is a digital block, the place and route boundary is snapped to the placement grid and the pin centers are snapped to the routing grid. Pins with placement status unplaced, fixed, locked, or unknown are not snapped, even if they were touching the boundary before the reshape.
5. Click *Hide*.
6. Select one or more objects.
7. Click to enter the first corner of the rectangle cutter.
8. Click to enter the opposite corner of the rectangle cutter.

   The second click completes the chop.

Chopping a Place and Route Boundary at Level 1

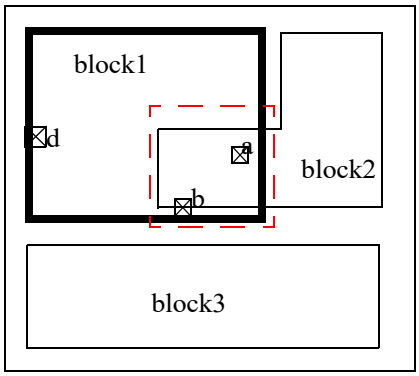
If the *Snap – Pins and boundary to grid* option is on, the chopping boundary is snapped depending on the block type.

Block Type	Snap Behavior
Analog	The chopping boundary is snapped to the manufacturing grid. Pin edges are snapped to the closest boundary edge.

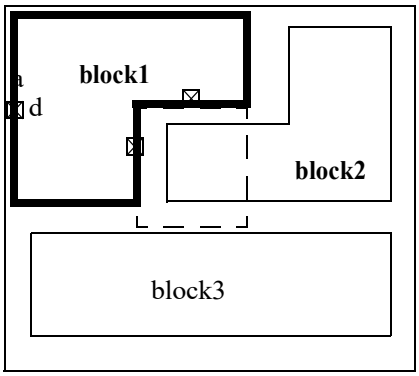
© 2023 Cadence Design Systems, Inc.

Block Type	Snap Behavior
Digital	The chopping boundary is snapped to the placement grid.

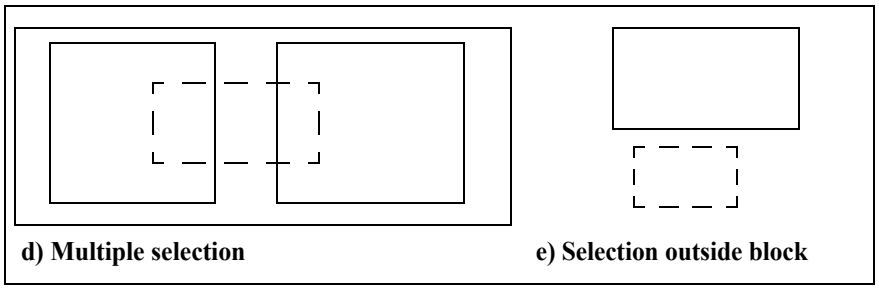
During boundary chopping, pins with placement status `placed` are automatically snapped to the closest edge. Pins with placement status `unplaced`, `fixed`, `locked`, or `unknown` are not moved, even if they were touching the boundary before it was chopped.



a) Initial block shape and pin placement



b) block after chop



d) Multiple selection **e) Selection outside block**

© 2023 Cadence Design Systems, Inc.

To chop a block in post-selection mode:

1. With Level-1 Editing mode enabled, choose *Edit – Basic – Chop* from the layout window menu bar.

   If the Chop form does not appear automatically, press `F3` to display it.



2. In the Chop form, set *Chop Shape* to *rectangle* and turn on *Pins and boundary to grid* in the *Snap* section to snap the object to its respective grid.
3. In the Palette assistant, make sure *Soft Blocks P&R Boundary* is selectable.

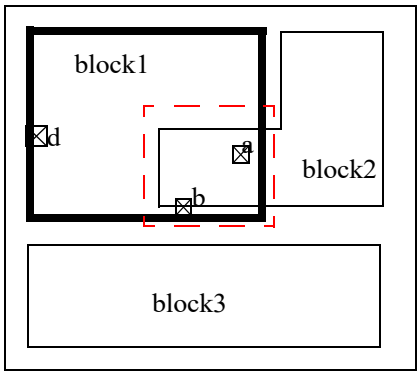
   You are prompted to point to the shape to be chopped.
4. Select the Level-1 PR Boundary of the object to chop.

   You are prompted to point to the first corner of the chop rectangle.
5. Point to the first corner.

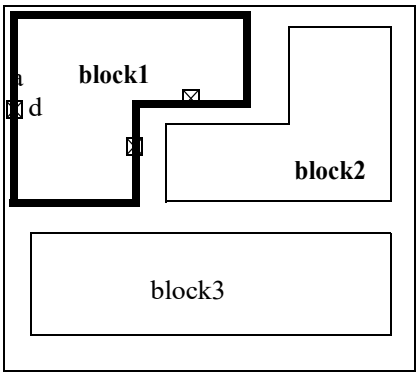
   You are prompted to point to the second corner of the chop rectangle.

© 2023 Cadence Design Systems, Inc.

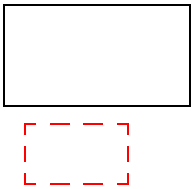
6. Point at the second corner.



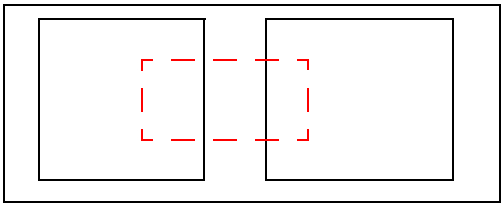
When the cutter is completed, the chop operation completes.



If the boundaries of the rectangle do not intersect the edges of the PR boundary to be modified, the place and route boundary is not chopped.



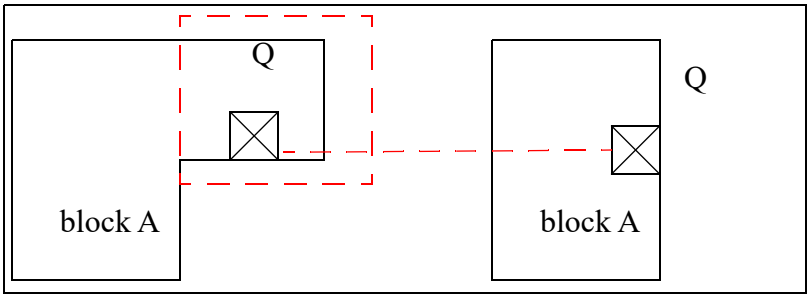
You can also chop multiple boundaries in a single operation.



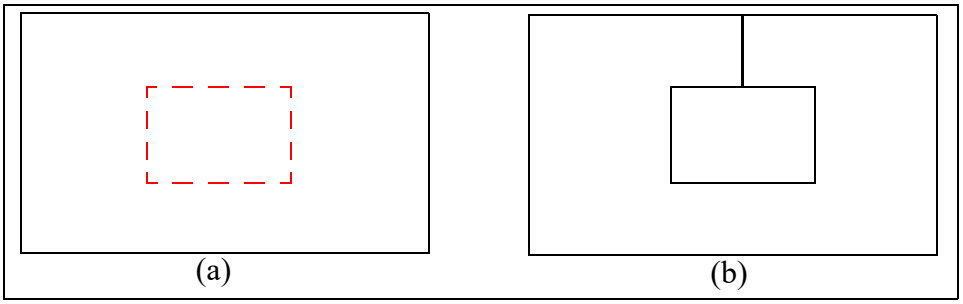
© 2023 Cadence Design Systems, Inc.

Examples

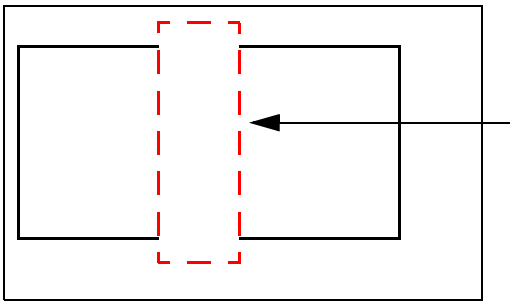
The picture below shows you what happens when you collapse three edges to form a single edge.



The picture below shows a donut-like shape created using the chop operation. You can select a place and route boundary and then draw a rectangle within it to chop it (a). This operation creates a polygon shape figure (b).

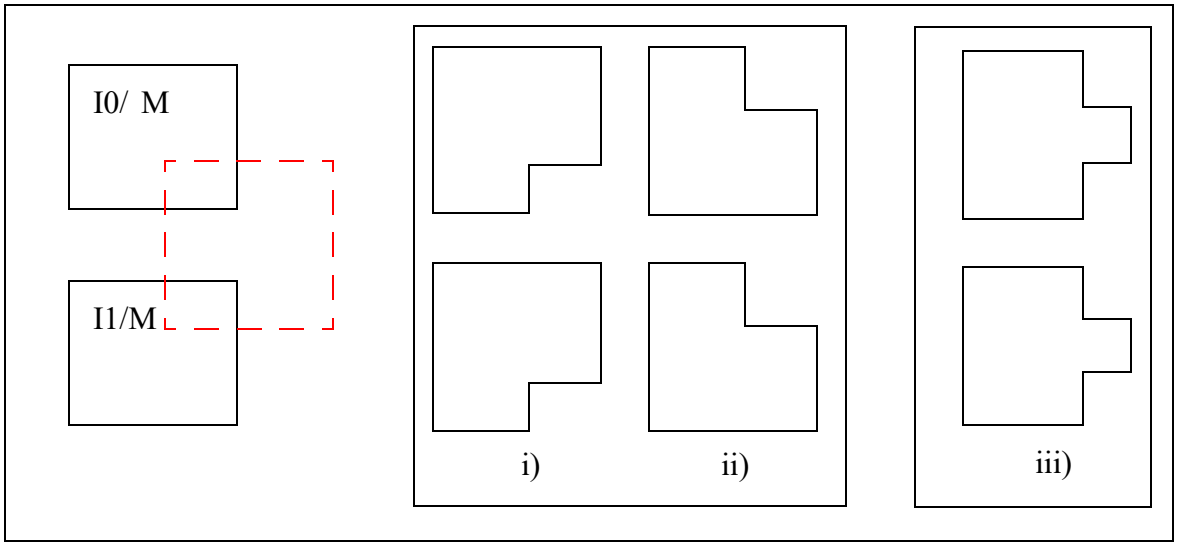


You cannot chop a place and route boundary into two or more pieces.



© 2023 Cadence Design Systems, Inc.

If you select multiple instances of the same master cellview, you need to chop only one of them and the changes are reflected on all the instances. The result, as shown in the picture below, can be `i` or `ii` but not `iii`.



Related Topics

Editing Objects

Check XL Compliance Form

Enabling Level-1 Editing Mode

Soft Block Pins

After you have configured the physical hierarchy, it might be necessary to create additional soft pins or feedthrough terminal block pins on your soft blocks to improve the connectivity of your design. The *Create – Soft Pins* command lets you create, delete, and update soft pins for your soft blocks. The *Create – Feedthrough Terminal Pins* lets you create feedthrough pins.

Both *Create Soft Pins* and *Create Feedthrough Terminal Pins* commands create strong pins. To create weak pins, use the *Create Pin* command instead.

© 2023 Cadence Design Systems, Inc.

Related Topics

Creating a Soft Pin

Updating Parameters for a Soft Pin

Deleting a Soft Pin

Updating Parameters for a Soft Pin

Updating Parameters for a Soft Pin

Updating Parameters for a Soft Pin

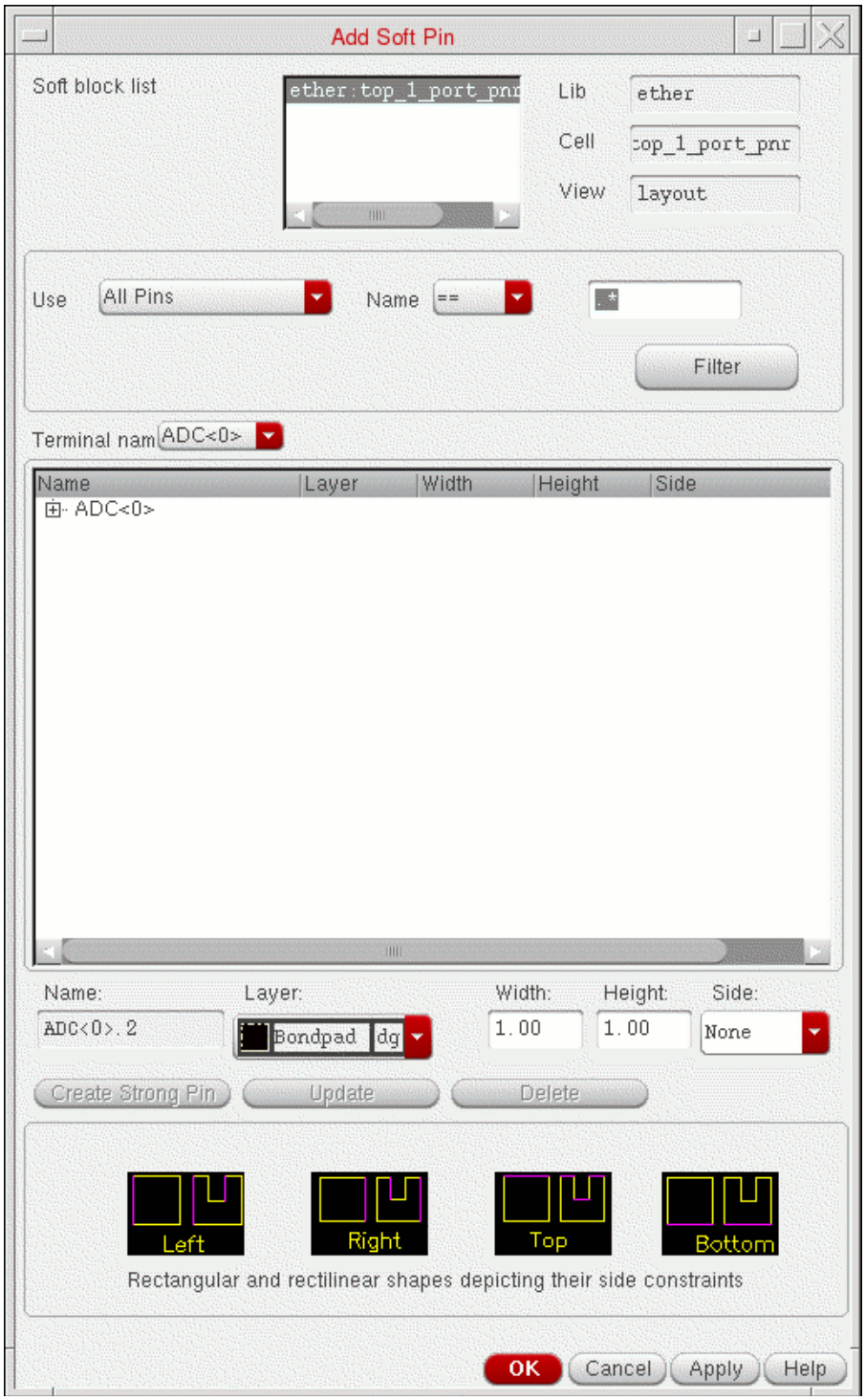
Creating a Soft Pin

To create a soft pin:

1. In the layout canvas, select the soft block for which the soft pin is to be created.
2. Click *Create – Soft Pins*.

© 2023 Cadence Design Systems, Inc.

The Add Soft Pin form is displayed.



If you selected multiple blocks before you launched the form, all the selected blocks are included in the soft block list at the top of the form. Select the one you want from the list.

© 2023 Cadence Design Systems, Inc.

3. Choose the *Terminal name* for which you want to create a soft pin from the drop-down list, which by default contains all the terminals in the currently selected block.

   You can filter the terminal names that are displayed by typing in the text box and filtering either *All Pins* or *Pins in Selected Set*.

   To assist you, the canvas shows flight lines to represent the nets between the terminals you select.

   The software automatically assigns a name for the new pin.

4. Choose the pin *Layer* purpose and specify the *Width* and *Height* of the pin in the fields provided.

5. Define the side constraints using an appropriate option from the *Side* drop-down list.

   - You can assign a side (Left, Right, Top, or Bottom) constraint for each pin figure from the *Side* drop-down. This creates an alignment (side) constraint, which you can view in the Constraint Manager. The alignment constraint, so created, is honored by the pin optimizer.
   - Selecting a value **None** from the *Side* drop-down implies that the pin figure is not constrained to any particular side.

6. Click *Create Strong Pin* to create a new pin.

   The soft pin is created in the specified block.

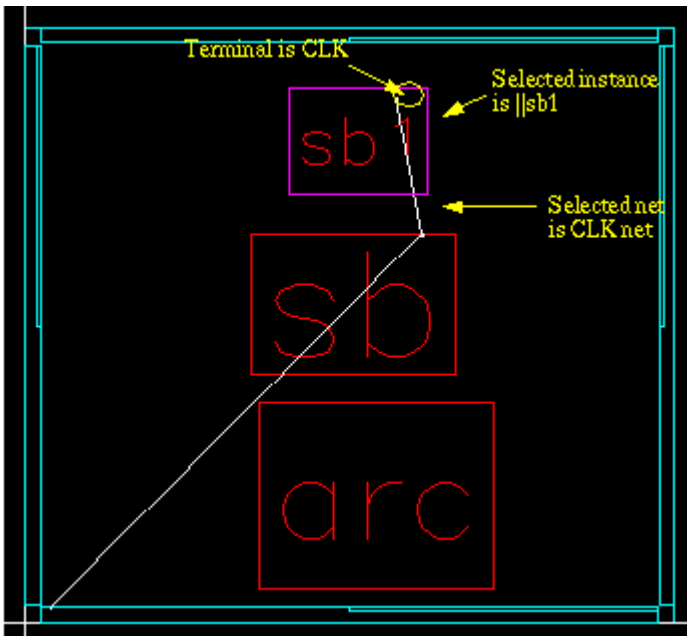
Examples

You want to create a soft pin for an existing terminal on a number of soft block instances.

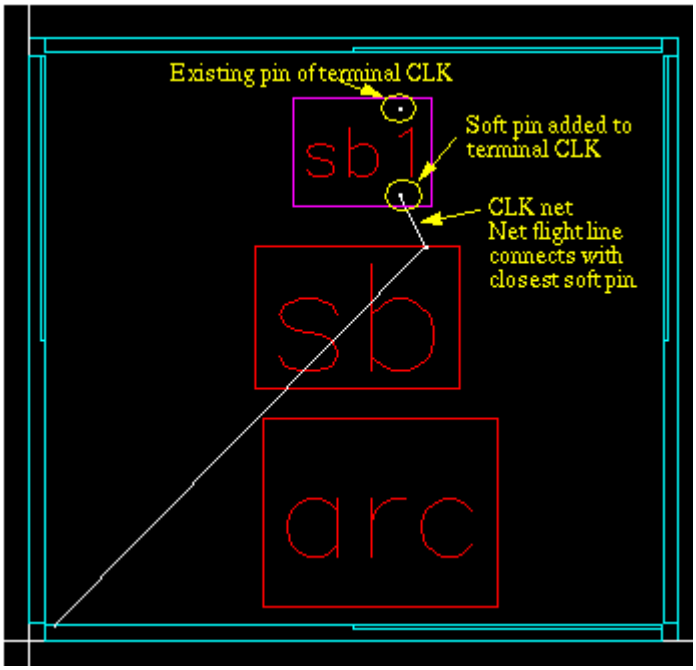
To do this, select the soft block instances and the name of the existing terminal on the selected set to create an extra soft pin for the selected terminal on each instance in the selected set.

© 2023 Cadence Design Systems, Inc.

Now you want to add a pin for the `CLK` terminal on instance `||sb1`.



After creating another pin for the existing terminal `CLK` on the instance `||sb1`, the `CLK` net changes as shown below.



© 2023 Cadence Design Systems, Inc.

Related Topics

Add Soft Pin Form

Updating Parameters for a Soft Pin

To update the parameters for one or more soft pins,

1. In the layout canvas, select the soft block for which the soft pin is to be created.
2. Click *Create – Soft Pins*.

   The Add Soft Pin form is displayed.
3. Select the pins you want to update from the list.
4. Use the controls provided to change the *Layer purpose* and specify the *Width* and *Height* of the pin, as required.
5. Define the side constraints using an appropriate option from the *Side* drop-down list.
   - You can assign a side (Left, Right, Top, or Bottom) constraint for each pin figure from the *Side* drop-down. This creates an alignment (side) constraint, which you can view in the Constraint Manager. The alignment constraint, so created, is honored by the pin optimizer.
   - Selecting a value **None** from the *Side* drop-down implies that the pin figure is not constrained to any particular side.
6. Click *Update*.

   The selected pins are updated.

Related Topics

Add Soft Pin Form

Deleting a Soft Pin

To delete one or more soft pins,

1. In the layout canvas, select the soft block for which the soft pin is to be created.
2. Click *Create – Soft Pins*.

   The Add Soft Pin form is displayed.

© 2023 Cadence Design Systems, Inc.

3. Select the pins you want to delete from the list.
4. Click *Delete*.

   The selected pins are deleted.

Related Topics

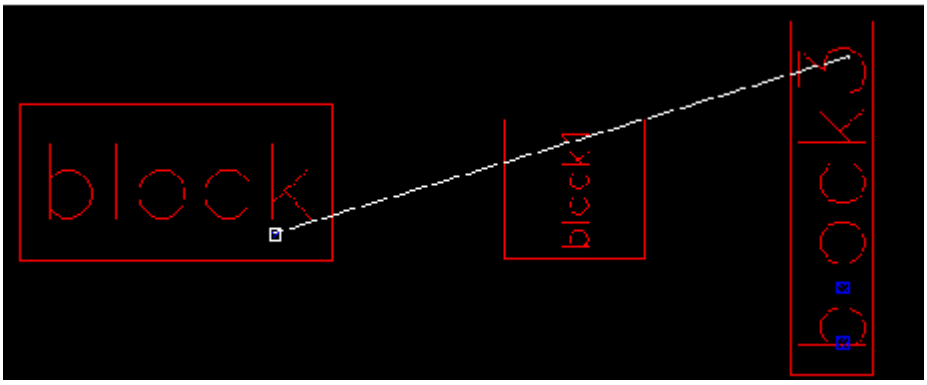
Add Soft Pin Form

Creating a Feedthrough Terminal Pin

To create a feedthrough terminal pin:

1. In the layout canvas, select the soft block for which the pin is to be created.

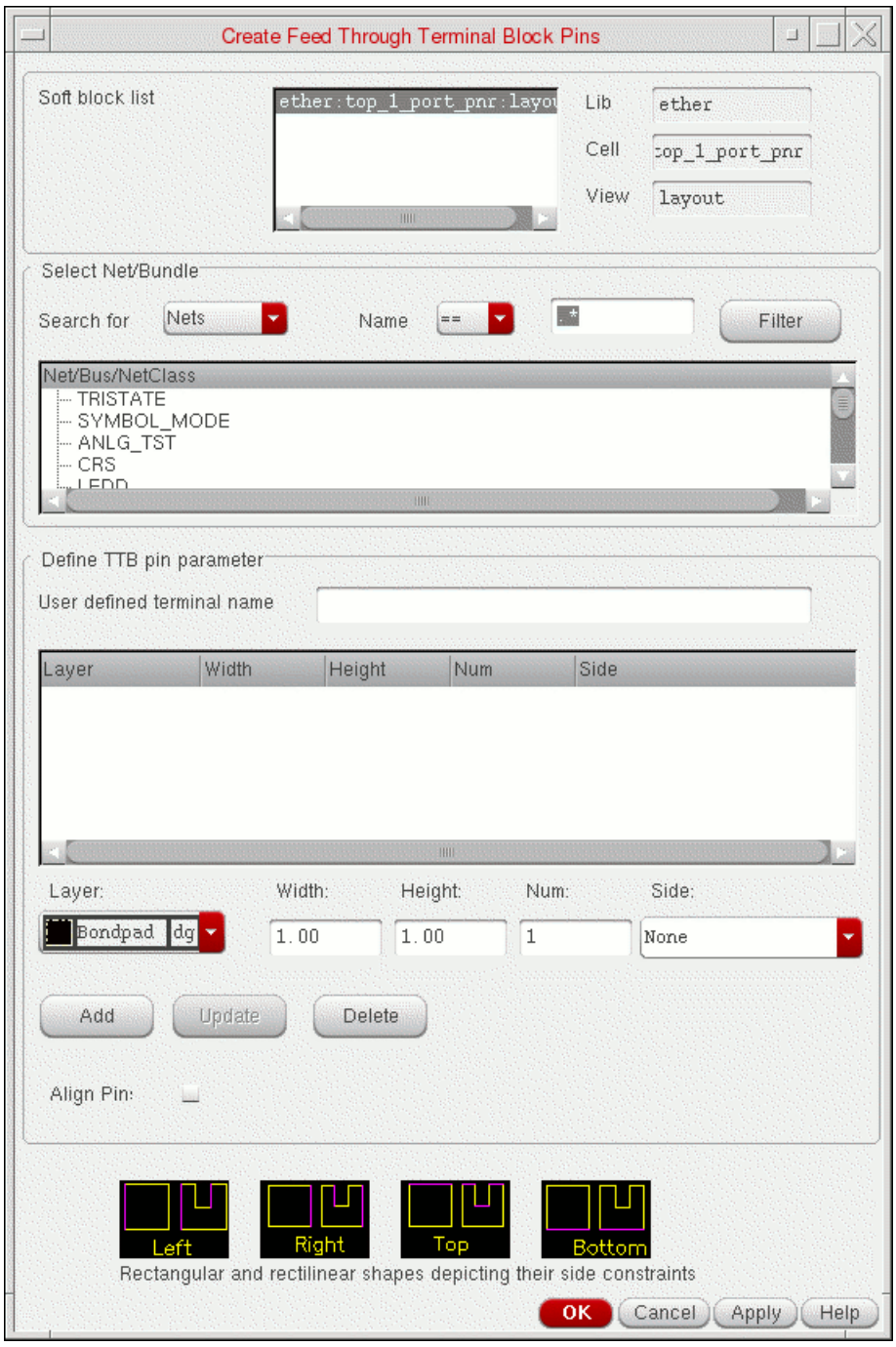
   In this example, we select `block1` as shown in the picture below.



2. Click *Create – Feed Through Terminal Pins*.

© 2023 Cadence Design Systems, Inc.

The Create Feed Through Terminal Block Pins form is displayed.

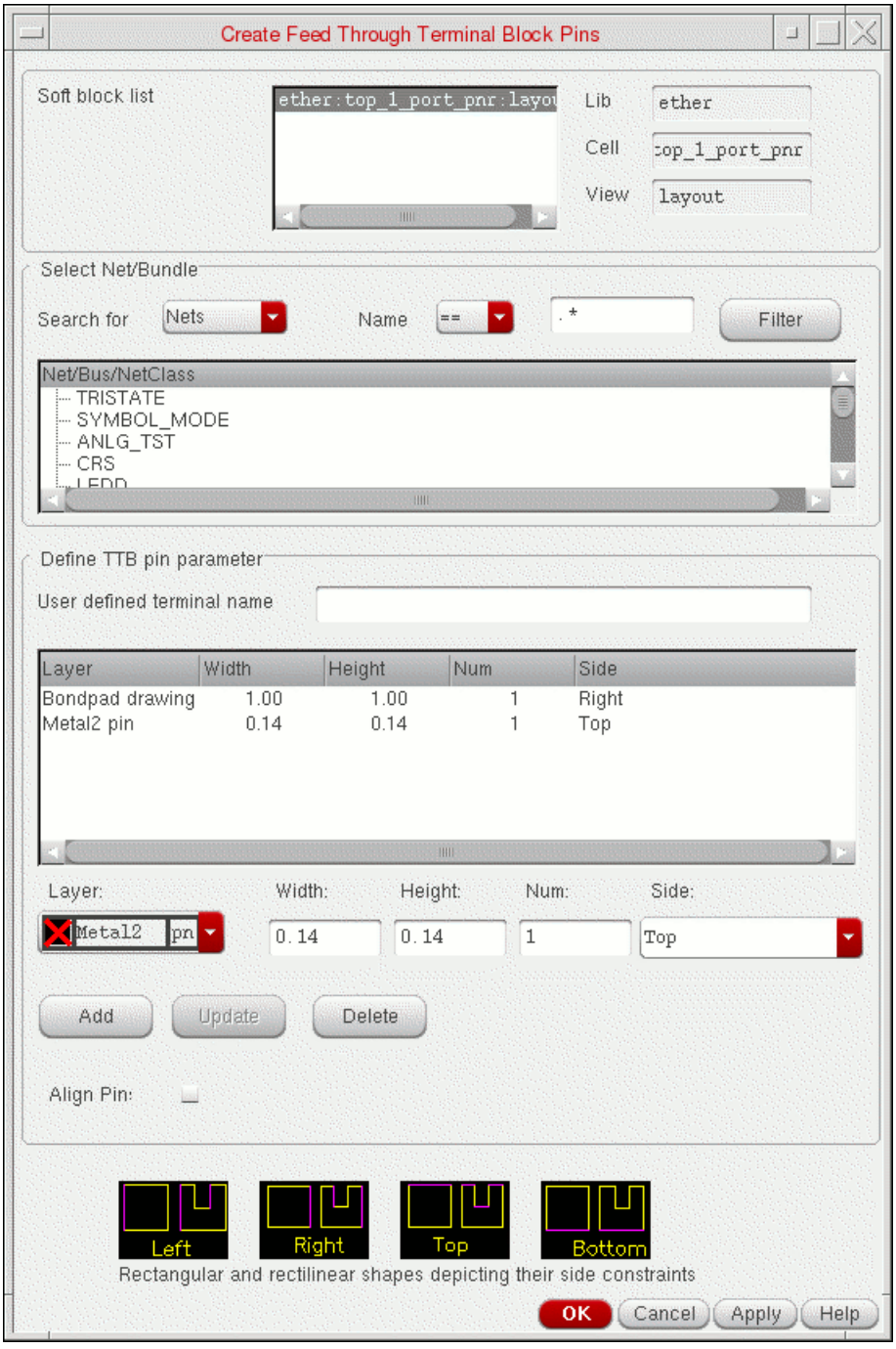


If you selected multiple blocks before you launched the form, all the selected blocks are included in the soft block list at the top of the form. Select the one you want from the list.

3. From the net table, select the net for which you want to create the feedthrough pin.

© 2023 Cadence Design Systems, Inc.

The system-generated terminal name is displayed as shown in picture below. The canvas shows flight lines to represent the nets between the terminals you select.



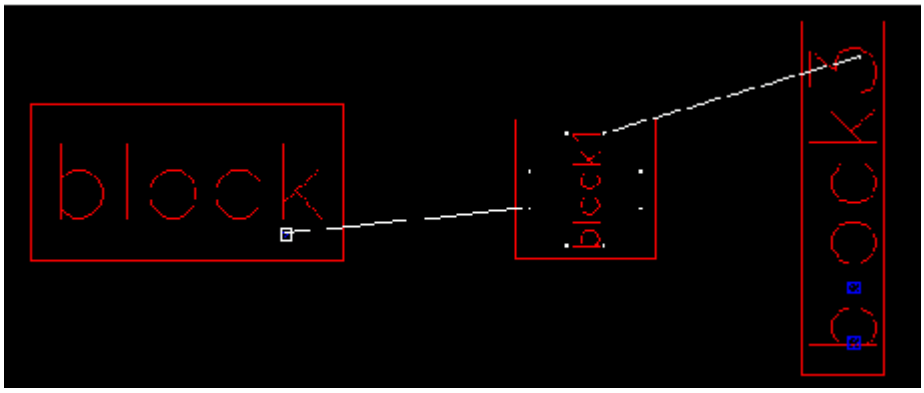
4. Choose the *Layer purpose* and specify the *Width* and *Height* of the pin in the fields provided.

© 2023 Cadence Design Systems, Inc.

5. Define the side constraints using an appropriate option from the *Side* drop-down list.
   - You can assign a side (Left, Right, Top, or Bottom) constraint for each pin figure from the *Side* drop-down. This creates an alignment (side) constraint, which you can view in the Constraint Manager. The alignment constraint, so created, is honored by the pin optimizer.
   - Selecting a value **None** from the *Side* drop-down implies that the pin figure is not constrained to any particular side.
6. Click *Add*.

   The feedthrough pin parameters are added to the table as shown in the picture below.
7. Click *OK*.

   The design view changes to show the feedthrough pin as shown in the picture below.



A new terminal `ftTerm_D` is created in soft block `block1` with two pins, Net `D` is broken into two parts; one part connecting `block` and `block1` and another connecting `block1` and `block3`.

Creating Feedthrough Terminal Pins for ordered nets

The steps to create feedthrough terminal pins for a group of ordered nets, which are grouped by Net Class or Bus constraints, are similar to the steps for creating a feedthrough terminal pin for a single net. The only difference is that you need to select a group of ordered nets in step 3 above.

Related Topics

Create Feed Through Terminal Block Pins Form

© 2023 Cadence Design Systems, Inc.

Updating Parameters for a Feedthrough Terminal Pin

To update a parameter:

1. In the layout canvas, select the soft block for which the pin is to be created.
2. Click *Create – Feed Through Terminal Pins*.

   The Create Feed Through Terminal Block Pins form is displayed.
3. Select the entry that you want to update in the table.
4. Change the values in the *Layer*, *Width*, *Height*, *Num*, and *Side* fields as required.
5. Click *Update*.

   The parameter is updated in the table.

Related Topics

Create Feed Through Terminal Block Pins Form

Deleting a Feedthrough Terminal Pin

To delete a pin,

1. In the layout canvas, select the soft block for which the pin is to be created.
2. Click *Create – Feed Through Terminal Pins*.

   The Create Feed Through Terminal Block Pins form is displayed.
3. Select the entries that you want to delete in the table.
4. Click *Delete*.

Related Topics

Create Feed Through Terminal Block Pins Form

Snapping Soft Block Pins

The *Move*, *Reshape*, *Stretch*, and *Chop* commands have options that let you snap boundaries and pins automatically to grid during interactive editing. To snap pins without

© 2023 Cadence Design Systems, Inc.

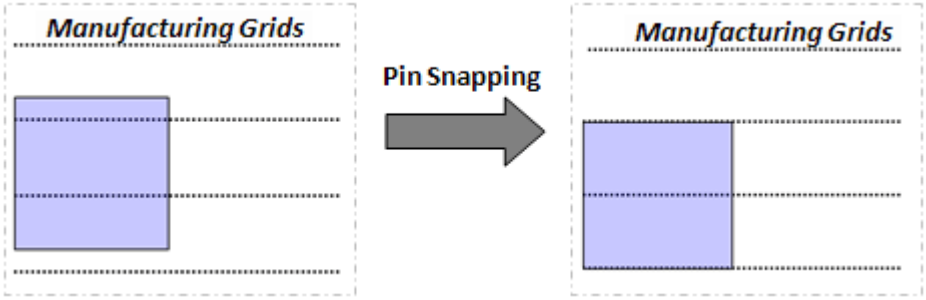
accessing these commands, use the *Connectivity – Pins – Snap Pins* command. This snaps both top-level and level-1 pins to the grid appropriate to the block type you are editing.

If the pins are placed outside the PR boundary, by default, they will be snapped to the PR boundary even if the Snap to Boundary option is OFF. This is because the grids outside the PR boundary are not initialized and the nearest grid that the pin locates, is the PR boundary.

Snap Behavior for Analog Pins

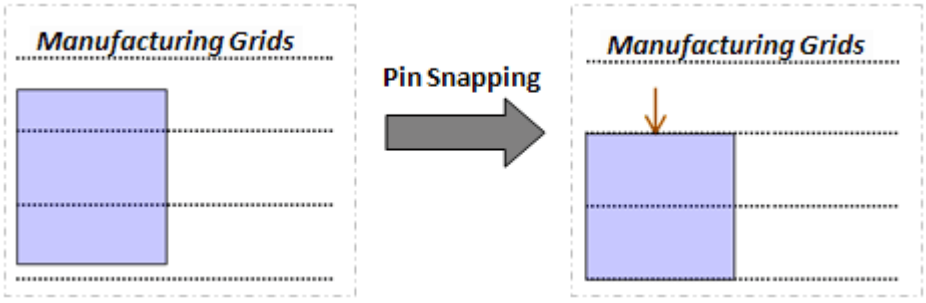
Situation 1: The pin size is a multiple of the manufacturing grid but the pin is located off grid.

Such pins are snapped to the nearest manufacturing grid.



Situation 2: The pin size is not a multiple of the manufacturing grid and the pin is located off grid.

The Lower Left Point (LLP) of the pin is snapped to the nearest manufacturing grid. Optionally, the pin can be resized and snapped to the nearest manufacturing grid.



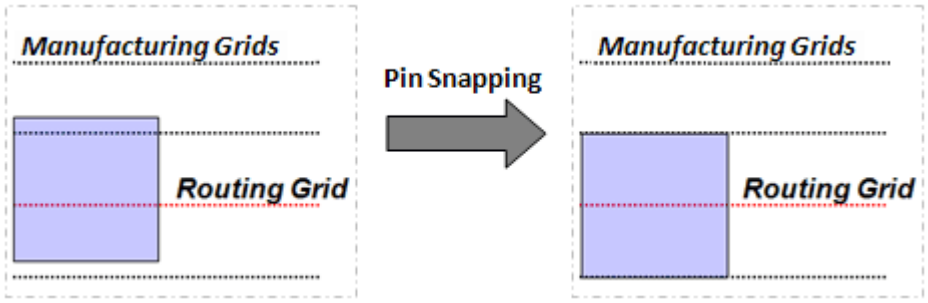
Snap Behavior for Digital Pins

The routing grid may be square or rectangular in shape. Pin snapping behavior is the same, regardless of the shape of the routing grid.

© 2023 Cadence Design Systems, Inc.

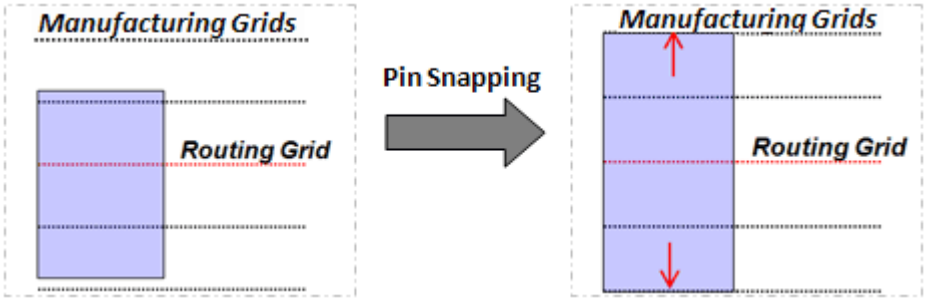
Situation 1: The pin size is an even multiple of the manufacturing grid but the pin is not centered on the routing grid.

Such pins are centered to the “nearest” routing grid.



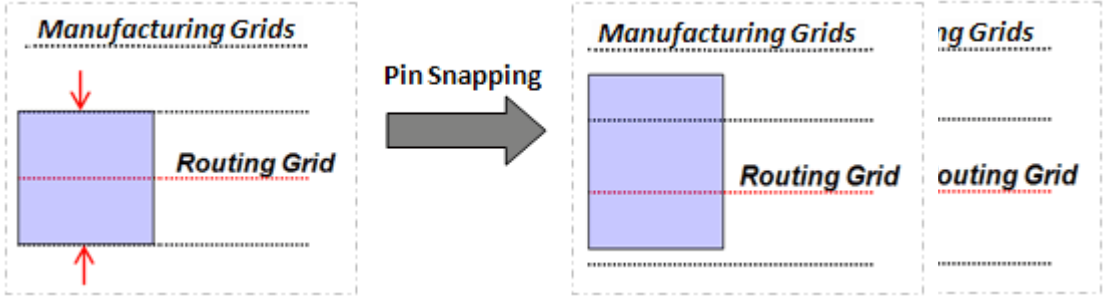
Situation 2: The pin size is an odd multiple of the manufacturing grid and the pin is not centered on the routing grid.

Such pins are centered to the “nearest” routing grid. Optionally, the pin can be resized and aligned with the nearest manufacturing grid.



Situation 3: Pin size is not a multiple of the manufacturing grid and the pin is not centered on the routing grid.

The pin is centered to the nearest routing grid. Optionally, the pin can be resized and aligned with the nearest manufacturing grid.



© 2023 Cadence Design Systems, Inc.

Related Topics

Moving an Object in Level-1 Editing Mode

Stretching an Object in Level-1 Editing Mode

Reshaping an Object in Level-1 Editing Mode

Chopping an Object in Level-1 Editing Mode

Snapping Soft Block Pins to Grid

Use the *Snap Pins* command to snap top-level and level-1 soft block pins to the nearest grids. This command works on both preselected and postselected pins.

At advanced nodes, pins are snapped to nearest WSP tracks. The options in the *Layer Awareness*, *Width Awareness*, and *Length Awareness* sections, which are available only at advanced nodes, let you control pin snapping.

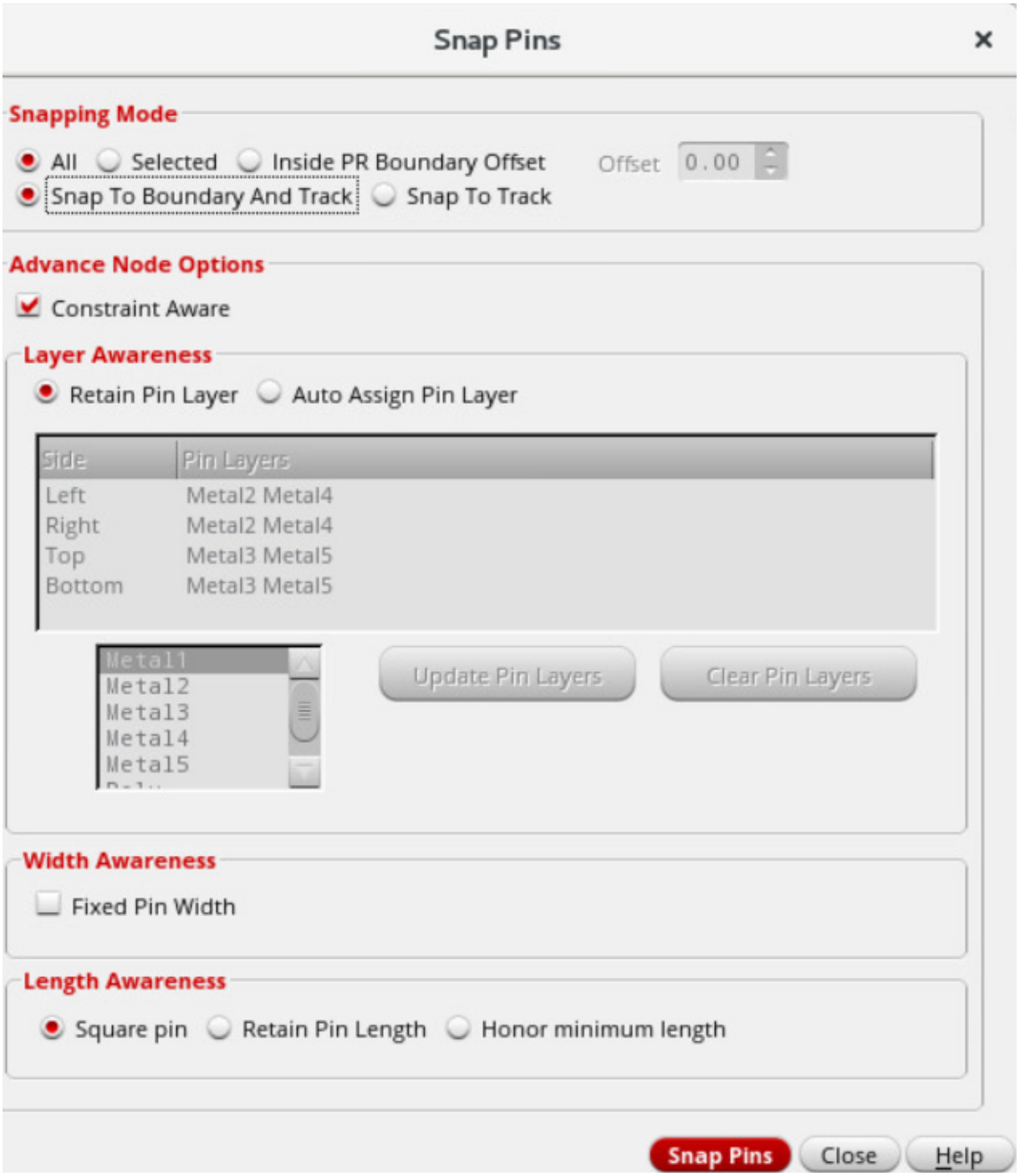
The *Advance Node Options* section is available only when there are WSP tracks in the design.

To snap soft block pins to appropriate grids:

1. Choose *Connectivity – Pins – Snap Pins*.

© 2023 Cadence Design Systems, Inc.

The Snap Pins form appears.



2. In *Snapping Mode,* select how the pins are to be snapped from the available options: *All* (default), *Selected*, *Inside PR Boundary Offset*, *Snap To Boundary And Track*, and *Snap To Track*.

© 2023 Cadence Design Systems, Inc.

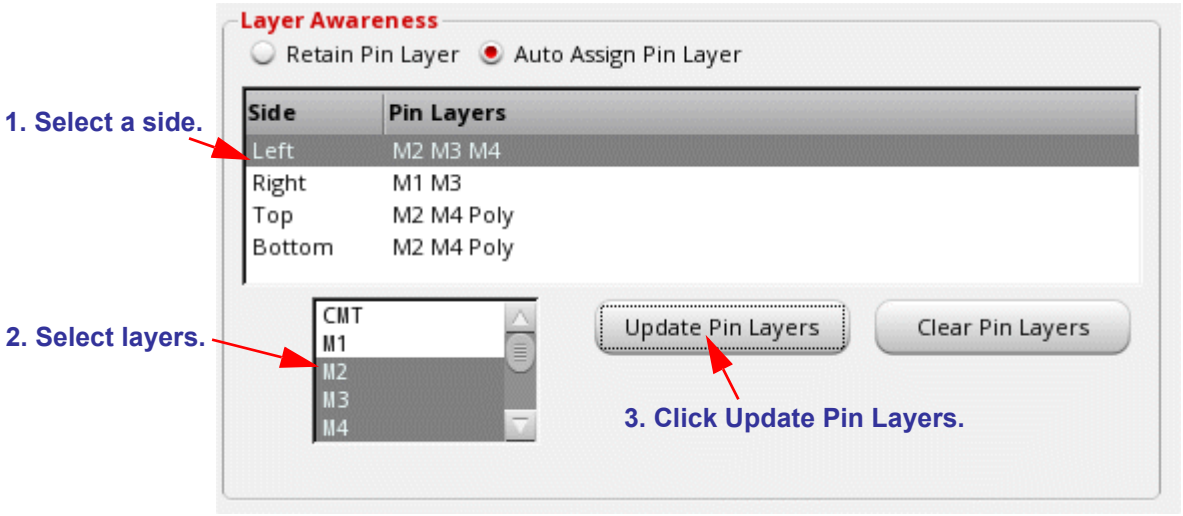
3. Select *Constraint Aware* in the *Advance Node Options* section to ensure that the following constraints are honored during pin snapping:

   Wire type, width, order, bus, side, edge, pin group guide, layer per side, blockages, and preplaced pins.

4. In the *Layer Awareness* section, specify the layers on which the pins are to be placed after they are snapped.

5. Select *Retain Pin Layer* to snap pins to the nearest WSP tracks, while retaining the pins in their original layers and honoring all applicable pin constraints.

6. Select *Auto Assign Pin Layer* to assign pin layers to the sides of the block.



The first box has two columns, *Side* and *Pin Layers*. *Side* lists the four sides of soft blocks. *Pin Layers* assigns pin layers to these sides. The sequence in which the layers are listed determines the layer priority for pin snapping.

To add or update the pin layer for a side:

a. Select the required side in the first box.

b. Select one or more layers in the second box.

c. Click *Update Pin Layers*. The selected pin layer is displayed in the *Pin Layers* column in the first box.

d. Click *Clear Pin Layers* to clear the pin layers of the selected side.

During pin snapping, when a pin layer is switched, the pin label associated with the pin is also switched to the same layer on which the pin is changed. However, if the pin layer

© 2023 Cadence Design Systems, Inc.

is not switched, the layer of the pin label will remain unchanged. Ensure that the pin label is associated with the pin.

7. Select *Fixed Pin Width* in the *Width Awareness* section to snap pins to the nearest WSP tracks with matching or higher pin widths.
8. In the *Length Awareness* section, specify the length of the snapped pins by selecting one of the options: *Square Pin*, *Retain Pin Length*, and *Honor Minimum Length.*
9. Click *Snap Pins*.

All the top-level and level-1 pins are snapped to appropriate grids.

The pins with placement status as `Fixed` are also snapped, and their statuses are retained as `Fixed`.

Related Topics

Snap Pins Form

Editing Objects

Snapping All Pins to Grid

To snap all soft block pins to the appropriate grid:

1. From the layout window menu bar, choose *Connectivity – Pins – Snap Pins*.

© 2023 Cadence Design Systems, Inc.

The Snap Pins form appears.



2. Ensure that *Snapping Mode* is set to *All*.
3. Click *Snap Pins*.

   All the top-level and level-1 pins are snapped to the appropriate grid.

© 2023 Cadence Design Systems, Inc.

The pins with placement status as `Fixed` are also snapped and the status is retained as `Fixed`.

Related Topics

Snap Pins Form

Snapping Soft Block Pins

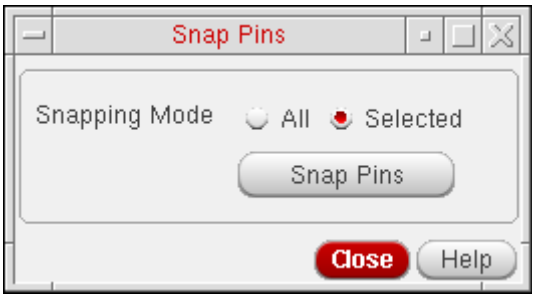
Snapping Selected Pins to Grid

Snap Pins can also snap selected top-level and level-1 pins to grid. The command works with both pre- and postselected pins.

To snap preselected pins to grid:

1. Select the pins you want to snap in the layout window or Navigator assistant.
2. From the layout window menu bar, choose *Connectivity – Pins – Snap Pins*.

   The Snap Pins form appears.
3. Ensure that *Snapping Mode* is set to *Selected*.



4. Click *Snap Pins*.

   The selected top-level and level-1 pins are snapped to the appropriate grid.

To snap postselected pins to grid:

1. From the layout window menu bar, choose *Connectivity – Pins – Snap Pins*.

   The Snap Pins dialog appears.
2. Ensure that *Snapping Mode* is set to *Selected*.
3. Select the pins you want to snap in the layout window or Navigator assistant.

© 2023 Cadence Design Systems, Inc.

4. Click *Snap Pins*.

   The selected top-level and level-1 pins are snapped to the appropriate grid.

In the *Selected* mode, when the pins with placement status as `Fixed` are snapped, the status of the pins is retained as `Fixed`.

Related Topics

Snap Pins Form

Snapping Soft Block Pins

© 2023 Cadence Design Systems, Inc.

Working With Pin Connectivity Models

The pin connectivity model for a design defines the connectivity pattern to be followed during routing. The model specifies how to connect nets to terminals, terminals to pins, and pins to figures.

Supported Pin Connectivity Models

Layout XL supports the following pin connectivity models:

- **Strong-connect model:** In the strong-connect model, a single connection point on a highly conductive metal is used to provide the required connectivity. You create a strong-connect group of the required pin figures.
- **Weak-connect model:** A weak-connect model is applicable in situations where a router must connect to a single pin, but the terminal has more than one pin, each of which has one or more shapes or pin figures. In this model, the router can connect to any shape of any pin, but it cannot use the pin as a feedthrough pin.

  This model is useful when two pins are connected using resistive wires, such as local interconnect.
- **Must-connect-model:** In the must-connect model, the drain, source, and gates of a MOS device are isolated. External, individual connections are made to these terminals.
- **Must-connect-all-pins model:** The must-connect-all-pins model is applicable to terminals that have more than one pin. This model applies the following settings:
  - Routers, such as Virtuoso Space-based Router (VSR), connect to all pins of the selected terminal.
  - The connectivity extractor flags all unconnected pins of the selected terminal.
  - Tools such as the wire editor display flight lines to all pins of the selected terminal.

Related Topics

Setting a Pin Connectivity Model

© 2023 Cadence Design Systems, Inc.

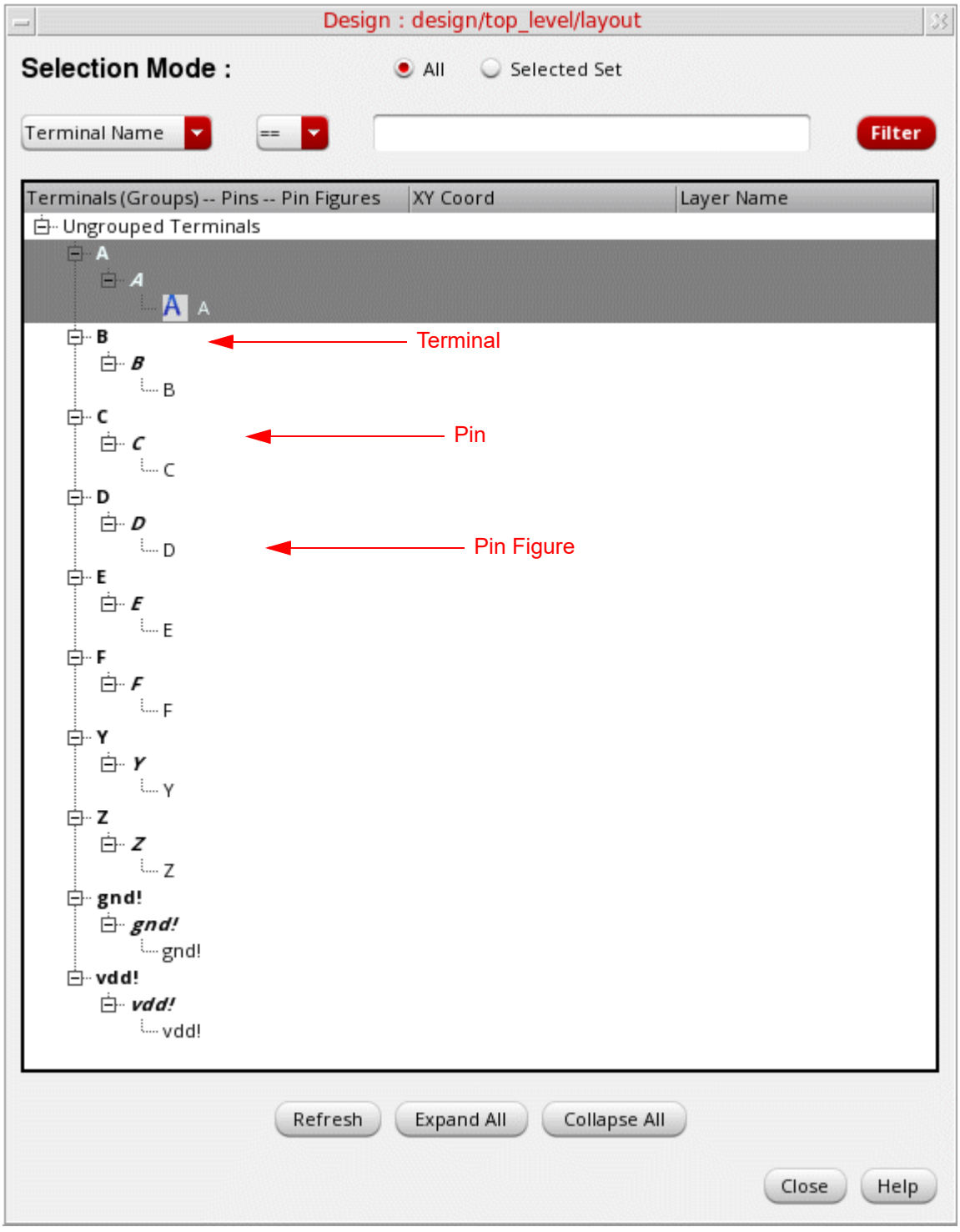
Setting a Pin Connectivity Model

To set a pin connectivity model:

1. Choose *Connectivity – Pins – Pin Connectivity Setting*.

© 2023 Cadence Design Systems, Inc.

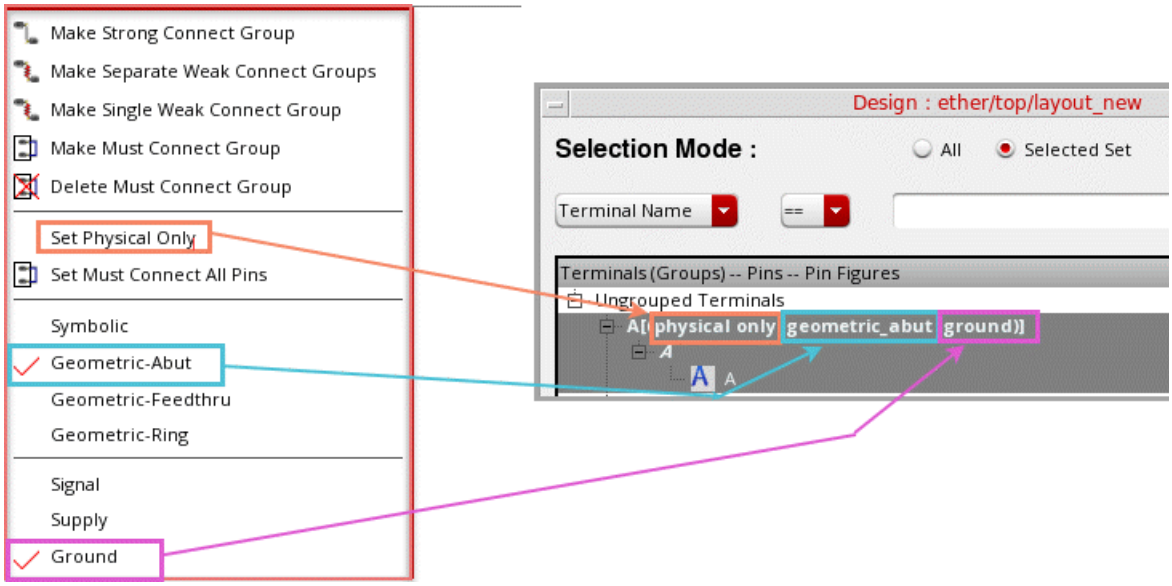
The Pin Connectivity Model form appears.



© 2023 Cadence Design Systems, Inc.

The Pin Connectivity Model form lists all terminals, pins, and pin figures in the current design in a tree structure.

- The first level in the tree structure lists the terminals in the design in bold, normal typeface, for example, A[(physical only|geometric_abut|ground)], which represents:
  - The terminal name
  - The physical configuration attribute (optional)
  - The route and pin connectivity method (optional)
  - The signal type (optional)



  - To show the physical configuration attribute, select the required terminal and choose *Set Physical Only* from the shortcut menu. To turn off this setting, select *Unset Physical Only* from the shortcut menu. By default, the setting is off.
  - To show the route and pin connectivity method, choose one of these options from the shortcut menu:

    - *Symbolic*

    - *Geometric-Abut*

    - *Geometric-Feedthru*

    - *Geometric-Ring*

© 2023 Cadence Design Systems, Inc.

- To show the signal type, choose one of the following options from the shortcut menu:

  - *Signal*

  - *Supply*

  - *Ground*

- When you expand the first level, the pins in each terminal are displayed in bold, italic typeface; for example, **A, B, C, D, E, F, Y, Z, gnd!***,* and **vdd!**.

- When you expand the pins, the pin figures are displayed in normal typeface; for example, **A, B, C, D, E, F, Y, Z, gnd!**, and **vdd!.**

2. Use the following options to filter the terminals, pins, and pin figures that are listed in the tree structure:

- *All* and *Selected*: By default all pins are listed. You can select the required terminals, pins, or pinFigs and then click *Selected* to restrict the list based on your selection.

- *Terminal Name* and *Pin Name*: Lets you filter pin names based on the specified terminal names or pin names.

- *Expand All* and *Collapse All*: Lets you traverse the tree structure.

The Pin Connectivity Model form is synchronized with the layout canvas. Any selection or change you make in the form is reflected in the layout canvas and conversely. For example, if you select a pin in the form, it is automatically highlighted in the layout canvas. After making changes to pins in the layout canvas, for example after creating a new pin, click *Refresh* in the Pin Connectivity Model form to see the changes in the form.

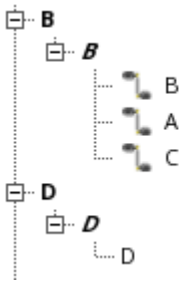
3. Right-click the individual terminals, pins, or pin figures for which you want to set a pin connectivity model.

4. Choose the required pin connectivity model. The available options are:

- ***Make Strong Connect Group:*** The pin figure you select first is considered the anchor pin, and the **A** symbol is displayed beside its name. The selected pins are

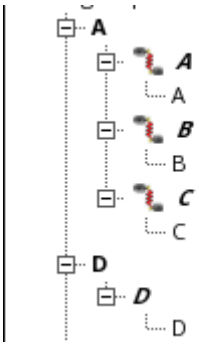
© 2023 Cadence Design Systems, Inc.

placed in a strong-connect group. All pins are moved to the terminal corresponding to the anchor pin. All empty terminals are deleted.



You can also drag-and-drop a pin on another to create a strong-connect group.

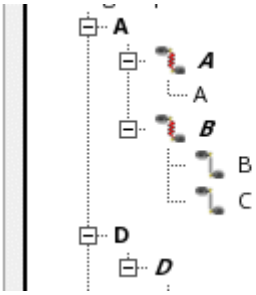
- ***Make Separate Weak Connect Groups:*** The pins are weakly connected and placed in separate pin groups. These pin groups are listed under the terminal of the anchor pin. All empty terminals are deleted.



Caution

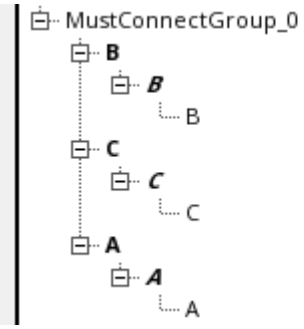
Poly, being resistive, cannot be used as a feedthrough.

- *Make Single Weak Connect Group*: The pins are weakly connected and placed in single groups.



© 2023 Cadence Design Systems, Inc.

- ***Make Must Connect Group:*** The terminals are connected and are placed together in a must-connect group.



  To delete a must-connect group, right-click a terminal in the group and choose *Delete Must Connect Group*.

- ***Set Must Connect All Pins:*** The must-connect-all-pins model is applied to the selected terminal. To remove the setting, right-click and choose *Unset Must Connect All Pins*.

Alternatively, you can use one of the following methods to set a pin connectivity model:

- Choose a model from the *Connectivity – Pins* menu.
- Use database SKILL functions.

Related Topics

Pins

Smart Display

Smart Display (SMD) is a layout editing environment that can be used to generate a symbolic representation of the layout by using one of the two modes, *Abstracted* or *Schematic*.

In *Abstracted* mode, you see a MOS representation of the layout. The layers visible in this mode are diffusion and poly. The diffusion layers are recolored to help identify PMOS and NMOS cells, red for PMOS and green for NMOS. This abstract representation helps in the placement of devices.

In *Schematic* mode, MOS devices are represented using their schematic symbols and connections are displayed using symbolic flight lines. All physical layers are hidden from view.

© 2023 Cadence Design Systems, Inc.

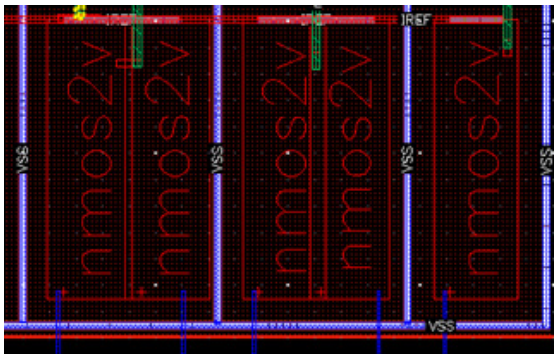
Because of the reduced data and symbolic connections, designs in this mode are simple to understand and debug. This makes it easier for users to exchange designs.

In *Abstracted* mode, if required, you can choose to abstract only a specific set of objects. For this, you need to select the objects before you enable *Abstracted* mode. In *Schematic* mode, the entire cellview is abstracted, and not just the objects that are selected in the layout.

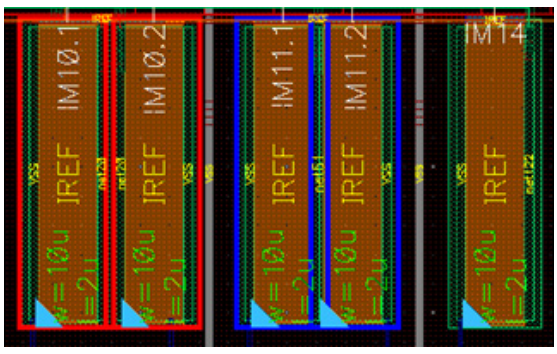
In both *Abstracted* and *Schematic* modes, you can also see an abstract representation of resistors, capacitors, and hierarchical blocks. Any other surrounding objects are by default dimmed. They can also be hidden from view. Moreover, you can choose to display a specific set of connectivity objects, such as wires, vias, flight lines, and pins, and information that you consider relevant, such as the instance name and device size.

Devices are shown using their real size, which helps circuit designers generate an accurate placement prototype. Flight lines and the number of tracks help estimate the channel size before routing.

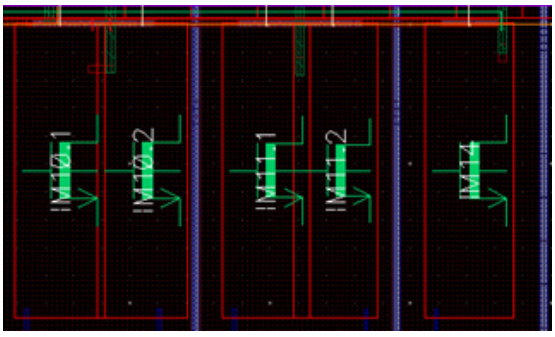
The following figure compares a design object in different modes:



Smart Display = Off



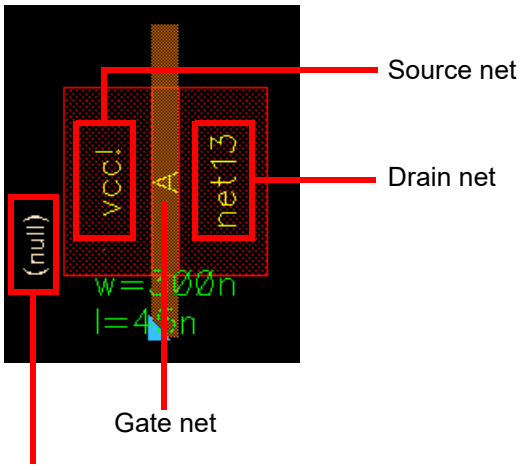
Smart Display = Abstracted



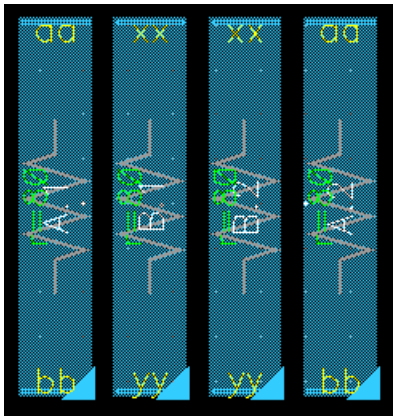
Smart Display = Schematic

© 2023 Cadence Design Systems, Inc.

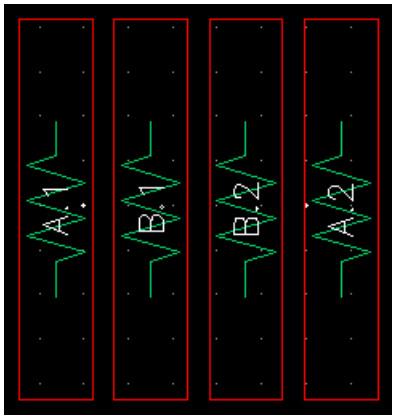
The following figure shows the net names for a PMOS device in *Abstracted* mode:



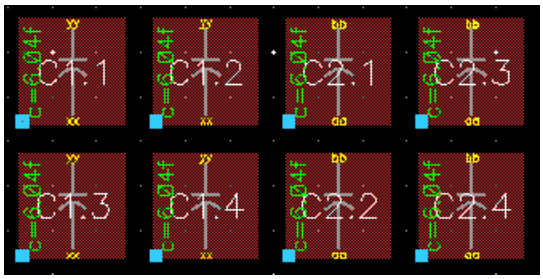
The following figure shows capacitors and resistors in *Abstracted* and *Schematic* modes:



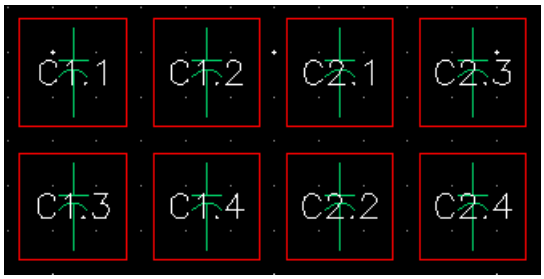
Resistors (Abstracted)



Resistors (Schematic)



Capacitors (Abstracted)



Capacitors (Schematic)

© 2023 Cadence Design Systems, Inc.

Exporting the Design in Smart Display Mode

To save the design in *Abstracted* or *Schematic* mode to an image file, choose *File – Export Image* in the layout window. In the Export Image form that is displayed, specify the required options, including the path for the file to which you want to save the design, and click *Save to File*.

Related Topics

Enabling Smart Display

Accessing Smart Display Options

Display of Mosaics in Abstracted Mode

Guard Rings in Abstracted Mode

Enabling Smart Display

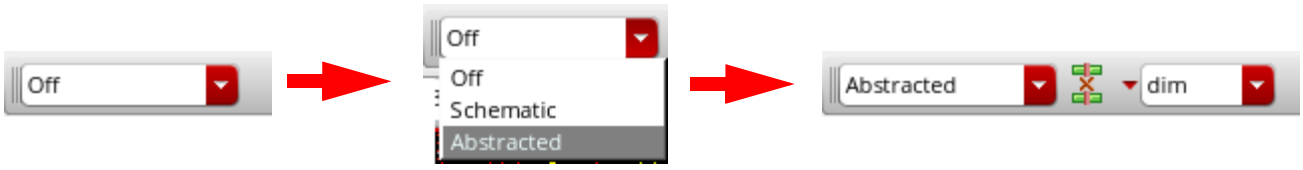
Hierarchical Blocks in Abstracted Mode

figGroups in Abstracted Mode

Smart Display Options Form

Enabling Smart Display

The *Smart Display* toolbar is displayed by default. To turn on Smart Display, select either *Abstracted* or *Schematic* from the *Display Mode* drop-down list, as shown below:



After you enable the required Smart Display mode, you can see listed on the toolbar the *Connectivity* and *Surround Display* options.

© 2023 Cadence Design Systems, Inc.

To display the Smart Display toolbar, do one of the following:

- Choose *Window – Toolbars – Smart Display*.
- Right-click anywhere in the menu bar, and then choose *Toolbars – Smart Display*.

Related Topics

Enabling Smart Display

Accessing Smart Display Options

Accessing Smart Display Options

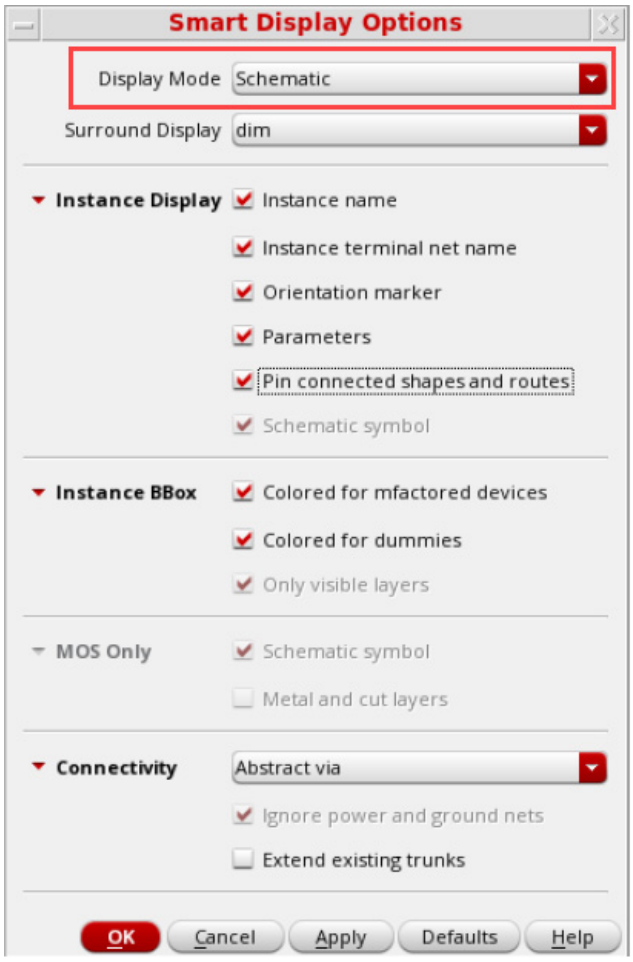
To display Smart Display Options form:

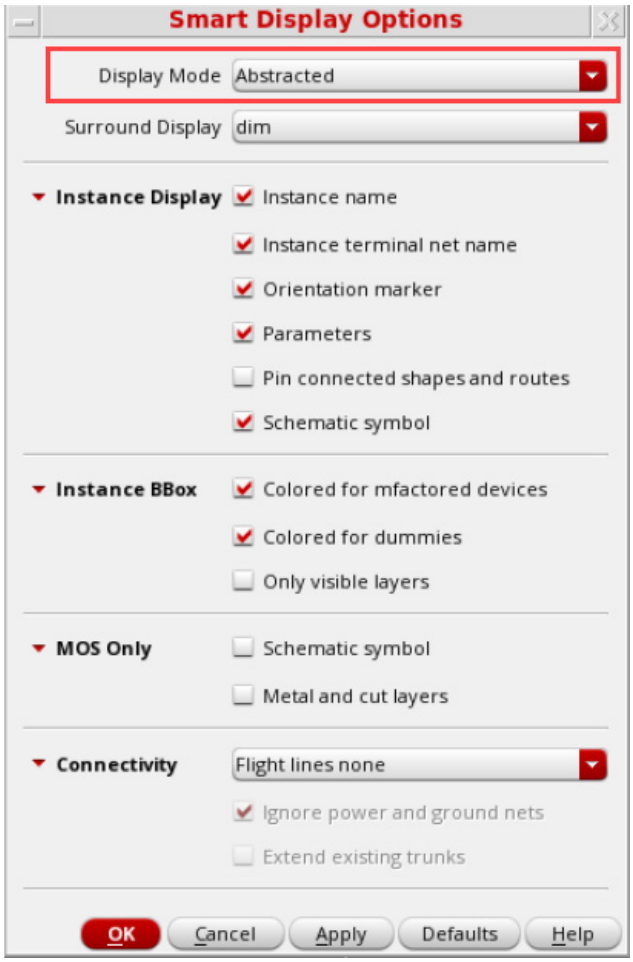
- Choose *Options – Smart Display*.

  The Smart Display Options form is displayed. The figures below show the default settings for *Abstracted* and *Schematic* modes.

By default, all option categories are in a collapsed state. To expand a category to view the options listed under it, click the category name.

© 2023 Cadence Design Systems, Inc.





The options that are available both in the form and on the toolbar are synchronized. For example, if you select the display mode as *Abstracted* in the form and click *Apply*, the value in the toolbar is updated to reflect this change.

Related Topics

Smart Display Options Form

Display of Mosaics in Abstracted Mode

The following features define the display of mosaics in *Abstracted* mode:

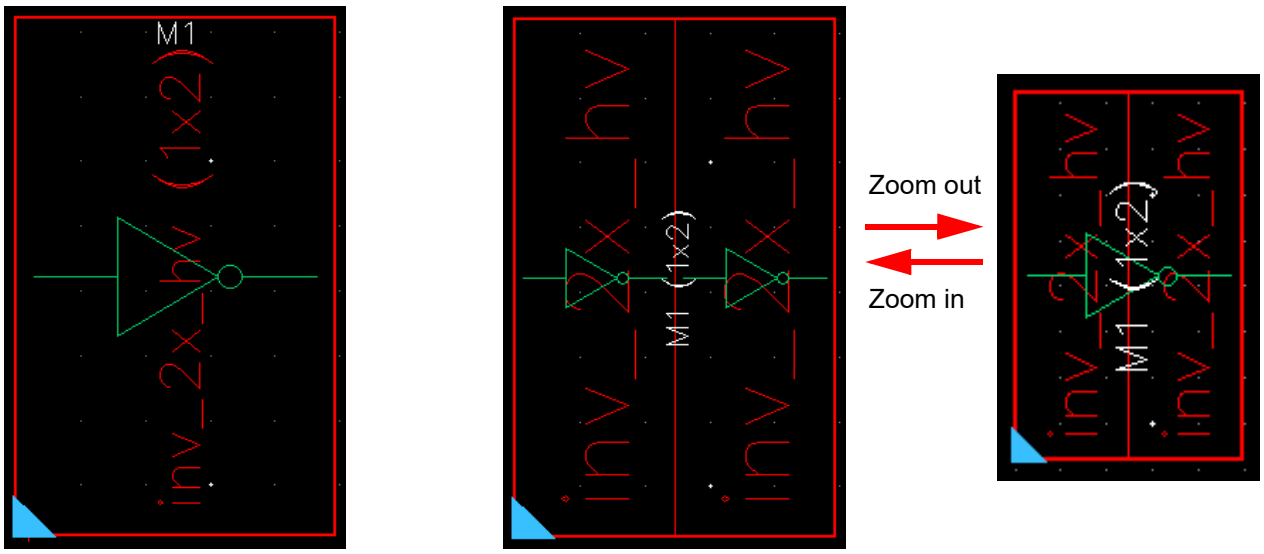
- An abstract representation of the unit cell is repeated.

© 2023 Cadence Design Systems, Inc.

- A bounding box is added for the mosaic on the boundary or visible layer.
- An array representation of the mosaic is displayed and a label specifying the array dimension is added to the mosaic instance name. For this, the *Array Icons* check box in the Display Options form must be selected.

  For information about the Display Options form, see Setting Display Options in *Virtuoso Layout Suite XL: Basic Editing User Guide*.

The abstract display changes with the display level. The following figure shows how a mosaic with a 1x2 array is displayed in *Abstracted* mode when the display level is set to 0.

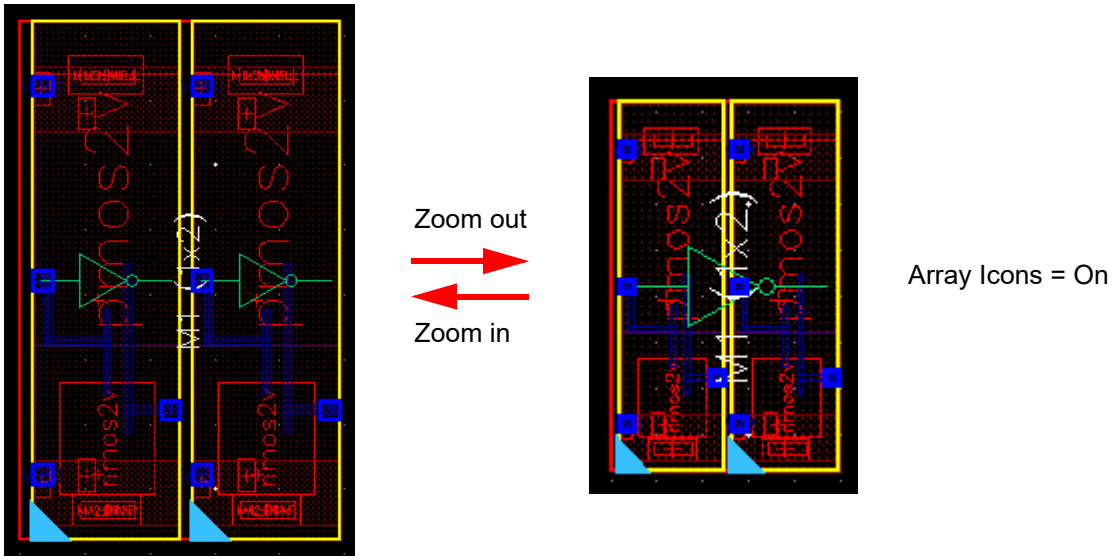


Array Icons = Off

Array Icons = On

© 2023 Cadence Design Systems, Inc.

The following figure shows how a mosaic with a 1x2 array is displayed in *Abstracted* mode when the display level is set to 1.



Zooming out the design displays one large symbol for the entire mosaic instance if the number of pixels defined for the schematic symbol of a unit cell goes below the threshold.

Related Topics

Setting Display Options

Guard Rings in Abstracted Mode

Supported objects include MPP guard rings and fluid guard rings. Substrate contact is not supported. When you turn on Smart Display in *Abstracted* mode, an abstract representation of guard rings is automatically displayed.

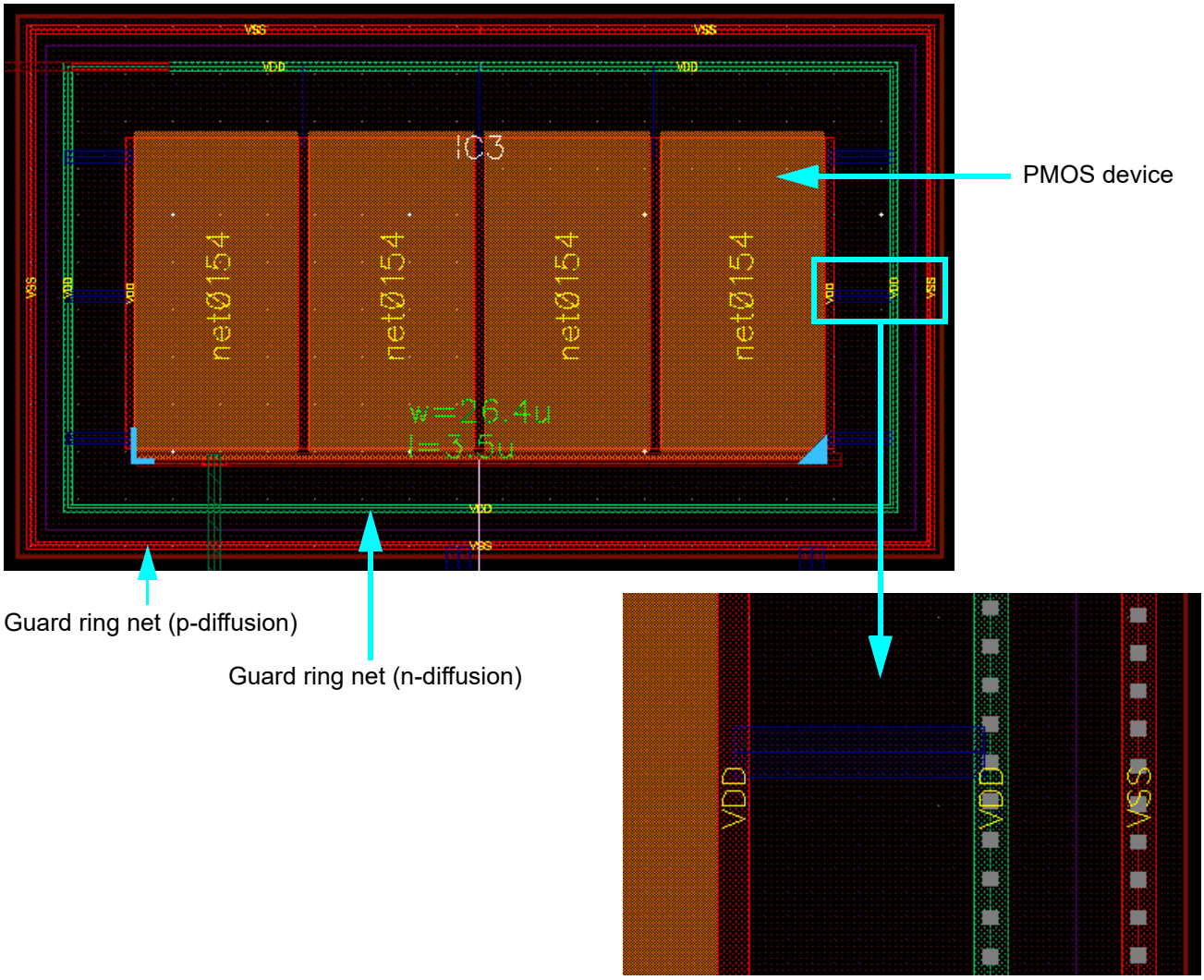
The following features define the display of guard rings in *Abstracted* mode:

- Visible layers include diffusion and contact. The diffusion layer is displayed in green for n-diffusion guard rings and in red for p-diffusion guard rings.
- Connectivity is shown using net labels.

© 2023 Cadence Design Systems, Inc.

MPP Guard Rings

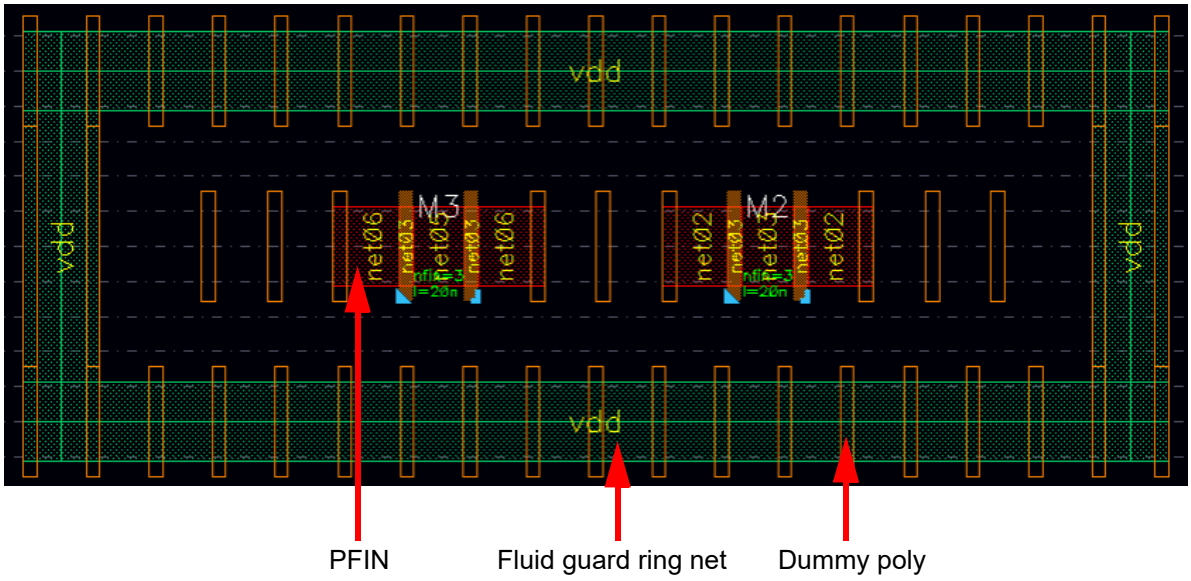
The following figure illustrates MPP guard rings in *Abstracted* mode:



© 2023 Cadence Design Systems, Inc.

FinFET Fluid Guard Rings

The following figure illustrates an n-diffusion FinFET fluid guard ring in *Abstracted* mode:



Related Topics

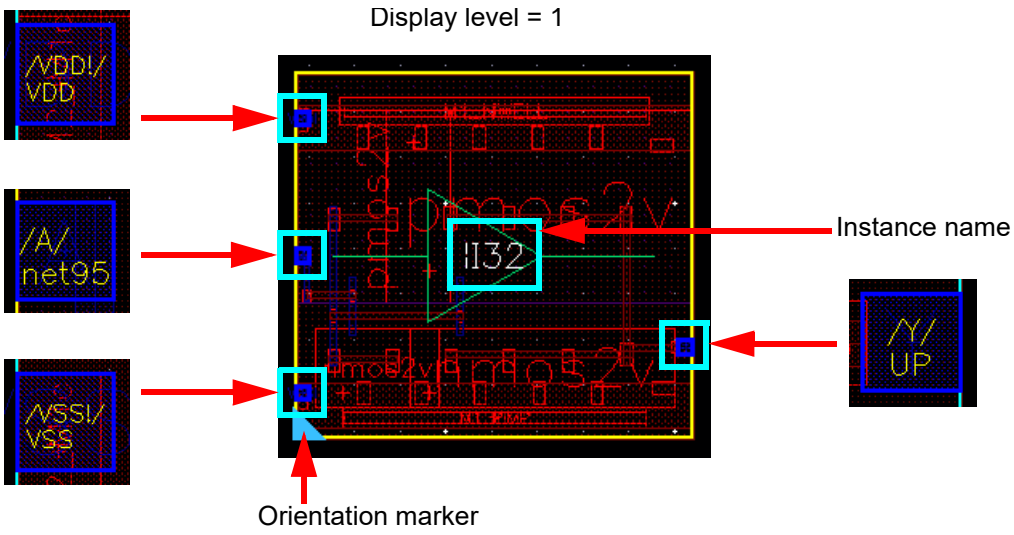
Smart Display

Hierarchical Blocks in Abstracted Mode

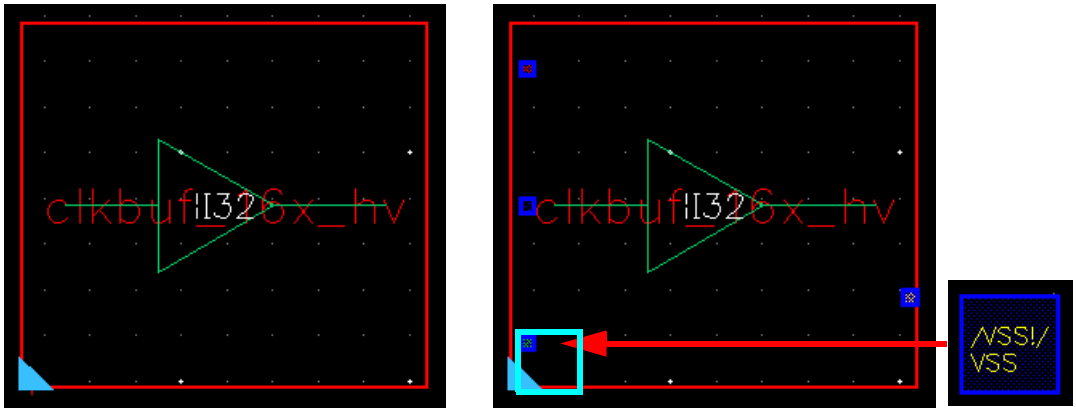
Smart Display lets you abstract pin connectivity information—the pin name and the name of the top-level net—from hierarchical blocks. Visible layers include level-1 pin layers and

© 2023 Cadence Design Systems, Inc.

boundary layers. You can also view the instance name and the orientation marker for hierarchical blocks, as shown in the figure below:

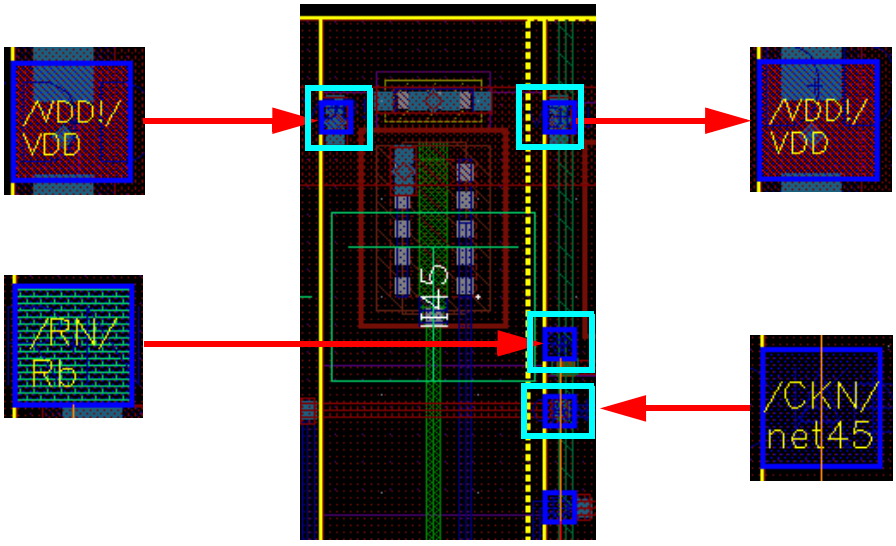


To view pin connectivity information with the display level set to 0, select the *Instance Pins* check box in the Display Options form.

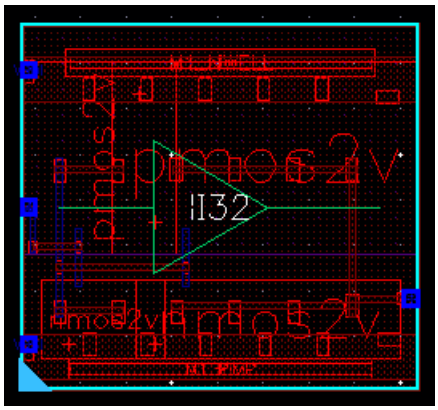


© 2023 Cadence Design Systems, Inc.

The following image shows pins inside a hierarchical block as seen in *Abstracted* mode with *Surround Display* set to *pins*:



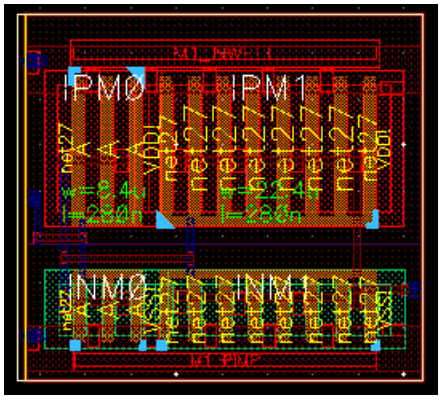
To view the PR boundary for instances, choose *P&R Boundary* from the *Instance Drawing Mode* list in the Display Options form. The PR boundary is displayed as shown below:



For information about the Display Options form, see Setting Display Options in *Virtuoso Layout Suite XL: Basic Editing User Guide*.

© 2023 Cadence Design Systems, Inc.

Pin connectivity information is not available when you run the *Descend edit* or *Edit In Place* commands on a hierarchical block, as shown below:



Related Topics

Smart Display

figGroups in Abstracted Mode

The following features define the display of figGroups in *Abstracted* mode:

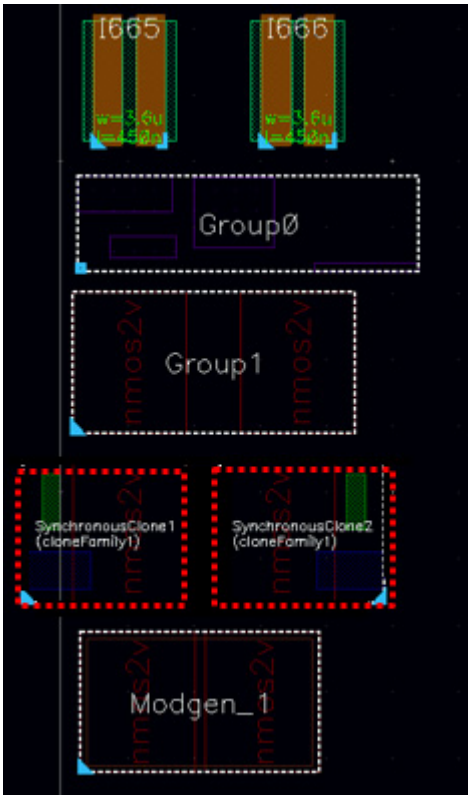
- An orientation marker is displayed in the abstracted view of a figGroup.
- A colored bounding box for each figGroup can be added.

The abstract display changes with the selected display level.

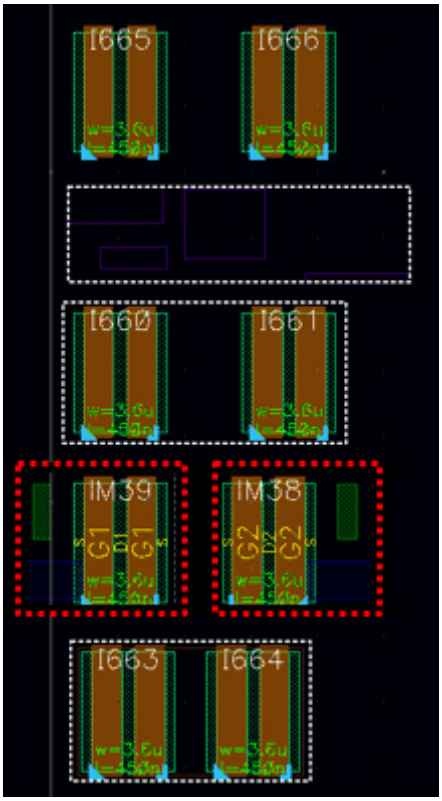
- At display level 0, the name of the figGroup is displayed but the visibility of instances within the figGroup is blocked, which is similar to the display of regular instances at that level.
- At display level 1, you can look inside the figGroup, although the figGroup name is not displayed any more and dimming is not supported.

© 2023 Cadence Design Systems, Inc.

The figures below illustrate how the abstract display of figGroups changes based on the selected level of display.



Display level = 0



Display level >0

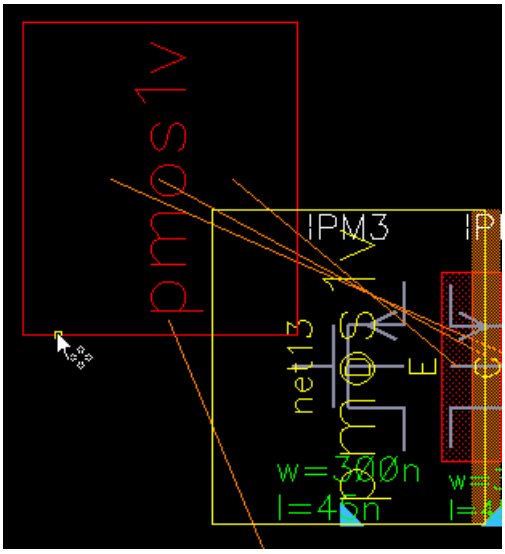
Related Topics

Smart Display

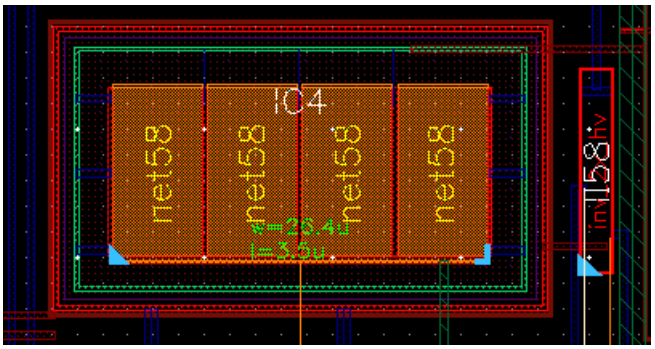
© 2023 Cadence Design Systems, Inc.

Limitations of Smart Display

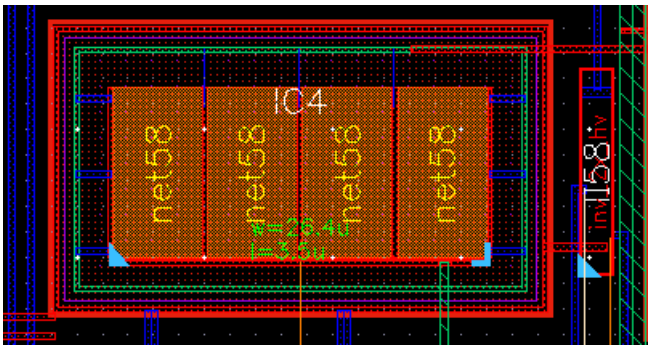
- The visible layers and smart labels are lost during a drag operation, as shown below:



- The dimming effect is lost when you export a design in *Abstracted* or *Schematic* mode to an image file, as shown below:



As seen in the canvas



As seen in the exported image

- Smart Display re-abstracts all the figures larger than 20 pixels every time the pan or zoom commands are invoked. To minimize the performance impact in a large design, the

© 2023 Cadence Design Systems, Inc.

abstraction stops with a warning when the maximum count of 500 instances is abstracted.

- The Smart Display device abstraction device abstraction displays MOS poly in solid stipple, making any text difficult to read.

  To change the poly stipple, use either:

  Method 1

  a. Enable Smart Display in *Abstracted* mode and open the Display Resource Editor.

  b. Change view mode from *All LPPs* to *All Packets*.

  c. Search for `smdPacketPoly` and select it.

  d. Change the stipple from *solid* to *dots*. Click *Modify Current Packet*.

  Method 2

  In the Display Resource Editor, choose the *File – Save/Load* to directly change the `smdPacketPoly` as:

  ```
  ( display smdPacketPoly   solid none orange orange stipple )
  ```

Related Topics

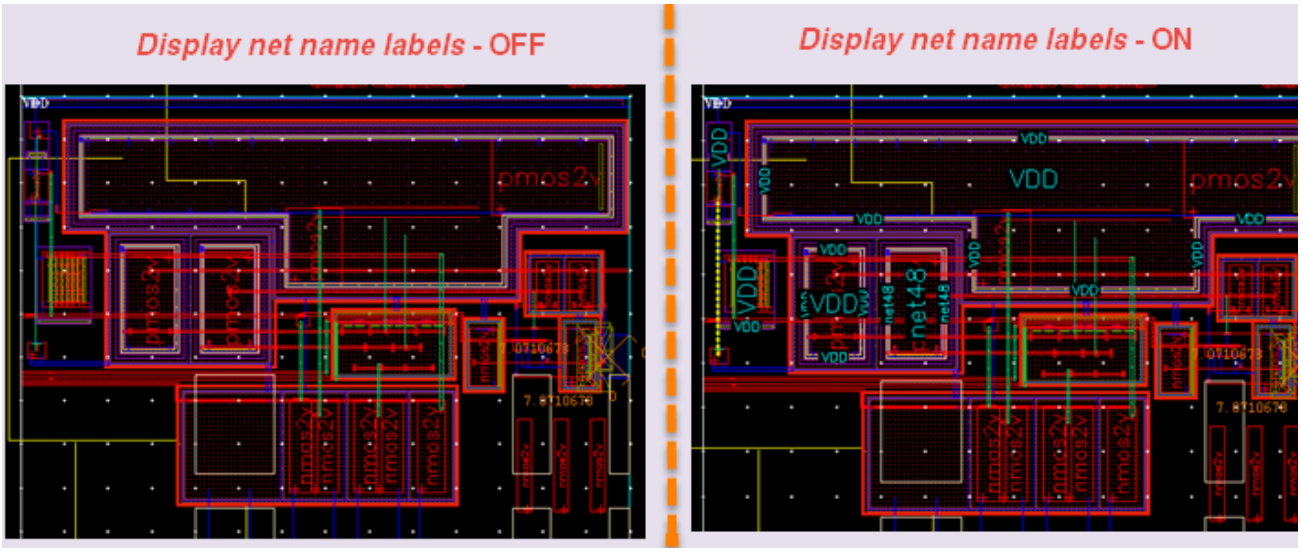
Smart Display

Net Name Display

Layout XL enables you to add net name labels in the layout canvas for shapes that have associated labels. Having the net name labels display in the layout canvas makes layout editing simpler, especially for large designs.

© 2023 Cadence Design Systems, Inc.

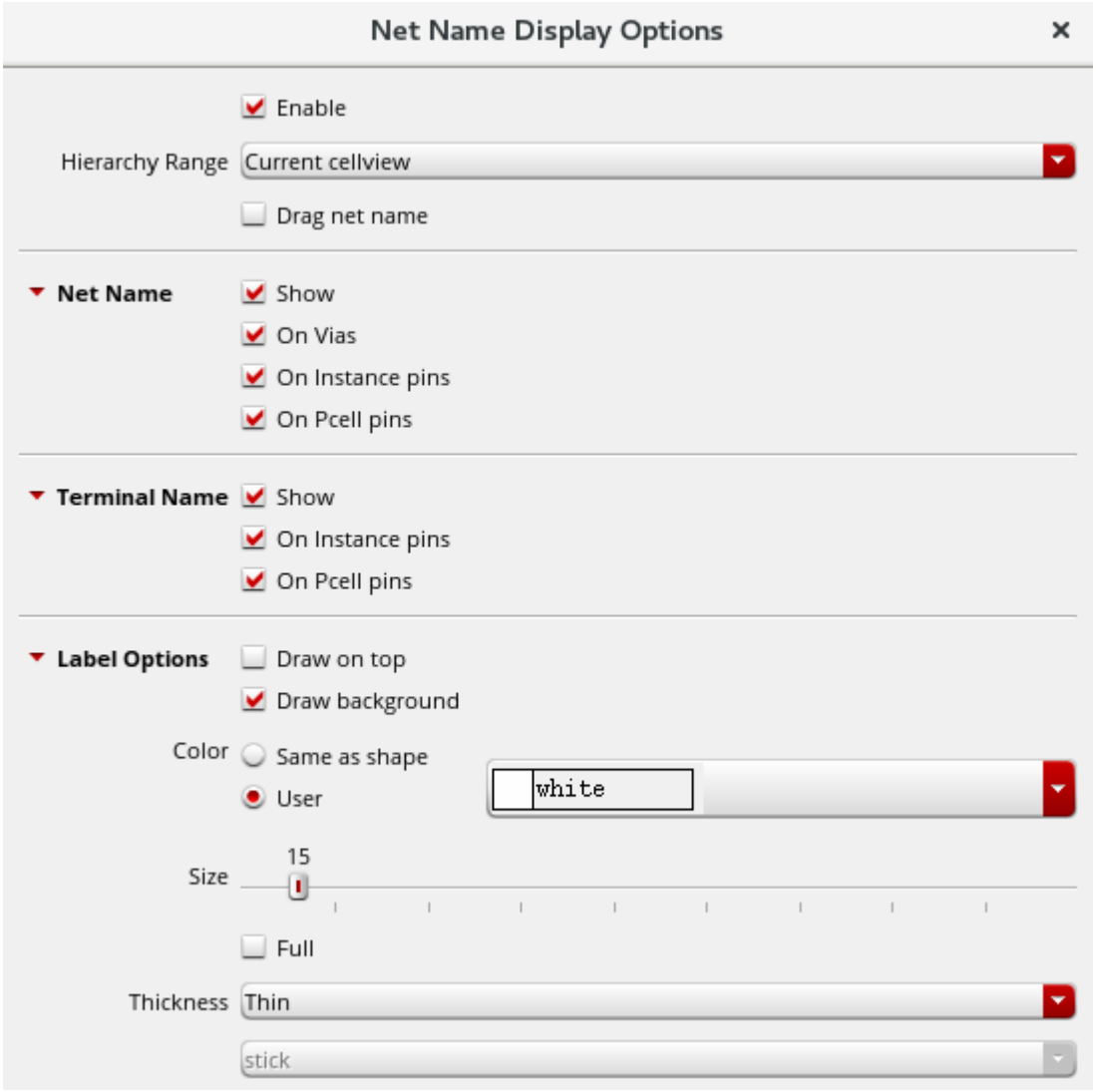
As illustrated in the figure below, having the net name labels display on the canvas make it easy to identify the nets that are associated with a shape in the cellview.



You can use the *Options – Net Name Display* command to enable the display of net name labels for shapes, or use the `geEnableNetNameDisplay` SKILL function.

© 2023 Cadence Design Systems, Inc.

The *Net Name Display* command opens the Net Name Display Options form that lets you customize the label display. Alternatively, you can use the `geNetNameDisplayOptionForm` SKILL function to open the form.



You can also use the associated Options toolbar icon  , to launch the Net Name Display Options form.

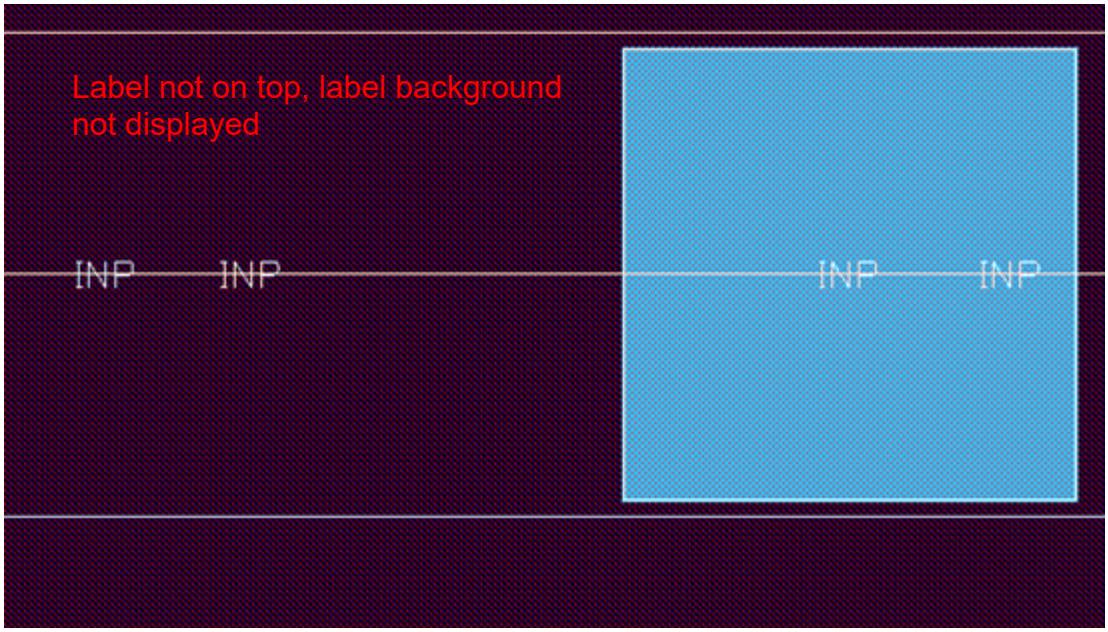
The toolbar displays a slightly different icon  , when *Name Filtering* is selected in the Net Name Display form. Net Name Filtering enables you to selectively display (or hide) the net name labels in the layout canvas.

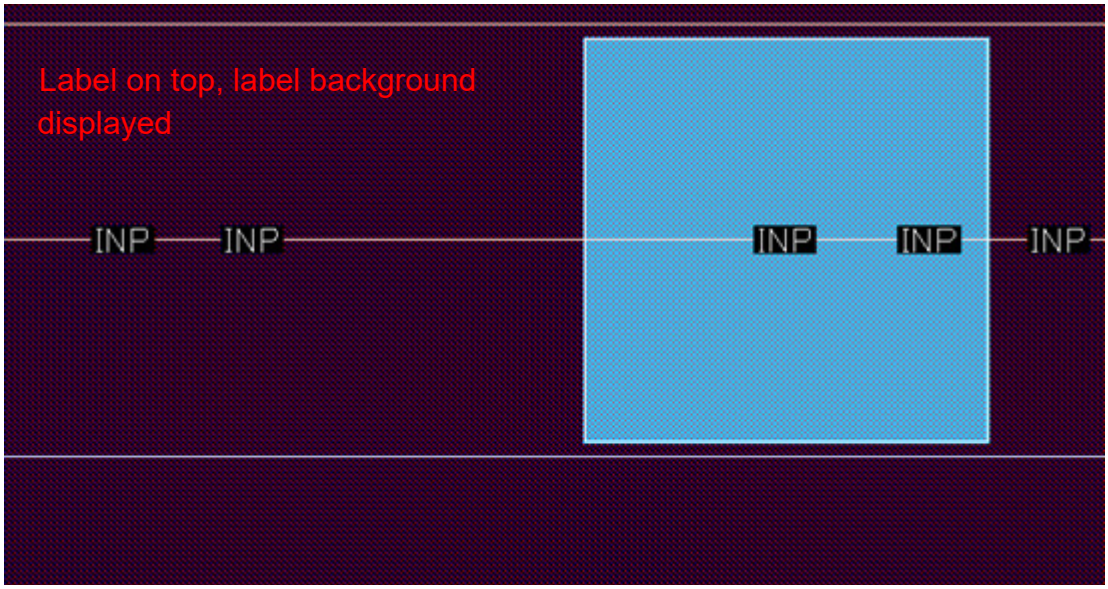
By default, the display of net name labels in the layout canvas is enabled. You can use the *Display net name labels* option on the form to turn the label display off. Alternatively, you

© 2023 Cadence Design Systems, Inc.

can use the `netNameDisplayEnabled` environment variable to control the net name display.

Layout XL lets you choose whether to display the label background as well on top of the shape, as displayed in the figure below. Displaying the label background makes the label appear prominently on the canvas. The figures below illustrate how adding a background can make the label appear more prominent.

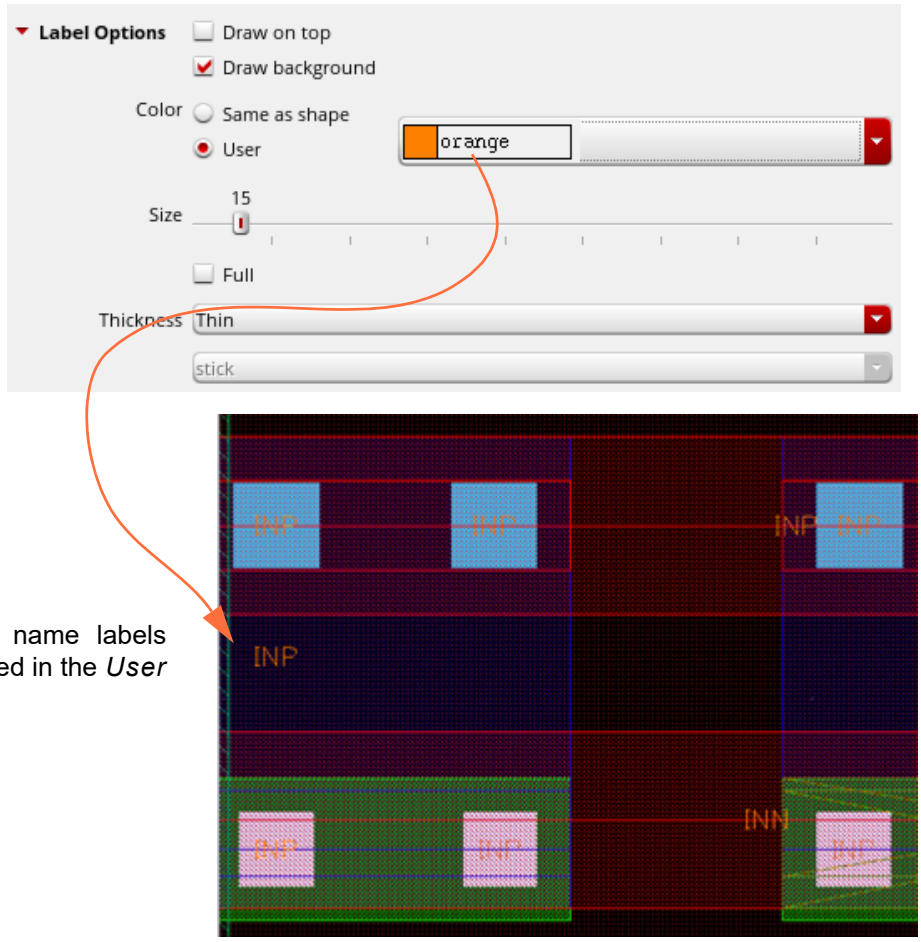




© 2023 Cadence Design Systems, Inc.

Depending on the device colors in the design, you can choose an appropriate color for the labels, or draw the label in the same color as that of the shape.

The figure below displays all the net labels in the specified color.



All net name labels displayed in the *User* color

Related Topics

Net Name Display Options Form

geEnableNetNameDisplay

geNetNameDisplayOptionForm

netNameDisplayEnabled

© 2023 Cadence Design Systems, Inc.

Net Name Filtering

Name filtering lets you selectively display the net name labels on the canvas. You can choose an appropriate filtering mode to selectively display net name labels on the canvas, or hide them. Alternatively, you can use the `netNameDisplayFilterMode` environment variable to specify the filter mode to use.

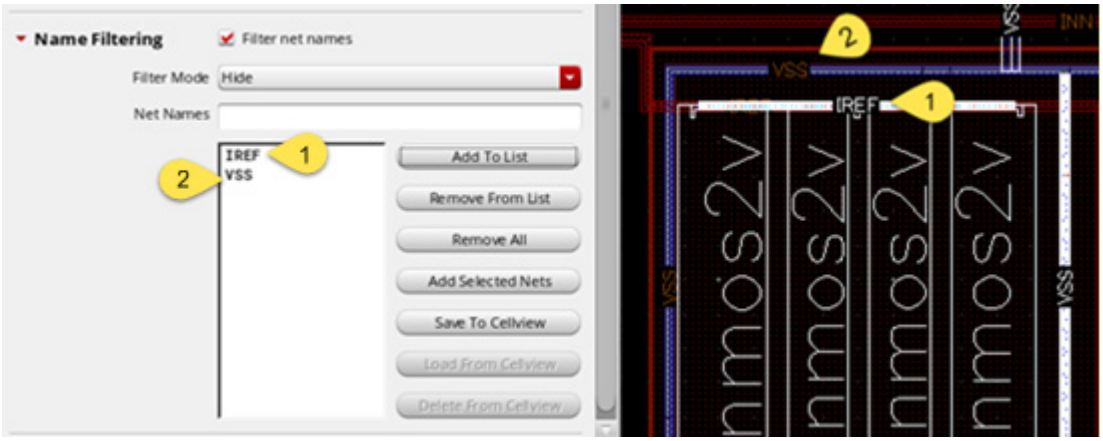
You can type in and add the names of the nets on which the name filter needs to be applied. Or, you can remove nets from an already existing filter list to keep them out of the filtering options applied. You can also specify a default filter list using the `netNameDisplayFilterDefaultList` environment variable, if no information is already saved in the design.

By default, name filtering is OFF. But, when enabled, the *Options* toolbar icon that controls the net name display functionality, looks slightly different,  , also displaying a filter symbol.

Let us consider the scenario represented by the figure below to explore how the *Name Filtering* options can be used to control the selective display of net name labels.

In the figure below, net name filtering is enabled and the filter mode is set to *Hide*. This implies that the net name labels for only the specified nets, `IREF` and `VSS`, are hidden. All other nets will have their net name labels displayed on the canvas.

If the filter mode was set to *Show*, only the nets in the specified filter list would have their net name labels displayed.



Related Topics

netNameDisplayFilterMode

netNameDisplayFilterDefaultList

© 2023 Cadence Design Systems, Inc.

Setting the Net Name Display Options

By default, net name labels are displayed in the layout canvas. You can turn off the display, if required, by deselecting the Display net name labels option in the Net Name Display Options form. Or, by setting the `netNameDisplayEnabled` environment variable to `nil`.

To specify the net name display:

1. Choose the appropriate options for displaying the labels.

   Choose whether you want the labels to be displayed on top of the shape and if you want the labels to be drawn with a background to make them more prominent.

2. Choose a color for the label, or choose to display the label in the same color as the shape.

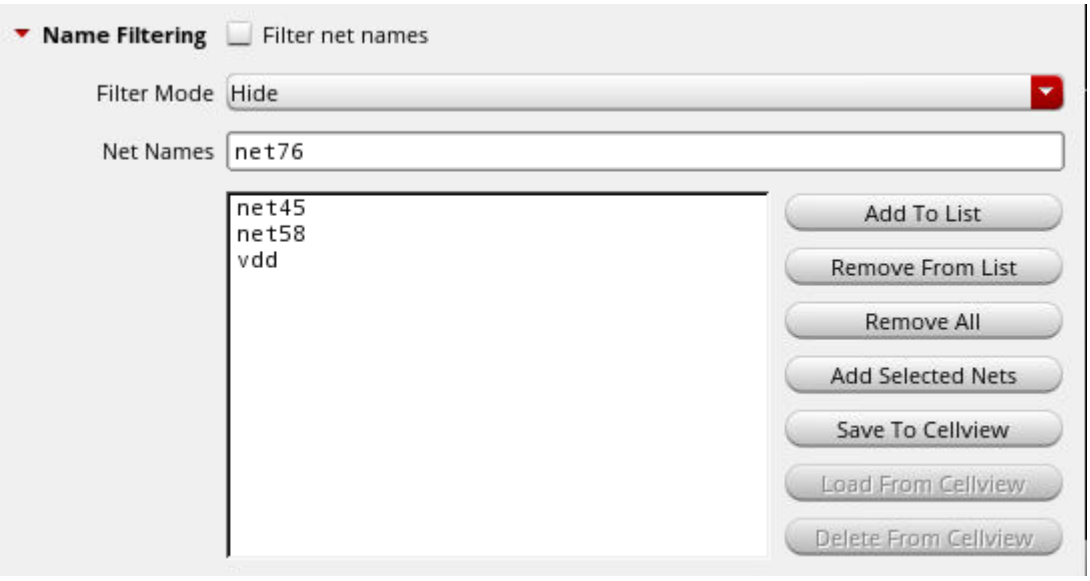
3. (*Optional*) Choose whether you want to display labels on vias, or skip them. By default, labels are displayed on vias.

4. (*Optional*) Select the *Filter net names* option to enable the name filtering functionality that allows you to selectively display or hide the net name labels for the specified nets.

5. Set the *Filter Mode* to *Show* or *Hide*.

   Environment variable: netNameDisplayFilterMode

6. (*Optional*) In the *Net Names* field, type the names of the nets for which the name filtering needs to be applied and click *Add To List,* as illustrated in the figure below.



- The *Net Names* field is case-sensitive.

© 2023 Cadence Design Systems, Inc.

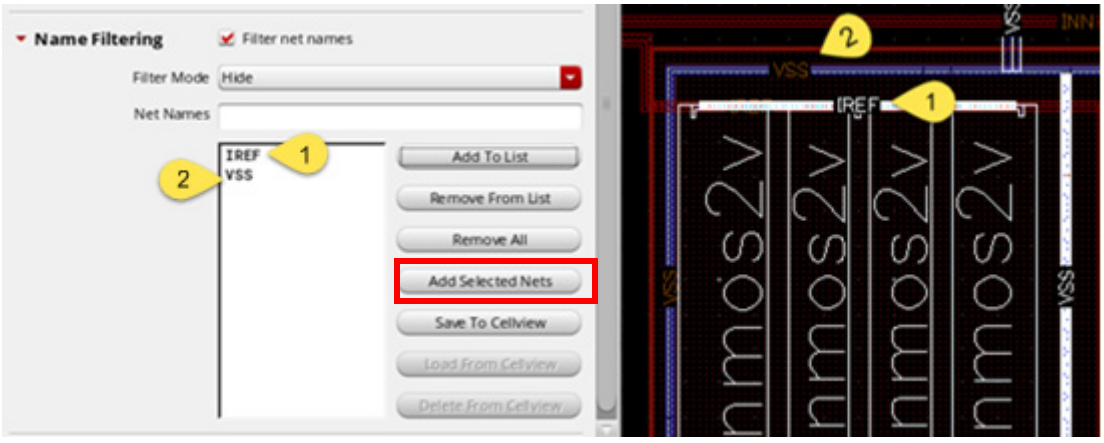
- You can type in multiple net names in the Net Names field and add them to the list in one go, but, keep the net names space-separated.
- Alternatively, you can specify the net names to be filtered by selecting the nets in the Navigator assistant or the layout canvas and clicking the *Add Selected Nets* button.

  See Adding Selected Nets.
- You can provide a default net name list to be used by using the netNameDisplayFilterDefaultList environment variable.

7. (*Optional*) To remove net names from an already existing filter list, select the net name and click *Remove From List*.
   - To remove the entire filter list, click *Remove All*.
8. (*Optional*) Click *Save To Cellview* to save the current filtering options to the cellview such that the same settings can be reloaded later when the cellview is opened in a different window.
   - The *Save To Cellview* option is enabled only when the current cellview is editable.
   - To revert to the filtering options previously saved for the cellview, click the *Reload From Cellview* button. This option is enabled only when some design information is saved.
   - To delete the existing filter settings saved in the cellview, click the *Delete From Cellview* button.

     See Saving To Cellview.

Adding Selected Nets



© 2023 Cadence Design Systems, Inc.

Add Selected Nets lets you add nets to the filter list by selecting the nets in the Navigator assistant or from the layout canvas. This option can be used when a large number of nets need to be added to the filter list.

To add selected nets to the filter list:

➡ Select the nets from the Navigator assistant or the canvas and click *Add Selected Nets*.

Alternatively, when the *Routing Object Granularity* option on the Selection Options form is set to *Net*, you can hover the mouse over a shape in the layout canvas and choose *Select – Net* from the shortcut menu. Then, clicking the *Add Selected Nets* option in the Net Name Display Options form adds the selected net (or nets) to the filter list.

When the number of nets to be added to the filter list is small, it might be quicker to type in the net names in the *Net Names* field and click *Add To List*.

Saving To Cellview

Save To Cellview lets you save the current filtering options to the cellview that you can reload later in a different window. Any edits that you make after the save cannot be reloaded.

Example

Let us consider a scenario that illustrates the use of the *Save To Cellview* and *Reload From Cellview* options.

Let us assume that you start with a clean filter list when you launch Virtuoso. As you edit the layout, you add the nets `VDD` and `VSS` to the filter list and click *Save To Cellview*. You have now saved the current filter list to the cellview.

If you were to now also add net `VDD2` to the filter list, you will have a filter list that displays all the three nets—`VDD`, `VSS`, and `VDD2`. But, your saved list will still only have nets `VDD` and `VSS`. Therefore, if you choose to revert to the saved list by clicking *Reload From Cellview*, the filter list in the form will be updated to display only `VDD` and `VSS`.

Related Topics

Selection Options Form

Net Name Display Options Form

netNameDisplayBackground

© 2023 Cadence Design Systems, Inc.

netNameDisplayDrawOnTop

netNameDisplayFontStyle

netNameDisplayUseShapeColor

netNameDisplayFilterMode

netNameDisplaySkipVias

netNameDisplayFilterDefaultList

geIsNetNameDisplayActiveOnWindow

geIsNetNameDisplayEnabled

Composite Waveguides (Virtuoso Photonics Option)

You will need the `Virtuoso_Photonics_Option` license to use the functionality discussed in this section. For information on obtaining the required license, contact your local Cadence representative.

If you are using the Virtuoso Photonics Solution and have the `Virtuoso_Photonics_Option` license checked out, Layout XL enables you to use optical connections called waveguides in your design. When several such optical connections or waveguides are used in a design, the resultant waveguide is called a composite waveguide.

The Virtuoso Photonics Solution supports the generation of such composite waveguides using the *Generate All From Source* command, and supports editing of the composite waveguides using the Composite Waveguide Editor. Each component of the composite waveguide is called an *Element*.

The Composite Waveguide Editor can be launched from both the editors, schematic and layout. If launched in the Schematic view, the Composite Waveguide Editor allows early estimation of the waveguide geometry.

The various ways you can launch the Composite Waveguide Editor in the schematic and layout view are:

- *From the Create Instance form*

  When creating a composite waveguide instance, click the Ellipses (…) button corresponding to the *Edit* label displayed just above the Waveguide parameter list.

- *From the Property Editor assistant*

© 2023 Cadence Design Systems, Inc.

In the Property Editor assistant, click the *Ellipses* (…) button corresponding to the *Edit* label displayed just above the *Waveguide* parameter list.

- *From the Property Editor form*

  In the *Parameters* tab of the Property Editor form, click the *Ellipses* (…) button corresponding to the *Edit* label displayed just above the *Waveguide* parameter list.

Editing a waveguide can involve the following:

- Adding a Waveguide Element
- Deleting a Waveguide Element
- Changing the Position of a Waveguide Element
- Adding an Optical Connector
- Waveguide Element Rotation

Adding a Waveguide Element

To add a waveguide element:

1. In the layout canvas, select the composite waveguide instance to update.

   The Property Editor assistant populates to display the properties of the selected optical instance.

2. In the Property Editor assistant, click the *Ellipses* (…) button corresponding to the *Edit* label displayed just above the *Waveguide* parameter list.

   The Composite Waveguide Editor displays.

3. In the Composite Waveguide Editor, right-click an existing waveguide element to use as the reference and choose *Add Before* or *Add After* to specify the position of the element to add.

   The Add Element form displays.

4. Specify the *Library*, *Cell*, and *View* to use for the new waveguide element.
5. Click *Apply* to apply the settings.
6. *Click OK to close the editor*.

A new waveguide element gets added at the position you specify.

© 2023 Cadence Design Systems, Inc.

Related Topics

Generating All Components from Source

Generation of Optical Pins and Chains (Virtuoso Photonics Option)

Property Editor Assistant

Deleting a Waveguide Element

To delete a waveguide element:

1. In the layout canvas, select the composite waveguide instance to update.

   The Property Editor assistant populates to display the properties of the selected optical instance.

2. In the Property Editor assistant, click the *Ellipses* (…) button corresponding to the *Edit* label displayed just above the Waveguide parameter list.

   The Composite Waveguide Editor displays, listing the various waveguide elements that comprise the composite waveguide.

3. In the Composite Waveguide Editor, right-click the waveguide element that you want to remove and choose *Delete* from the shortcut menu.

   The selected element is removed from the waveguide element list and from the layout canvas.

4. Click *Apply* to apply the settings.

5. *Click OK to close the editor*.

Related Topics

Property Editor Assistant

Changing the Position of a Waveguide Element

To change the position of an element in the composite waveguide:

1. In the layout canvas, select the composite waveguide instance to update.

   The Property Editor assistant populates to display the properties of the selected optical instance.

© 2023 Cadence Design Systems, Inc.

2. In the Property Editor assistant, click the *Ellipses* (…) button corresponding to the *Edit* label displayed just above the *Waveguide* parameter list.

   The Composite Waveguide Editor displays, listing the various waveguide elements that comprise the composite waveguide.

3. In the Composite Waveguide Editor, right-click the waveguide element for which the position needs to be changed and choose *Move Up* or *Move Down* to specify the new position for the element.

The selected element is moved to the new position in the element list. The composite waveguide displayed in the layout canvas also reflects the element at its new position.

Related Topics

Property Editor Assistant

Adding an Optical Connector

Optical connectors are complex waveguides that can automatically generate a waveguide layout by using basic input information such as curve type.

To add an optical connector:

1. In the layout canvas, select the composite waveguide instances to connect.
2. Right-click and choose *Optical Connector* from the context-sensitive instance menu.

   The Waveguide Connector for Optical Connections dialog box opens.

3. From the *Connector* drop-down list, choose a connector type to use for connecting the selected waveguides.

   The second value in the listed *Connector* options indicates the style of the connector that will be created—*curve*, *sine*, and so on.

4. Click *OK*.

An optical connector is created in the layout canvas based on the connector style you selected.

Related Topics

Property Editor Assistant

© 2023 Cadence Design Systems, Inc.

Waveguide Element Rotation

If a composite waveguide has the rotation parameter set, you can rotate all the elements inside the composite waveguide by the specified rotation value. The rotation of the waveguide elements is determined based on the rotation value set on the individual waveguide elements and the value set on the composite waveguide.

Let us consider that a composite waveguide includes two `waveguideStraight` elements with rotation on each element set to `0`. Let us also assume that the rotation of the composite waveguide is set to `45`.

On canvas, both the waveguide elements are rotated by `45` degrees, which puts them at an effective rotation of `45` degrees. If the `waveguideStraight` elements were set to rotation of `45` and the composite waveguide was set to rotate by `45` degrees, the effective, on-canvas rotation of the `waveguideStraight` elements will be `90` degrees.

However, when you edit a composite waveguide using the Composite Waveguide Editor, the composite waveguide is considered to be at rotation `0`. This allows editing the rotation of the individual waveguide elements without taking into account the rotation of the composite waveguide.

Related Topics

Composite Waveguide Editor Form (Virtuoso Photonics Option)

Generation of Optical Pins and Chains (Virtuoso Photonics Option)

Object Rotation

© 2023 Cadence Design Systems, Inc.

© 2023 Cadence Design Systems, Inc.

7

Connectivity Extraction in Layout XL

This topic explains how the Virtuoso® Layout Suite XL layout editor (Layout XL) processes connectivity information and presents several ways to prepare your design for routing.

The Layout XL connectivity extractor verifies that the connectivity of your physical implementation is legal and generates violation markers indicating open and short circuits, weak and must connect violations, and illegal overlaps in the design.

You can specify the types of markers that are created and limit the number of open violations displayed using the controls on the Connectivity of the Connectivity form. You can view and manage the violation markers in the *Connectivity* tab of the Annotation Browser Assistant assistant.

Related Topics

Connectivity Extraction Use Model

Connectivity Extraction from Substrate and Well Layers

Connectivity Extraction from Mosaics

Connectivity Extraction for Stacked Cut Layers

Support for Shape Trimming during Connectivity

Support for Color-Dependent Shape Trimming

Enabling Virtual Connections

Markers Generated by Connectivity Extractor

© 2023 Cadence Design Systems, Inc.

Connectivity Extraction Use Model

When you open a design in Layout XL, by default, connectivity extraction is not run. If you are opening the design for the first time, you can run the extractor explicitly using the *Connectivity – Update – Extract Layout* command to generate an up-to-date connectivity model for the entire design. The same applies if you have edited your design in a non-connectivity-aware environment such as VLS L.

- **Automatic Connectivity Extraction:** To automatically run connectivity extraction when Layout XL is launched, use the `extractRunOnInit` environment variable. However, this means that extraction is run even if it is not required and can adversely affect the performance of Layout XL during startup.

  Cadence recommends that you do not enable the `extractRunOnInit` environment variable by default, but that you use the *Connectivity – Update – Extract Layout* command when you know it is required.

- **Incremental Connectivity Extraction:** By default, Layout XL updates the connectivity and generates violation markers when you create, delete, or otherwise modify the shapes in your design either manually or automatically, using editing tools. This incremental extraction capability is controlled by the *Update connectivity when design is modified* option on the Connectivity Form.

- The connectivity and violation markers generated are saved in the design database and, therefore, are present the next time you open the design in Layout XL. Unless you edit your design outside Layout XL, you do not need to run extraction again when you re-open a design.

- **Bottom-Up Connectivity Extraction:** Bottom-up connectivity extraction is enabled by setting the extraction *Scope* to *Current Cellview and Cellviews in Hierarchy* in the Extract Layout Form.

  During bottom-up connectivity extraction, a series of connectivity extraction runs are performed on lower-level cellviews in a bottom-up order. This means the connectivity extraction starts from the lower-level cellviews in the hierarchy up to the current edited cellview. The advantage of running a bottom-up-extraction is that it does not require you to open each of these cellviews in a graphical window before being able to extract them. However, the extraction run can only extract connectivity from the cellviews that are within the specified hierarchical depth below the current edited cellview.

  Each bottom-up connectivity extraction run verifies the connectivity of the objects in the cellview being extracted. During the verification, if a connectivity mismatch is found for an object, the extractor either updates the connectivity on the object, if possible, or creates a marker in the cellview. The markers generated by the extractor can be viewed

© 2023 Cadence Design Systems, Inc.

from the current edited cellview by setting the scope of the Annotation Browser to *Current Cellview to Depth*.

- **Extractor Performance:** The performance of the connectivity extractor could be impacted if the design has many overlapping, identical shapes. To prevent the performance impact, ensure that overlapping, identical shapes are removed before connectivity extraction is run.

Related Topics

extractRunOnInit

Connectivity

Extract Layout Form

Annotation Browser Assistant

Enabling and Disabling Incremental Extraction

Extracting a Top-Level Design

Extracting Hierarchical Cellviews

Connections Recognized by the Extractor

When extracting connectivity, Layout XL detects:

- Shorts and opens on nets by recognizing connections between adjacent shapes, vias, and instance pin figures, including microwave objects. These connections can exist on the same layer or on electrically connected layers, as reported by the Diagnostic Report, which is available through the Connectivity Form.
  - Instance pin figures are considered by the extractor even when the default extract stop level is `0`. The connectivity checked in this case is that of the nets attached to the corresponding instTerms.
  - Connectivity of shapes inside global routes is not extracted because these represent coarse routing, which is usually only used to get an estimate about the congestion of the design.
  - An empty net—a net that does not own a figure, an instTerm, a terminal, or a non-empty constraint —can be deleted by the extractor.

© 2023 Cadence Design Systems, Inc.

- Connections between a shape and a label, where the label is the source of connectivity for the shape, are recognized. Such shapes are considered to be sticky shapes that have the `lxStickyNet` property, therefore, they stay connected to the net referenced by the label text. The label can be attached to the shape or it can be a stampLabel overlapping the shape.

  See Connectivity from Attached Labels to Consider During Extraction and Connectivity from stampLabels During Extraction.

- Connections between dedicated abutment shapes (dummy poly or dummy diffusion) of overlapping devices, even if the default extract stop level is 0, are recognized.
- Connections to instances and nets within a transparent layout hierarchy.

When extracting designs that have been edited outside Layout XL, the connectivity extractor silently cleans up the unmatched instance terminals that have no associated terminal in the cell master. This means that you can be asked to save your design, even though nothing has visibly changed.

Assigning Connectivity to a Via

You can assign connectivity directly to a via in the same way as you assign connectivity to paths, polygons, and rectangles. The `lxStickyNet` property is automatically added to the via, which retains the net assignment until you remove the property or override it with a new net assignment.

If you remove the `lxStickyNet` property, Layout XL assigns connectivity to the via based on overlaps.

Related Topics

Connectivity

lxStickyNet Property

Transparent Layout Hierarchy Generation

Instance Terminal Connectivity Considered During Extraction

The connectivity of an instance terminal can come from the schematic (if the corresponding schematic instance has connectivity) or from the first overlap detected in the layout. Once assigned, Layout XL considers the connectivity of an instance terminal as *sticky*, which

© 2023 Cadence Design Systems, Inc.

means that when an instance terminal is assigned to a net, it stays assigned to that net unless you change it using, for example, the *Propagate Nets* command.

For example, assume you have an instance, `I2`, with no connectivity in Layout XL. When you move this instance to overlap a shape assigned to `net2`, Layout XL creates an instance terminal for `I2` and assigns it to `net2`. If you now move `I2` to overlap a shape on `net1`, a short is created.

Depending on the pin connectivity of the terminals, pins, and pinFigs of an instance, the extractor considers these connections differently.

- **Strong-connected pinFigs**: Two pinFigs are strongly connected when they belong to the same pin. The connectivity extractor considers such pins as *connected* in the master of an instance. Therefore, no shorts are reported for these pins in the Annotation Browser.
- **Weakly-connected pinFigs**: Two pinFigs are weakly connected when they belong to different pins of the same terminal. The connectivity extractor considers such pins as *connected* in the master of an instance. However, Layout XL runs additional checks to ensure no current flows between such pinFigs. No shorts are reported for these pins in the Annotation Browser.
- **Must-connect pins**: When a terminal is set as *must-connect-pins*, two pinFigs of different pins must be connected together in the design of the instance. If the pinFigs are not routed together, the connectivity extractor considers the connection as incomplete. An open marker is reported in the Annotation Browser.
- **Must-connect terminals:** When two terminals are set as *must-connect*, pinFigs of the terminals must be routed together in the design of the instance. If the pinFigs are not routed together, the connectivity extractor considers the connection as incomplete. An open marker is reported in the Annotation Browser.

The following exceptions apply:

- Where an instance terminal is assigned to a net with a must-join terminal at the current editing level, the extractor does not consider the instance terminal sticky. However, if the instance terminal is moved away from an overlap, it keeps its connectivity until the extractor finds an overlap with a different must-join net that is better suited for the connection. If such an overlap is found, the instance terminal is reassigned to the new net.

  This behavior has been implemented to support the must-join pin model at different levels of hierarchy.
- When you move a physical-only instance, which is an instance for which the `lvsIgnore` property is set to `t`, Layout XL automatically reassigns the connectivity based on any

© 2023 Cadence Design Systems, Inc.

new overlap. The only exception to this is when the physical-only instance has the `lxStickyNet` property set to `t`, in which case the extractor creates a short marker if the new connectivity does not match the existing connectivity.

If there is no new overlap after the move and the physical-only instance has the `lxStickyNet` property set to `t`, the existing connectivity is retained. If the `lxStickyNet` property is not set to `t`, the existing connectivity is deleted.

Related Topics

Supported Pin Connectivity Models

lvsIgnore Property

lxStickyNet Property

Support for Diffusion Layers During Extraction

Some manufacturing processes define diffusion shapes that must be explicitly implanted by n-type or p-type implant shapes to get their own type. If the implant is p-type, the diffusion shape is considered to be p-type. If the implant is n-type, the diffusion shape is considered to be n-type.

Other processes derive the implant type from the definition of the diffusion layer, “ndiff” or “pdiff”, in the technology file. This is the case when an explicit implant does not exist. The connectivity extractor supports both types of diffusion layers.

The connectivity extractor does not modify the connectivity of implant shapes.

- Analog processes typically feature a single diffusion layer and two or more layers for implants. The extractor recognizes shapes on layers with function `diff` as unimplanted diffusion: they need an explicit ‘n’ or ‘p’ implant shape to connect to the appropriate `pwell` or `nwell`.

  The `functions` table for such a process might include the following definitions.

```
functions(
;( layer          function         [maskNumber])
;( -----          --------         ------------)
  ( pwell          "pwell"          10          )
  ( nwell          "nwell"          20          )
  ( diff           "diff"           30          )
  ( poly           "poly"           40          )
  ( pimp           "pplus"          50          )
  ( nimp           "nplus"          60          )
);functions
```


© 2023 Cadence Design Systems, Inc.

Only `diff` diffusion is defined, so there are also `pimp` and `nimp` layers defined explicitly for the implants. A shape on `diff` must overlap with a shape on `pimp` to connect to a shape on `pwell` or with a shape on `nimp` to connect to a shape on `nwell`. The connectivity stack is derived as follows:

```
metal -> cut -> (diff & nplus) -> nwell or ntype substrate
metal -> cut -> (diff & pplus) -> pwell or ptype substrate
```

- Manufacturing processes used in digital design often define separate layers for n and p diffusion. A shape on these layers is considered to have an *implicit implant* whose type is derived from the layer function.

  Here, the functions table might include the following definitions.

```
functions(
;( layer          function         [maskNumber])
;( -----          --------         ------------)
  ( lv_nwell       "nwell"          10           )
  ( hv_nwell       "nwell"          20           )
  ( hv_pwell       "pwell"          30           )
  ( ndiff          "ndiff"          40           )
  ( ndiff_hv       "ndiff"          50           )
  ( pdiff          "pdiff"          60           )
  ( odiff_hv       "pdiff"          70           )
  ( poly           "poly"           80           )
);functions
```

  There are no layers defined for implants, the extractor differentiates between p-type and n-type implants based on whether the function is `pdiff` or `ndiff`. An `ndiff` shape connects to shapes that have the `nwell` function; a `pdiff` shape connects to shapes that have the `pwell` or `substrate` function.

  The connectivity stack is derived as follows:

```
metal -> cut -> ndiff -> nwell or ntype substrate

Metal -> cut -> pdiff -> pwell or ptype substrate
```

- Some processes have a single active diffusion layer, such as “`ndiff`”, which is used to connect to both types of wells and substrates. These processes also have a single implant, such as “`pimp`”, which takes priority over the diffusion type.

  The functions table for such a process might include the following definitions:

```
functions(
;( layer          function         [maskNumber])
;( -----          --------         ------------)
  ( M1             "metal"          10           )
  ( CO             "cut"            20           )
  ( NOD            "ndiff"          30           )
  ( PP             "pplus"          40           )
  ( NW             "nwell"          50           )
  ( PW             "pwell"          60           )
 );functions
```


© 2023 Cadence Design Systems, Inc.

Both, a diffusion layer, `ndiff`, and an implant layer, `pimp`, are defined. An `ndiff` shape connects to shapes that have the `nwell` function. An `ndiff` shape when overlapped by a `pimp` shape connects to shapes that have the `pwell` function.

The connectivity stack is derived as follows:

```
metal -> cut -> ndiff -> nwell or ntype substrate
metal -> cut -> (ndiff & pplus) -> pwell or ptype substrate
```

Related Topics

Annotation Browser Assistant

Weakly-Connected Instance Pins During Extraction

The extractor also considers the external loops of current that can exist in the connections made to weakly-connected instance pins. This means it is legal to route two weakly-connected instance pin figures, provided the connections on these pins are routed together. If this is the case, the extractor does not detect a violation because it is assumed that the current will not flow through the resistive path between the two pin figures inside the instance.

When the extractor detects an illegal weak-connect violation, the generated marker is a line marker drawn between the two instance pin figures that are overlapped by unconnected top-level shapes on the same net. This provides better visibility of the problem and eases the process of fixing it. The Annotation Browser highlights both the line marker and the two instance pin figures.

Any incremental updates that you make are also considered for the detection of illegal weak-connect violations.

Related Topics

Annotation Browser Assistant

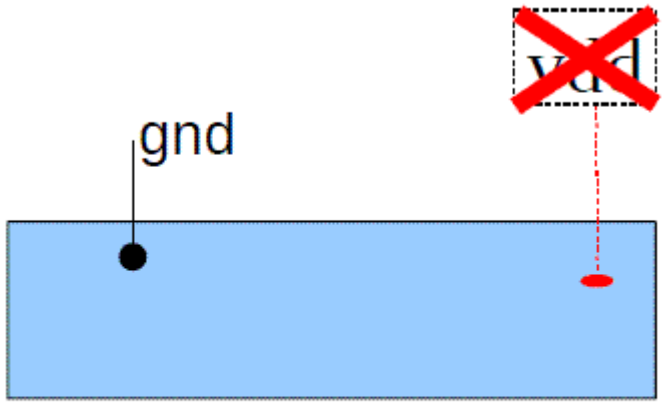
Connectivity from Attached Labels to Consider During Extraction

If a top-level shape has an attached label, you can choose to automatically connect the shape to the net referenced by the label text. To do this, you must enable the *Assign shapes from attached labels* option on the Connectivity Form.

However, in some situations, as those described below, the connection may result in violation markers being displayed in the Annotation Browser.

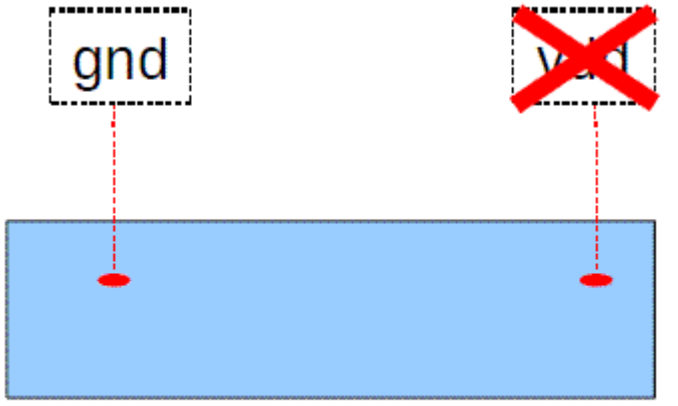
© 2023 Cadence Design Systems, Inc.

Scenario 1: If a label is attached to a shape that has sticky connectivity to another net, a short marker is created overlapping the label.



Short between:
- net "vdd" and attached label with text "vdd"
- net "gnd" and a shape on layer-purpose "Metal1 drawing"

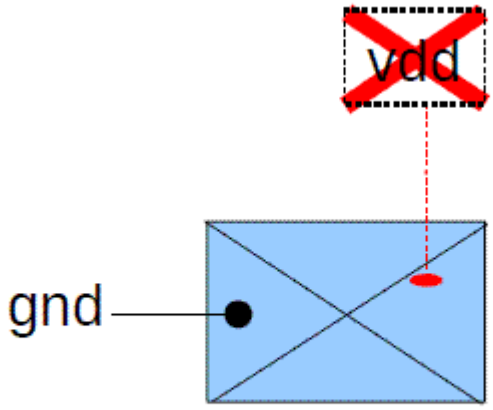
Scenario 2: If several labels with different text are attached to a single shape, the labels that create a short with the shape are overlapped by a short marker.



Short between:
- net "vdd" and an attached label with text "vdd"
- net "gnd" and a shape on layer-purpose "Metal1 drawing"

© 2023 Cadence Design Systems, Inc.

Scenario 3: If a label is attached to a pin shape that has a terminal name different from the label text, a short marker is created overlapping the label. If the label text and the terminal name are the same, no violations are reported.



Short between:
- net "vdd" and an attached label with text "vdd"
- net "gnd" and a pin figure (a rectangle on Metal1 pin) of terminal 'gnd'

Related Topics

Connectivity

Annotation Browser Assistant

Create Label Form

Connectivity from stampLabels During Extraction

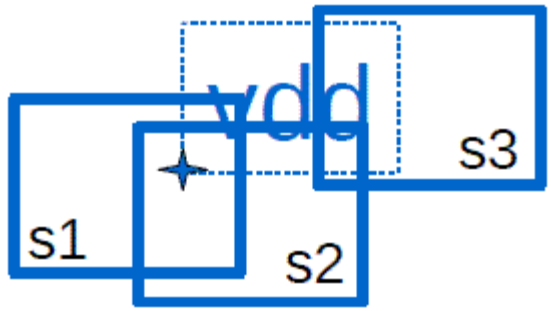
If a top-level shape or a substrate region overlaps the origin of a label, you can choose to automatically assign the shape (or the substrate region) to the net referenced by the label text. However, for the shape (or the substrate region) to be connected to the label, a valid `stampLabelLayer` association should be defined in the technology file for the layer of the shape (or the substrate region) and that of the label.

To enable the feature, you must select the **Assign shapes from overlapping labels as defined by 'stampLabelLayers' rule** option on the Connectivity Form. If the `stampLabelLayers` rule is not defined in the technology file, the option appears disabled.

© 2023 Cadence Design Systems, Inc.

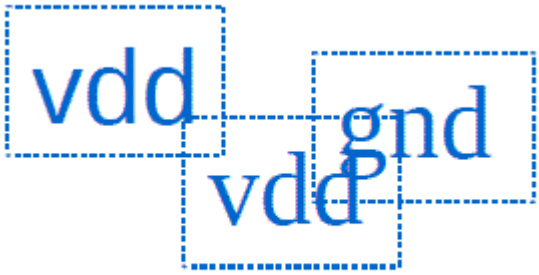
Let us now consider some scenarios that detail how the overlap with a label is processed by the extractor.

Scenario 1: If a label is overlapped by several shapes, only those shapes that overlap the origin of the label are connected to the net referenced by the label text.



Only s1 and s2 are assigned to net "vdd"

Scenario 2: If a label is overlapped by other labels due to space constraints, no shorts are reported.



No shorts are reported

© 2023 Cadence Design Systems, Inc.

Scenario 3: If a shape overlaps a label that has sticky connectivity to another net, a short marker is created.



Marker overlapping the intersection of the label and the shape

For more information about stamp labels, see stampLabelLayers in the *Virtuoso Technology Data ASCII Files Reference User Guide* and the *Auto* option under *Label Layer/Purpose* in the *Create Label form*

Related Topics

Connectivity

Create Label Form

Extracting a Top-Level Design

When extracting a top-level design, Layout XL extracts

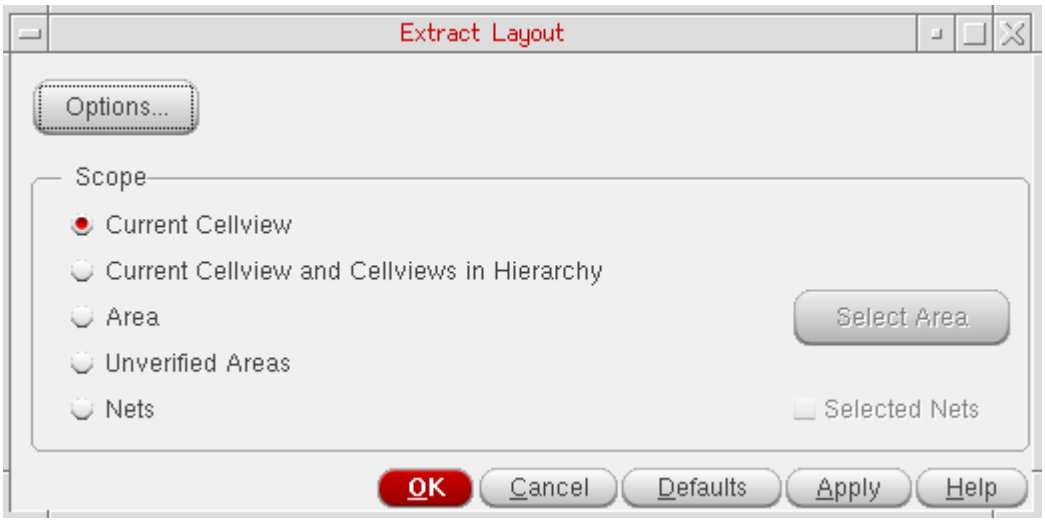
- Top-level objects—shapes, vias, and pin figures.
- Pin figures of top-level instantiated masters. These pin figures represent the interface between the top level instances and the lower level instances in the design and are, therefore, considered to be at the top level. The connectivity of these pin figures in the context of top level extraction is represented by the nets attached to the instance terminals corresponding to these pin figures.

To extract the current top-level design:

1. From the CIW, open a layout cellview and choose *Connectivity – Update – Extract Layout*.

© 2023 Cadence Design Systems, Inc.

The Extract Layout Form form is displayed.



2. Make sure the *Scope* is set to *Current Cellview* and click *OK* or *Apply*.

   The current cellview is extracted.

Related Topics

Extract Layout Form

Extraction Stop Level

The `extractStopLevel` environment variable controls the *Extract connectivity to level* option in the Connectivity form, increasing the visibility of the extractor into the design hierarchy.

For example, an overlap between a top-level shape and a hierarchical shape at level “N” is processed by the extractor only if the extraction stop level is equal to or greater than “N”. However, if the hierarchical shape is a pin figure, it is visible from the top level even if the extraction stop level is “N-1”.

Likewise, an overlap between a hierarchical shape at level “N” and a hierarchical shape at level “M” in the hierarchies of two different top-level instances is processed by the extractor only if the `extractStopLevel` is equal to or greater than the greater of the two levels, “N” and “M”.

In the context of top-level extraction, the connectivity of a hierarchical shape is the net attached to the corresponding top-level instance terminal (if any). This top-level connectivity

© 2023 Cadence Design Systems, Inc.

is established by recursively iterating in a bottom-up series from terminal to instance terminal, starting from the net attached to the hierarchical shape.

The greater the `extractStopLevel`, the more accurate is the extraction. For example:

- Top-level nets that are incomplete with `extractStopLevel=0` may become complete with `extractStopLevel>0`. For example, a top-level shape on a given net overlaps a hierarchical shape that has its top-level connectivity defined by an instance terminal on the same net. The resolution of the open violation, in this case, happens only if the hierarchical shape is visible, implying that the extraction stop level is sufficient, which is achieved at an extraction stop level greater than 0.
- There is a better detection of shorts between top-level and hierarchical shapes.
- There is a better detection of shorts between different instances.
- Trim shapes have the same visibility as regular shapes. They are visible down to `extractStopLevel`.

  Trim shapes on a pin are visible down to `extractStopLevel+1`. There is an exception for a pinFig at `extractStopLevel+1`. Here, the trim shapes in the same master are seen as stopping the pinFig even when the trim shapes are actually at `extractStopLevel+1`.

The extractor attempts to bind the instances that are in the hierarchical depth defined by the extract stop level. By default, the extract stop level is 0. Therefore, the extractor tries to bind all the top-level instances to their corresponding master.

Whatever the `extractStopLevel`, the extractor only ever extracts the current top level; that is, it only changes the connectivity of objects or creates markers at the top level of the design.

Related Topics

extractStopLevel

Connectivity

Extracting Hierarchical Cellviews

Extracting Hierarchical Cellviews

Hierarchical cellview extraction lets you extract either an entire hierarchical design or selected lower-level cellviews from the current top level of the design. The extraction is

© 2023 Cadence Design Systems, Inc.

performed bottom-up, starting with the cellview lowest in the hierarchy and ending with the top-level cellview. This is why the extraction is also called bottom-up extraction.

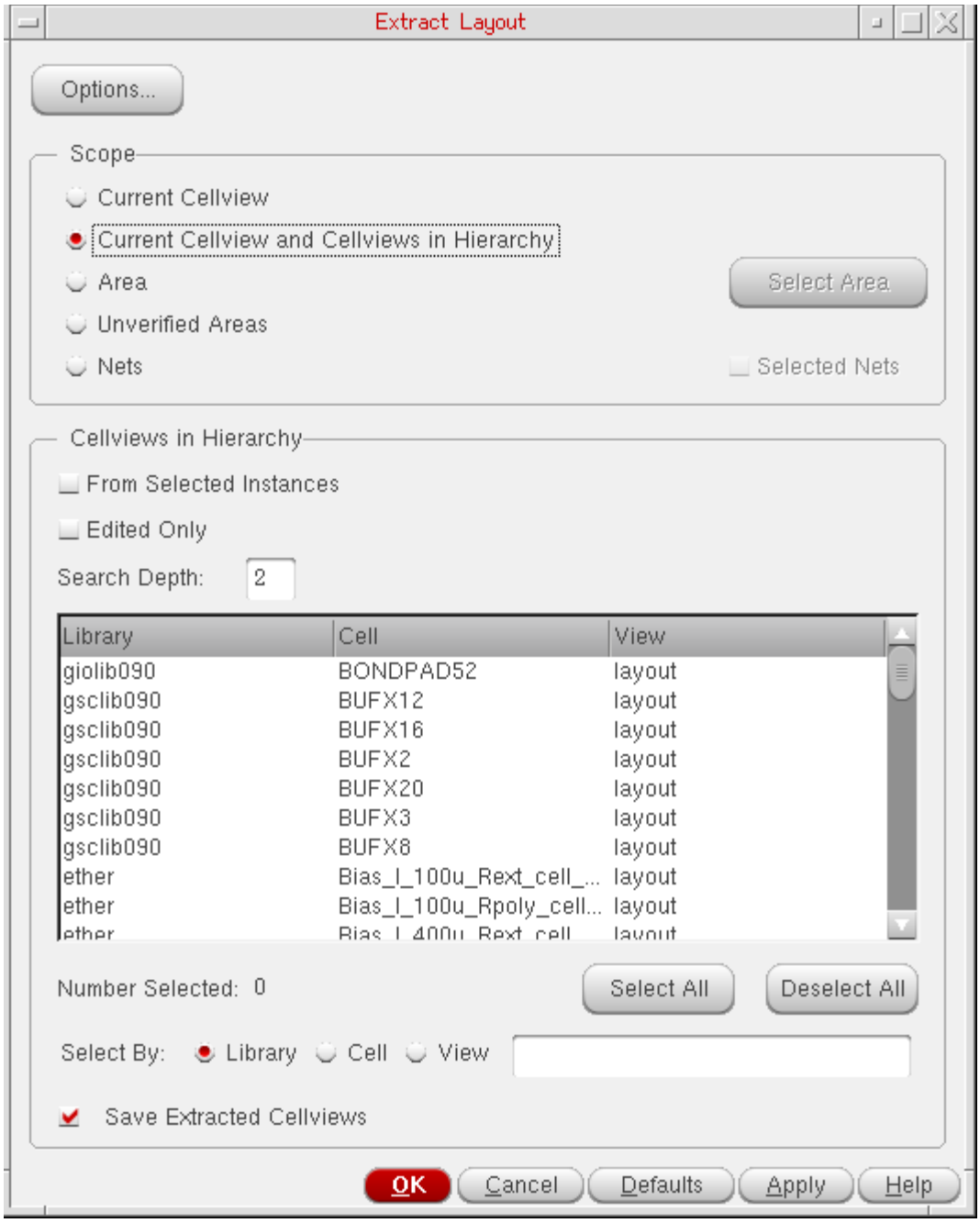
If you manually disable the extractor while editing a lower-level cellview, the extractor will be automatically disabled for the entire hierarchy. To resume incremental extraction for the edited cellview, the extractor must be manually re-enabled when needed.

To perform bottom-up extraction:

1. From the CIW, open a layout cellview and choose *Connectivity – Extract Layout*.

© 2023 Cadence Design Systems, Inc.

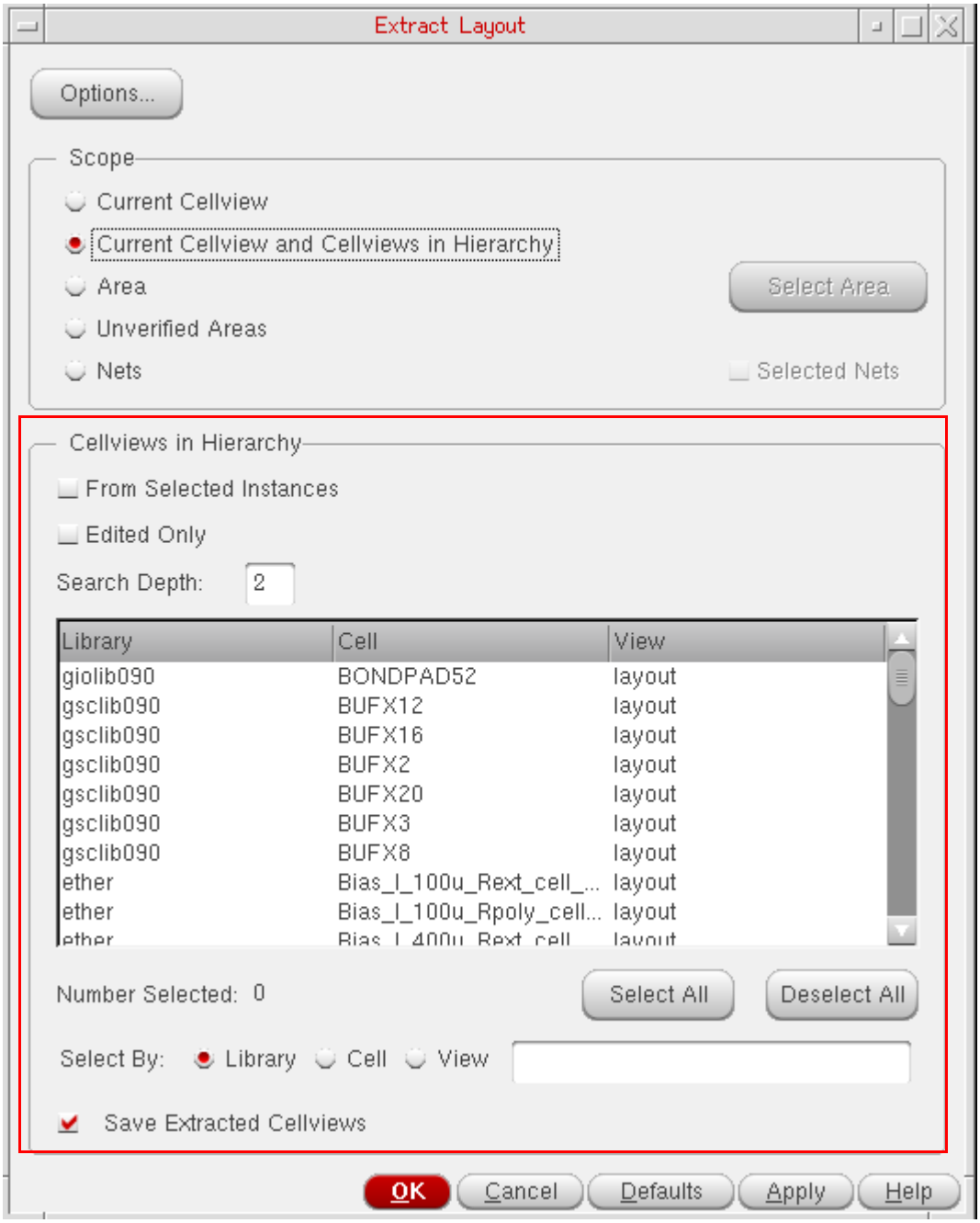
The Extract Layout Form form is displayed.



2. Set the *Scope* field to *Current Cellview and Cellviews in Hierarchy,* which means the extractor will extract connectivity from the current top-level cellview and the specified cellviews in the hierarchy.

© 2023 Cadence Design Systems, Inc.

The controls in the *Cellviews in Hierarchy* group box are enabled, as displayed in the figure below.



3. hierarchical cellviews are listed as extractable:

   - Select *From Selected Instances* if you want hierarchical cellviews only for the selected instances from the current edited cellview to be displayed.

© 2023 Cadence Design Systems, Inc.

- Select *Edited Only* if you want hierarchical cellviews only for the cells edited since the last extraction run to be displayed.

4. In the *Cellviews in Hierarchy* group box, specify how far down in the hierarchy to look for the cells to extract.

   Depending on the specified search depth, if lower-level extractable cellviews are found, they are listed in the table.

5. Choose the cellviews you want to re-extract from the list.

   You can filter the list by library name, cell name, and view name or click *Select All* to select all the cellviews listed. Click *Deselect All* to deselect all the listed cellviews.

   The filter mechanism has no wildcard capability. For example, to select a cell called '`resistor`', you must type '`r`', '`re`', '`res`', and so on. If you type '`sis`', the cell is not selected.

6. (Optional) To change the way the extraction is performed, click *Options* at the top of the form to open the *Connectivity* tab of the Connectivity form, where you can set the following extraction parameters for the design.

   - To increase or decrease the extractor's visibility into the hierarchy of the design, change the value of the *Extract connectivity to level* option.

     See Extracting a Top-Level Design.

   - To change the list of extractable layers, choose a different constraint group from the *Derive extractable layers from constraint group* list.

     See Prerequisites for Using the Layout XL Connectivity Extractor.

   - To show weak-connect and must-connect violations in the design, switch on the *Verify weak-connect violations* and *Verify must-connect violations* options.

   - To show the opens in the design, switch on the *Verify open violations* option.

     To improve performance, you can limit the numbers of violations displayed by net and by cellview using the *Maximum number of open violations by net* and *Maximum number of open violations by cellview* options.

     You can also include or exclude instance terminals with non-extractable pin shapes using the *Verify unimplemented instance terminals* option.

7. Select *Save Extracted Cellviews* to automatically save each cellview after it has been extracted.

8. Click *OK* or *Apply* to extract the selected lower level cellviews.

© 2023 Cadence Design Systems, Inc.

The selected cellviews are extracted all the way up to the top level of the design.

Parameterized Cells

The connectivity extractor does not extract Pcell masters during bottom-up extraction.

Related Topics

Extracting Hierarchical Cellviews

Extract Layout Form

Connectivity

Virtuoso Parameterized Cell Reference

Extracting Connectivity by Area

You can limit the extraction to a specified area of the design. The results are the same as those derived from a full batch extraction, but the process is often faster for small areas because the extraction is limited to the shapes enclosed by and overlapping the specified area.

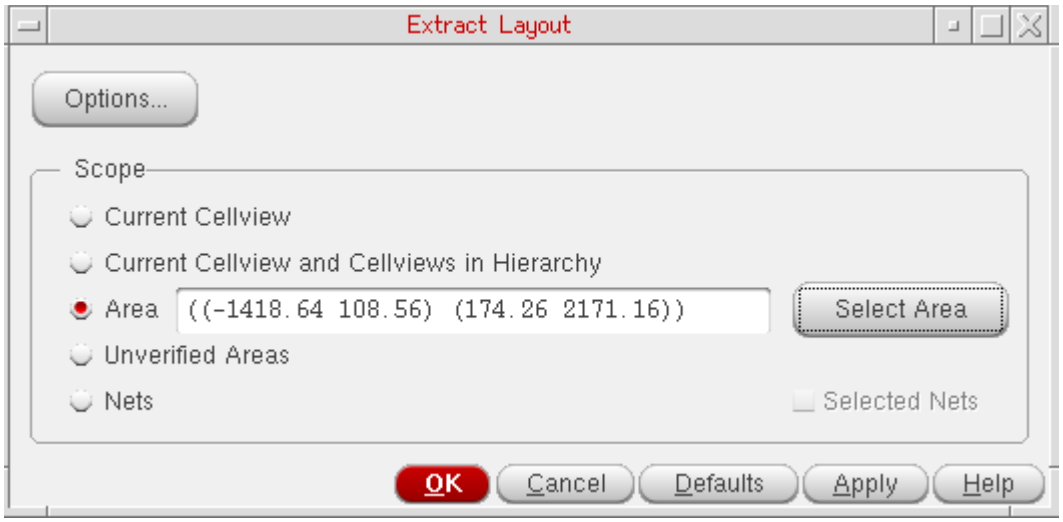
To extract connectivity for a specified area,

1. From the CIW, open a layout cellview and choose *Connectivity – Extract Layout*.

   The Extract Layout Form form is displayed.

© 2023 Cadence Design Systems, Inc.

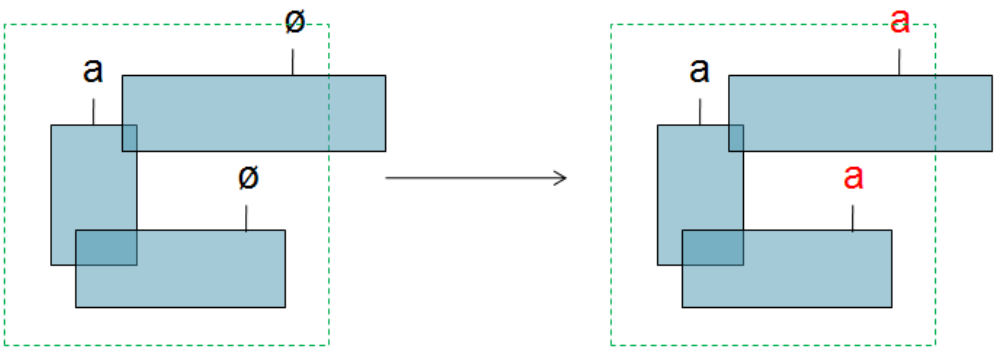
2. Set the *Scope* to *Area* to enable the area extraction controls.



3. Type the coordinates of the area to be extracted in the text field or click *Select Area* and draw the area you require in the canvas.
4. Click *OK* or *Apply* to extract connectivity for the specified area.

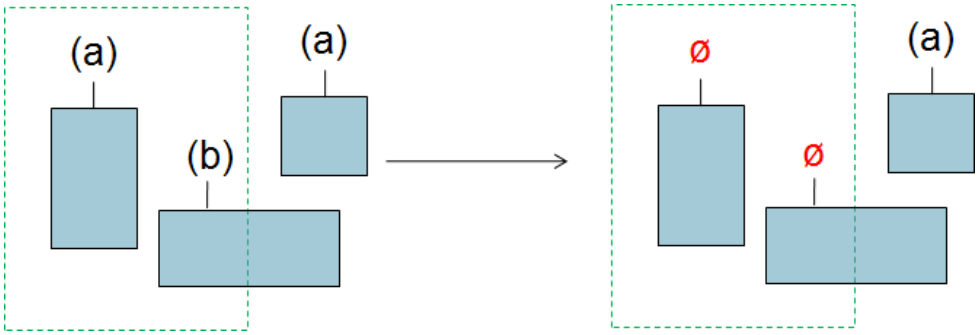
Area Extraction Examples

Example 1 shows connectivity propagated to all the connected shapes enclosed by or overlapping the specified area.

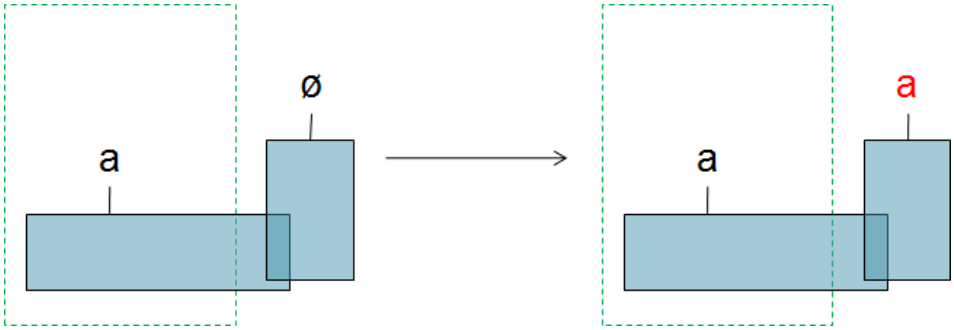


© 2023 Cadence Design Systems, Inc.

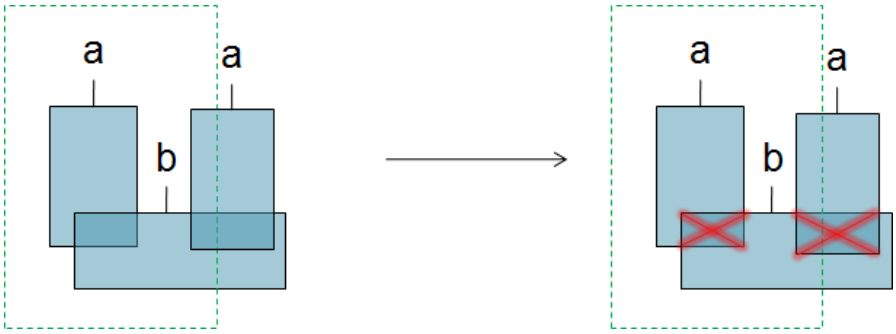
Example 2 shows connectivity removed from the unconnected shapes enclosed by or overlapping the specified area.



Example 3 shows connectivity propagated to the connected shapes outside the specified area.

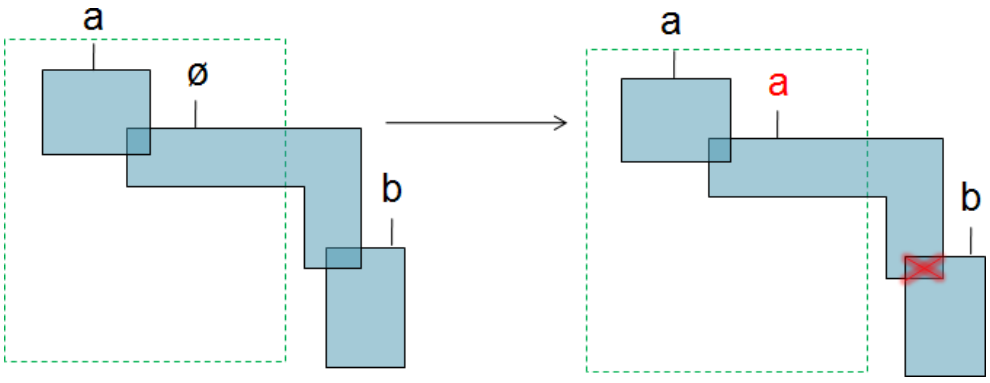


Example 4 shows markers created for violations on the objects enclosed by or overlapping the specified area.

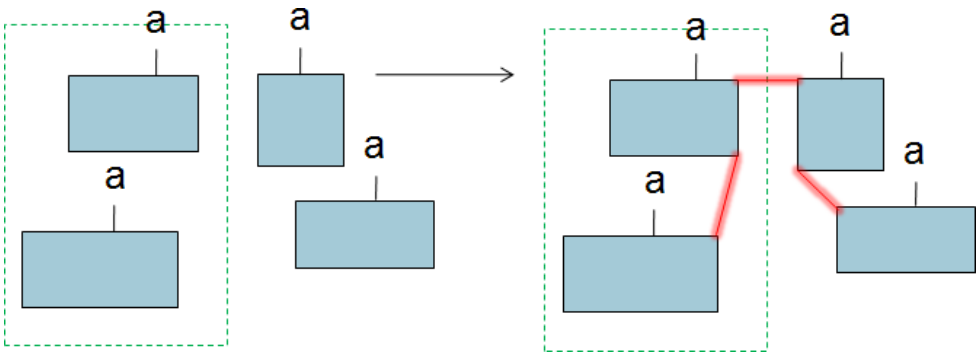


© 2023 Cadence Design Systems, Inc.

Example 5 shows markers created outside the specified area because connectivity was updated for overlapping shapes.



Example 6 shows open markers created between all the unconnected shapes on a given net regardless of whether the shapes overlap the specified area.



Related Topics

Extract Layout Form

Extracting Connectivity for Unverified Areas

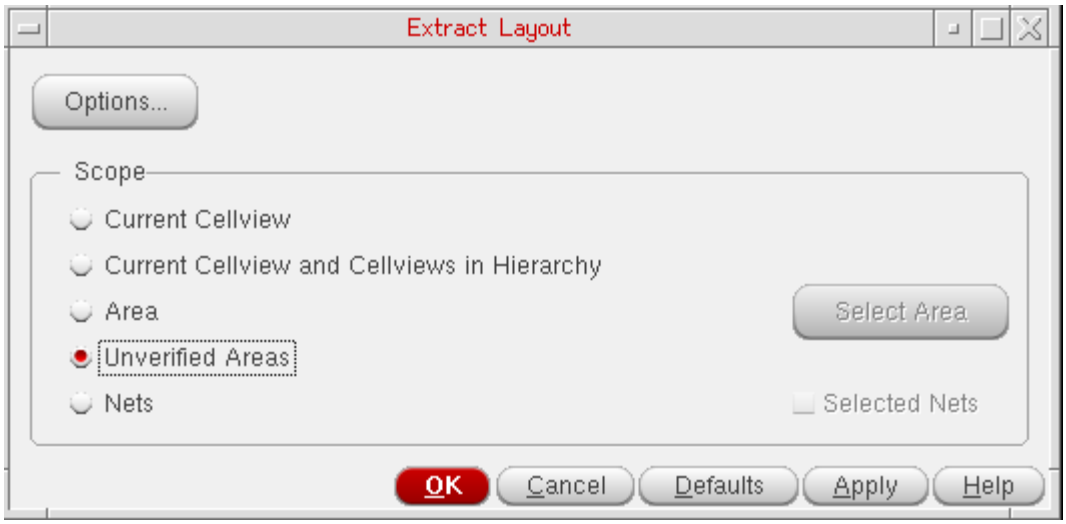
To extract connectivity for previously unverified areas,

1. From the CIW, open a layout cellview and choose *Connectivity – Extract Layout*.

   The Extract Layout Form form is displayed.

© 2023 Cadence Design Systems, Inc.

2. Set the *Scope* to *Unverified Areas* to enable the unverified area extraction controls.



3. Click *OK* or *Apply* to extract connectivity for previously unverified areas.

   Connectivity extraction is performed for areas that are defined by Unverified Area Markers.

Related Topics

Extract Layout Form

Unverified Area Markers

Extracting Connectivity by Nets

Connectivity extraction by a net (or nets) lets you limit the scope of extraction to a specified net or a set of nets. You can type in the net names directly in the *Nets* field or choose the *Selected Nets* check box in the *Scope* group box to enable extraction for all the nets already selected.

Alternatively, you can use the `lceExtractNet` and `lceExtractNets` SKILL functions, as appropriate, to enable extraction by nets.

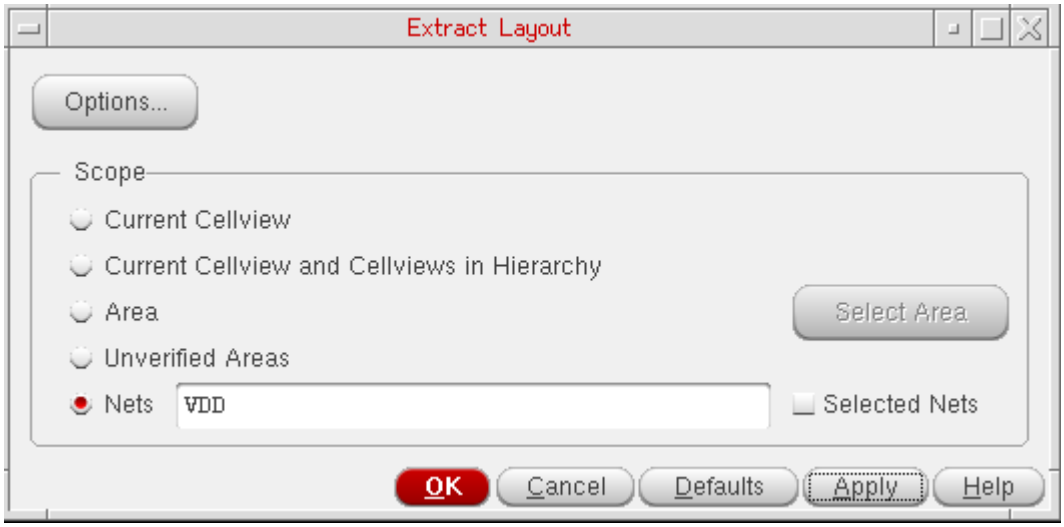
- The net names specified in the *Nets* field should be space separated.
- Choosing *Selected Nets* overrides any net names already specified and performs the extraction on all the selected nets.

To enable net-based connectivity extraction:

© 2023 Cadence Design Systems, Inc.

1. From the CIW, open a layout cellview and choose *Connectivity – Extract Layout*.

   The Extract Layout Form form is displayed.



2. Set the *Scope* to *Nets* to enable controls for extraction by nets. To extract connectivity only for specific nets, type the net names in the associated field. Keep the net names space separated.
3. Choose the *Selected Nets* check box to extract connectivity for all the selected nets. Selecting this option overrides any net names, if specified, in the *Nets* field and enables extraction for all the selected nets.
4. Click *OK* or *Apply* to extract connectivity for nets.

Connectivity extraction is performed for nets and the results are displayed in the CIW.

Related Topics

Extract Layout Form

lceExtractNet

lceExtractNets

Enabling and Disabling Incremental Extraction

Incremental extraction automatically updates the connectivity whenever the design is modified manually or by using automatic tools. This capability is switched on by default.

© 2023 Cadence Design Systems, Inc.

To enable and disable incremental extraction:

1. From the layout window menu bar, choose *Options – Connectivity* and select the *Connectivity* tab from the Connectivity form.
2. In the *Verification* tab, select the *Update connectivity information when design is modified* option.
3. Click *OK* or *Apply* in the Connectivity form to accept the change.

   Alternatively, click the *Update connectivity information when design is modified* icon,  , in the *Options* toolbar.

Unverified Area Markers

If the number of edited shapes is large, the extractor does not extract all the edited shapes. In this case, one or several markers of type *Unverified Area* are created in the Annotation Browser, indicating the layout areas that have not been completely extracted.

The unverified areas can be extracted later either directly by setting the scope of the extraction to *Unverified Areas* or by performing a complete extraction of the design.

Note: Potential open markers are no longer supported. You might still be able to view old markers in Annotation Browser. They can be removed by running extraction again.

Related Topics

Connectivity Form

Connectivity Extraction from Substrate and Well Layers

The extractor fully recognizes soft connections to the substrate and well layers.

- The soft connections are established using tap vias. Tap vias can be defined as connections to the substrate or a well layer on which they are placed.
- The substrate and well layers form an area on which you can place devices. The extractor verifies the connections made to the bulk terminal of these devices.
- The extractor evaluates the soft connections to appropriately propagate the connectivity. The extractor supports connectivity extraction of the bulk terminal of a device even when the bulk is not implemented by a physical shape. This is typically the case when an `n-type` transistor is placed over a `p-type` substrate and no physical shape represents the bulk area of the transistor for connectivity extraction.

© 2023 Cadence Design Systems, Inc.

- The extractor allows the substrate and well layers to be partitioned into discrete areas. This enables users to limit the connectivity extraction to a local substrate or a well area created by a partition.

Before these enhancements were implemented, the connectivity of the substrate was not checked at all and any physical shape on a well layer was considered as metal in terms of connectivity propagation. Likewise, the connectivity of shapeless bulk terminals was not checked.

Now, the extractor checks connectivity for the substrate, well layers, and bulk terminals and subsequently creates appropriate open and short markers in the Annotation Browser.

Related Topics

Annotation Browser Assistant

Soft Connection Extraction

Bulk Area Connectivity Extraction

Connectivity Propagation through the Substrate and Well Areas

Soft Connection Extraction

A soft-connect object can be:

- The entire substrate or an isolated area of the substrate
- A physical shape on a well layer or an isolated portion of the well shape
- A bulk instance terminal

A soft connection is a physical overlap between a soft-connect object and a tap via connecting to it. For a well layer, a soft-connection can also be made directly from a physical layer connecting to it, such as a diffusion shape.

A soft-connect object must be polarized from a tap via or a connecting layer. The polarization in such a connection is unidirectional—from the polarization source to the soft-connect object.

In terms of connectivity checking, it is valid to connect several soft-connect objects together, such as two well shapes.

The extractor does not report soft-connect design errors, such as a floating well shape or substrate area or a layout pattern that forces the current to flow out of a soft-connect object.

© 2023 Cadence Design Systems, Inc.

To detect such errors, tools such as Cadence PVS (Physical Verification System) can be used.

Related Topics

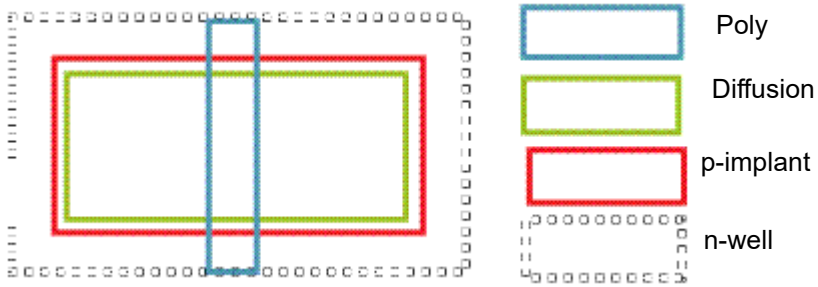
Annotation Browser Assistant

Bulk Area Connectivity Extraction

Connectivity Propagation through the Substrate and Well Areas

Bulk Area Connectivity Extraction

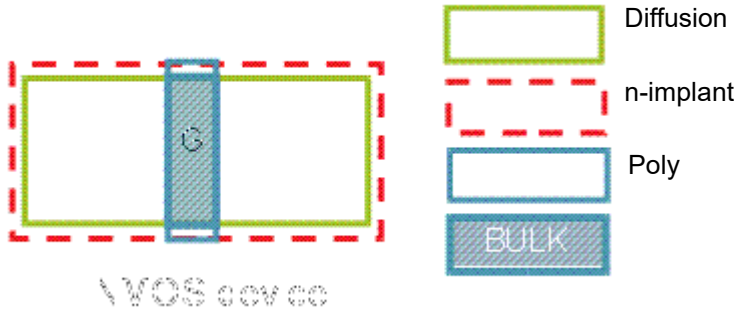
The bulk area of a transistor can be defined using a physical and extractable pin figure or using a shapeless terminal. If the bulk is implemented using a physical shape, the corresponding pin figures define the bulk area and the extractor uses the corresponding layer(s) to determine the top-level connections. If the bulk is a shapeless terminal, the bulk area of the device is usually the active layer that lies under its gate, between the source and the drain. The following figure displays the bulk area of a PMOS device, which is a physical shape on the Nwell layer.



Implemented bulk terminal of a PMOS device for connectivity extraction

© 2023 Cadence Design Systems, Inc.

The following figure displays the bulk area of an NMOS device for a shapeless bulk terminal.



Bulk area of an NMOS device for connectivity extraction

When the bulk terminal is shapeless, the corresponding bulk area needs to be derived by the extractor. For more information, see Identifying the Bulk Area.

Bulk area connectivity extraction means that the extractor recognizes the bulk area as an extractable layer even when the terminal is shapeless.

The bulk area is considered as “fully” extractable because the bulk is recognized by the extractor even if it does not have a physical shape drawn. When an instance terminal without an extractable layer has its pinFig-less layer defined as "substrate", the PR boundary in the master is used to define the geometry of the bulk area.

For the extractor to derive connectivity information from the bulk area, here are the tasks that need to be performed:

- Bulk Area Connectivity Extraction - Identifying the Substrate Area
- Bulk Area Connectivity Extraction - Isolating the Substrate and Well Layers
- Bulk Area Connectivity Extraction - Identify the Bulk Area
- Bulk Area Connectivity Extraction - Identify the Substrate and Well Tap Vias

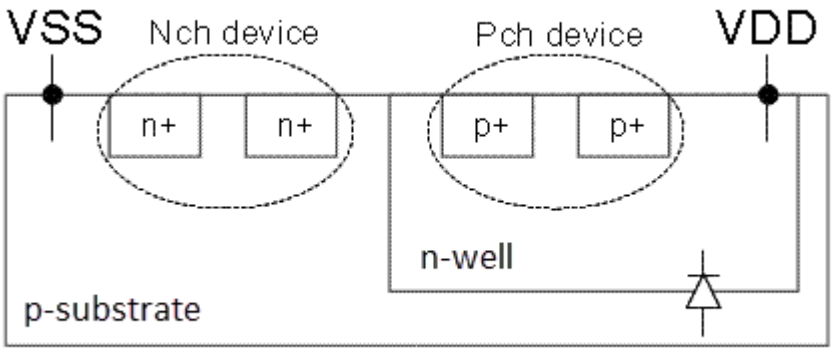
Bulk Area Connectivity Extraction - Identifying the Substrate Area

Before we discuss why it is important to identify the substrate area for bulk area extraction, let us find out what a substrate means in the context of chip fabrication.

A substrate is the base layer of a chip on which the various devices are placed to form a functional model. The substrate can be `n-type` or `p-type`. Depending on the substrate type, devices can be placed directly on the substrate or inside a well shape.

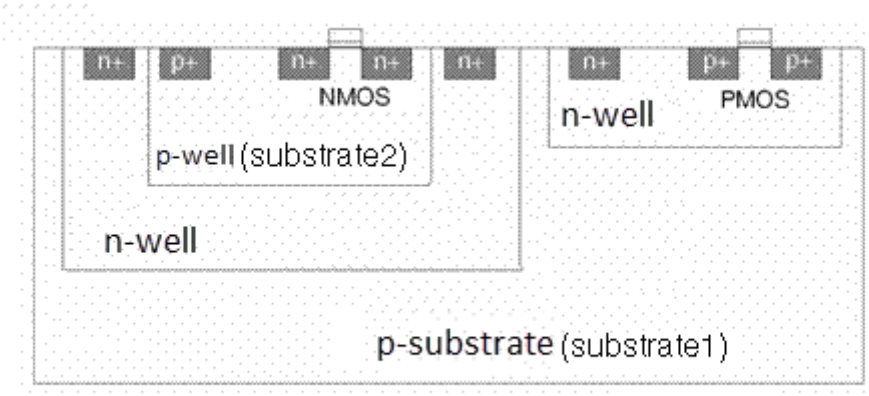
© 2023 Cadence Design Systems, Inc.

For example, in the figure below, the substrate is `p-type`. So, the NMOS (Nch) devices are directly placed over the substrate. Conversely, the PMOS (Pch) devices are placed over an n-well shape. The p-substrate and the n-well shape form a diode that must be reverse-biased to prevent the current to flow from one side to the other. This is why the `p-type` substrate is polarized to a net such as VSS, similar to the bulk terminal of the Nch devices. Likewise, the n-well shape is polarized to a net such as VDD, similar to the bulk terminal of the Pch devices.



n-type process

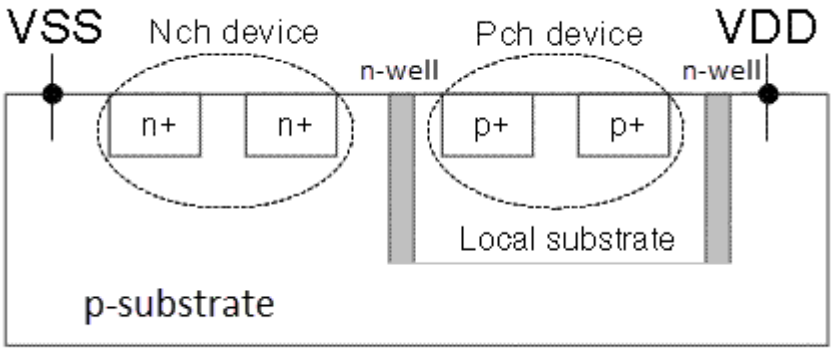
- The n-well shape drawn over the substrate isolates the enclosed devices from the main substrate.
- The n-well shape can be a guard ring. In such a case, the isolation created by the Nwell guard ring divides the substrate into two areas—substrate1 and substrate2—as displayed in the figure below.



Local and global substrates

© 2023 Cadence Design Systems, Inc.

The main substrate is the *Global* substrate that belongs to the entire chip. The *Local* substrate, which is the area enclosed within the n-well guard ring, is shown in the figure below. The local and global substrates are used in a design configuration to separate the digital and analog parts on a chip.



With this kind of isolation, the bulk instance terminals of devices placed on one substrate area are not connected to the bulk instance terminals of devices placed on the other substrate area.

See Bulk Area Connectivity Extraction - Isolating the Substrate and Well Layers.

The n-well guard ring can additionally be overlapped by an `n-buried` shape. This increases the isolation between the global and the local substrate by preventing the current to flow vertically between the two substrate areas.

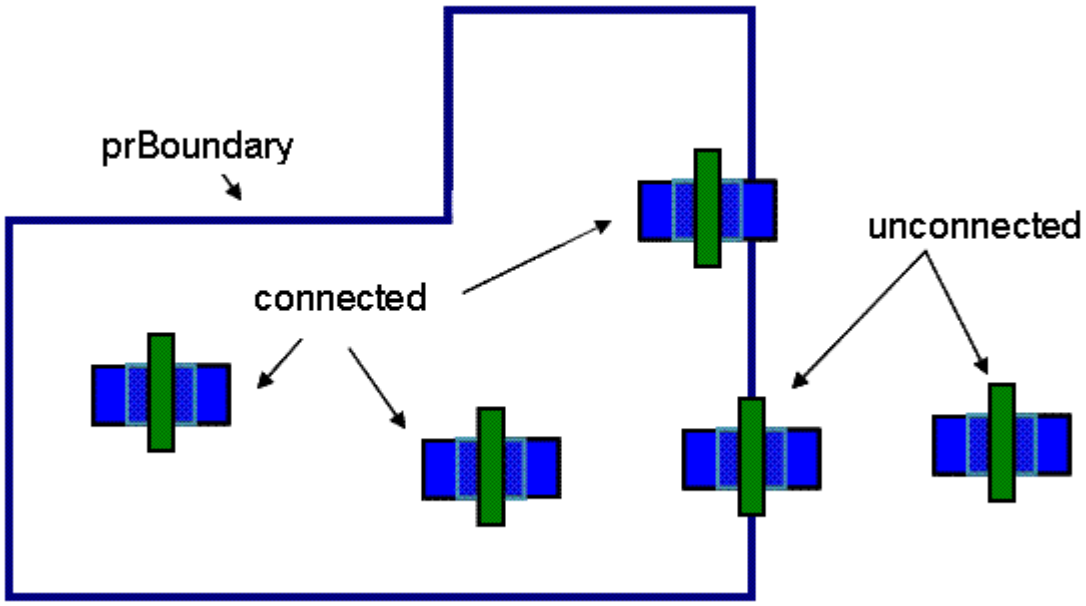
See Substrate and Well Isolation Using a Buried Layer.

When extracting connectivity, you identify the substrate area by representing it using the PR boundary. If there is no PR boundary in the design, the substrate is considered to extend over the entire area of the current layout.

You can then partition the global substrate into specific local substrates. Only those devices on the given substrate area that have their bulk area (physical shape or derived area) fully enclosed within the PR boundary (if present) are considered for extraction.

© 2023 Cadence Design Systems, Inc.

For example, in the figure below, the two devices that are labeled as “unconnected” do not have their bulk area fully inside the PR boundary. So, these devices are not considered for connectivity extraction.



Nch devices inside and outside the substrate area identified for extraction

Related Topics

Annotation Browser Assistant

Bulk Area Connectivity Extraction - Identify the Bulk Area.

Soft Connection Extraction

Bulk Area Connectivity Extraction

Connectivity Propagation through the Substrate and Well Areas

© 2023 Cadence Design Systems, Inc.

Bulk Area Connectivity Extraction - Isolating the Substrate and Well Layers

Isolating a portion of the substrate requires that the technology file mentions a given well layer as “stopping” the substrate. This helps the extractor recognize any shape drawn on a well layer as either:

- ”Connecting” to the substrate. For example, a p-well shape connects to a p-substrate.
- “Stopping” the substrate—providing an area isolated from the substrate or splitting it into two parts. For example, an n-well shape stops a p-substrate.

Isolating a portion of a well layer is similar to isolating the substrate because it requires some other well layers to be defined as “stopping” it. These “stopping” well layers should all be of the same type, opposite to the type of the well layer to be stopped.

You can use dedicated isolation layers to stop the substrate or well layers. In that case, these isolation layers should be defined as “recognition” layers in the technology file.

The details about a layer cutting or stopping another layer are specified in the technology file by using derived layers. For more information about derived layers, see Prerequisites for Using the Layout XL Connectivity Extractor.

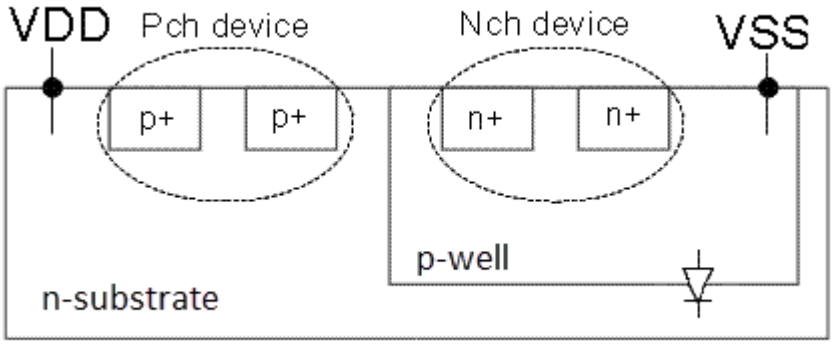
If the technology file does not define a well layer of the opposite type as the substrate as “stopping” the substrate, the extractor considers the corresponding well shapes as shorting to the substrate. In such a case, the extractor might produce some incorrect results during connectivity extraction. For more information, see Checking Connectivity.

Let us consider the following example.

Consider a highly doped “P” region formed by a p-well shape on top of an n-substrate, as in the figure below. If you do not specify, by using appropriate derived layers, that the p-well layer stops the substrate, the extractor assumes that the p-well layer “shorts” to it. The

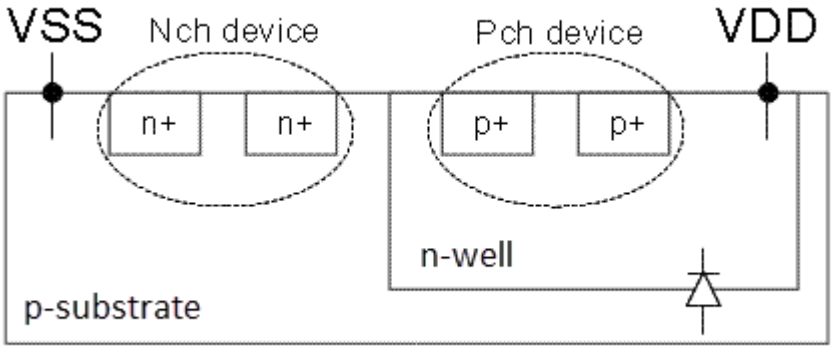
© 2023 Cadence Design Systems, Inc.

extractor, therefore, propagates the connectivity between the p-well shape and the substrate, generating incorrect connectivity extraction results.



p-type process

It is, therefore, required that the technology file defines any well layer, which is of the reverse doping type compared to the substrate, as stopping the substrate. For example, the figure below displays a p-substrate stopped by an n-well shape. For the extractor to establish correct connectivity between the two, the n-well layer must be set as stopping the substrate.



To summarize, if a well layer is n-type and the substrate is p-type or vice versa, the technology file must indicate that the well layer stops the substrate.

The various methods that you can use to isolate a substrate and well layer are:

- Substrate and Well Isolation by Drawing a Physical Shape on a Layer
- Substrate and Well Isolation Using an Isolation Layer
- Substrate and Well Isolation Using a Pseudo (Dummy) Isolation Layer

© 2023 Cadence Design Systems, Inc.

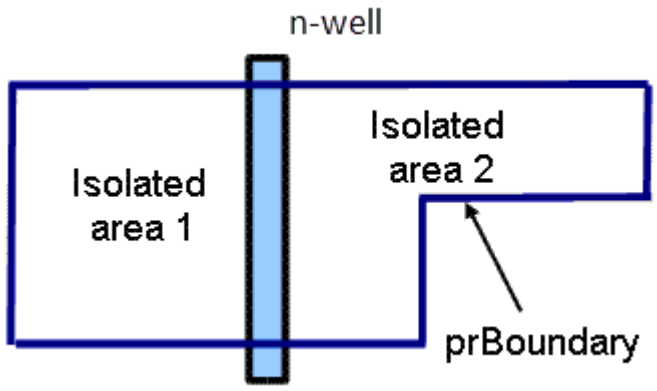
■ Substrate and Well Isolation Using a Buried Layer

Substrate and Well Isolation by Drawing a Physical Shape on a Layer

You can isolate an area of the substrate by creating a physical shape on a well layer. The well shape can be a simple rectangle or a guard ring. If you use a guard ring for isolation, the well shape that is formed isolates the substrate area lying inside the guard ring from the substrate area outside.

If an Nwell layer completely cuts the substrate into two distinct parts, the two substrate areas are electrically isolated. For example, in the following figure, the n-well shape splits the substrate into two physically and electrically separated areas. If you now place “Nch” devices on the two substrate areas, there is no connectivity propagation across the bulk instance terminals of these devices.

The two substrate areas must be physically distinct—without overlap or abutment—for the isolation to be effective and recognized by the extractor.

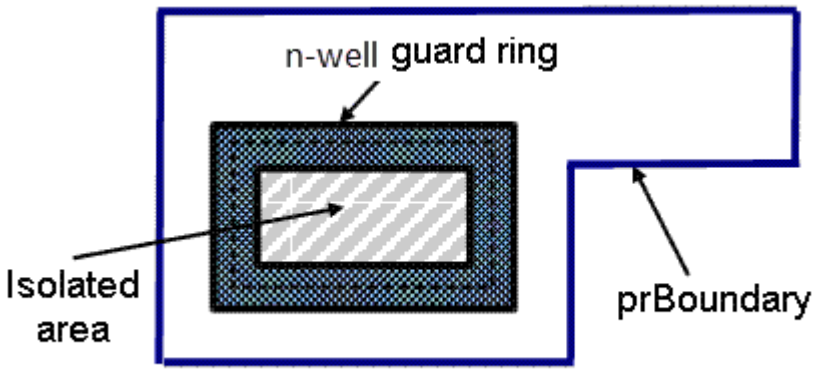


n-well splits the substrate, defines two p-type areas

Let us now consider another example. In this case, an n-well guard ring is drawn on the substrate. This guard ring cuts the substrate and, therefore, provides a local substrate within, which is isolated from the global substrate. The bulk instance terminals of devices placed

© 2023 Cadence Design Systems, Inc.

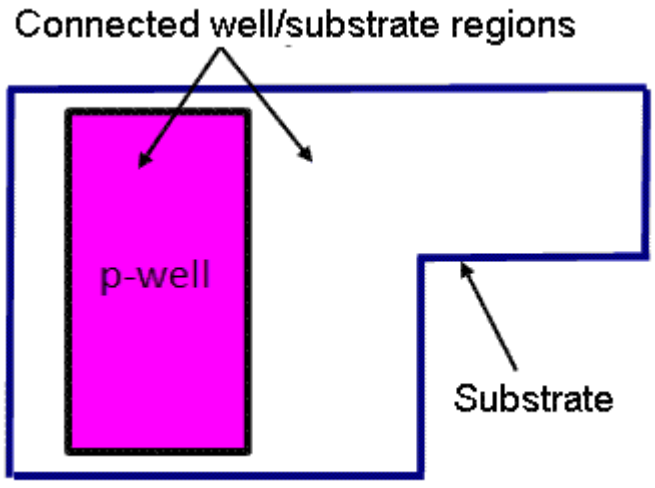
inside the guard ring are not connected to the bulk instance terminals of devices placed outside the guard ring.



n-well “cuts” substrate, defines a local substrate

If the n-well guard ring does not form a closed ring, the two substrate areas formed by the guard ring are not electrically isolated. Therefore, the bulk instance terminals of devices placed inside the guard ring are, in this case, connected to the instance terminals of devices placed outside the guard ring.

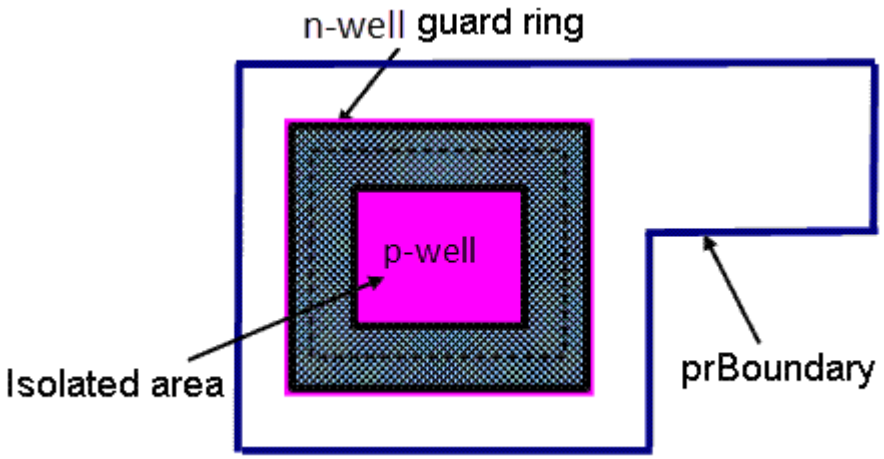
Let us next consider an example where a p-well shape is placed over a p-substrate. In this case, the extractor considers the well shape as connecting to the substrate and propagates connectivity between the two.



p-well shorts to substrate

© 2023 Cadence Design Systems, Inc.

If you want to isolate the p-well shape from the substrate, you can surround it with an n-well guard ring as shown in the figure below.



n-well “cuts” substrate, defines a local substrate above p-well

Although not strictly required by the connectivity extractor for connectivity checking, you can additionally increase the isolation of the above p-well shape by overlapping it with an n-buried layer. For more information about increased substrate isolation using buried layers, see Substrate and Well Isolation Using a Buried Layer.

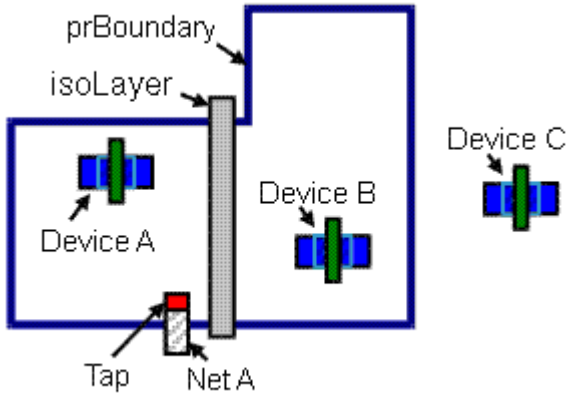
Substrate and Well Isolation Using an Isolation Layer

In addition to using a well shape for isolating an area of the substrate or another well layer, you can use a dedicated isolation layer. An isolation layer can be a shape—a rectangle, a guard ring, or any other shape—that cuts the substrate or the well layer into discrete areas.

When isolating an area of the substrate or a well layer using an isolation layer, the bulk instance terminals of the devices on different substrate or well areas are not connected. Let

© 2023 Cadence Design Systems, Inc.

us consider an example of a well-formed isolation made up of an isolation layer that completely cuts the substrate into two distinct parts.



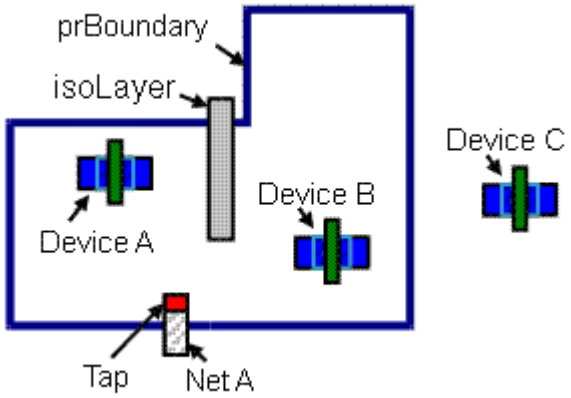
isoLayer cuts substrate, well-formed isolation

Here:

- The isolation layer cuts the substrate into two discrete areas.
- The bulk instance terminal of device A is assigned to net A from the tap via through the substrate.
- The bulk instance terminal of device B is left unassigned because it lies in a different substrate region.
- Device C is not considered for connectivity extraction of the bulk terminal because it lies outside the PR boundary. Therefore, the bulk instance terminal of the device is left unassigned.

© 2023 Cadence Design Systems, Inc.

Let us next consider an example of a malformed isolation in which the isolation layer does not completely cut the substrate into two different parts.



isoLayer partly cuts substrate, malformed isolation

Here:

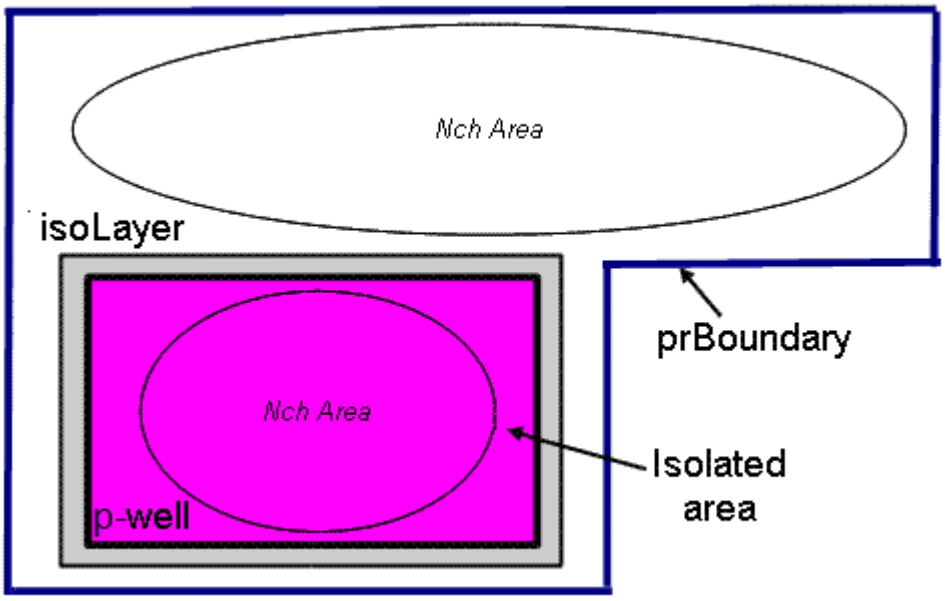
- The isolation layer does not cut the substrate into two discrete areas.
- The bulk instance terminal of devices A and B is assigned to net A from the tap via through the substrate.
- Device C is not considered for connectivity extraction of the bulk terminal because it lies outside the PR boundary. Therefore, the bulk instance terminal of the device is left unassigned.

During connectivity checking, the extractor does not consider the distances between a tap via placed over a substrate or well area and the bulk terminal of the devices placed within this area. All the devices that are placed on this substrate or well area are polarized, irrespective of their distance from the tap via.

An isolation layer can also be used to isolate a well layer from the substrate if the well layer is of the same type as the substrate. In the following figure, the Pwell shape is totally enclosed inside the isolation shape and it is, therefore, isolated from the substrate outside. In this example, the isolation layer is set as ‘stopping’ the substrate but the extractor only considers

© 2023 Cadence Design Systems, Inc.

its edges as cutting the substrate. Therefore, the extractor considers the inner area as an isolated portion of the same doping type as the outer area.

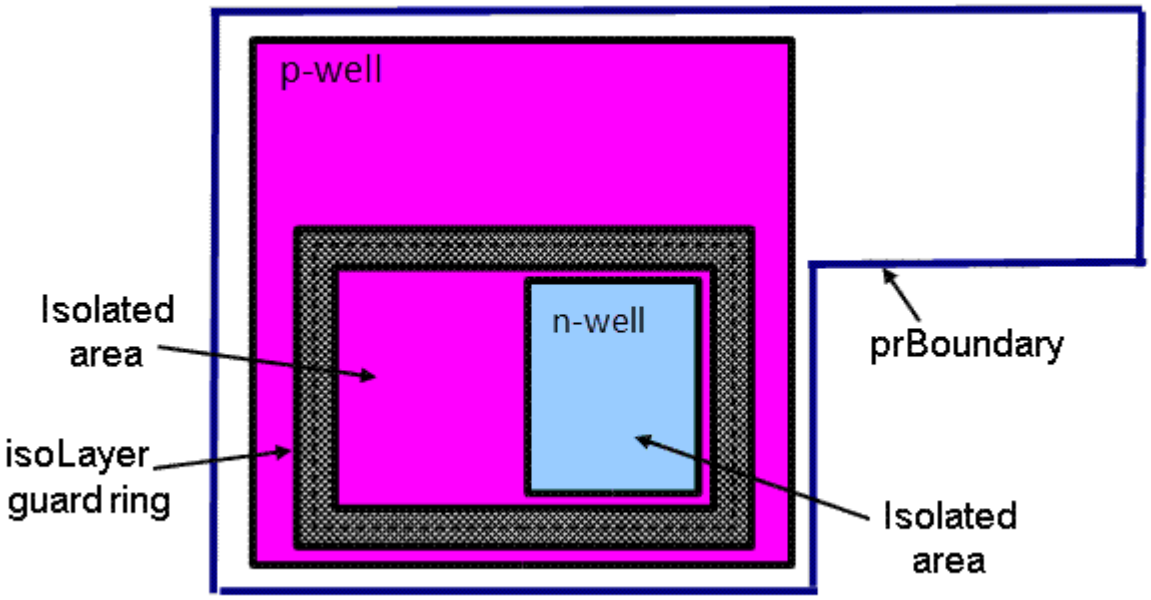


isoLayer isolates p-well by cutting substrate

For further isolation within a well area, you can use a guard ring. In the figure below, a guard ring containing an isolation layer is placed over a p-well shape. This isolates the p-well area inside the guard ring from the p-well area outside. The p-well outside the guard ring shorts to

© 2023 Cadence Design Systems, Inc.

the substrate. However, the p-well inside the guard ring forms an isolated area, which can be cut further by an n-well shape to provide an isolated area of the opposite polarity.



isoLayer guard ring cuts p-well and n-well cuts substrate

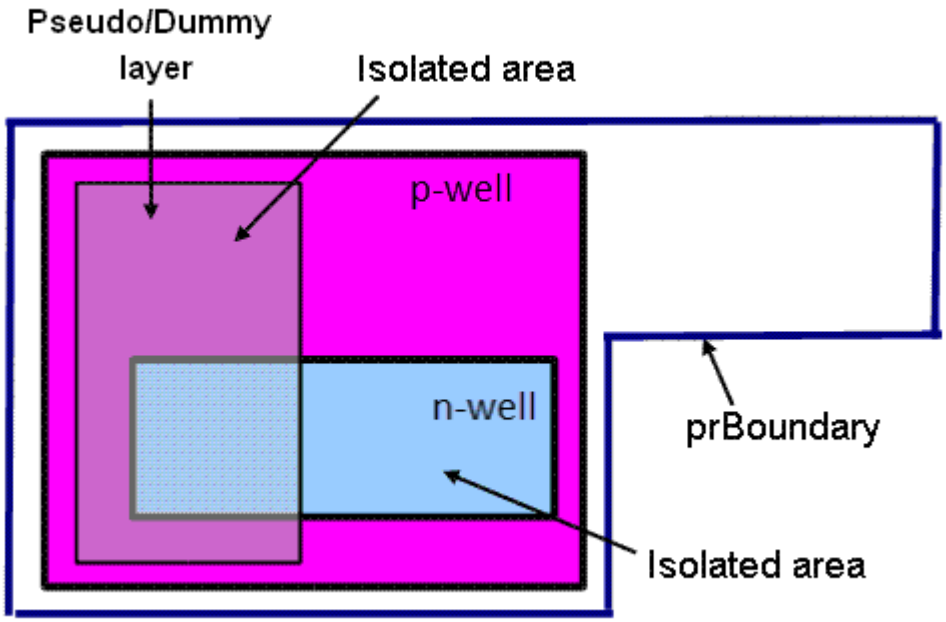
Substrate and Well Isolation Using a Pseudo (Dummy) Isolation Layer

This methodology is not recommended because the pseudo isolation layer is not manufactured. Therefore, some real short violations can potentially be left undetected by the tool. However, if a pseudo isolation layer is used, the same extraction requirements and connectivity checking results will be observed as for a real isolation layer.

A pseudo isolation layer is processed by the extractor the same way as a real isolation layer. For example, in the figure below, the pseudo isolation layer isolates the p-well area below it

© 2023 Cadence Design Systems, Inc.

from the p-well area outside. The n-well shape further stops the p-type area and provides an isolated n-type area by itself.



Dummy layer isolates p-well; n-well cuts the substrate

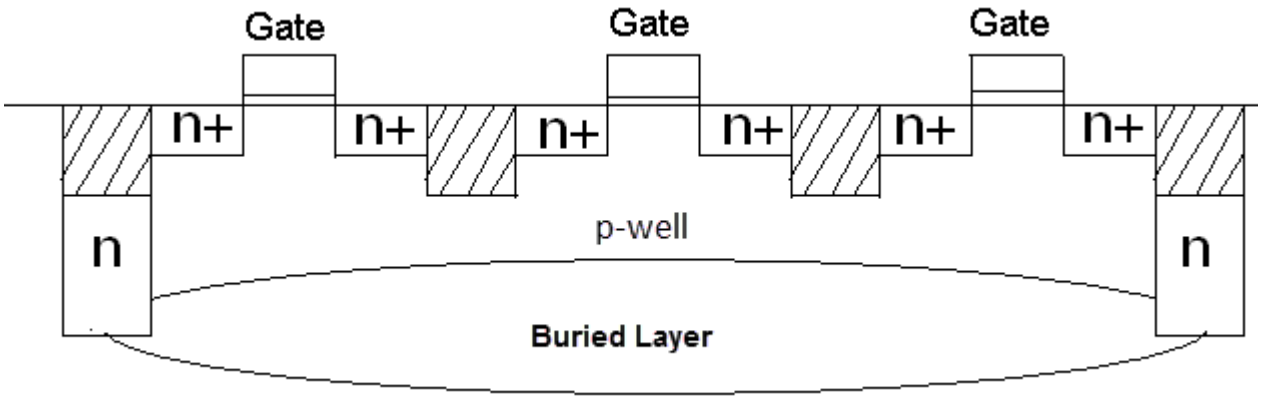
Substrate and Well Isolation Using a Buried Layer

Buried layers allow users to provide a more efficient isolation when used in addition to the isolation methods based on physical shapes or isolation layers.

In the technology file, a buried layer has function “n-buried” or “p-buried”. Its polarity should be the opposite of the polarity of the substrate. A buried well layer is a well layer with a lower mask number compared to another well layer of the same polarity.

© 2023 Cadence Design Systems, Inc.

A local p-substrate is formed by a p-well shape over an n-buried shape, as shown in the figure below, with the p-well shape surrounded by an n-well guard ring.



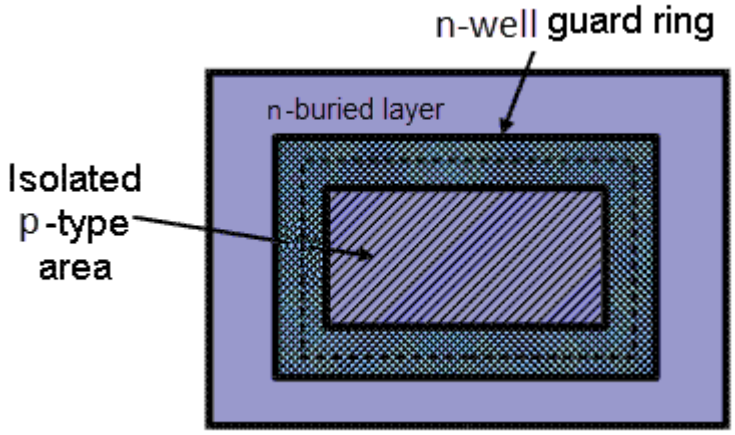
The n-well guard ring provides sufficient isolation to prevent the current from flowing horizontally between the p-well and the substrate. The n-buried layer further increases the isolation by preventing the current from flowing vertically between the p-well and the substrate. Therefore, if an “Nch” device is placed over the p-well that lies above the n-buried layer, the bulk instance terminal of this “Nch” device is isolated from any other “Nch” device placed outside of the buried layer.

For the isolation due to the buried layer to be considered effective, the buried shape must be assigned to a net.

In the example below, the n-well guard ring defines a local substrate, which is further isolated from the global substrate by an n-buried shape.

© 2023 Cadence Design Systems, Inc.

In this example, a p-well shape is not drawn. This could be intentional because the technology file does not define an explicit p-well layer.



Local substrate with a buried Layer

Related topics

Bulk Area Connectivity Extraction

Bulk Area Connectivity Extraction - Isolating the Substrate and Well Layers

Substrate and Well Isolation Using a Pseudo (Dummy) Isolation Layer

Substrate and Well Isolation Using a Buried Layer

Bulk Area Connectivity Extraction - Identify the Bulk Area

For the extractor to recognize the bulk area of a device placed over the substrate or a well area, the bulk terminal should either have:

- Physically drawn shapes that represent the areas to be extracted.
- A derived region that represents the areas to be extracted.

In both cases, the bulk terminal must be explicitly set as soft-connect for the extractor to recognize it as a bulk connection.

If the bulk terminal is implementing a tap to the substrate or the well, it is being used to polarize the substrate or the well, so the bulk terminal should not be set as soft-connect.

© 2023 Cadence Design Systems, Inc.

You can create a soft-connect bulk terminal by using the dbSetTermSoftConnect SKILL API. Because the terminal is soft-connect, it does not draw current and is used only to polarize the bulk area of the corresponding device. The supported direction for polarization is from the substrate or well layer toward the bulk instance terminal. This allows the bulk area of the device to be polarized from an appropriate external source, but does not allow the connectivity to propagate from the bulk instance terminal to the outside of the device.

- As usual, the connectivity of a bulk terminal is provided by the associated top-level net attached to the instance terminal.
- Even though it is recommended to explicitly set bulk area terminals as soft-connect, a terminal that has only pins on well layers is considered to be effectively soft-connect even if the bulk is not explicitly defined.

The connectivity extraction of a bulk terminal varies on a case basis:

- If the bulk terminal is implemented by at least one extractable pin figure, the extractor uses the corresponding geometries to detect any top-level overlap on these physical layers.
- If the bulk terminal is not implemented by any extractable pin figure (shapeless bulk terminal), the extractor derives the bulk area for that terminal using a derived layer. The derived layer represents the intersection between two or more physical layers, typically poly and diffusion. To set the connection between the bulk terminal and this derived layer, use the dbSetSoftConnectTermPinlessLayer SKILL function.

  For example, the bulk area of an NMOS device placed on a p-type substrate can be a shapeless bulk for which the bulk area is derived based on the “gate area”. In that case, you can use the following definition to derive the bulk area:

  ```
  techDerivedLayers((gateArea 10001 (PO 'and OD)))
  ```

  Important

  For the extractor to recognize the derived bulk area for connectivity extraction, the derived layer that represents the bulk area must be defined as a valid layer and used as a parameter for dbSetSoftConnectTermPinlessLayer.

  By default, a shapeless bulk terminal connects to the substrate. However, if you need to redefine this connection, you can do so using the dbSetSoftConnectTermConnectToLayer SKILL function.

- For a device having several bulk areas, either on the same bulk terminal or on different ones, the bulk areas should be physically connected so that they are properly polarized. Otherwise, the extractor creates open markers to indicate that the bulk areas should be connected. In the case of a device having several bulk terminals, such as when

© 2023 Cadence Design Systems, Inc.

implementing multi-fingered devices, all the bulk terminals must be set as must-connect terminals to allow for proper detection of the unconnected bulk areas by the extractor.

Related topics

Bulk Area Connectivity Extraction

Bulk Area Connectivity Extraction - Isolating the Substrate and Well Layers

Substrate and Well Isolation Using a Pseudo (Dummy) Isolation Layer

Substrate and Well Isolation Using a Buried Layer

Bulk Area Connectivity Extraction - Isolating the Substrate and Well Layers

Bulk Area Connectivity Extraction - Identify the Substrate and Well Tap Vias

Bulk Area Connectivity Extraction - Identify the Substrate and Well Tap Vias

A tap via (pVia or nVia) is a via that connects to a substrate area or a well layer and polarizes it.

In a p-type substrate, the technology file defines a pVia as a standard via that has its optional well or substrate layer set to “substrate”. A pVia connects to the p-type local or global substrate or to a p-well shape on which the via is placed.

The following is a sample via definition that defines a via polarizing the substrate from the `metal1` layer:

```
metal1 - > cutLayer - > (diffusionLayer && PImplantLayer) - > substrate
```

Unlike a pVia, an nVia has its optional well or substrate layer set to a physical n-well layer. An nVia connects to an n-well shape on which it is placed.

The following is a sample via definition that defines a via polarizing an n-well shape on which it is placed:

```
metal1 - > cutLayer - > (diffusionLayer && NImplantLayer) - > Nwell
```

- The extractor recognizes an nVia defined as connecting to an n-well layer even if the via master does not implement a physical n-well shape.
- An nVia does not connect to the n-buried layer, if it is defined.

© 2023 Cadence Design Systems, Inc.

The definitions for an n-type substrate are similar to those for a p-type substrate except that it is an nVia that connects to the substrate instead of a pVia. Conversely, a pVia is defined as connecting to a p-well shape.

Related Topics

Bulk Area Connectivity Extraction

Connectivity Propagation through the Substrate and Well Areas

Connectivity Propagation through the Substrate and Well Areas

As already mentioned, the extractor does not propagate the connectivity from a soft-connect object; such as a substrate area, a well shape, or a device bulk terminal to a polarizing object, such as a tap via or a diffusion shape. However, it does support connectivity to propagate between two soft connect objects that directly overlap.

For example, if you polarize a substrate or a well area and place multiple devices on it, you will observe that the connectivity propagates to the bulk instance terminals of all the devices. As earlier mentioned, no connectivity propagation is observed to the bulk instance terminals for the devices placed outside the polarized substrate or well area.

Conversely, during a *Pick From Schematic* command, if you place multiple devices whose bulk instance terminal is assigned to a net and place them over a floating substrate or a well area—an area which is not assigned to a net—you will observe no connectivity propagation from the bulk instance terminals to the substrate or well. This enforces the requirement that the substrate or the well must be polarized from an external source, such as a tap via or a diffusion shape.

Checking Connectivity

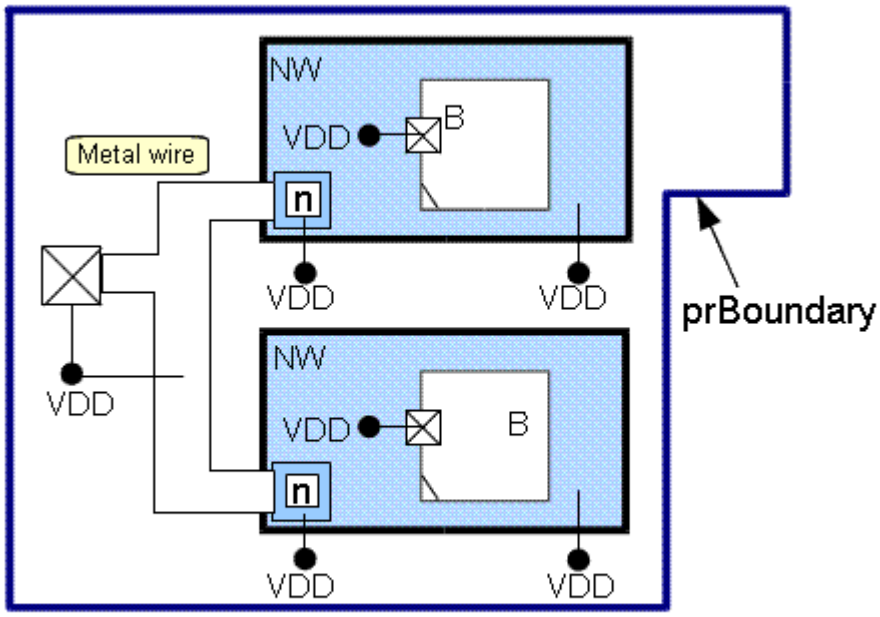
During connectivity extraction, if the extractor identifies any design violations related to substrate, well shapes, or bulk terminals, it generates appropriate open and short markers in the Annotation Browser.

- All the devices placed on a polarized substrate or well area should have their bulk instance terminal assigned to the same net. If this is not done, the extractor creates short markers between the various nets.
- If a net polarizes the bulk instance terminal of a device through the substrate or a well, the connection is considered valid irrespective of the distance between the bulk terminal and the tap via on the net polarizing it. This means that for connectivity checking, a single

© 2023 Cadence Design Systems, Inc.

tap via is sufficient to polarize the bulk of all the devices placed on a given substrate or a well area.

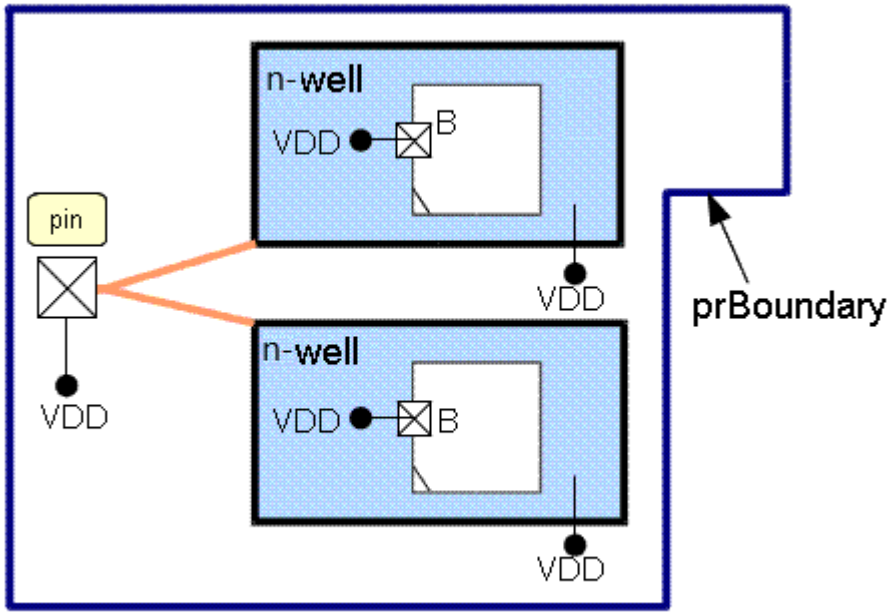
For example, in the figure below, a metal path on net VDD polarizes two well shapes from a pin on that net by using Nvias that connect the metal to the wells.



Well shapes properly polarized by a metal pin

© 2023 Cadence Design Systems, Inc.

Now, let us assume that the user deletes the VDD path. Although the two well shapes are close to each other, the extractor creates the open markers with the top-level VDD pin to indicate the preferred routing path. This is displayed in the figure below.



Open marker between a pin and two well shapes

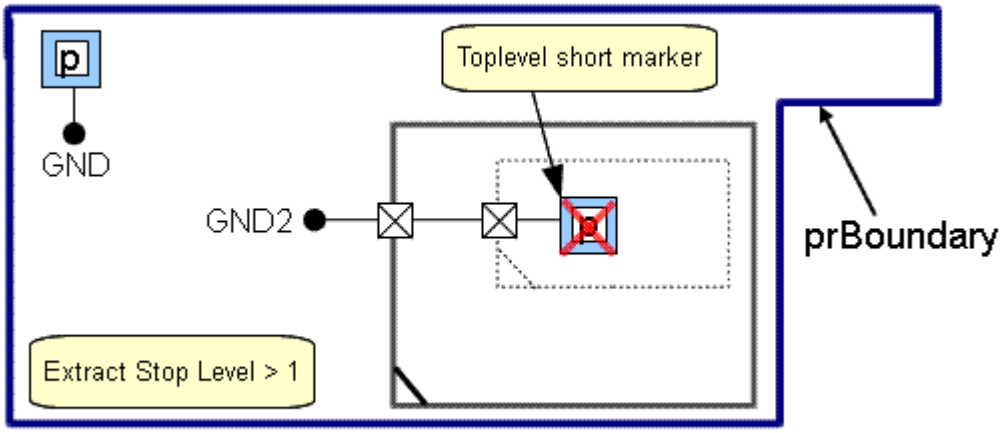
Impact of the Hierarchy on the Top-Level Connectivity Checking

When the extract stop level is increased from 0 to a higher value, the extractor is able to create appropriate connectivity violation markers for objects located in the hierarchy and, in particular, for objects that create a soft connection with a top-level object.

For example, in the following figure, the substrate is polarized on net GND by a top-level pVia, as displayed in the upper left of the figure. This creates a short condition with the pVia located

© 2023 Cadence Design Systems, Inc.

in the hierarchy and connected to the top-level net, GND2. The extractor, therefore, creates a short marker to report this design error, as shown in the figure below.

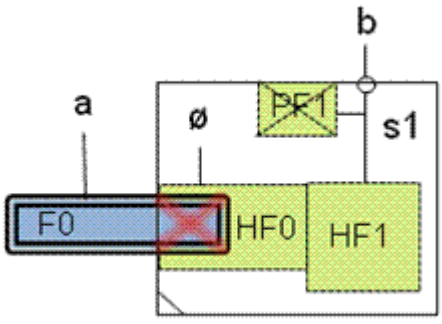


Hierarchical pVia over a global substrate

Hierarchical Overlaps Processed During Extraction

The connectivity extractor processes various types of overlaps with unassigned hierarchical shapes. Some of these scenarios are explained below.

Scenario 1: F0 overlaps HF0, HF0 overlaps HF1



ECV shape, F0, overlaps unassigned hierarchical shape, HF0

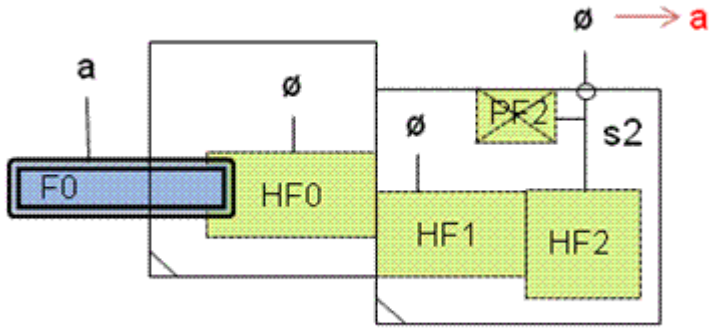
© 2023 Cadence Design Systems, Inc.

In this scenario, the ECV shape F0 assigned to net a overlaps the unassigned hierarchical shape HF0. If the extractor stop level is greater than 0, the connectivity extractor detects the overlap and chases hierarchical shapes connected to this unassigned hierarchical shape.

Here, HF0, which is an unassigned hierarchical shape, overlaps another hierarchical shape HF1, which is connected to net s1 that has terminal s1 in the instance master. Therefore, HF0 is effectively connected to net b and the overlap of HF0 and F0 creates a short between net a and net b.

If the extractor stop level is 0, no shorts are detected because the extractor ignores the hierarchical shapes HF0 and HF1.

Scenario 2: F0 overlaps HF0, HF0 overlaps HF1, HF1 overlaps HF2



F0 overlaps with HF0, HF0 overlaps HF1,
HF1 overlaps HF2

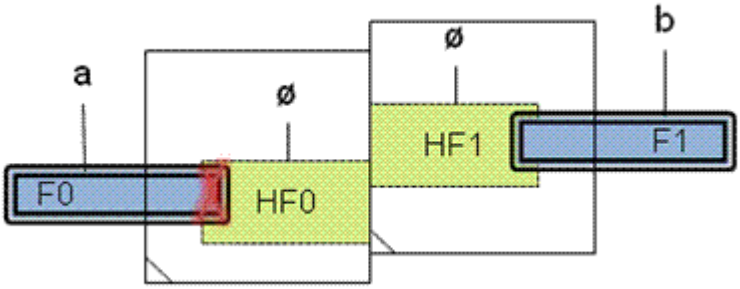
In this scenario, the ECV shape F0 assigned to net a overlaps the unassigned hierarchical shape HF0. If the extractor stop level is greater than 0, the connectivity extractor detects the overlap and chases hierarchical shapes that are connected to this unassigned hierarchical shape.

Here, HF0, which is an unassigned hierarchical shape, overlaps another unassigned hierarchical shape HF1, and HF1, in turn, overlaps HF2. The hierarchical shapes HF0, HF1, and HF2 are, therefore, effectively connected to net a and the instance terminal for terminal s2 is assigned to the same net.

If the extractor stop level is 0, the hierarchical shapes—HF0, HF1, and HF2—are ignored by the extractor and the instance terminal for terminal s2 remains unassigned.

© 2023 Cadence Design Systems, Inc.

Scenario 3: F0 overlaps HF0, HF0 overlaps HF1, and HF1 overlaps F1



F0 overlaps HF0, HF0 overlaps HF1, HF1 overlaps F1

In this scenario, the ECV shape F0 overlaps an unassigned hierarchical shape HF0. If the extract stop level is greater than 0, the connectivity extractor detects the overlap and chases hierarchical shapes that are connected to this unassigned hierarchical shape.

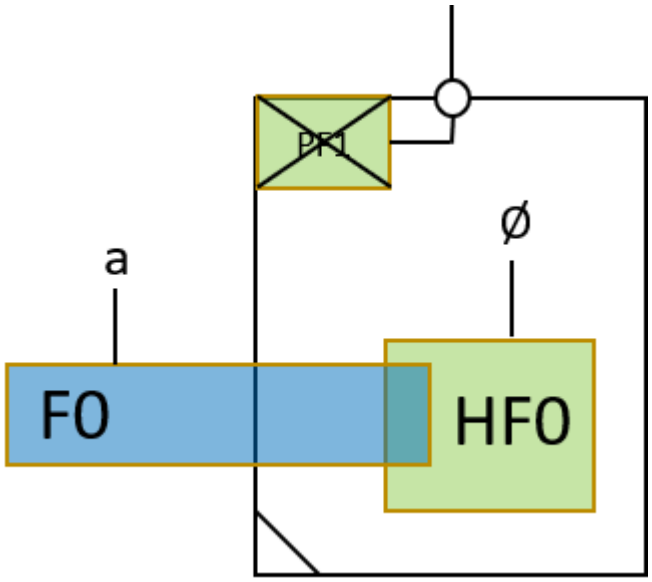
Here, HF0 overlaps another hierarchical shape HF1, which is effectively connected to net b due to its overlap with the ECV shape F0. Therefore, HF0 is also effectively connected to net b.

Because HF0 also overlaps with ECV shape F0, which is connected to net a, a short marker is created between F0 and HF0.

If the extract stop level is 0, HF0 and HF1 are ignored by the extractor. Therefore, the short is not detected.

© 2023 Cadence Design Systems, Inc.

Scenario 4: F0 overlaps HF0



F0 overlaps HF0

In this scenario, the ECV shape F0 overlaps the unassigned hierarchical shape HF0.

An illegal hierarchical overlap is reported because the ECV shape is assigned and the:

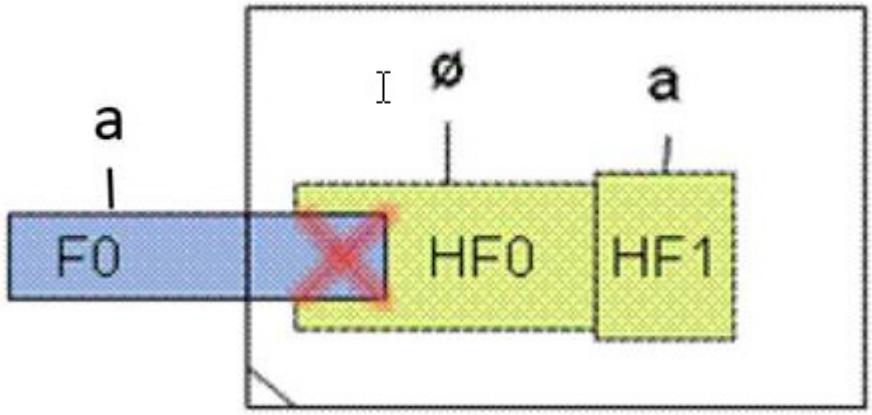
- hierarchical shape is metal or poly
- instance master has at least one implemented terminal
- instance is not transparent

If the ECV shape is unassigned, no error is reported. If the extract stop level is 0, hierarchical shape HF0 is ignored and no error is reported.

The environment variable `extractVerifyIllegalHierConnectionsIgnoreUnassignedHierShapes` controls whther an error is reported this scenario.

© 2023 Cadence Design Systems, Inc.

Scenario 5: F0 overlaps HF0, HF0 overlaps HF1

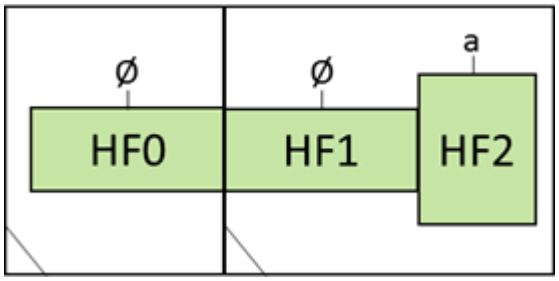


F0 overlaps HF0, HF0 overlaps HF1

In this scenario, HF0 and HF1 are two hierarchical shapes in the master of the same instance. HF1 is assigned to an internal net, a. However, this net is not connected to a terminal in the master. So, the connectivity of the hierarchical shape cannot be compared to the connectivity of the ECV shape. Therefore, an “illegal hierarchical connection” marker is created.

If the ECV shape is unassigned, the connection is considered valid and no marker is created. If the extract stop level is 0, hierarchical shapes, HF0 and HF1, are ignored by the extractor and no marker is created.

Scenario 6: HF0 overlaps HF1, HF1 overlaps HF2

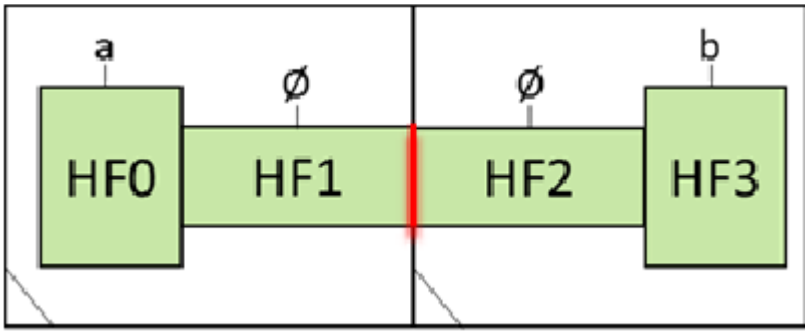


HF0 overlaps HF1, HF1 overlaps HF2

In this scenario, because HF0 is unassigned, it is effectively connected to the overlapping HF1. Therefore, an illegal hierarchical connection marker is not created, and the connection is considered valid.

© 2023 Cadence Design Systems, Inc.

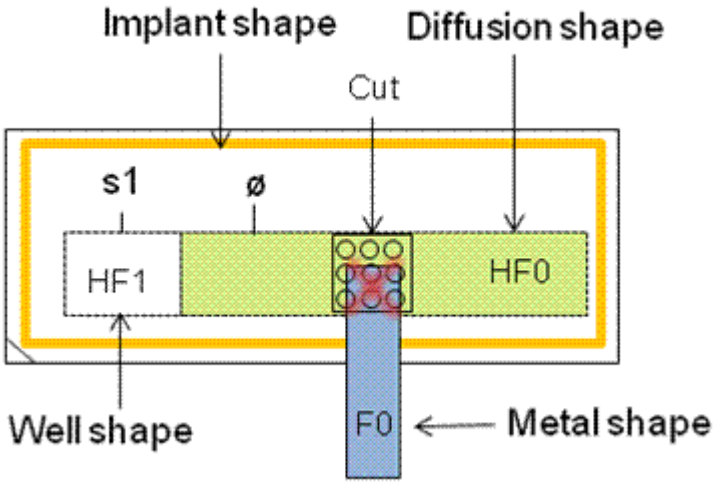
Scenario 7: Illegal hierarchical connection between two instances



Illegal hierarchical connection between two instances

In this scenario, an illegal hierarchical connection marker is created because two internal nets are connected at the overlap of the two instances.

Scenario 8: F0 connects to hierarchical well shape HF1



F0 connects to hierarchical well shape HF1

In this scenario, HF1 is a hierarchical well shape and HF0 is a hierarchical diffusion shape. In addition, an implant shape exists that establishes connectivity between the cut, diffusion, and the well shapes.

F0 is an ECV shape on a metal layer that connects to HF1 through the cut and the diffusion in accordance with the following via definition:

© 2023 Cadence Design Systems, Inc.

```
Metal -> Cut -> (Diffusion && N/P Implant) -> N/P Well
```

Since F0 is assigned to a net and HF1 is assigned to an internal net s1, the connection is considered invalid and an illegal hierarchical connection marker is created on the overlap of F0 and HF0.

If the extract stop level is 0, the hierarchical shapes, HF0, HF1, the cut shape, and the implant shape are ignored by the extractor. Therefore, the illegal hierarchical connection marker is not created.

Related Topics

Bulk Area Connectivity Extraction - Identify the Substrate and Well Tap Vias

extractVerifyIllegalHierConnectionsIgnoreUnassignedHierShapes

Connectivity Extraction from Mosaics

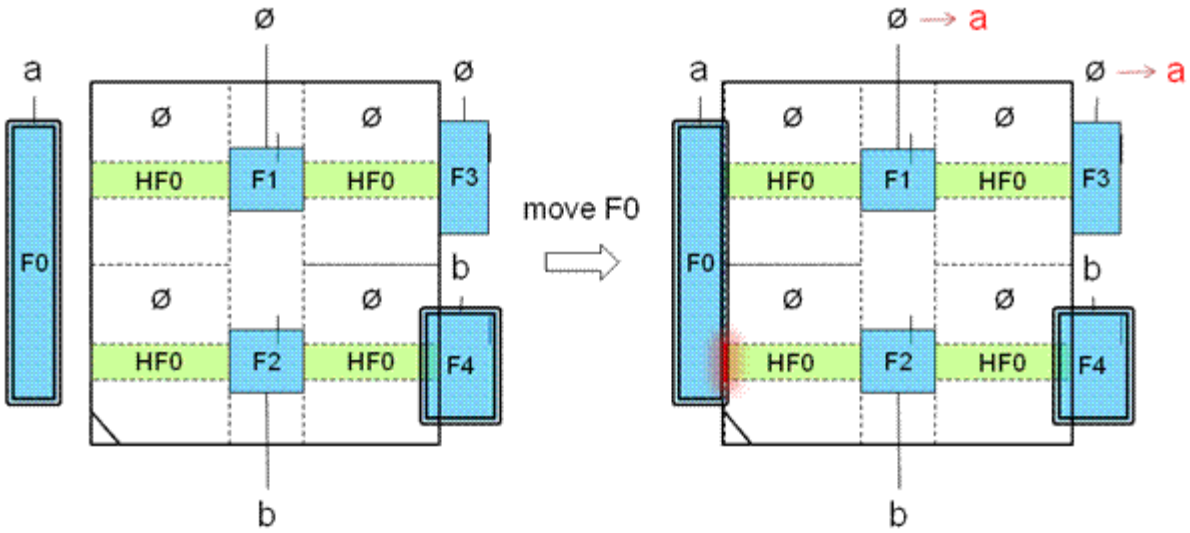
The connectivity extractor processes overlaps with each tile of a mosaic (`oaArrayInst` object) in the same way as it processes overlaps with regular instances.

- Overlaps are processed between tiles of the same mosaic.
- For pin figures that are not routed together, open markers are created if the pin figures are:
  - On the same terminal but on different tiles
  - On different terminals with instTerms on the same ECV net

© 2023 Cadence Design Systems, Inc.

Let us now consider some examples that illustrate how the connectivity extractor processes overlaps with mosaics:

Example 1: Consider a 2x2 physical-only mosaic with vertically abutting tiles, as shown in the figure below.



2x2 mosaic with vertically abutting tiles

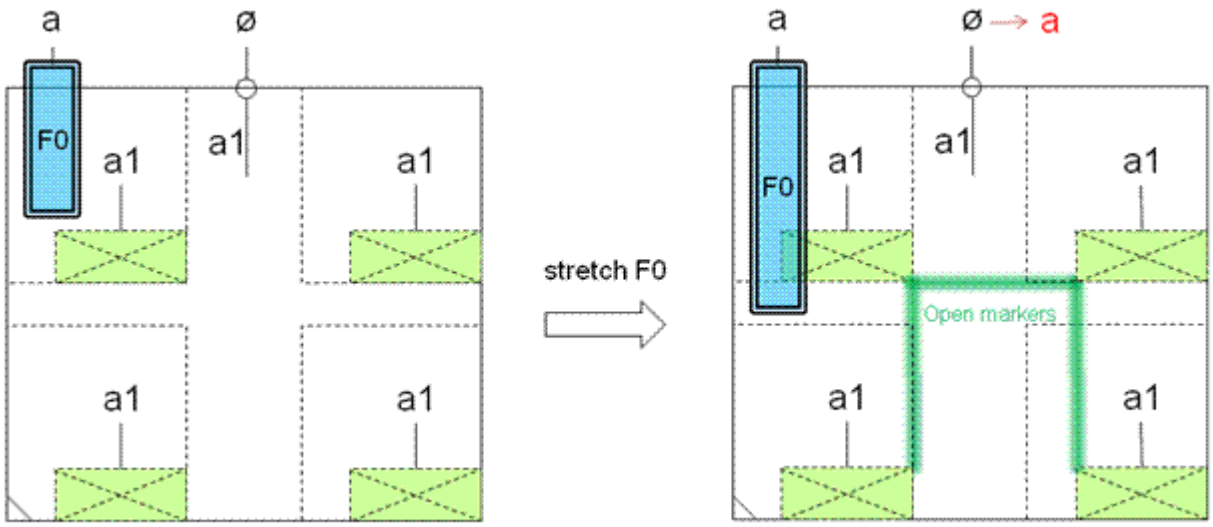
Here, F1 and F2 are ECV shapes that connect the hierarchical shapes between the tiles.

An ECV shape (edited cellview shape) is a shape in the edited cellview or a pin figure of a master that has been instantiated in the edited cellview.

If ECV shape F0 is moved horizontally to overlap the hierarchical shapes in the two left tiles of the mosaic, connectivity is propagated from F0 to F1 and F3. In addition, a short marker is created between F0 and F2 because the two are assigned to different nets, resulting in connectivity violation due to the overlap.

© 2023 Cadence Design Systems, Inc.

Example 2: Consider a 2x2 same-connectivity mosaic, as shown in the figure below.

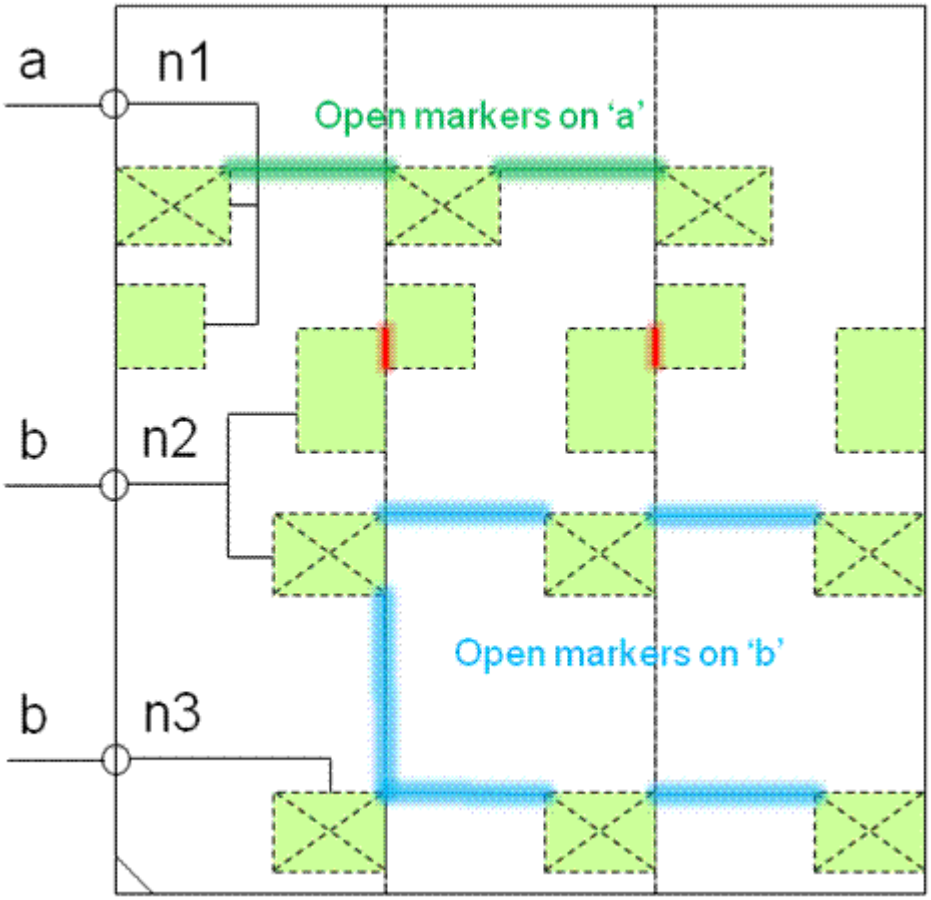


2x2 mosaic with same connectivity

Here, the ECV shape F0 is stretched to overlap the pin figure in the top-left tile of the mosaic. As a result, the unassigned instTerm, a1, which is common to all the tiles of the mosaic, is assigned to net a. Since the pin figures on terminal a1 are all on different tiles and not overlapping, open markers are created between these pin figures.

© 2023 Cadence Design Systems, Inc.

Example 3: Consider a 3x1 same-connectivity mosaic with horizontally-abutting tiles, as shown in the figure below.



3x1 same-connectivity mosaic with horizontally-abutting tiles

Here, the master has three terminals, n1, n2, and n3,connected to nets a, b, and c, respectively.

If the tiles are made to abut horizontally, overlaps are observed between hierarchical shapes that are connected to instance terminals n1 and n2. Since these overlapping shapes draw their connectivity from different ECV nets, net a and net b, two short markers are created for net a and net b at the overlap of these hierarchical shapes. In addition, two open markers are created on net a for the pin figures of terminal n1 because these are pin figures from the same terminal that lie on different tiles.

© 2023 Cadence Design Systems, Inc.

Open markers are also created for net b. These include:

- One open created between pin figures of terminals n2 and n3 as these are pin figures of different terminals lying on the same tile.
- Two opens created between the pin figures of terminal n2 as these are pin figures of the same terminal but on different tiles. Likewise, two opens are created between the pin figures of terminal n3 as the other corresponding pin figures of this instance terminal are on different tiles.

Related Topics

Mosaic Instances Generation

Connectivity Extraction for Stacked Cut Layers

A `cdsGenViaDef` is used to define a connectivity stack where several cut layers are required to connect two layers. The `cutExtraLayers` field is used to setup the additional cut layers.

The Layout XL connectivity extractor can parse such viaDefs and extract these configurations.

Example of Technology Setup

Cut layers `Via1` and `Via2` are required to connect `Metal1` to `Metal6`.

```
layerRules(

 functions(
  ( Metal1                  "metal"     12          )
  ( Metal2                  "metal"     13          )
  ( Metal3                  "metal"     14          )
  ( Via1                    "cut"       15          )
  ( Metal4                  "metal"     16          )
  ( Metal5                  "metal"     17          )
  ( Via2                    "cut"       18          )
  ( Metal6                  "metal"     19          )
) ;functions

) ;layerRules

viaDefs(
```


© 2023 Cadence Design Systems, Inc.

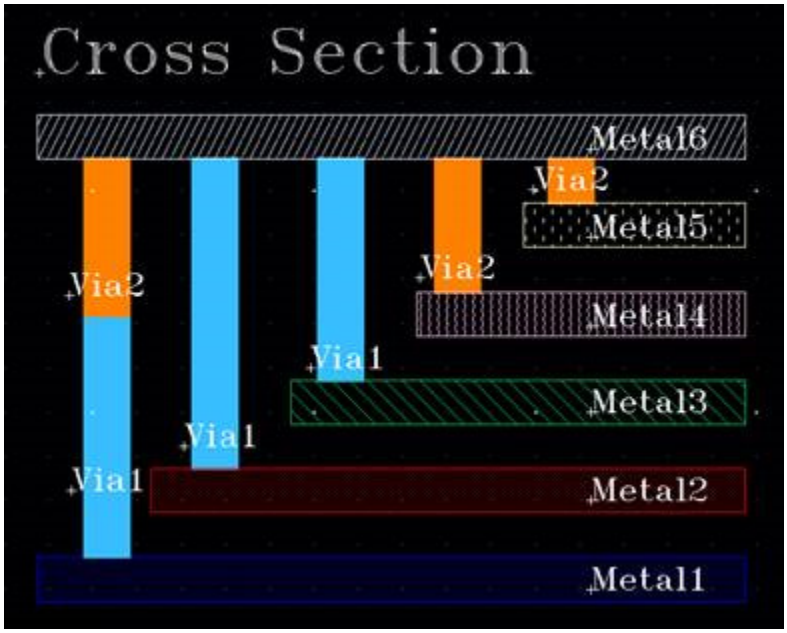
```
 standardViaDefs(
  ( M6_M5       Metal5      Metal6      ("Via2" 0.07 0.07 5.0)
     (1 1 (0.07 0.07))
     (0.005 0.03)(0.005 0.03)(0.0 0.0)(0.0 0.0)(0.0 0.0)
  )
  ( M6_M4       Metal4      Metal6      ("Via2" 0.07 0.07 5.0)
     (1 1 (0.07 0.07))
     (0.005 0.03)(0.005 0.03)(0.0 0.0)(0.0 0.0)(0.0 0.0)
  )
  ( M6_M3       Metal3      Metal6      ("Via1" 0.07 0.07 5.0)
     (1 1 (0.07 0.07))
     (0.005 0.03)(0.005 0.03)(0.0 0.0)(0.0 0.0)(0.0 0.0)
  )
  ( M6_M2       Metal2      Metal6      ("Via1" 0.06 0.06 41.0)
     (1 1 (0.06 0.06))
     (0.02 0.03)(0.0 0.03)(0.0 0.0)(0.0 0.0)(0.0 0.0)
  )
 ) ;standardViaDefs
cdsGenViaDefs(
  (M6_M1cdsg
    (layers
      (layer1      Metal1)
      (layer2      Metal6)
      (cutLayer    Via1)
    )
    (extraLayers
      (cutExtraLayers    (Via2))
    )
    (parameters
      (layer1Enc            (0.1  0.1  0.1  0.1))
      (layer2Enc            (0.1  0.1  0.1  0.1))
      (cutWidth             0.3)
      (cutHeight            0.3)
    )
  )
 ) ;cdsGenViaDefs

) ;viaDefs
```


© 2023 Cadence Design Systems, Inc.

```
constraintGroups(
  ( "virtuosoDefaultExtractorSetup"nil

    interconnect(
     ( validLayers    (Metal1 Metal2 Metal3 Via1 Metal4 Metal5 Via2 Metal6) )
     ( validVias      (M6_M1cdsg M6_M2 M6_M3 M6_M4 M6_M5) )
    ) ;interconnect
  ) ;virtuosoDefaultExtractorSetup
) ;constraintGroups
```



Related Topics

cdsGenViaDef

© 2023 Cadence Design Systems, Inc.

Connectivity Extraction for Recognition Layer

A `stdViaDef` is used to define a connectivity model in which an extra layer is needed to connect to the diffusion layer. This extra layer is called the recognition layer.




```
("MD"    "OD"    ("MD") "NP"      "RECOGNITION" "NW"           )

(layer1 layer2  cut    implant1 implant2       well/substrate)
```

To prevent unintended changes to the behavior with existing technologies:

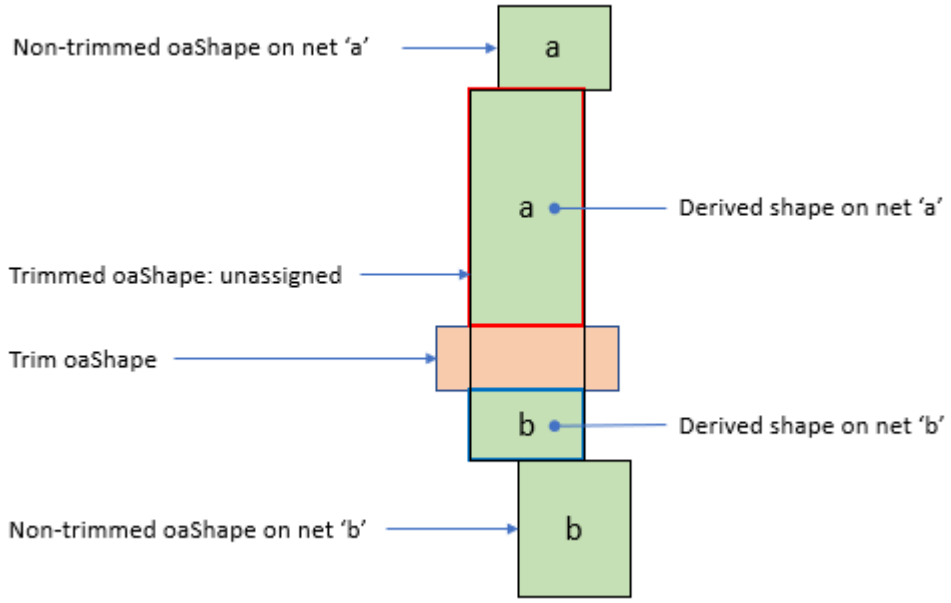
- The extra layer is only considered:
  - if it is a recognition layer.
  - for a connection between diffusion and well or substrate.
- Checks are done only to ensure that layers OD, NP, and NW have an intersection and layers OD, RECONGITION, and NW have an intersection. No check is done to ensure that all four layers (OD, NP, RECONGITION, and NW) have an intersection.
- Slots implant1 and implant2 are interchangeable.

Support for Shape Trimming during Connectivity

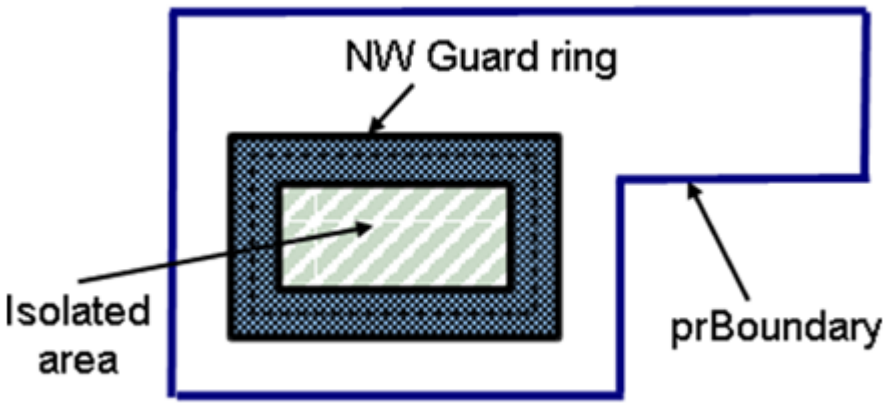
A trimmed shape or a trimmed substrate has one or several areas that can have different connectivity. Trimmed `oaShapes` and `oaVias` with a trimmed shape are set as *unassigned*.

© 2023 Cadence Design Systems, Inc.

The figure below represents a shape trimmed into two parts by a trim shape. In this example, there are two derived shapes, one assigned to net `a` and the other assigned to net `b`. The trimmed shape is set as unassigned in the database.

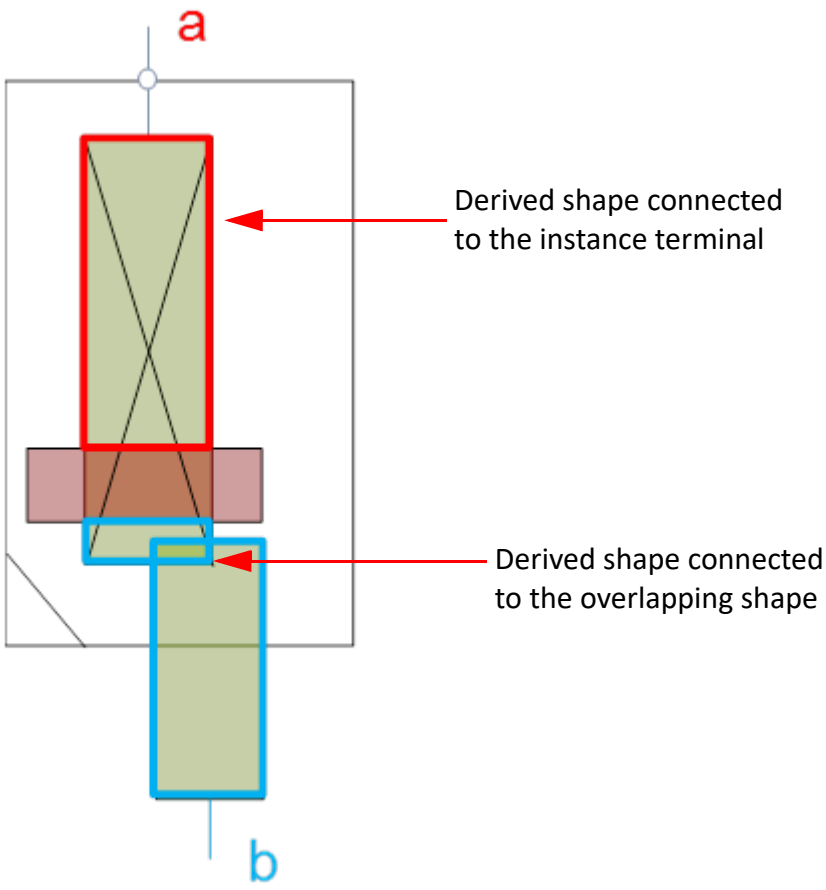


The figure below represents a p-substrate divided into two areas by a nwell guard ring.



© 2023 Cadence Design Systems, Inc.

If an instance pinFig is trimmed into several parts by a trim shape, as displayed in the figure below, only one of the derived shapes is considered to be connected to the instance terminal. The other derived shape takes the connectivity of an overlapping shape, if any.



The geometry and connectivity of the derived shapes is preserved between Layout XL sessions as this information is saved in the database.

Related Topics

Bulk Area Connectivity Extraction - Isolating the Substrate and Well Layers

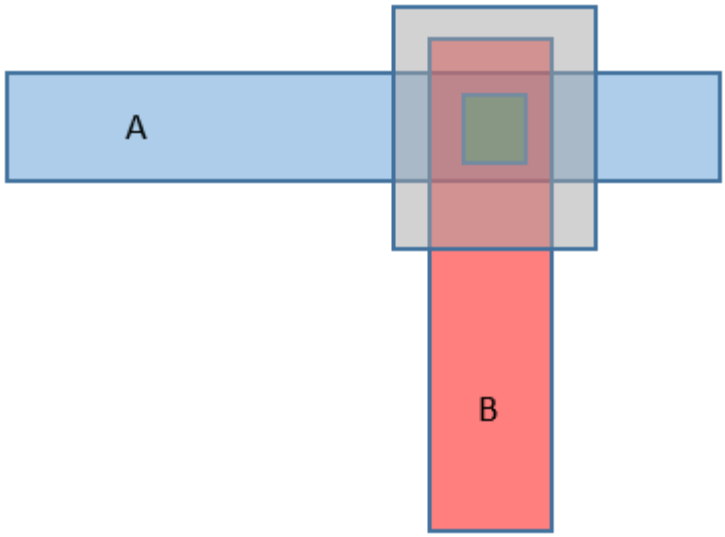
Support for Cut Shape Trimming during Extraction

Let us consider a few scenarios illustrating how cut shapes on a `validLayer` can be trimmed, and how the extractor treats such connections.

© 2023 Cadence Design Systems, Inc.

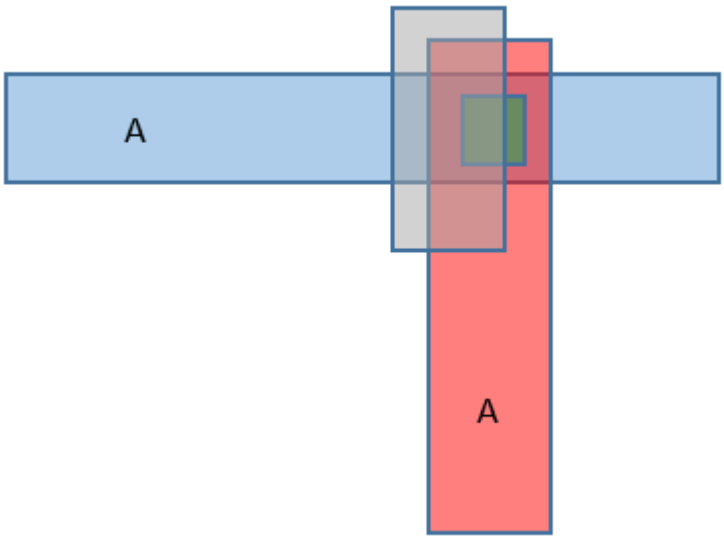
Scenario 1: Two metal shapes that are disconnected because the cut shape is completely trimmed

In the figure below, the two metal shapes are disconnected. The cut shape is completely trimmed, it does not gain connectivity from either net and exists as a floating shape.



Scenario 2: Two metal shapes connected by a partially-trimmed cut shape

In the figure below, two metal shapes are connected by a shape that is partially cut. Partially-trimmed cut shapes acquire the connectivity of the connected metal shapes. In this case, the cut shape is assigned to net `A`.



© 2023 Cadence Design Systems, Inc.

Related Topics

Extracting a Top-Level Design

Support for Color-Dependent Shape Trimming

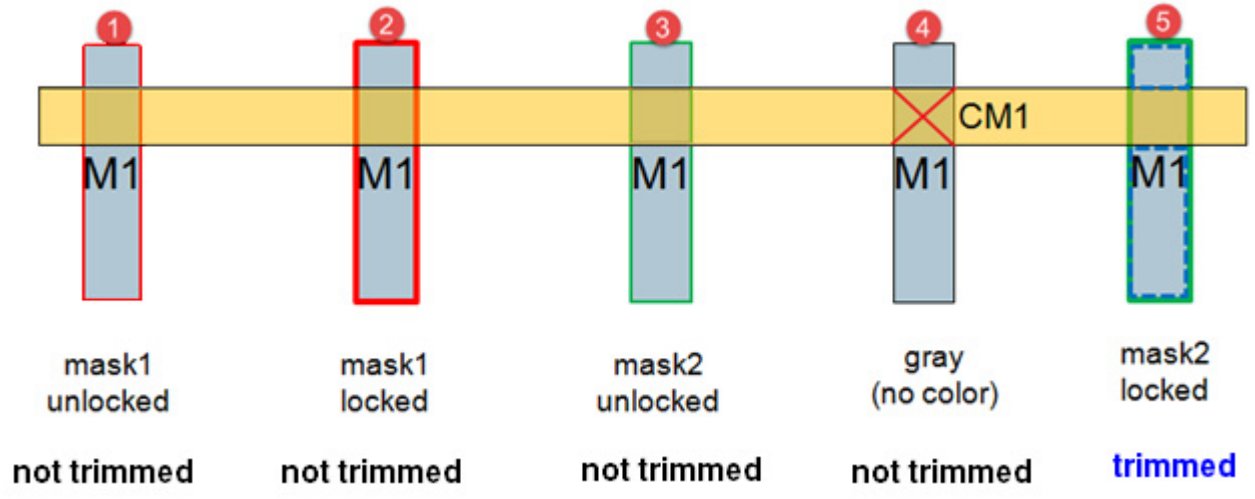
Layout XL recognizes color-dependent stopped and stop layers.

You can use the `techDerivedLayers` section of the technology file to create derived layers for trimming shapes on a specific color mask and color state.

Defining Color-Dependent Stopped Layers

You can set up the technology file to define that only the shapes belonging to a particular color and color state are trimmed by shapes on another layer.

Let us consider that only the M1 shapes *locked* on a particular color—`mask2`—are trimmed by the shapes on layer `CM1`, as displayed in the figure below. This means that `M1` shapes that are *unlocked* or on another color—`mask1`—or without a color are not trimmed by the `CM1` shapes.



In the figure above:

1. is not trimmed because it is neither *locked* nor on `mask2`.
2. is not trimmed because it is not on `mask2`, although it is *locked.*

© 2023 Cadence Design Systems, Inc.

3 is not trimmed because it is *unlocked*, although it is on `mask2`.

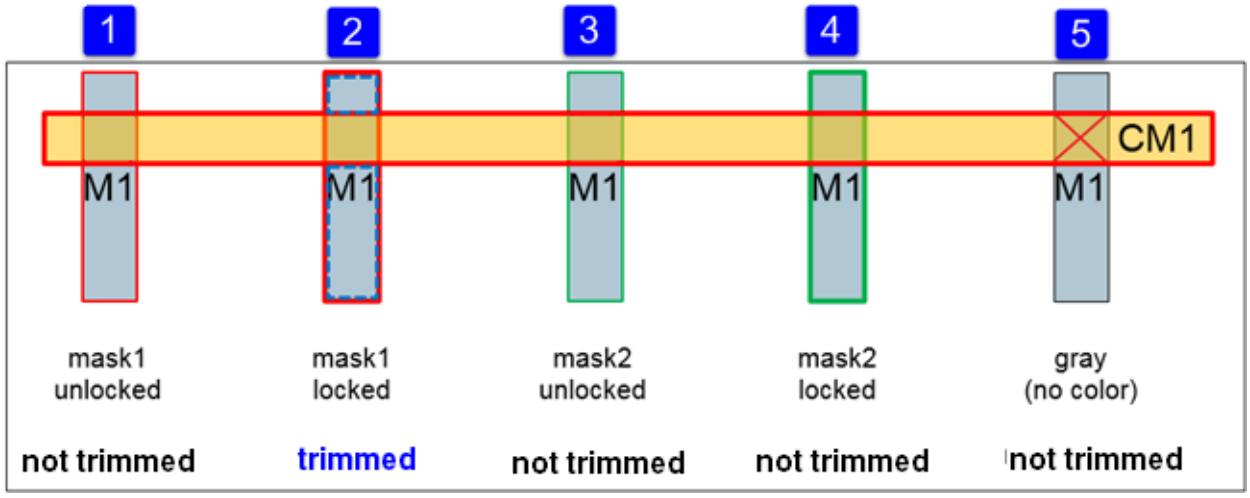
4 is not trimmed because it has no color specified (gray).

Because the mask color is not specified; the shape may be on any mask, `mask1` or `mask2`, at the time of fabrication. Therefore, the connectivity extractor reports the overlap as invalid and generates an illegal overlap marker.

5 is trimmed because the shape meets both the conditions—`mask2` color and color status set as *locked*.

Defining Color-Dependent Stop Layers

You can set up the technology file to define that only the shapes belonging to a particular color and color state are trimmed by shapes belonging to a particular color and color state on another layer.
Let us consider that *locked* shapes on `mask1` of the stopped layer are only trimmed by *locked* shapes on `mask1` of the stop layer, as displayed in the figure below.



In the figure above:

1 is not trimmed because it is *unlocked*, although the shape is on `mask1`.
2 is trimmed because the shape meets both the conditions—`mask1` color and color status set as *locked*.

3 is not trimmed because it is neither *locked* nor on `mask1`.

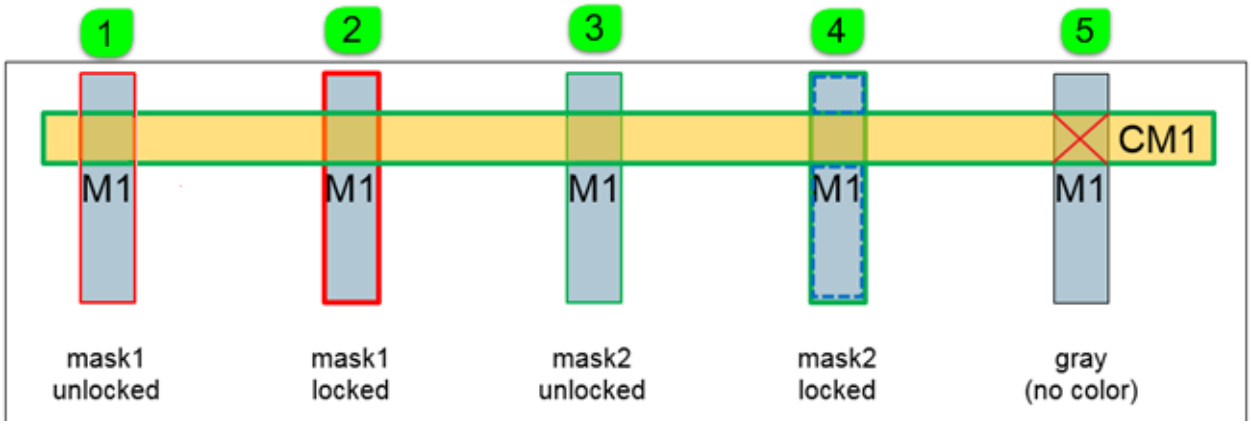
4 is not trimmed because it is not on `mask1`, although it is *locked.*

5 is not trimmed because it has no color specified (gray).

© 2023 Cadence Design Systems, Inc.

Because the mask color is not specified; the shape may be on any mask, `mask1` or `mask2`, at the time of fabrication. Therefore, the connectivity extractor reports the overlap as invalid and generates an illegal overlap marker.

Also, let us consider that *locked* shapes on `mask2` of the stopped layer are only trimmed by *locked* shapes on `mask2` of the stop layer, as displayed in the figure below.



In the figure above:

1 is not trimmed because it is neither *locked* nor on `mask2`.

2 is not trimmed because it is not on `mask2`, although it is *locked.*

3 is not trimmed because it is not *locked*, although it is on `mask2`.

4 is trimmed because the shape is `locked` on `mask2`.

5 is not trimmed because it has no color specified (gray).

Because the mask color is not specified; the shape may be on any mask, `mask1` or `mask2`, at the time of fabrication. Therefore, the connectivity extractor reports the overlap as invalid and generates an illegal overlap marker.

Related Topics

validLayers – Color-Dependent Stopped and Stop Layers Specification

© 2023 Cadence Design Systems, Inc.

Enabling Virtual Connections

Virtual connections are logical electrical connections between sets of physically connected shapes.

Listed below are some key terms to be understood before learning about virtual connections and how these connections are processed by the Layout XL connectivity extractor.

- **Attached Label**: Label (`oaText`) attached to a shape in the edited cellview. For information about how the Layout XL connectivity extractor processes connectivity from attached labels, see Connectivity from Attached Labels to Consider During Extraction.
- **Stamp Label**: Label (`oaText`) stamping a shape either in the edited cellview or in the hierarchy. For a label to stamp a shape, the origin of the label must overlap the shape. In addition, in the technology file `stampLabelLayers` rule, the stamp layer must be defined as stamping the shape layer-purpose pair. For information about how the Layout XL connectivity extractor processes connectivity from stamp labels, see Connectivity from stampLabels During Extraction.
- **Virtual-connect label**: Label (`oaText`) that has the specific syntax to tag shapes as virtually connected. By default, for a label to be recognized as a virtual-connect label, the label must have the virtual-connect separator, (`:`), such as `vdd:`. A virtual-connect label can be attached or stamped to the shape.

  See Connectivity from Attached Labels to Consider During Extraction and Connectivity from stampLabels During Extraction.

  You can change the virtual-connect separator to a value other than colon (:). See Connectivity Form in the Connectivity form for more information.

Layout XL recognizes virtual connections. This implies physically disconnected shapes that are on the same net are recognized as *connected*, if the shapes have a label containing the net name followed by a colon (:), such as `vdd:`. During connectivity extraction, no open markers are created between virtually connected shapes, if the connectivity extractor is run with the *Use (identifier) to define nets as virtually connected* selected.

For more information about the *Use (identifier) to define nets as virtually connected* option, see Connectivity tab in the Connectivity form.

However, for the virtual-connect labels to be recognized by the connectivity extractor, the labels must exist in the edited cellview. The shapes that these virtual-connect labels tag can exist in the edited cellview or in the hierarchy.

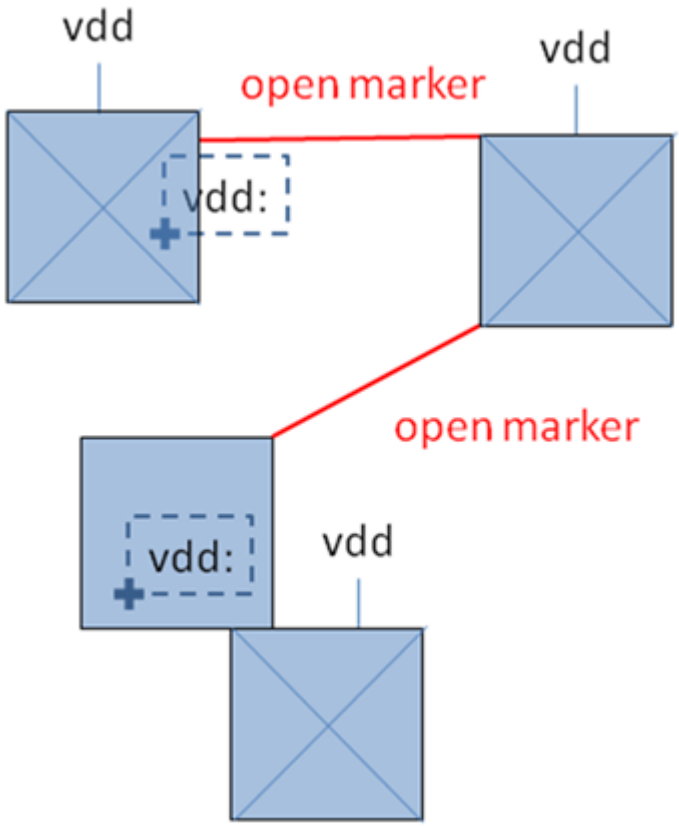
Depending on the factors listed below, the connectivity extractor processes connectivity between shapes on the same net differently.

© 2023 Cadence Design Systems, Inc.

- Existence of virtual-connect labels on the extractable shapes
- Status of the *Use (identifier) to define nets as virtually connected* option
- Physical connection between the extractable shapes

Let us consider some scenarios.

- *Case 1 – Two virtually-connected sets of shapes,* **Use (identifier) to define nets as virtually connected** *is deselected*

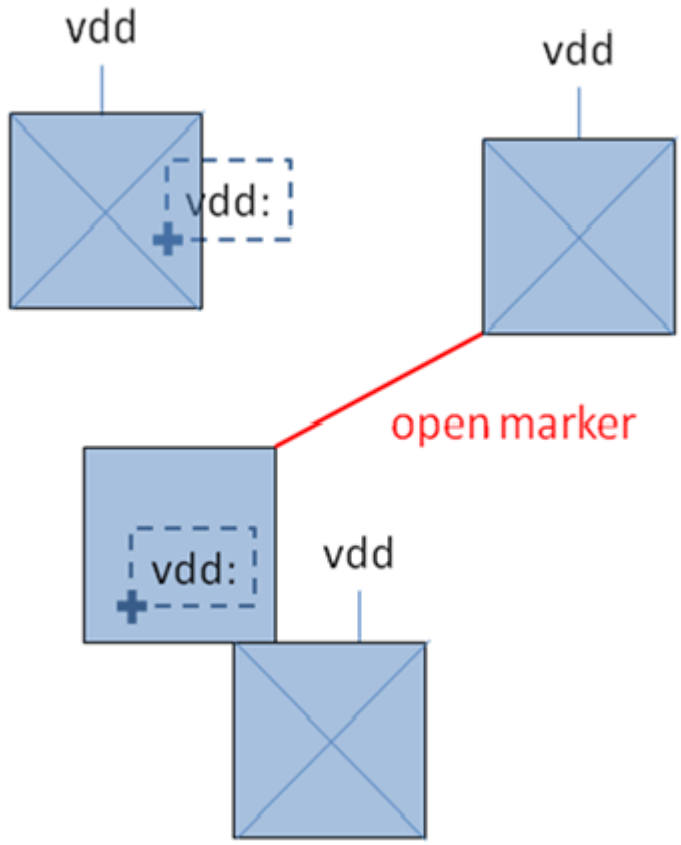


Use (identifier) to define nets as virtually connected deselected

In the figure above, there are three sets of physically connected shapes on net `vdd`. Two of these sets are tagged with a virtual-connect label, `vdd:`. But, the *Use (identifier) to define nets as virtually connected* option in the Connectivity tab of the Connectivity form is disabled. Therefore, the connectivity extractor processes the virtual-connect labels as regular labels and creates two open markers, as displayed.

© 2023 Cadence Design Systems, Inc.

- *Case 2 – Two virtually-connected sets of shapes,* **Use (identifier) to define nets as virtually connected** *is selected*



Use (identifier) to define nets as virtually connected selected

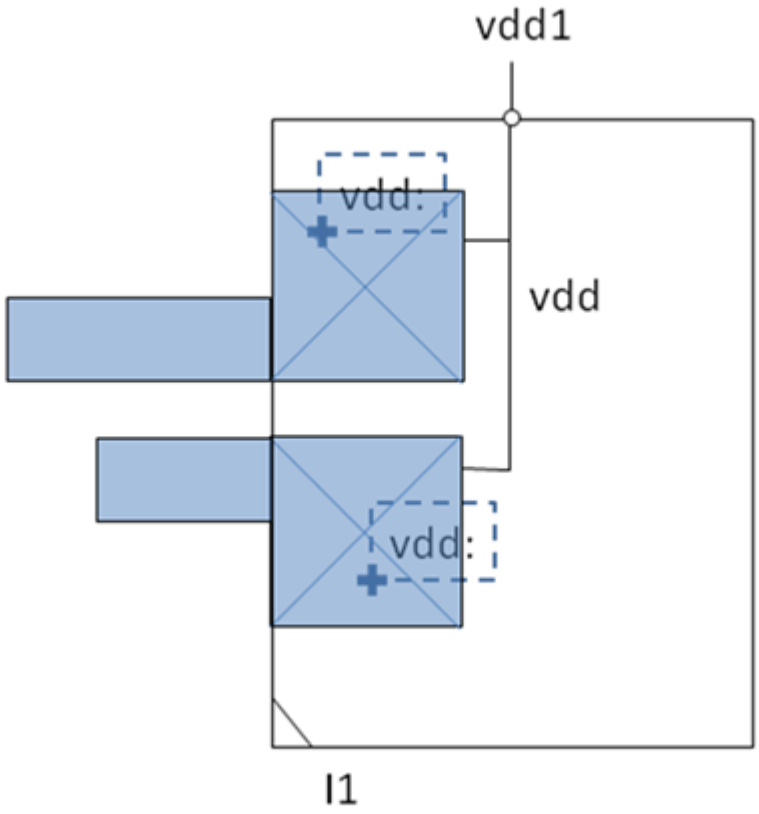
In the figure above, since the *Use (identifier) to define nets as virtually connected* option is selected, the extractor recognizes the virtual-connect labels on two sets of physically connected shapes and considers the sets as virtually-connected. Therefore, only one open marker is created with the non-virtually connected set of shapes, as shown in the figure above.

- *Case 3 – Two instance pinFigs virtually-connected in master,* **Use (identifier) to define nets as virtually connected** *is deselected*

  If *Use (identifier) to define nets as virtually connected* is deselected, the connectivity extractor ignores the virtual-connect labels on the pinFigs of an instance. Therefore, no open markers are created on instance pinFigs that are not physically connected, as shown in the figure below.

© 2023 Cadence Design Systems, Inc.

- *Case 4 – Instance pinFigs virtually connected in the master and the design, Use*



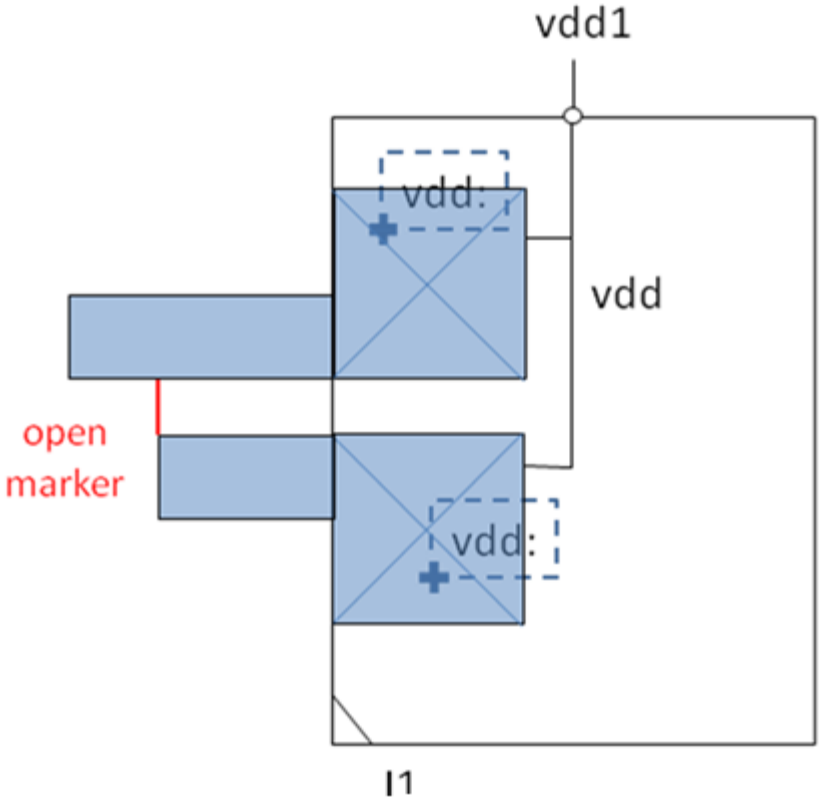
Instance pinFigs virtually connected in master, **Use (identifier) to define nets as virtually connected** de*selected*

(identifier) to define nets as virtually connected is selected

If the pinFigs of an instance are virtually connected in the master and *Use (identifier) to define nets as virtually connected* is selected, the connectivity extractor considers the pinFigs as physically disconnected in the master even though they belong to the same terminal. Because the pinFigs are defined as virtually connected in the master, they must be connected higher up in the hierarchy to complete the connection. Else, an

© 2023 Cadence Design Systems, Inc.

open marker is created between the two shapes in the edited cellview to indicate that the shapes must be connected.

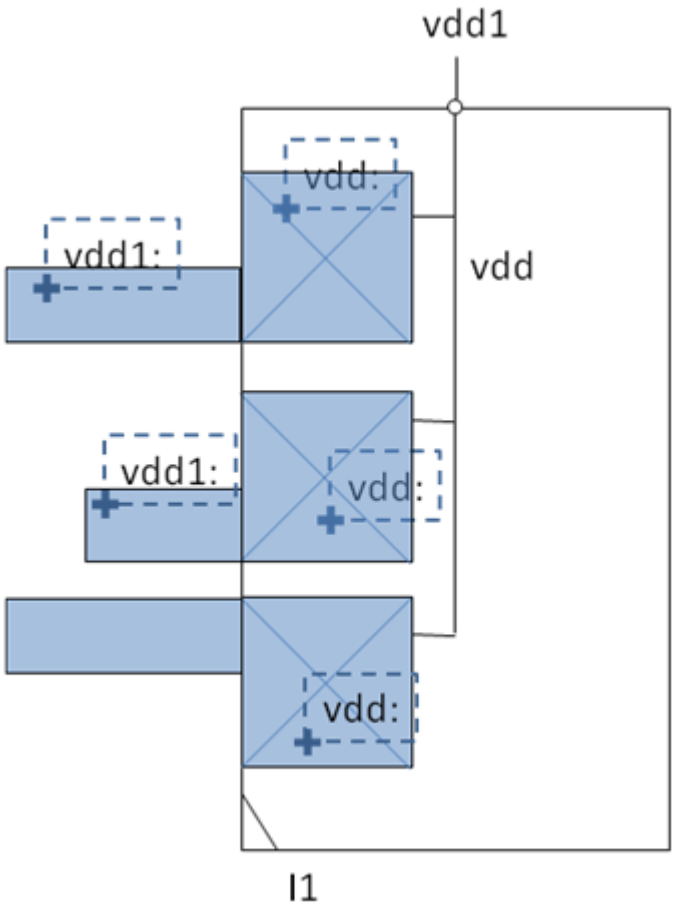


- *Case 5* – Instance pinFigs virtually connected in the master and the edited cellview, U*se (identifier) to define nets as virtually connected* is deselected

  When two sets of physically connected shapes are virtually connected in the master, they must be physically connected in the top-level design for the connection to be considered complete. Else, an open marker is created during the connectivity extraction.

© 2023 Cadence Design Systems, Inc.

But, when *Use (identifier) to define nets as virtually connected* is deselected, the connectivity extractor ignores virtual-connect labels.In this case, no open markers are reported for virtually connected instance pinFigs.



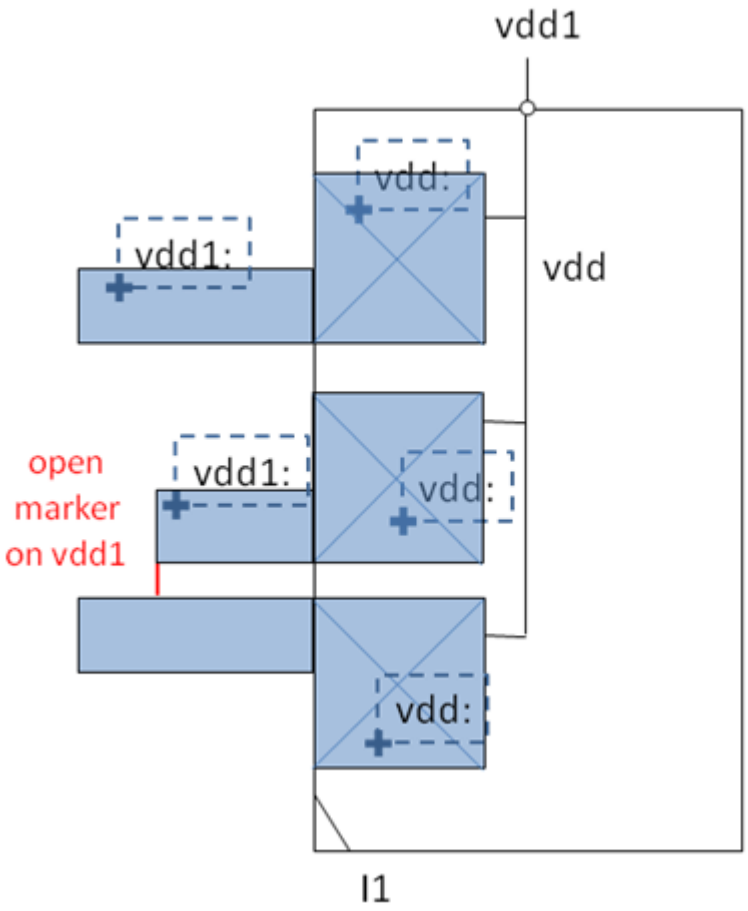
Instance pinFigs virtually connected in the master and edited cellview, **Use (identifier) to define nets as virtually connected** d*eselected*

- *Case 6 – Instance pinFigs virtually connected in the master and the edited cellview,* **Use (identifier) to define nets as virtually connected** *is selected*

  When *Use (identifier) to define nets as virtually connected* is selected, the instance pinFigs virtually connected in the master must either be physically connected or virtually

© 2023 Cadence Design Systems, Inc.

connected in the edited cellview. Else, the connectivity extractor creates an open marker between the instance pinFigs, as shown in the figure below.



Instance pinFigs virtually connected in the master and edited cellview, **Use (identifier) to define nets as virtually connected** *selected*

Related Topics

Connectivity

© 2023 Cadence Design Systems, Inc.

Markers Generated by Connectivity Extractor

Markers are flashing boxes in the layout canvas that indicate electrical shorts or invalid overlaps. In Layout XL, markers are generated by the connectivity extractor, and these markers are displayed in the *Connectivity* tab of the Annotation Browser Assistant assistant.

The Annotation Browser is replete with features to allow quick access and simple control of the displayed markers. You can use the:

- Annotation Browser Toolbar to filter and manage the violation markers displayed in the browser pane, and to define how the markers are displayed in the design window.
- Annotation Browser Paneto display the markers corresponding to each application category supported by a tab. A marker selected in the browser pane is automatically highlighted in the layout canvas, and vice versa.
- Annotation Browser Shortcut Menus to list the options to enable marker viewing and sorting.
- Annotation Browser Options Form to control the marker highlighting.
- Incomplete Net Filter Form to control the display of incomplete net markers for the specified nets.

Related Topics

Grouping Markers in the Annotation Browser

Sorting Markers in the Annotation Browser

Filtering Markers in the Annotation Browser

Assigning Colors to Markers

Zooming In on Markers

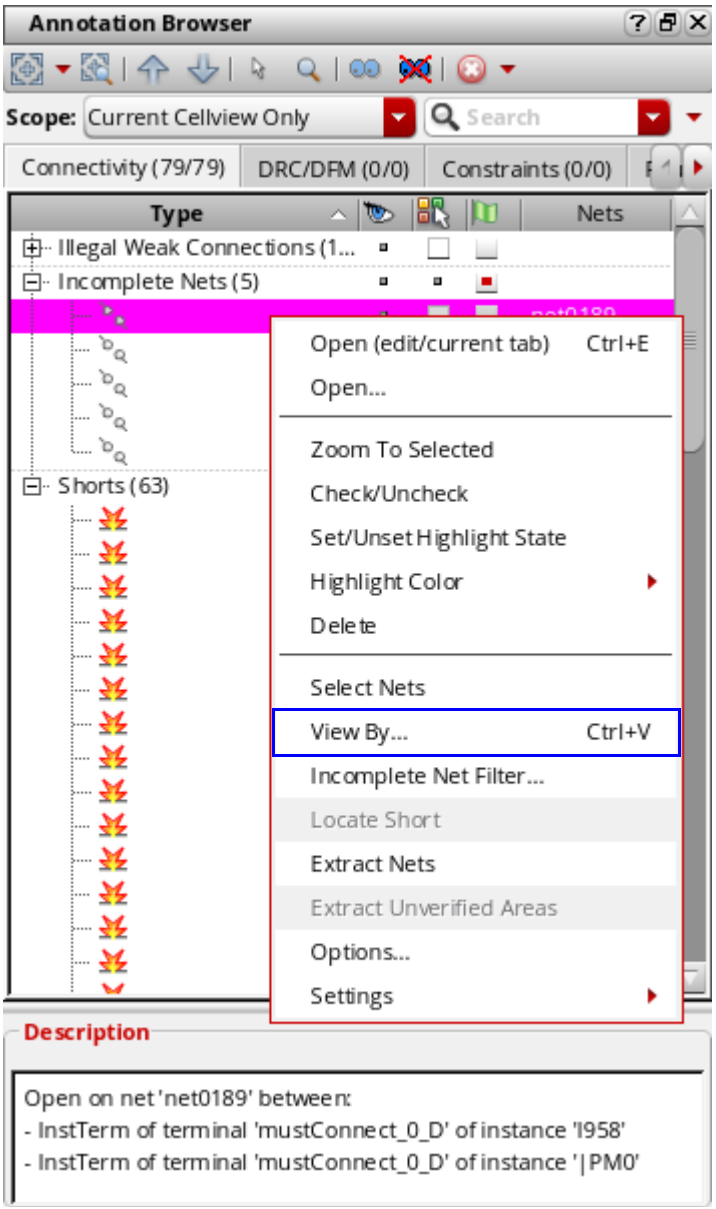
Fixing Markers Interactively in Annotation Browser

© 2023 Cadence Design Systems, Inc.

Grouping Markers in the Annotation Browser

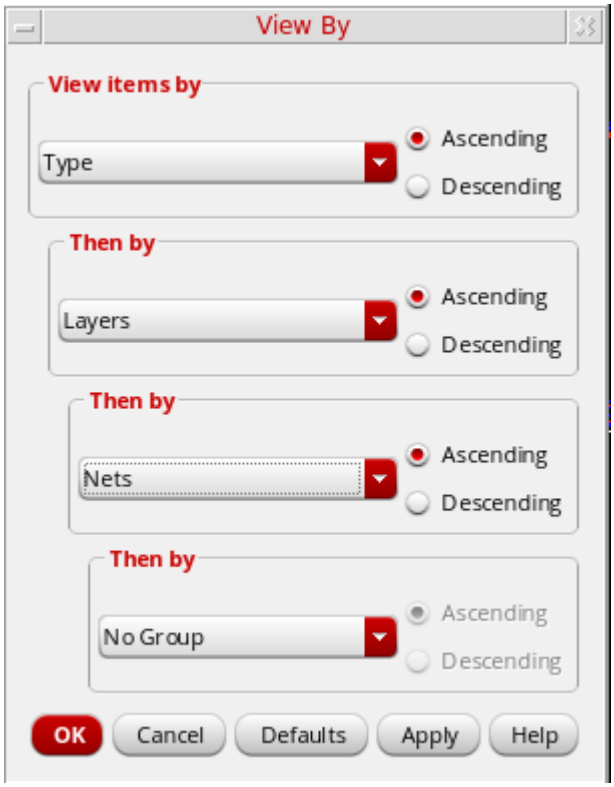
To change the way the markers displayed in the Annotation Browser are grouped:

1. In the browser pane, right-click anywhere and choose



© 2023 Cadence Design Systems, Inc.

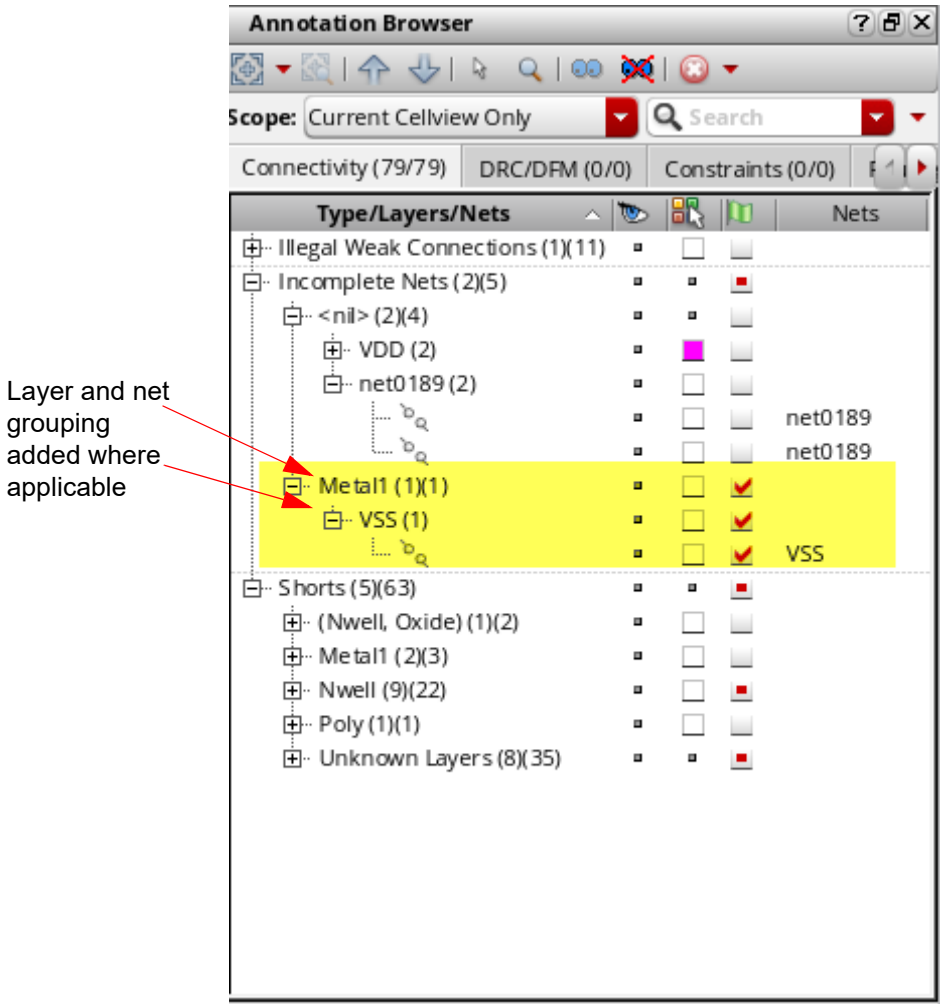
The View By form is displayed showing the current groupings. In this example, we will change these to group first by *Type*, then by *Layers*, and finally by *Nets*, with each group sorted in *Ascending* order.



2. Click *OK* to apply the changes and to close the form.

© 2023 Cadence Design Systems, Inc.

Where the layer information is available, the markers in the browser pane are grouped by layers.

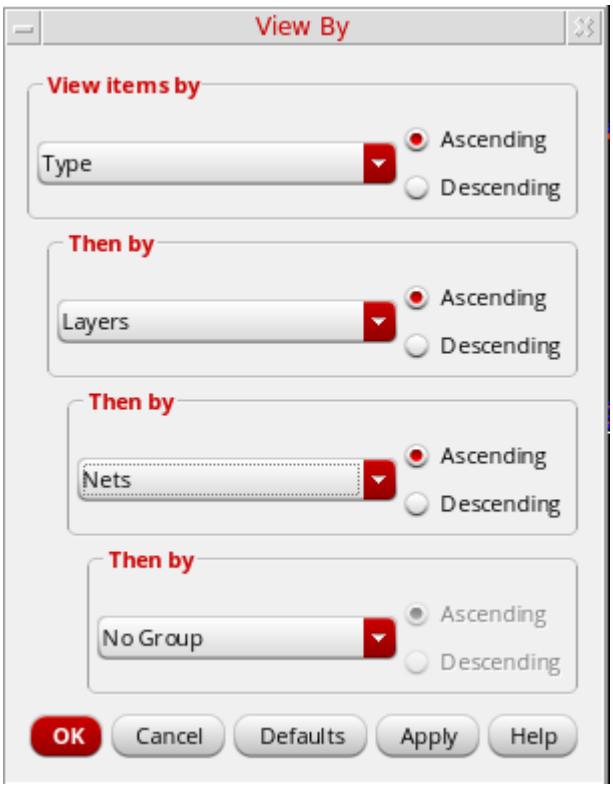


Restoring Default Groupings in the Annotation Browser

To restore the default grouping defined for the tab:

© 2023 Cadence Design Systems, Inc.

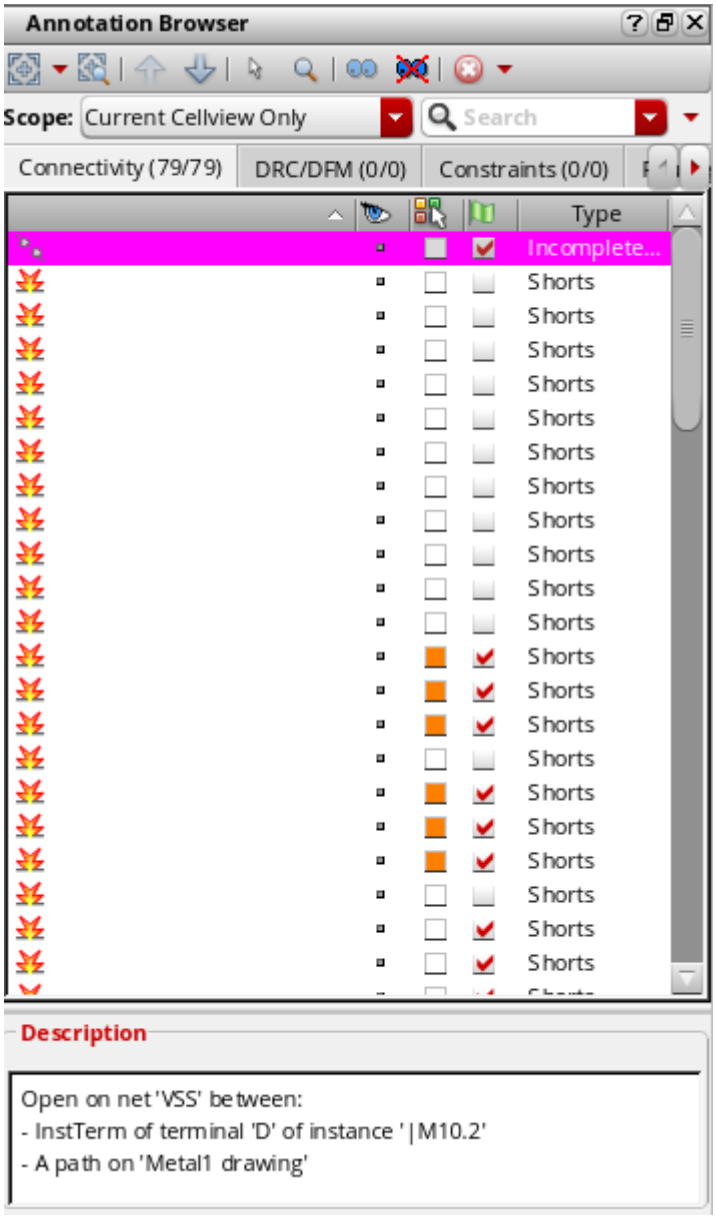
➤ Click *Defaults* in the View By form.



© 2023 Cadence Design Systems, Inc.

Annotation Browser Display with No Grouping Rules

If you specify no grouping, the Annotation Browser tree view is flattened, the tree column is removed, and the previous Top grouping column is restored and used to define how the markers are sorted.



Related Topics

View By Form

© 2023 Cadence Design Systems, Inc.

Annotation Browser Assistant

Annotation Browser Tabs

Annotation Browser Shortcut Menus

Grouping Markers in the Annotation Browser

Sorting Markers in the Annotation Browser

Filtering Markers in the Annotation Browser

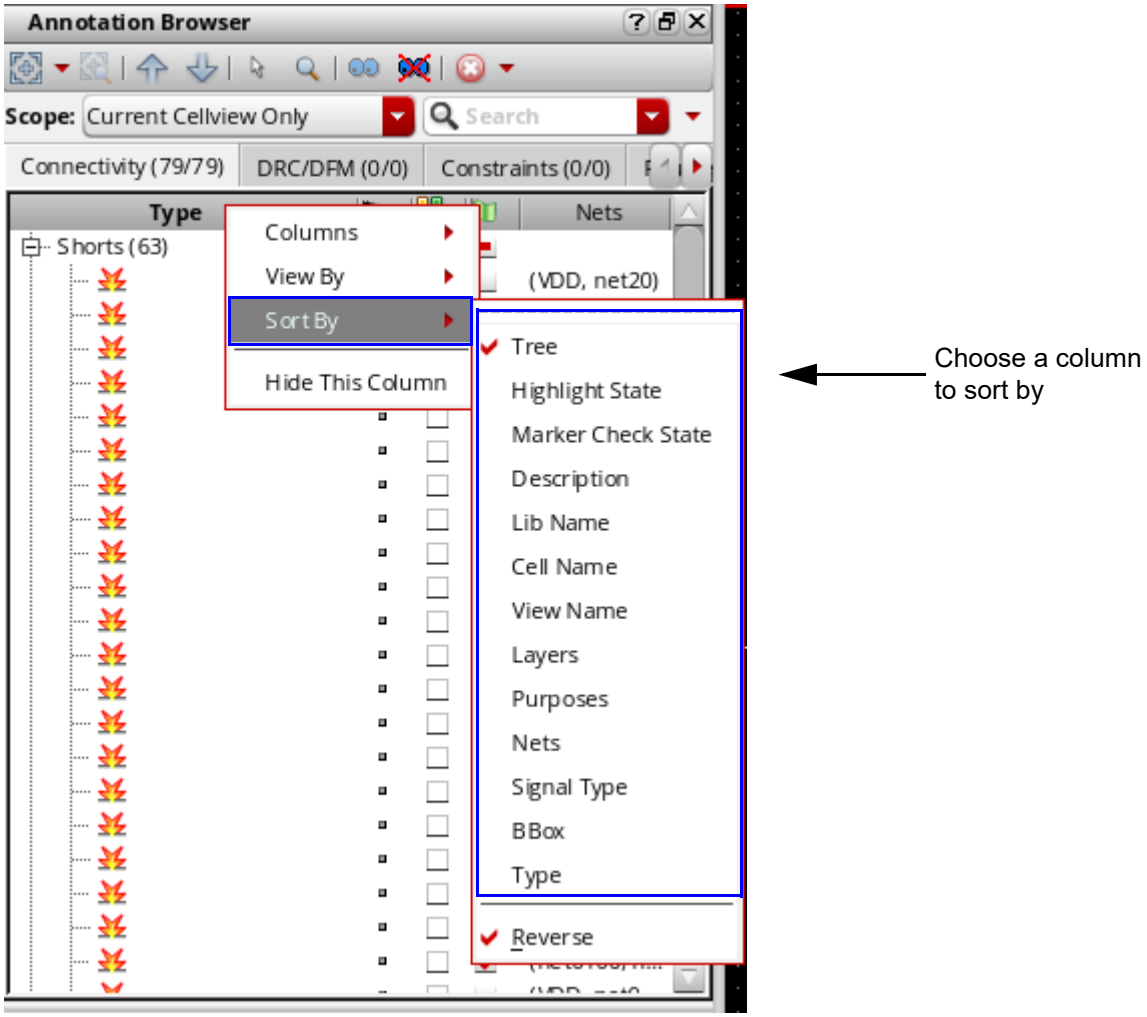
Sorting Markers in the Annotation Browser

To sort the markers in the browser pane by a different column than the one currently set:

1. Right-click the header of the column for which you want to sort the markers.

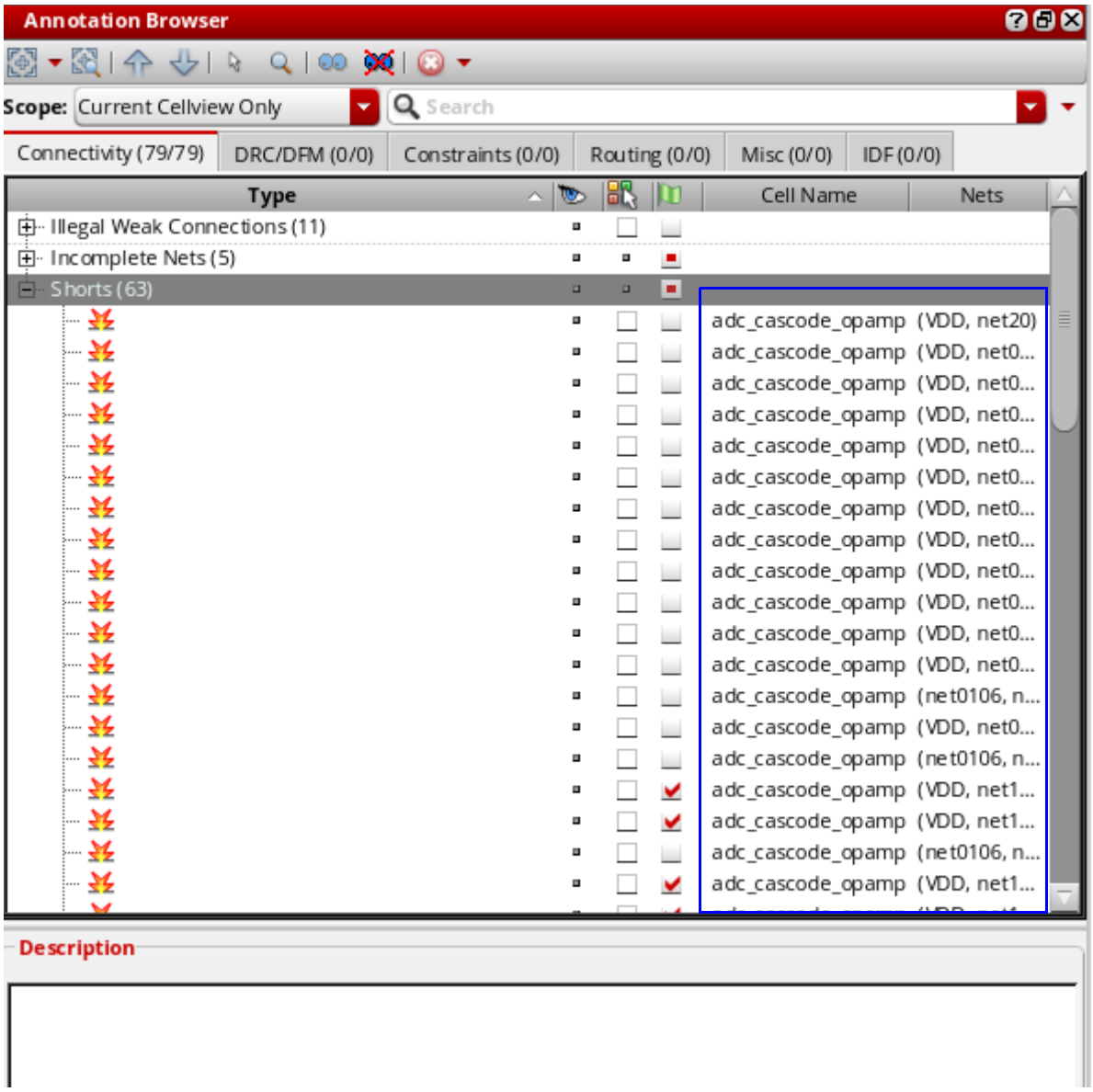
© 2023 Cadence Design Systems, Inc.

2. Choose *Sort By* from the shortcut menu and then select the column by which to sort the markers. For this example, let us sort by *Cell Name*.



© 2023 Cadence Design Systems, Inc.

The table entries are re-sorted based on your selection.



3. To sort the entries by a different column, repeat steps **1** and **2** for the column in question.

To sort markers by *Highlight State* or *Checked State,* keeping the `Ctrl` key pressed, click the corresponding column header. Sorting markers based on highlight color is not supported.

Related Topics

Annotation Browser Assistant

© 2023 Cadence Design Systems, Inc.

Annotation Browser Tabs

Annotation Browser Shortcut Menus

Annotation Browser Options Form

Filtering Markers in the Annotation Browser

Filtering Markers in the Annotation Browser

You can filter the markers shown in the Annotation Browser display using either a predefined column filter, or by defining your own filter using the Custom Show Filter form. Additionally, you can use the Incomplete Net filter form to filter incomplete net markers based on the set of currently selected nets in the layout canvas or Navigator assistant.

Setting a Predefined Column Filter for Markers

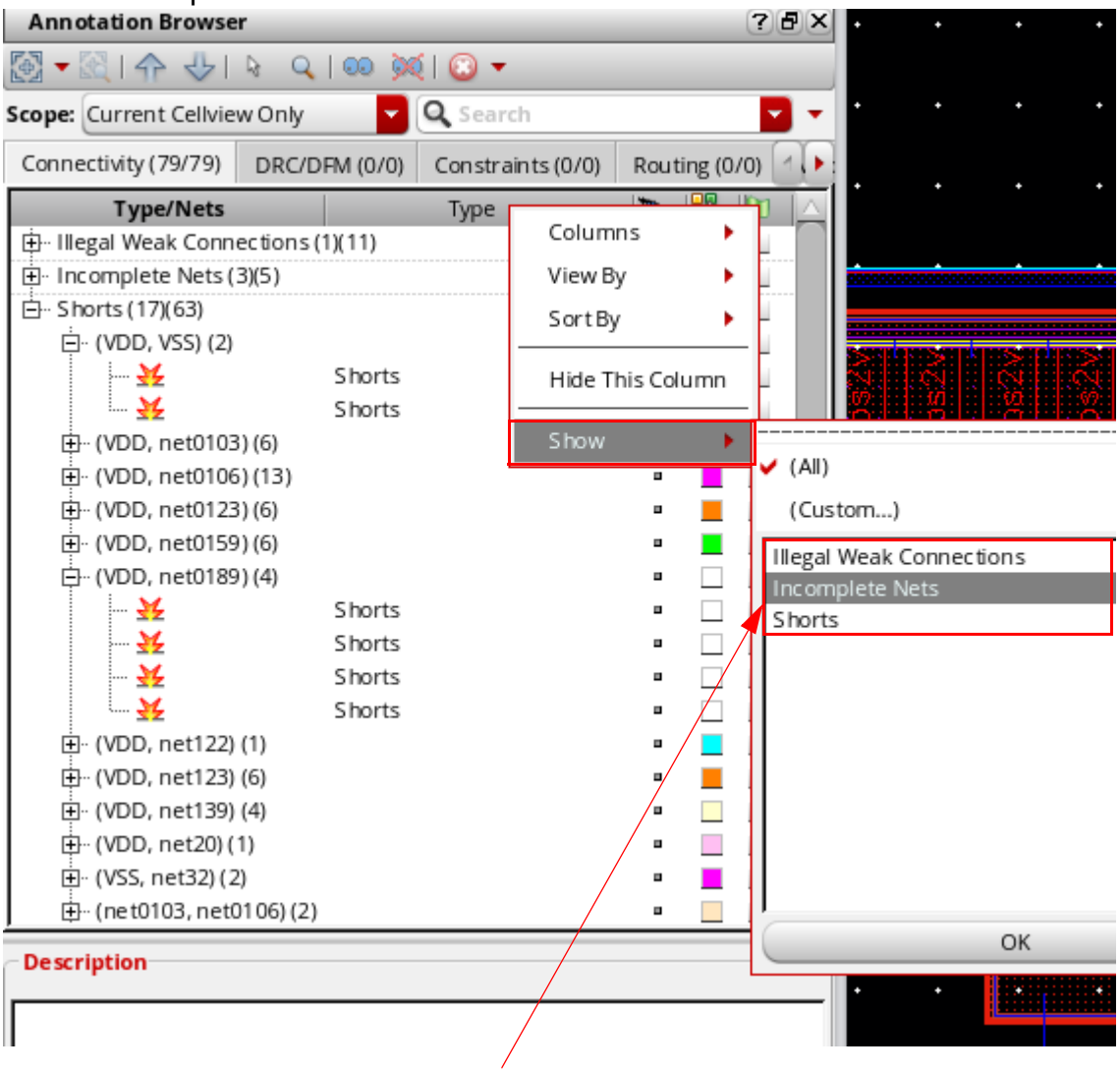
You can set a predefined filter for a column to selectively show the markers in that column. For example, you can set a predefined filter on the *Type* column to only show the markers of a particular type, such as *Incomplete Nets or Shorts*.

To set a predefined filter for a Column:

1. Right-click the header of the column to which you want to apply the filter.

© 2023 Cadence Design Systems, Inc.

2. In the shortcut menu, choose *Show* from the list of simple filters and click an appropriate filter from the predefined filter list.



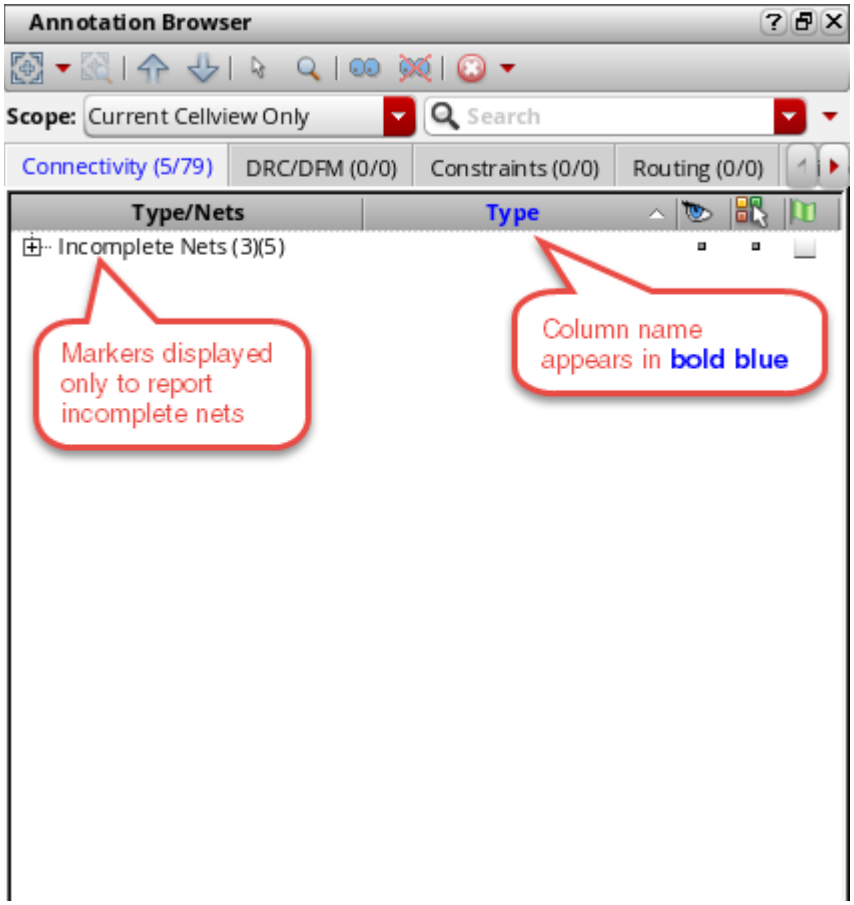
Choose the predefined filter you want to apply. The choices are determined automatically based on the contents of the column.

3. Click *OK* to proceed.

   The browser updates to show only the markers corresponding to incomplete nets.

© 2023 Cadence Design Systems, Inc.

The column header displays in **bold blue** text to indicate that a filter is set on the column.



If required, you can apply a predefined or custom filter on another column. The browser shows only those markers that match all the filter criteria for all the columns.

To remove the applied filter and to display all the markers, choose *Show – All*.

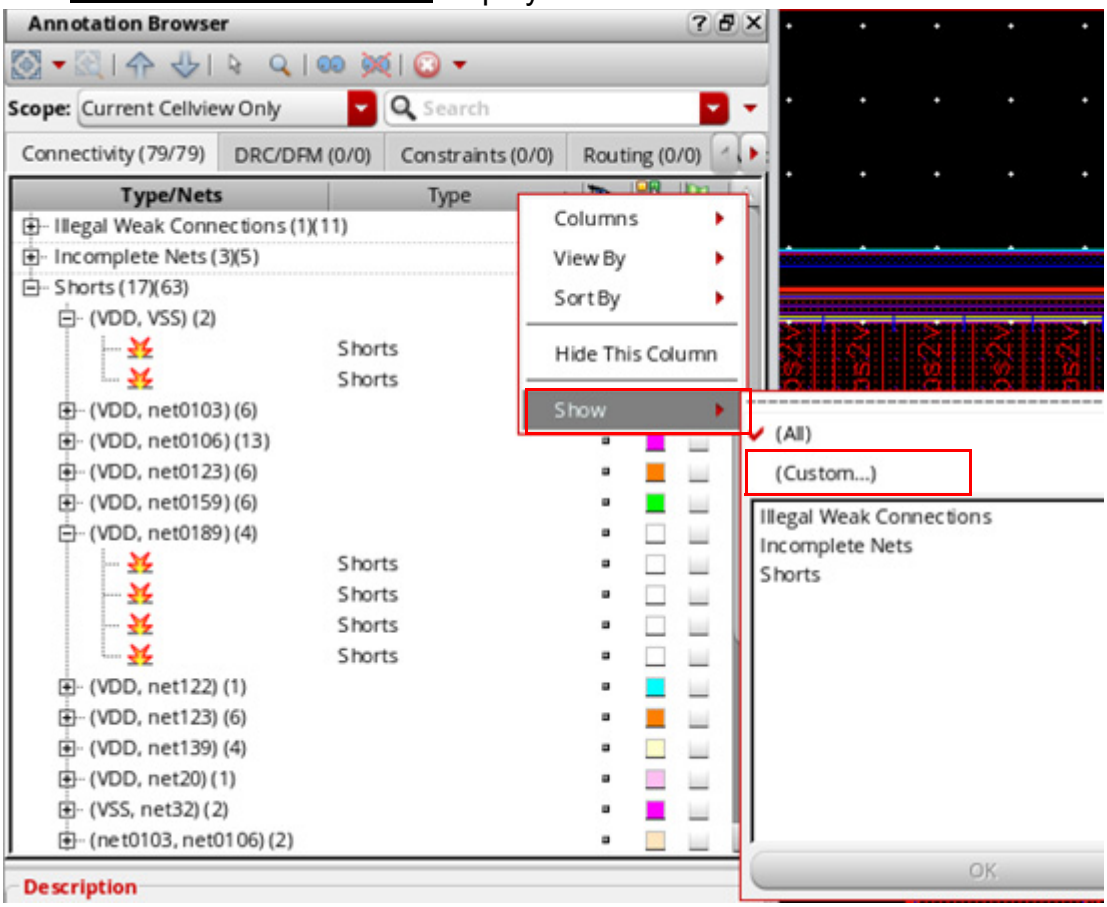
Defining and Applying a Custom Filter

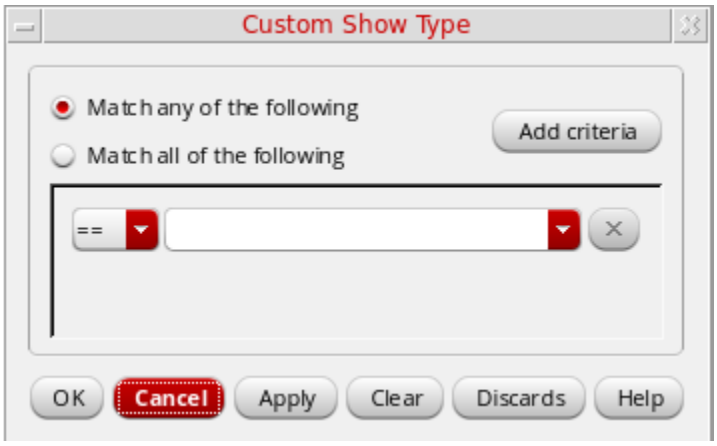
To define and apply a custom filter:

1. Right-click the header of the column to which you want to apply the filter.

© 2023 Cadence Design Systems, Inc.

2. In the shortcut menu, choose *Show – Custom* to define a custom filter for the column. The Custom Show Filter Form displays.



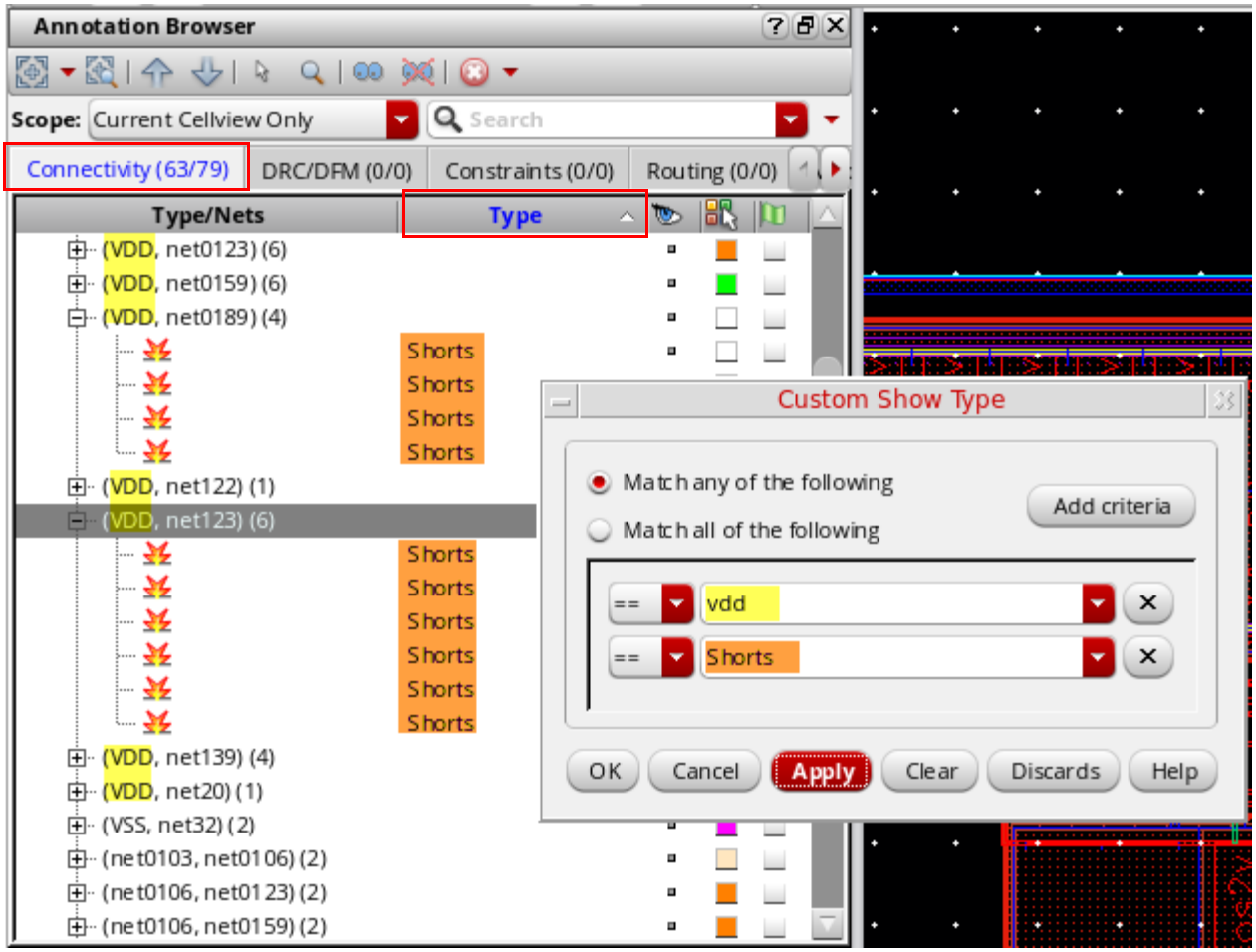


3. Click *Add criteria* to add a criterion for filtering the markers, such as *vdd*.

4. For each criterion, choose the operator from the first pull-down list and select the column content to filter by from the second pull-down list.

© 2023 Cadence Design Systems, Inc.

For example, the custom filter shown below displays all the *shorts* on net *vdd*.



5. Click *OK* to apply the changes and close the Custom Show Filter Form.

   The column header is drawn in **bold blue** text, as shown in the figure above, to indicate that a filter is set on the column.

In addition, the tab in which you applied the custom filter also has its header text turned blue.

If required, you can apply a predefined or custom filter on another column. The browser shows only those markers that match all the filter criteria for all the columns.

To remove the applied filter(s) and to display all the markers, choose *Show – All*.

© 2023 Cadence Design Systems, Inc.

Setting an Incomplete Net Filter

You can set an incomplete net filter for the selected nets in a window. This ensures that the Annotation Browser assistant displays (or hides) the incomplete net markers only for the specified list of nets in a given window.

To set an incomplete net filter:

- Use the abSetIncompleteNetFilter SKILL function.

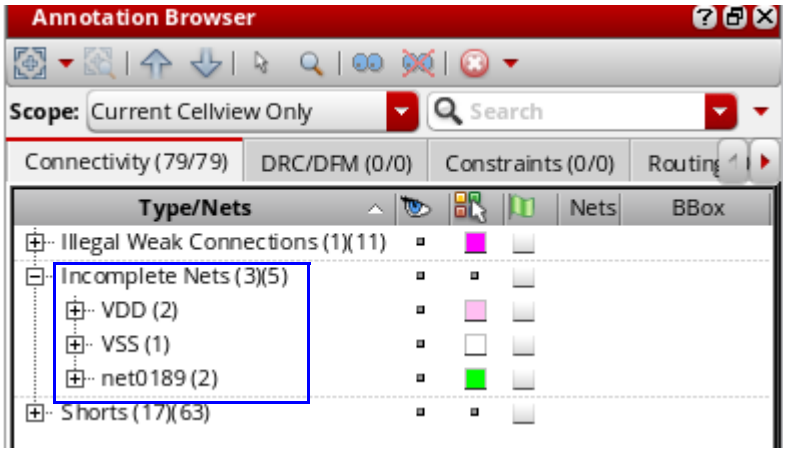
  The SKILL function enables you to specify a list of net names in a window for which a specific filter state—Show or Hide—needs to be set. Relatedly, you can use the abGetCurrentIncompleteNetFilter SKILL function to determine the current incomplete net filter state applied.

- Use the *Incomplete Net Filter* command in the shortcut menu of the browser pane.

Alternatively, you can update the incompleteNetFilter environment variable to set the filter, if the environment variable is set otherwise.

Let us see how an incomplete net filter state can be set using the *Incomplete Net Filter* command.

Consider the image below, which shows the Annotation Browser with incomplete net markers associated with 3 nets: `VDD`, `VSS`, and `net0189`.



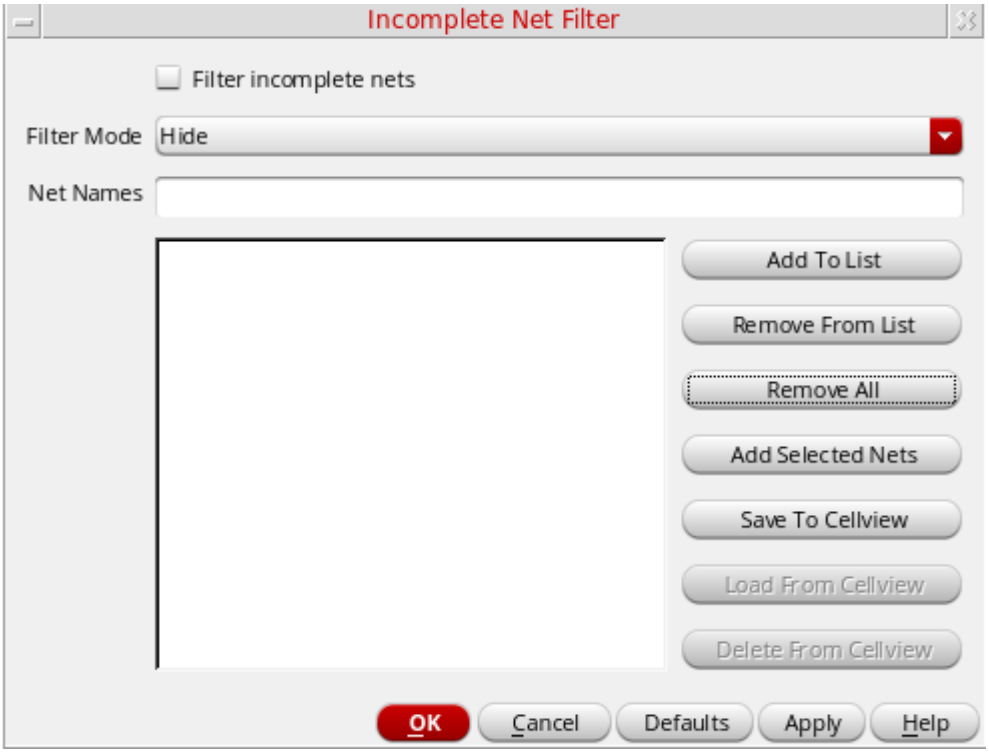
Now, select nets `VDD` and `net0189` from the layout Navigator assistant. To filter the incomplete net markers shown in the Annotation Browser based on the selected net in the layout window:

1. In the Annotation Browser, right-click any of the markers associated with the selected nets and choose *Incomplete Net Filter*.

© 2023 Cadence Design Systems, Inc.

The markers associated with the nets selected in the Navigator assistant (or the layout canvas) are cross-selected in the Annotation Browser.

The Incomplete Net Filter form displays.

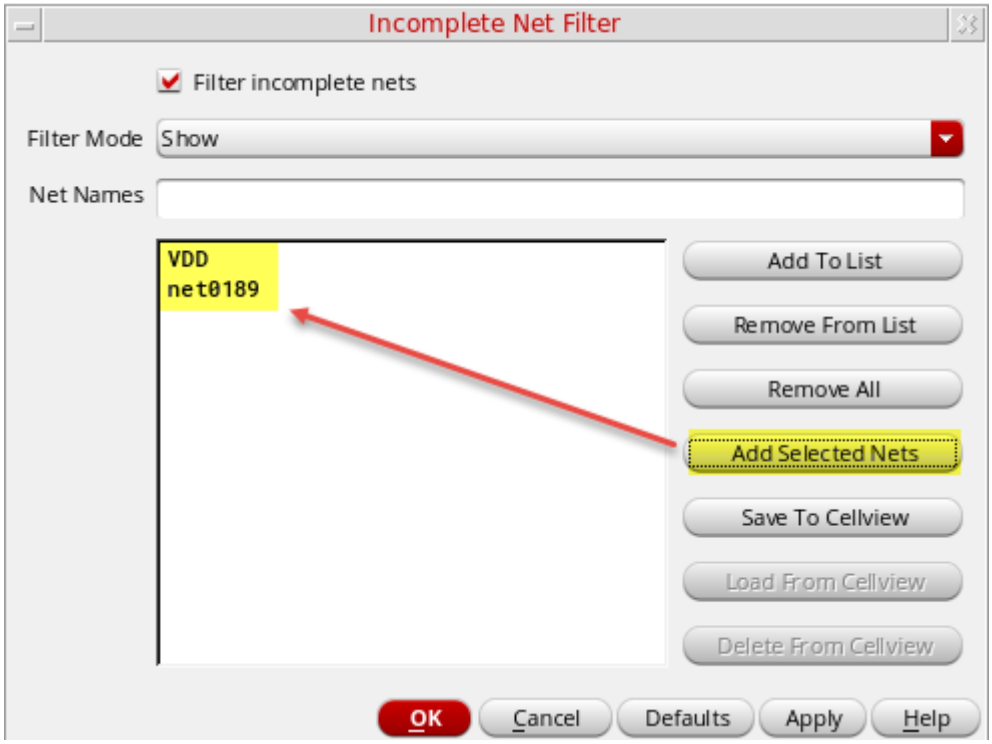


2. Select the *Filter incomplete nets* check box to specify that filtering needs to be driven by the values of the incompleteNetFilterDefaultList and incompleteNetFilterMode environment variables.
3. Choose an appropriate *Filter Mode* to either *Show* or *Hide* the markers for the specified nets.
4. Type the *Net Names* to apply the filtering and click *Add To List*.

   If you already have the nets on which to apply filtering selected in the Navigator assistant or the layout canvas, instead of manually typing the net names in the Incomplete Net Filter form, you can click the *Add Selected Nets* button.

© 2023 Cadence Design Systems, Inc.

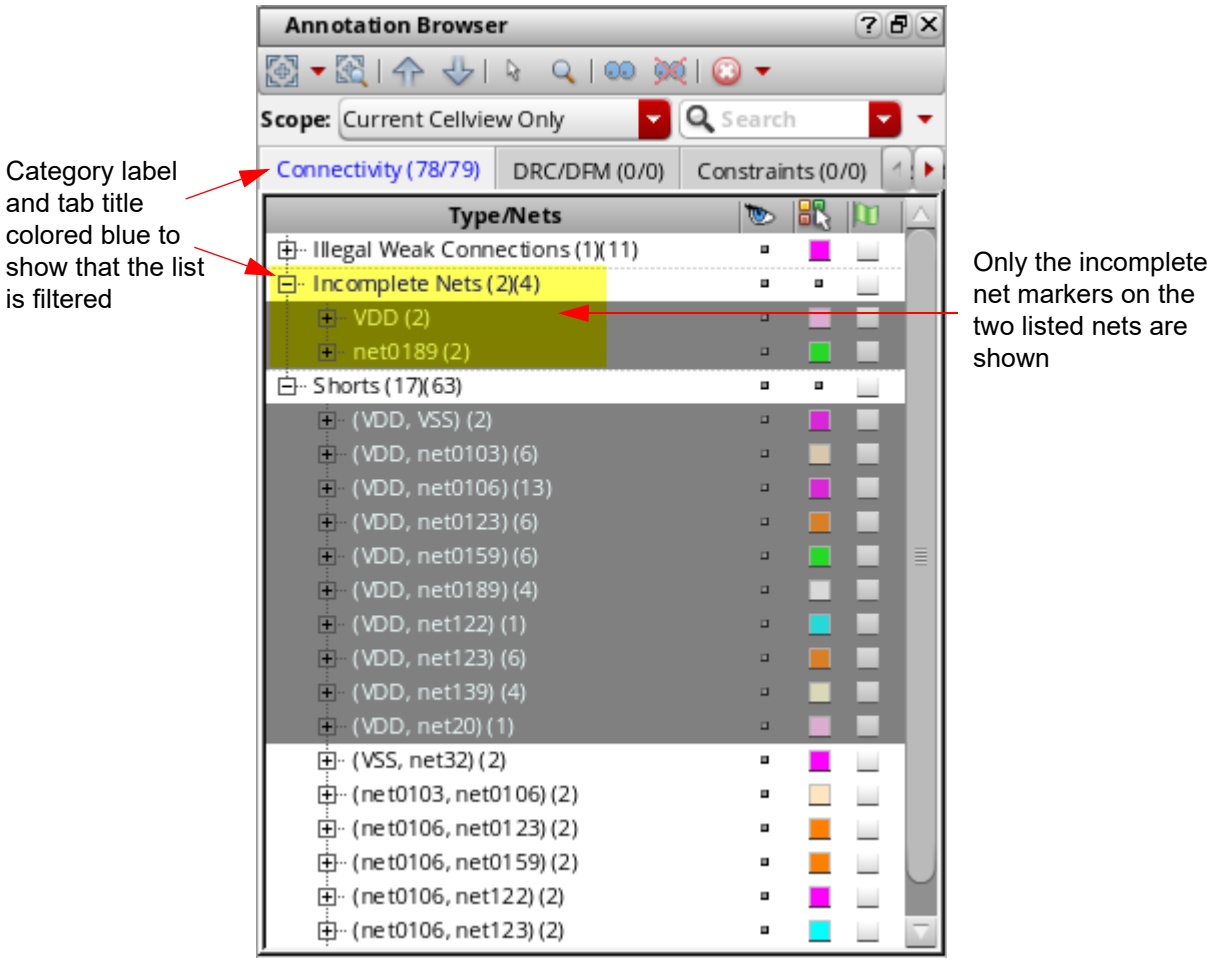
The net names are added to the form as a list, as shown in the figure below.



5. Click *OK*.

© 2023 Cadence Design Systems, Inc.

The browser updates to show only the incomplete net markers associated with the listed nets. All other incomplete net markers are hidden.

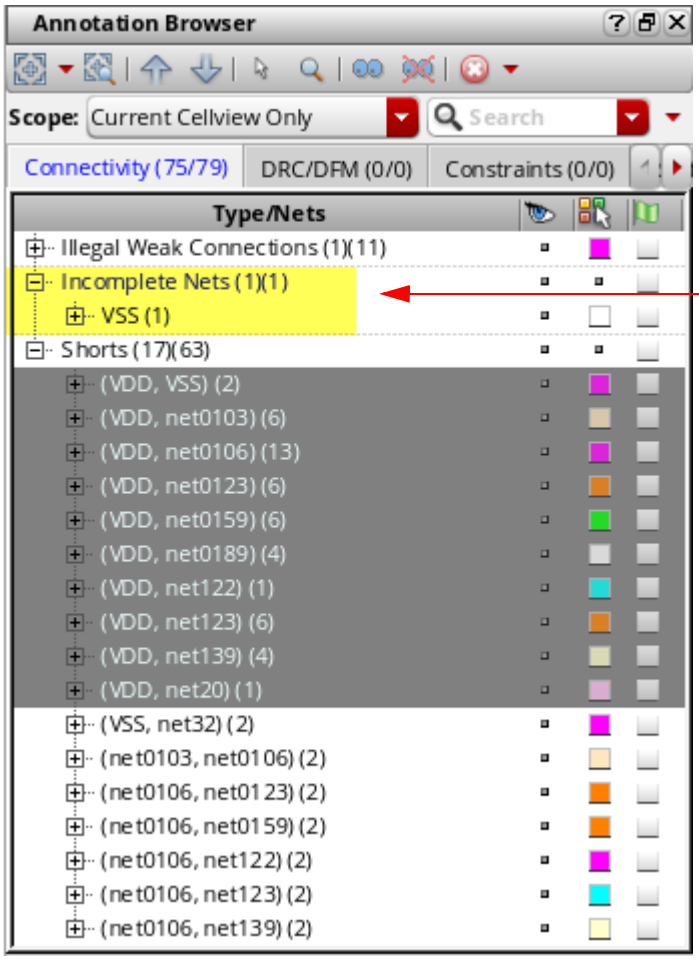


The label for the *Incomplete Nets* category and the tab label are colored blue to indicate that a filter has been applied. The marker count in the tab label also indicates that some markers are hidden.

6. Bring up the Incomplete Net Filter form again and this time choose *Hide nets from list* from the pull-down list at the top.
7. Click *OK*.

© 2023 Cadence Design Systems, Inc.

The browser updates to hide the incomplete net markers associated with the listed nets. All the other incomplete net markers are shown.



Category label colored blue to show that the list is filtered. Incomplete net markers on the two listed nets are hidden.

The label for the *Incomplete Nets* category and the tab label are colored blue to indicate that a filter has been applied. The marker count in the tab label also indicates that some markers are hidden.

Related Topics

Custom Show Filter Form

Incomplete Net Filter Form

View By Form

Annotation Browser Options Form

© 2023 Cadence Design Systems, Inc.

Setting a Predefined Column Filter for Markers

Defining and Applying a Custom Filter

Setting an Incomplete Net Filter

Annotation Browser Assistant

Annotation Browser Tabs

Annotation Browser Shortcut Menus

Grouping Markers in the Annotation Browser

Sorting Markers in the Annotation Browser

Filtering Markers in the Annotation Browser

Assigning Colors to Markers

Zooming In on Markers

Assigning Colors to Markers

To assign the same color to all the markers displayed in the browser:

1. In the Annotation Browser assistant, click the header of the *Set Highlight Color* column and choose a color from the color menu.To assign colors to all the markers under a particular node in the browser:
2. In the Annotation Browser assistant, click the *Set Highlight Color* column for the parent node of the markers you want to change.
3. Do one of the following.
   - To show all the markers contained in the node in the same color, choose the color you want from the drop-down list.
   - To show each of the markers contained in the node in a different color, choose *cycle*. The markers are assigned colors automatically based on the predefined list.

The markers change according to the colors you specified.

© 2023 Cadence Design Systems, Inc.

To assign a color to multiple markers in the layout canvas:

1. In the Annotation Browser assistant, keeping the `Ctrl` key pressed, click the *Set Highlight Color* column next to the entry for the marker, and choose a color from the color menu.

   The markers in the canvas change to the color you specified.

To assign a different color to a marker in the layout canvas:

1. In the Annotation Browser assistant, click the *Set Highlight Color* column next to the entry for the marker whose color you want to change.
2. Do one of the following.
   - Choose the color you want from the predefined list.
   - Choose *cycle* to automatically select the next color in the predefined list.

   The marker in the canvas changes to the color you specified.

Related Topics

Annotation Browser Assistant

Annotation Browser Tabs

Annotation Browser Shortcut Menus

Annotation Browser Options Form

Grouping Markers in the Annotation Browser

Sorting Markers in the Annotation Browser

Filtering Markers in the Annotation Browser

Zooming In on Markers

Zooming In on Markers

To zoom in on specific markers in the layout canvas:

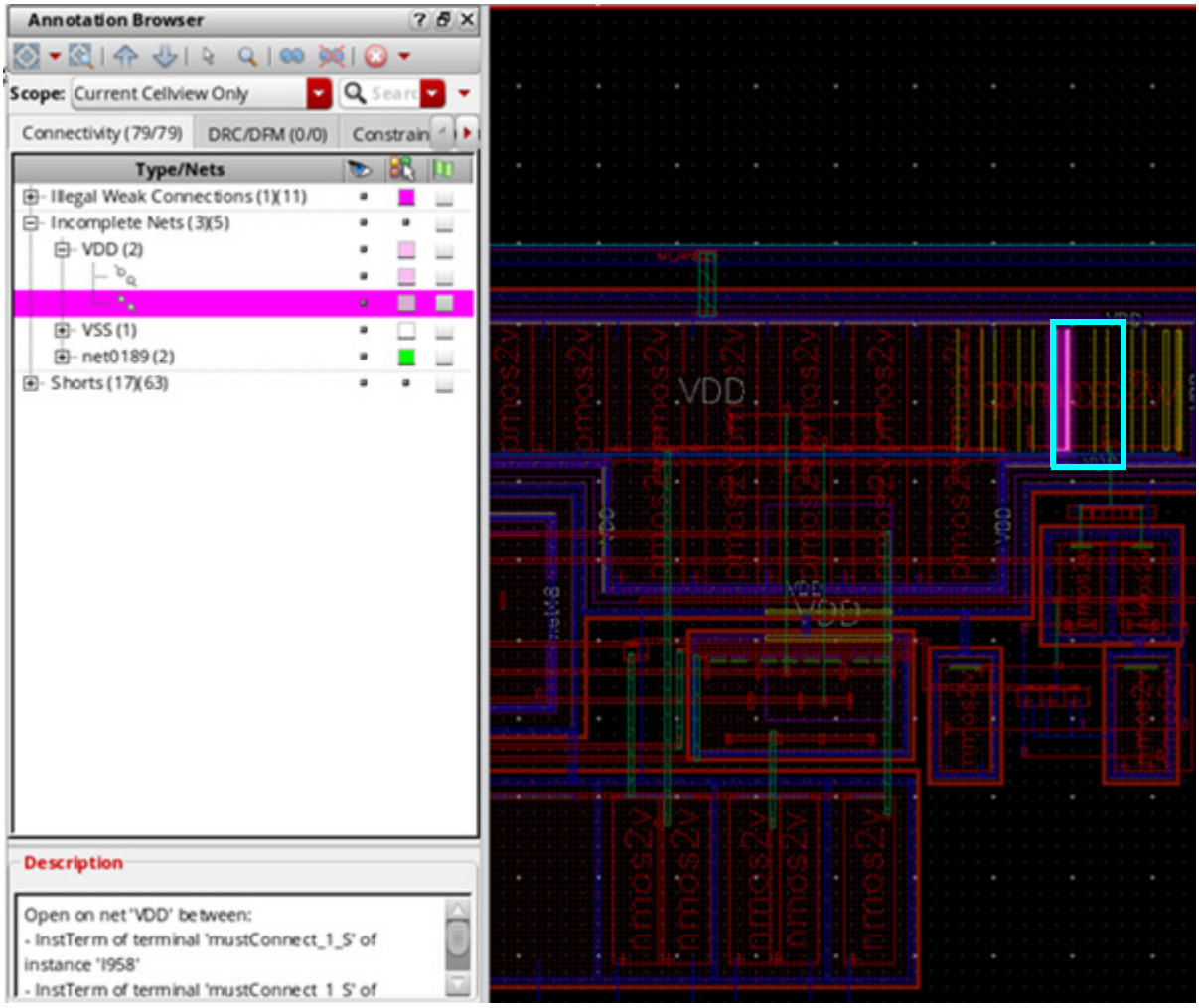
1. In the Annotation Browser assistant, select the markers you are interested in.

© 2023 Cadence Design Systems, Inc.

2. Click the *Zoom To Selected* button in the Annotation Browser toolbar.



The canvas display zooms to the flight lines for the selected nets, as shown in the figure below.



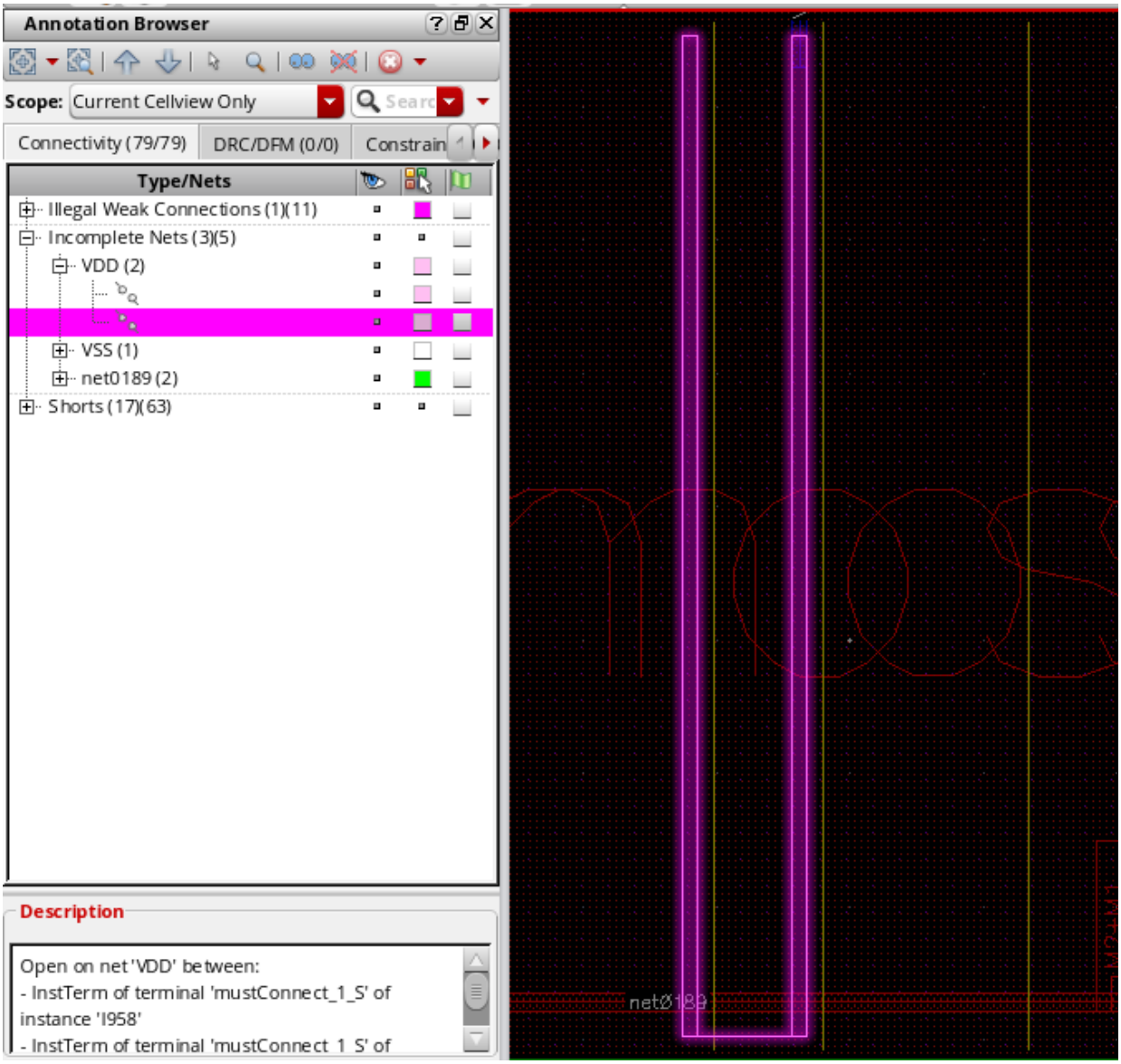
The level of detail you see depends on how many nets are selected and their relative locations in the design.

To zoom in on a specific marker in the layout canvas:

➡ Double-click the marker in the Annotation Browser assistant.

© 2023 Cadence Design Systems, Inc.

The display automatically zooms and pans to the marker representing the currently selected entry in the Annotation Browser, as shown in the figure below.



When the selection changes, the display updates accordingly.

To automatically zoom in on the canvas representation of the currently selected marker:

➤ Click the *Auto Zoom* button in the Annotation Browser assistant toolbar.



© 2023 Cadence Design Systems, Inc.

The display automatically zooms and pans to the marker representing the currently selected entry in the Annotation Browser. When the selection changes, the display updates accordingly.

Related Topics

Annotation Browser Assistant

Annotation Browser Options Form

Annotation Browser Tabs

Annotation Browser Shortcut Menus

Grouping Markers in the Annotation Browser

Sorting Markers in the Annotation Browser

Filtering Markers in the Annotation Browser

Fixing Markers Interactively in Annotation Browser

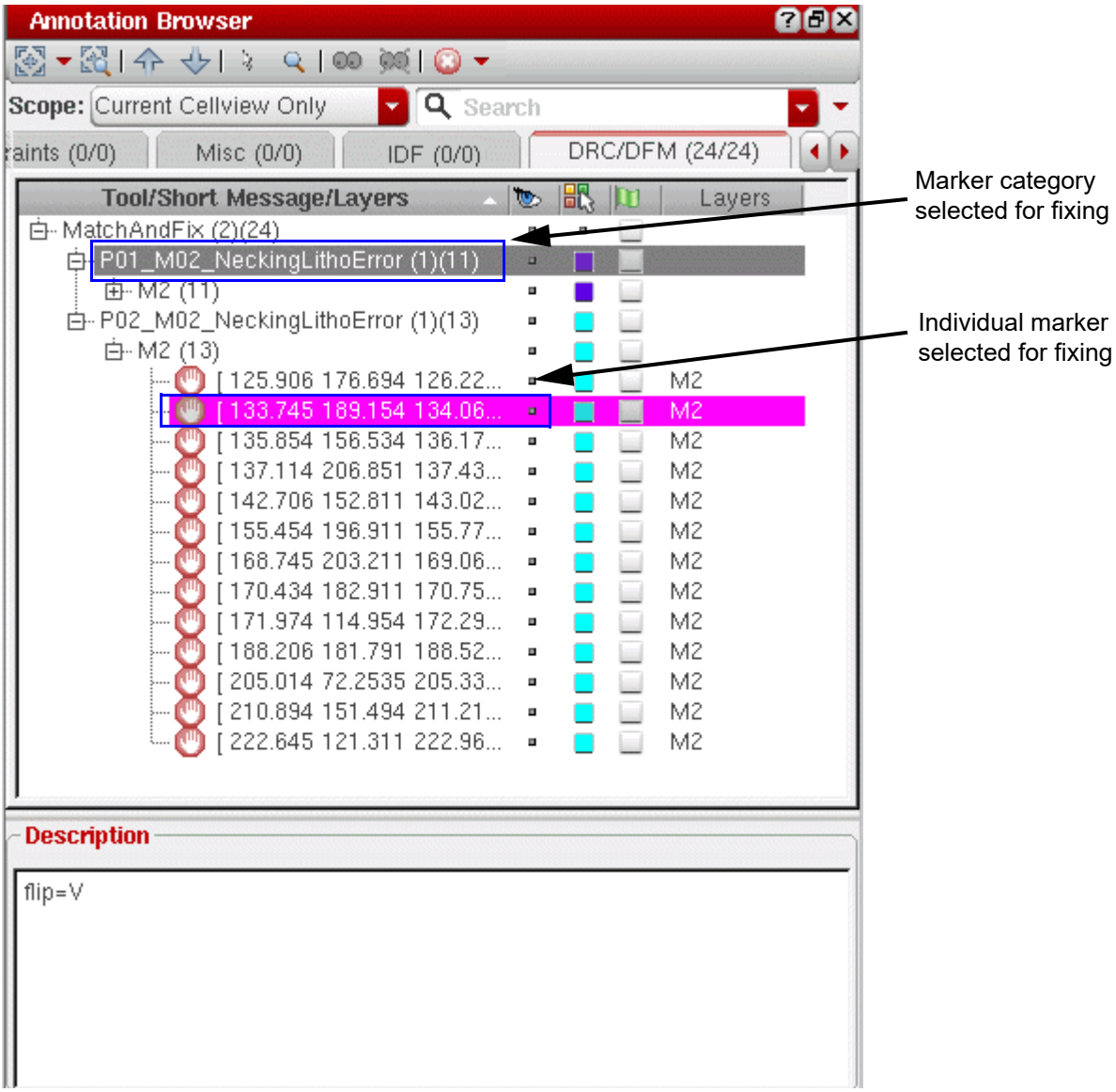
You can interactively fix markers that are generated in the DRC/DFM tab and CAS tab of Annotation Browser, by using the appropriate shortcut commands.

Let us see how the markers displayed in the DRC/DFM tab can be fixed interactively.

1. In the Annotation Browser assistant, select a marker category or an individual marker that you want to fix.

© 2023 Cadence Design Systems, Inc.

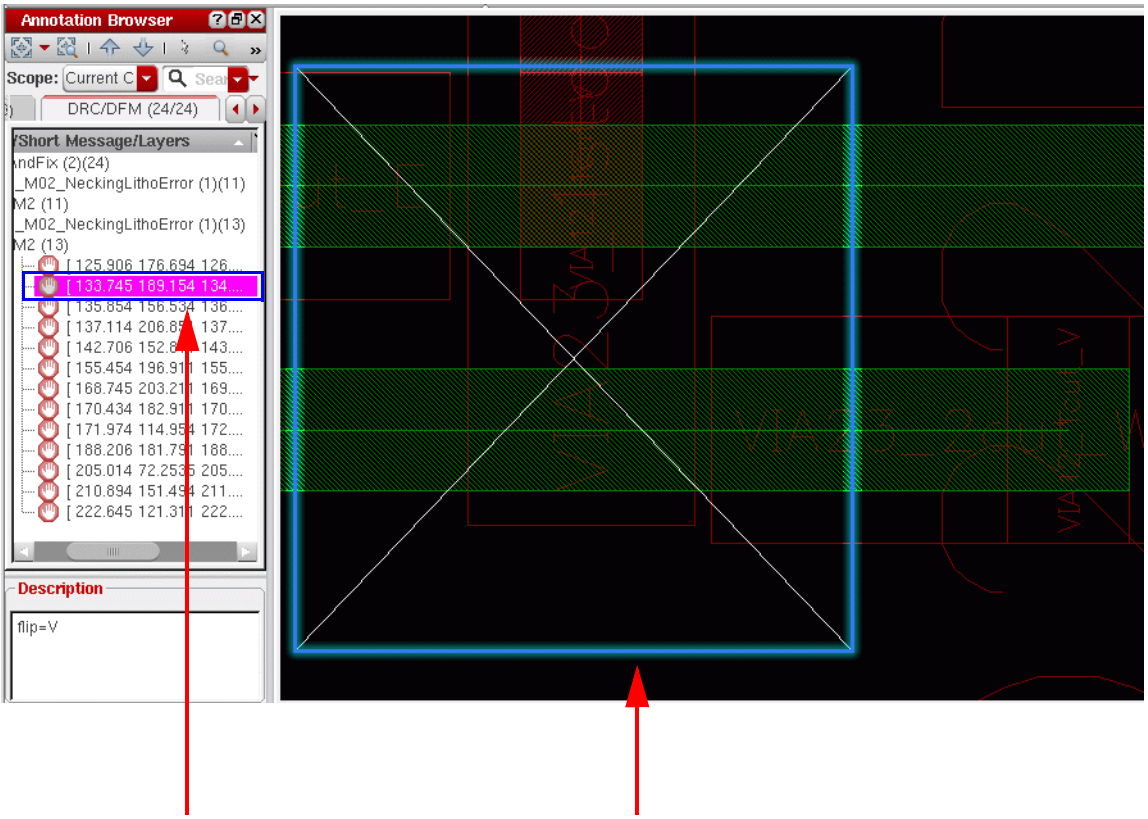
To select multiple markers or marker categories to be fixed, hold down the Ctrl key while selecting the markers.



1. To zoom into a selected marker to view the violation in the layout, select the marker and click the *Zoom To Selected* button in the Annotation Browser toolbar.

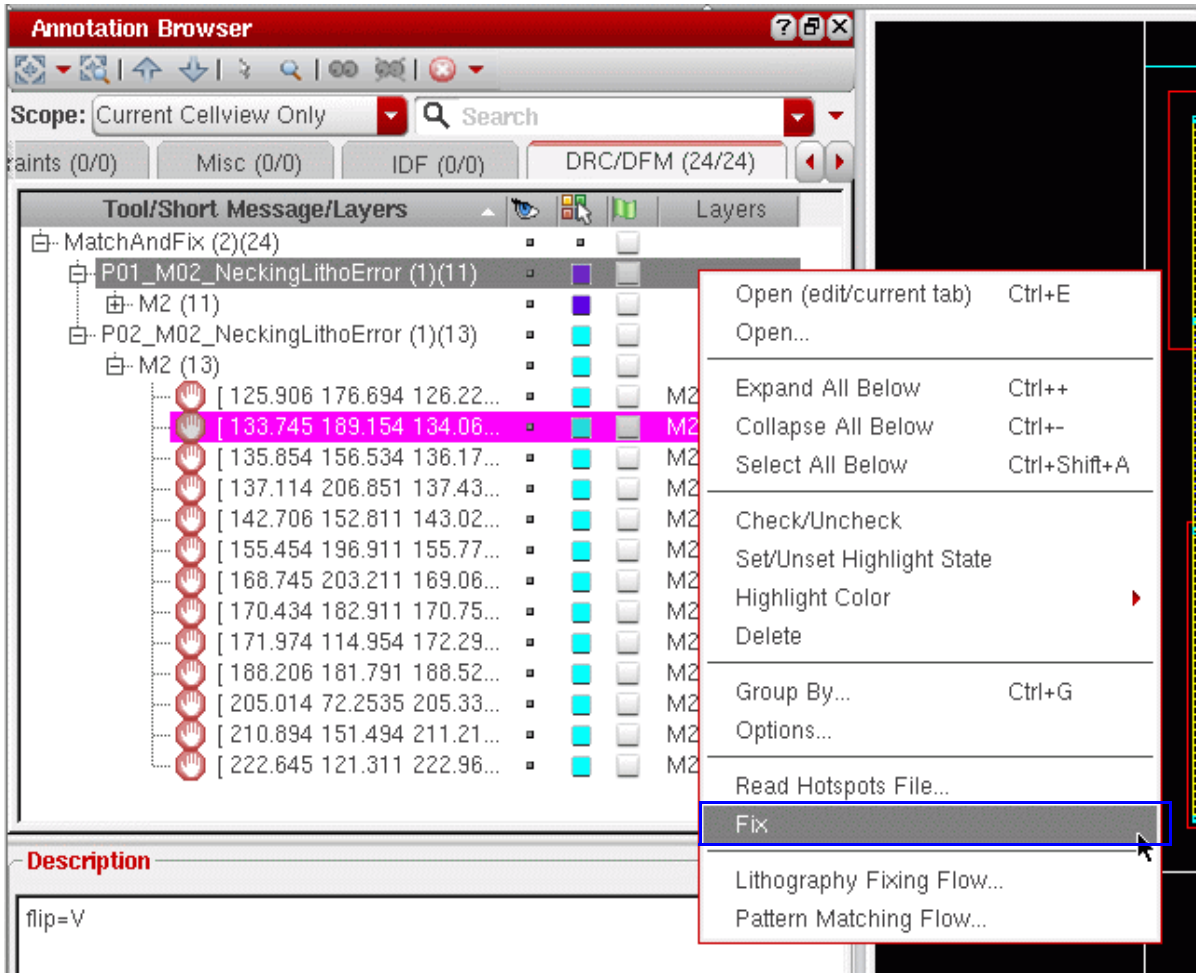
© 2023 Cadence Design Systems, Inc.

The layout canvas displays a representation of the selected marker as shown in the figure below.



© 2023 Cadence Design Systems, Inc.

2. To fix a marker, right-click the marker in the Annotation Browser and select the *Fix* command.

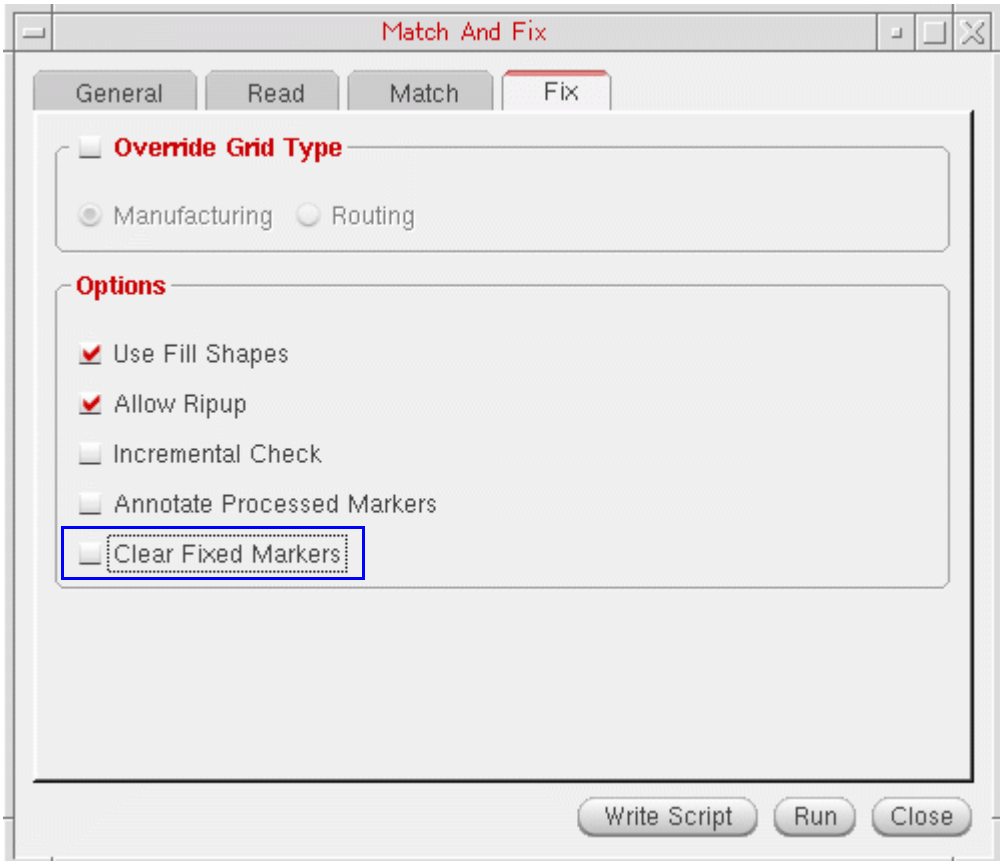


Depending on the lithography flow that generates the markers in the DRC/DFM tab, the fixing options in the Annotation Browser are made available from the corresponding fixing form. For example, if the DRC/DFM tab displays markers generated during a pattern matching run, the corresponding fixing options that are made available in the Annotation Browser are based on the Fixing options available in the Match And Fix form.

By default, the selected markers are automatically fixed and removed from the Annotation Browser. However, you can choose to retain the fixed markers in the Annotation Browser by deselecting the Clear Fixed Markers check box on the Fix tab of the corresponding form.

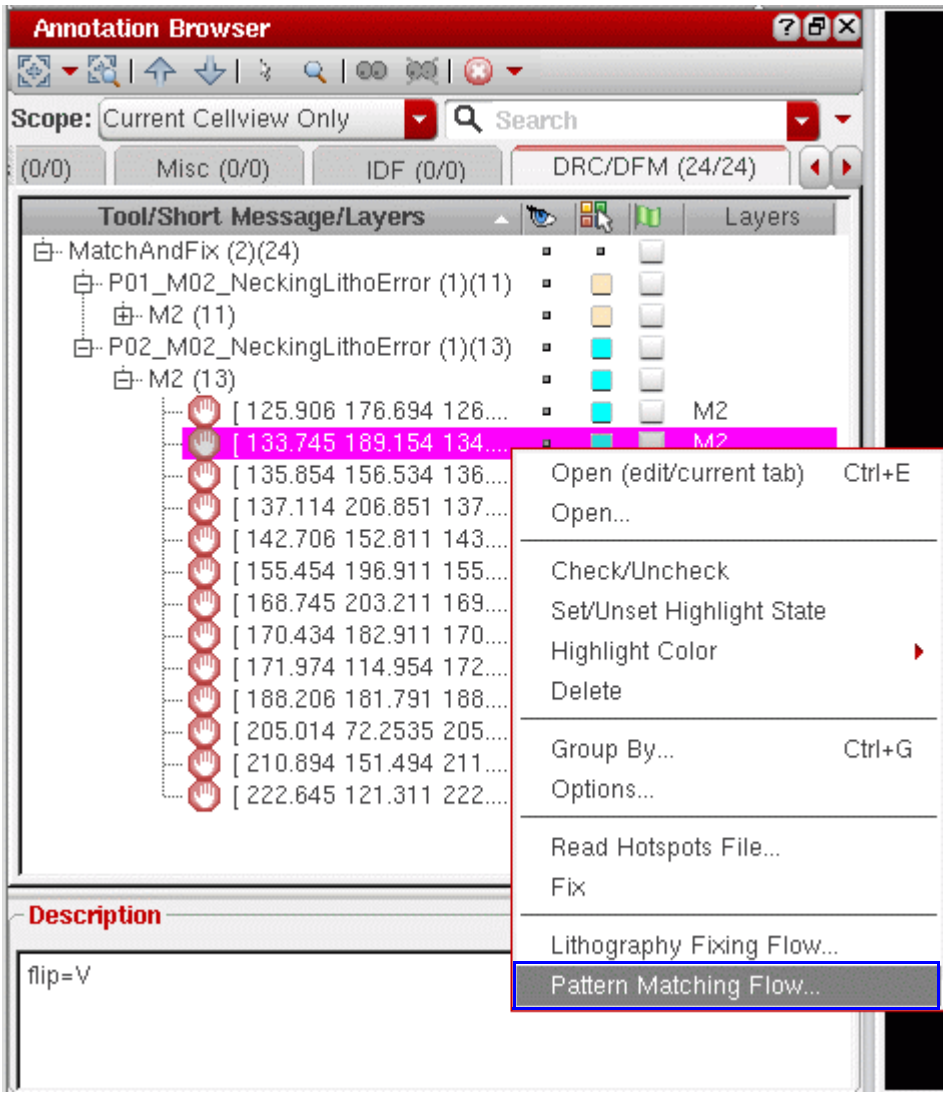
© 2023 Cadence Design Systems, Inc.

For example, to retain the fixed markers that have been generated during a Match and Fix run, you must deselect the Clear Fixed Markers check box on the Fix tab of the Match and Fix form.



© 2023 Cadence Design Systems, Inc.

To access the Match And Fix form from the Annotation Browser, right-click a marker and choose *Pattern Matching Flow*, as shown in the figure below.



To access the Litho Fixing form from the Annotation Browser, right-click a marker and choose *Lithography Fixing Flow*.

Related Topics

Match And Fix Form

Annotation Browser Assistant

Annotation Browser Tabs

© 2023 Cadence Design Systems, Inc.

Annotation Browser Shortcut Menus

Annotation Browser Options Form

Grouping Markers in the Annotation Browser

Sorting Markers in the Annotation Browser

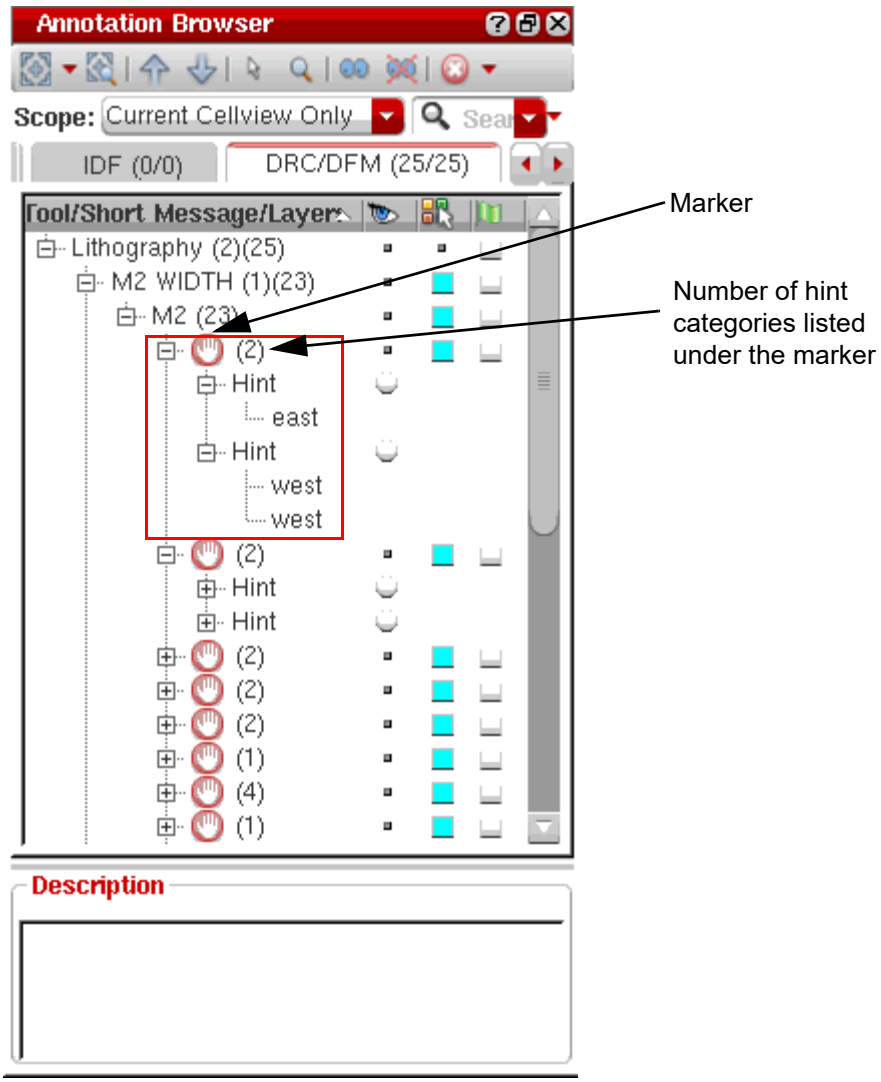
Filtering Markers in the Annotation Browser

Hint Support for DRC/DFM Tab of Annotation Browser

In addition to displaying the markers, the *DRC/DFM* tab displays a hint for some markers. The hint gives information about fixing the error reported by the marker. Depending on the error that a marker flags, several hints may be associated with the marker.

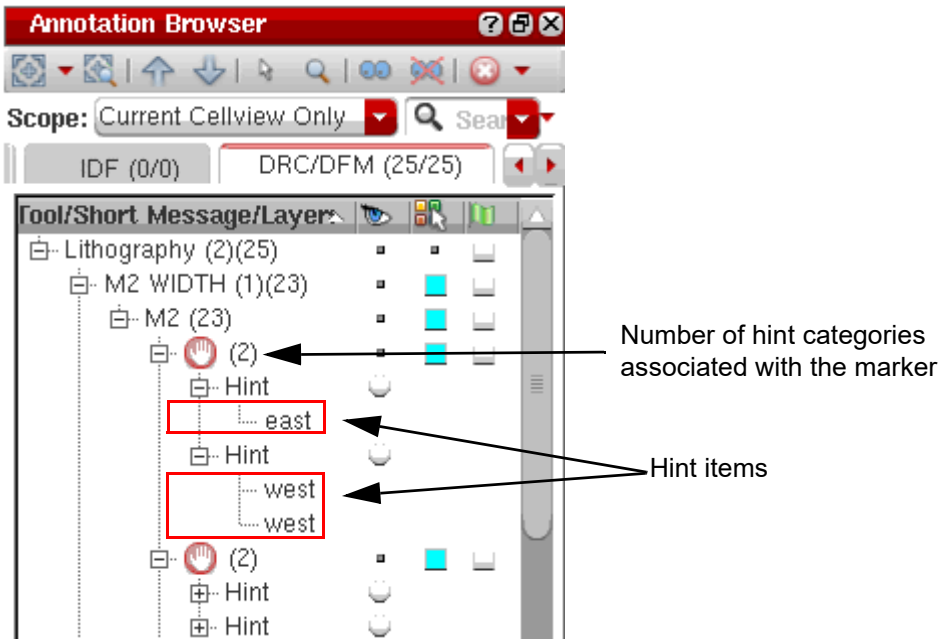
For example, if a design has an error that can be fixed in two ways, the Annotation Browser not only displays a marker to flag the error but also two hint categories, as shown in the figure below.

© 2023 Cadence Design Systems, Inc.



Each hint category is composed of one or several hint items, as displayed in the figure below, that provide information about the shapes that need to be moved to fix the error.

© 2023 Cadence Design Systems, Inc.

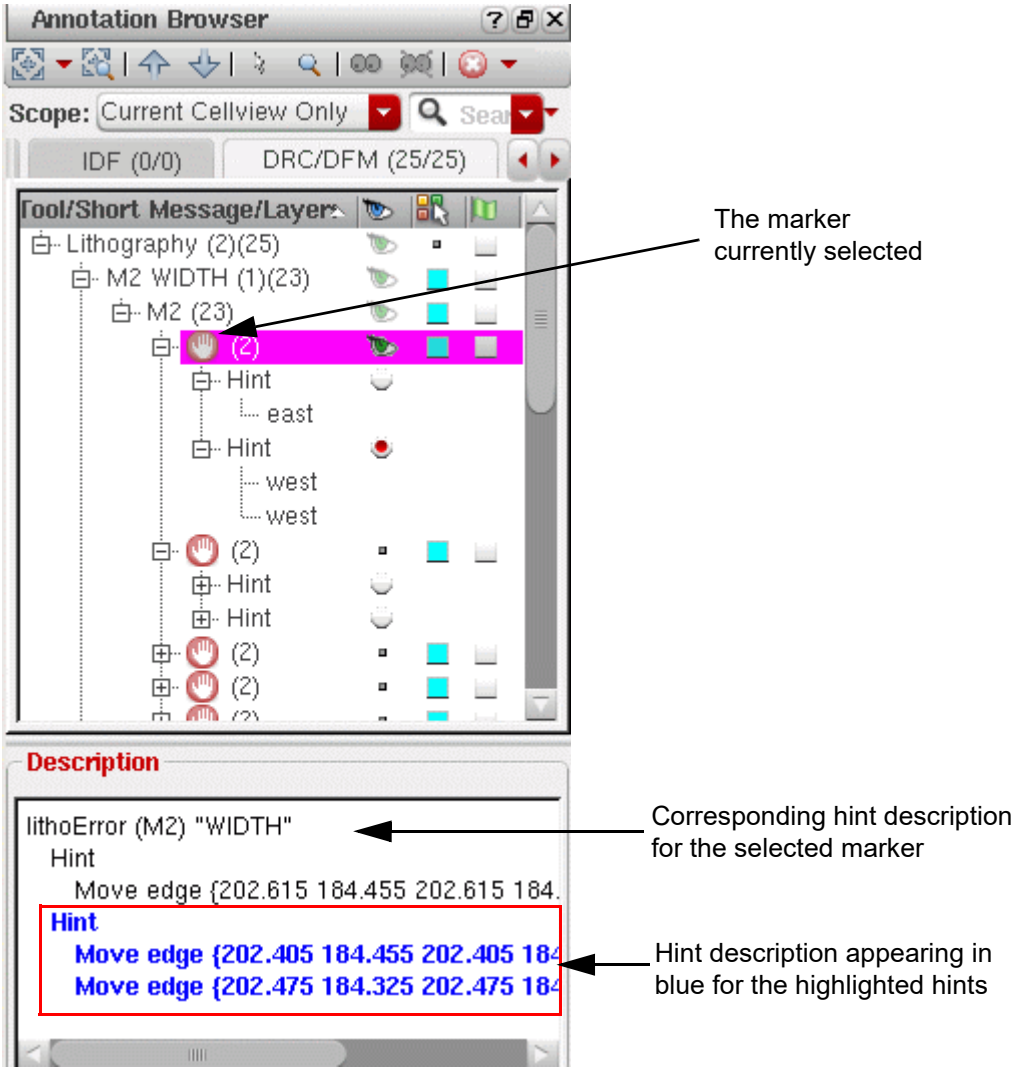


For the previous example, as displayed in the figure above, the first error flagged by the marker can be fixed by following either of the hint items:

- **The first hint item** requires only one shape to be moved to the east
- **The second hint item** requires two shapes to be moved to the west

To view the hint text, you can select either the hint item, the hint category, or the marker, as appropriate. The hint details are displayed in the *Description* box, as shown in the figure below. However, if you select multiple hint items or hint categories belonging to different markers, no information is displayed in the *Description* box.

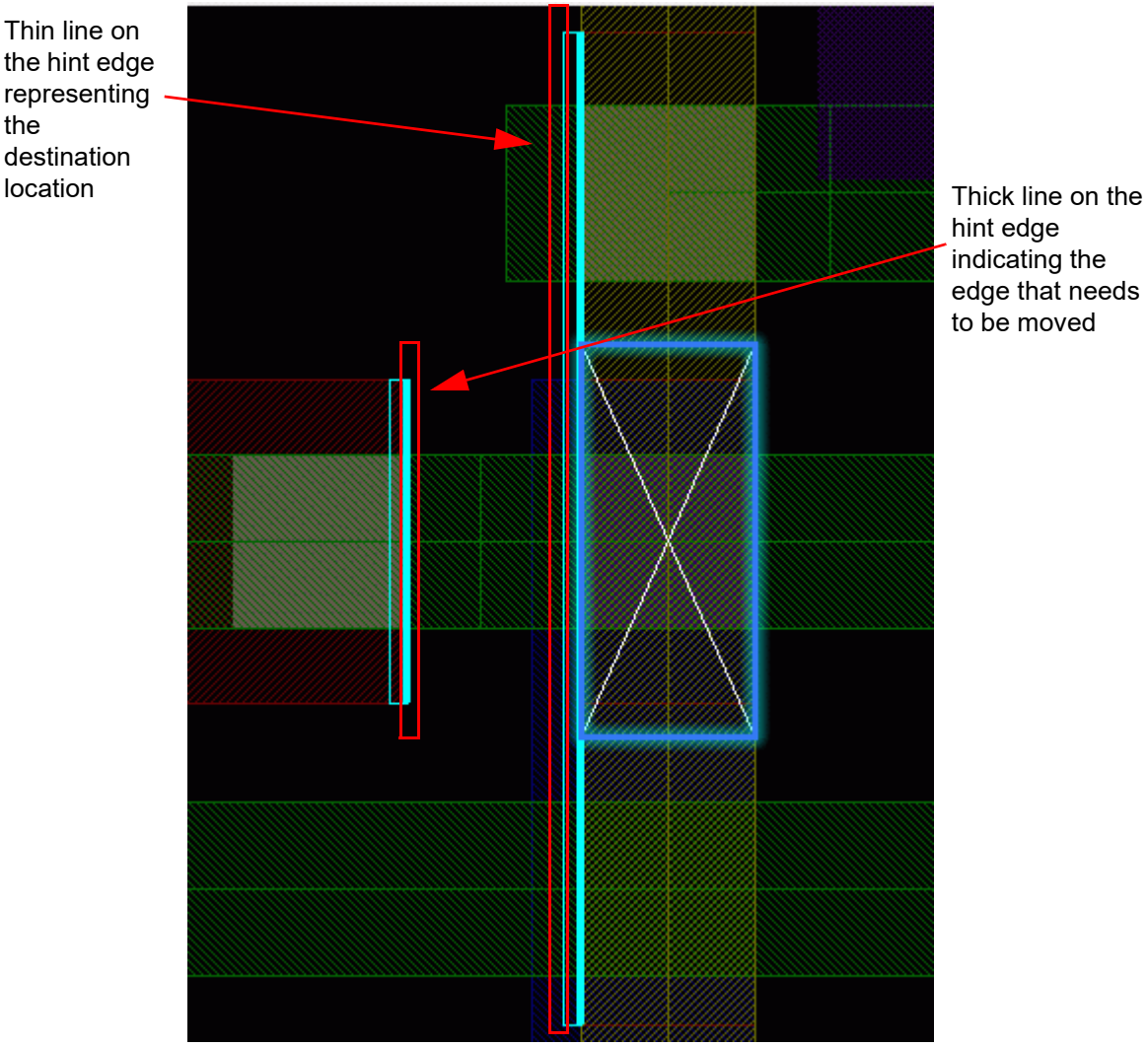
© 2023 Cadence Design Systems, Inc.



If you choose to highlight the hints for the selected marker, the description for these highlighted hints appears in blue, as shown above.

For a highlighted hint item, a thick line on the hint edge, as shown below, represents the edge that needs to be moved to fix the error. The thin line, on the other hand, represents the destination location for the edge.

© 2023 Cadence Design Systems, Inc.



Related Topics

Annotation Browser Tabs

CAS TAB Marker Categories

Annotation Browser Options Form.

Annotation Browser Assistant

© 2023 Cadence Design Systems, Inc.

© 2023 Cadence Design Systems, Inc.

8

Design Data Checks

This topic summarizes how to use the Virtuoso® Layout Suite XL (Layout XL) to check your design as you work.

- Selecting and Deselecting Nets in Navigator - Select a net in the Navigator assistant and all the shapes associated with that net in the design.
- Design Probing - Select an instance, net, or pin in the layout or schematic window to highlight the corresponding element in the other window.
- Tracing Nets - Trace the physical and logical connectivity of nets in a design.
- Checking Shorts, Incomplete Nets, and Illegal Overlaps - Check the number of incomplete nets, shorts, and illegal overlaps in your design using Annotation Browser.
- Design Check Against Source - Check mismatches between the schematic and layout.
- Design Check for Layout XL Compliance - Check if your design fulfills the compatibility criteria that allow it to fully leverage the connectivity-driven features of Virtuoso Layout Suite XL
- Design Check for Manufacturability - Check manufacturability of a design for optimum device yield and performance.
- Batch Checker - Use the Batch Checker tool to check for routability, process rules, connectivity, specialty routing, placement, fabrication, and WSP | SP Active.

© 2023 Cadence Design Systems, Inc.

Selecting and Deselecting Nets in Navigator

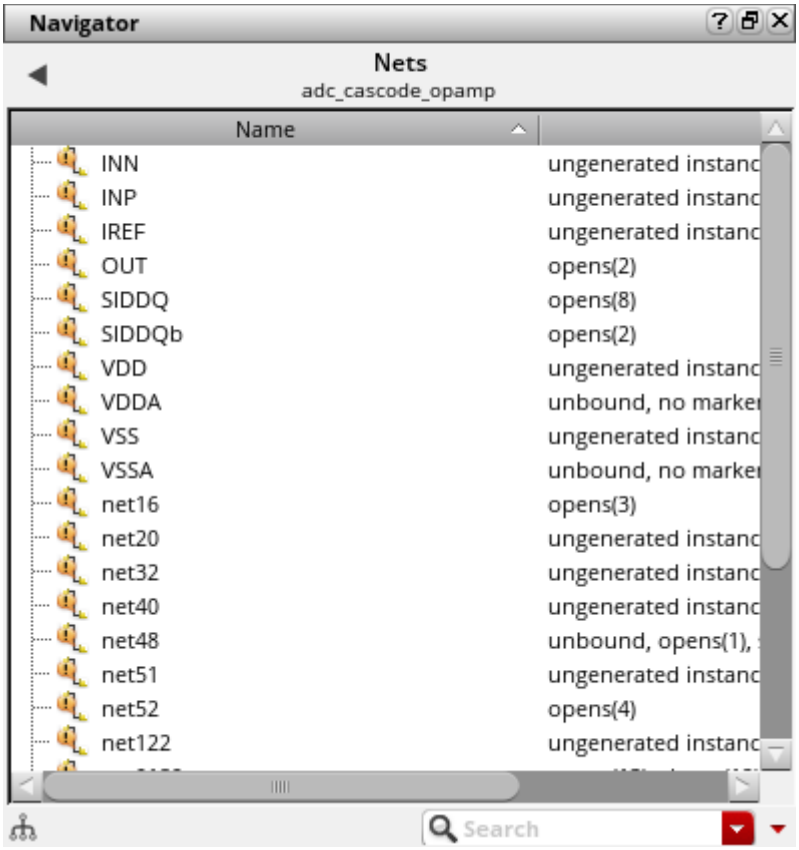
When you select a net in the Navigator assistant, Layout XL selects the net (or its flight line if it is not yet routed) and all the shapes associated with that net in the design. Additionally, any lower-level nets associated with the selected net are highlighted on the canvas. Likewise, when a top-level net is deselected, highlights on its associated lower-level nets are removed. For a better view of the highlighted nets, *Display Depth* should be set to a value higher than `0` and *Hierarchy Depth* should be set to display a higher number of hierarchy levels.

You need to select nets to perform pin optimization tasks, edit net criticality, and so on.

To select a net:

1. From the layout window menu bar, choose *Windows – Assistant – Navigator*.

   The Navigator assistant is displayed.



2. Use the *Show* drop down in the toolbar to display only the *Nets* in the design.
3. Locate the net you want to work on in the list and click it to select it.

© 2023 Cadence Design Systems, Inc.

The net you selected (or its flight line if it is not yet routed) is highlighted in the design window, along with all the shapes on that net.

The option to probe flight lines is off by default. You set it on the Highlight Options form available using the *Options – Highlight* menu pick.

To deselect all nets:

➤ Click in an empty area of the design window.

Related Topics

Highlight Options

The Navigator Assistant

© 2023 Cadence Design Systems, Inc.

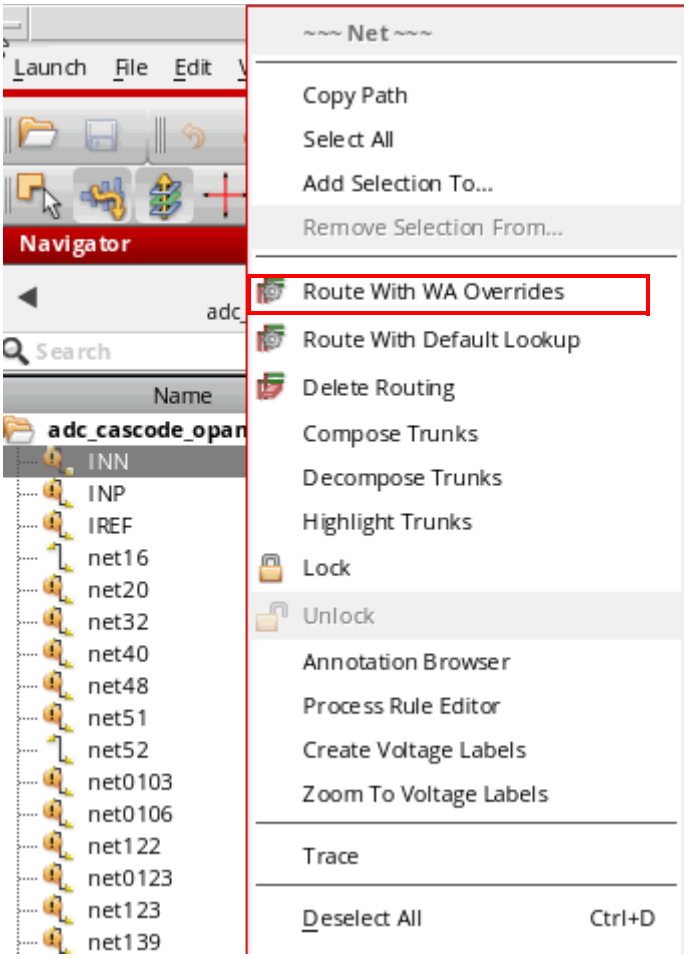
Selected Net Routing

To route a selected net, you can choose from the following routing commands, as appropriate:

- *Route With WA Overrides*

  Use this command to route the selected nets with the Wire Assistant Overrides taking precedence over the default constraint settings as defined by the Wire Editor Default Constraint Group, or any Net Specific Constraints.

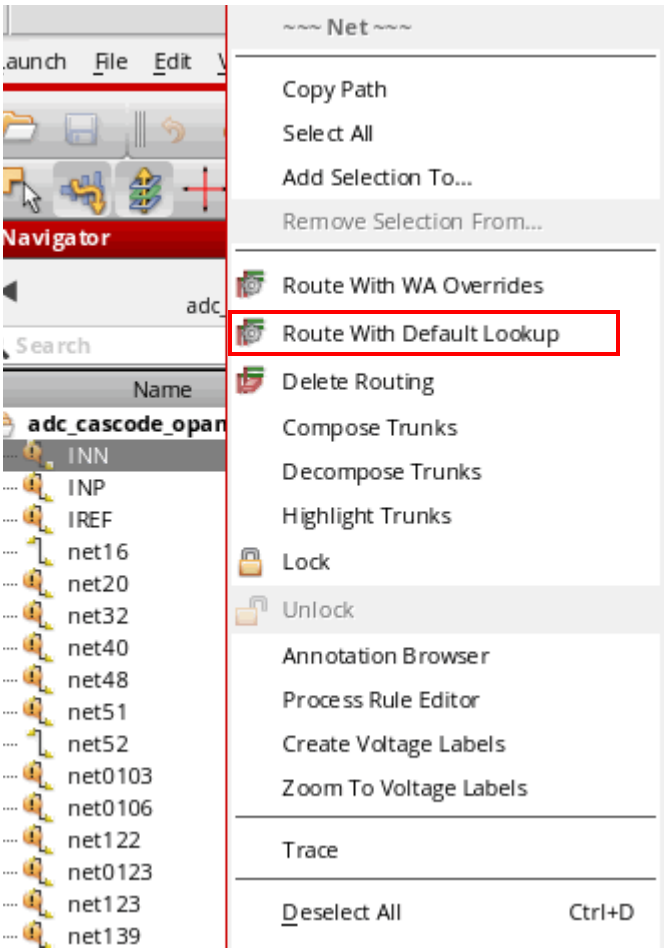
  .



© 2023 Cadence Design Systems, Inc.

- *Route With Default Lookup*

  Use this command to route the selected net by using the default constraint lookup for routing.



To route all the nets in a cellview, right-click the cell and choose *Route All Nets*.

Important

Cadence recommends that you lock the nets after performing pin-to-trunk routing to ensure that any subsequent routing by assisted routing commands does not rip the nets and re-route them as regular nets.

© 2023 Cadence Design Systems, Inc.

To delete the routing information and shielding wires on a net, you can right-click the net in the navigator and choose *Delete Routing* from the shortcut menu. Any routes or nets that are locked, are not deleted.

Related Topics

Override Constraints

Locking and Unlocking Nets to Set Routing Status

Deleting Routes

The Navigator Assistant

Locking and Unlocking Nets to Set Routing Status

You can lock and unlock nets to set the routing status on paths, pathSegs, and vias. In other words, by locking a net, you can implement a constraint on its position in the design. For more information on the *Locked* constraint, see Locked.

To lock or unlock a net, you can use the appropriate shortcut command from the Navigator assistant or the canvas.

Locking and Unlocking a Net from the Navigator Assistant

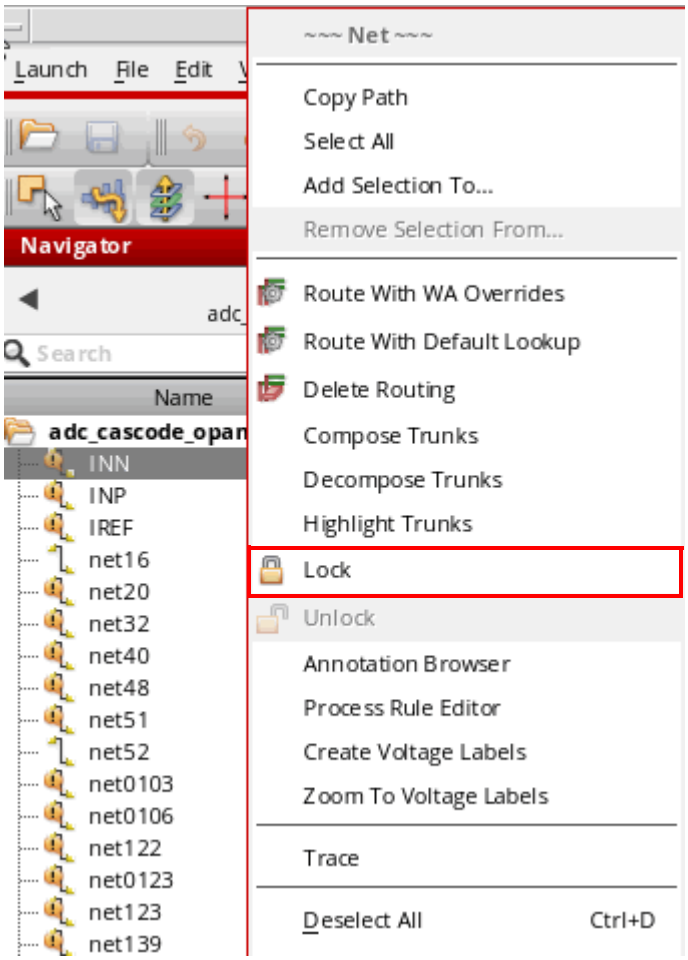
To lock a net by using the *Lock* command in the Navigator assistant shortcut menu:

1. Right-click a net or pin in the Navigator assistant.

   You can select multiple nets (or pins) to simultaneously lock (or unlock) them.

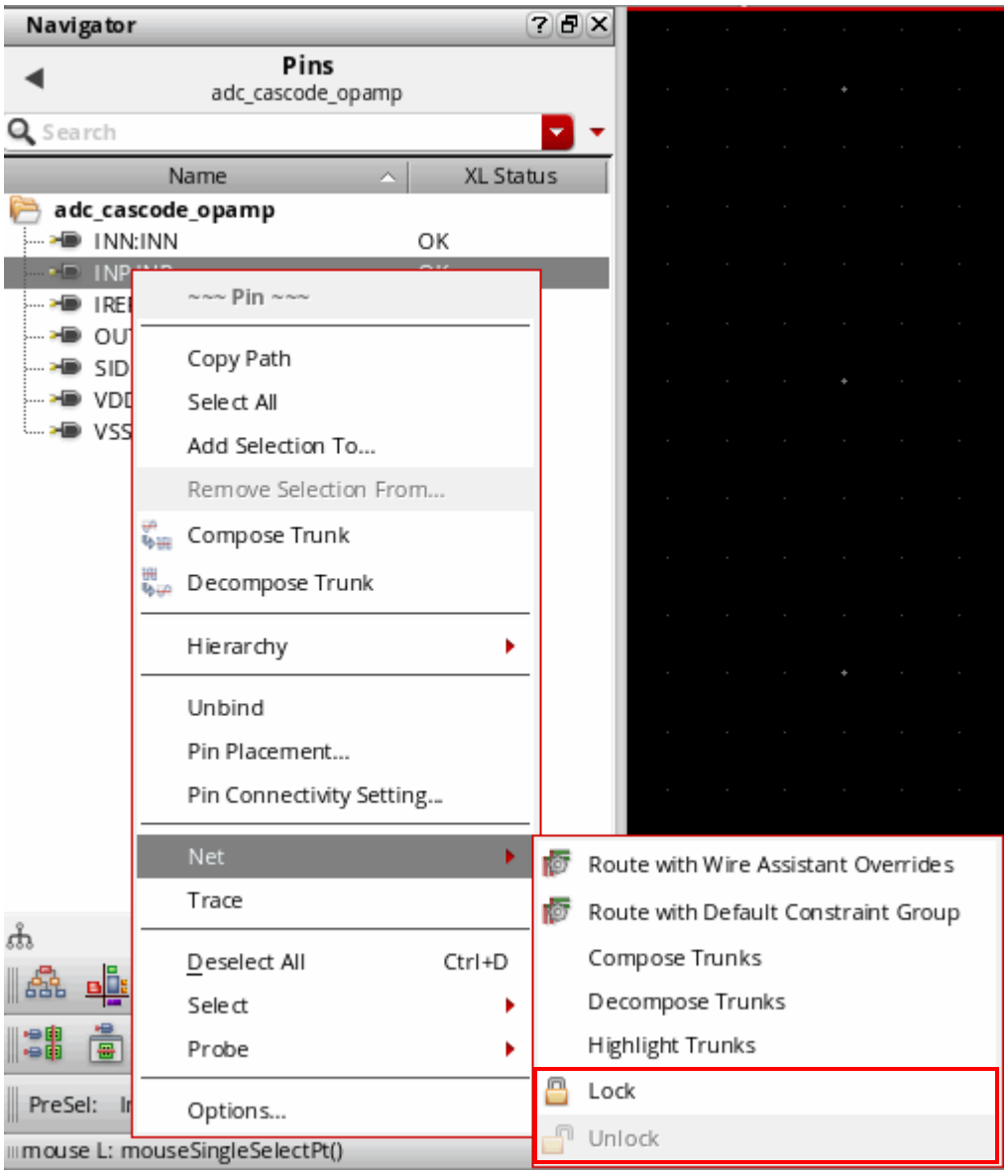
© 2023 Cadence Design Systems, Inc.

If you selected a net, the *Lock* (or *Unlock)* commands are displayed in the Navigator assistant shortcut menu.



© 2023 Cadence Design Systems, Inc.

If you selected a pin, the *Lock* and *Unlock* commands appear under the **Net** sub menu.



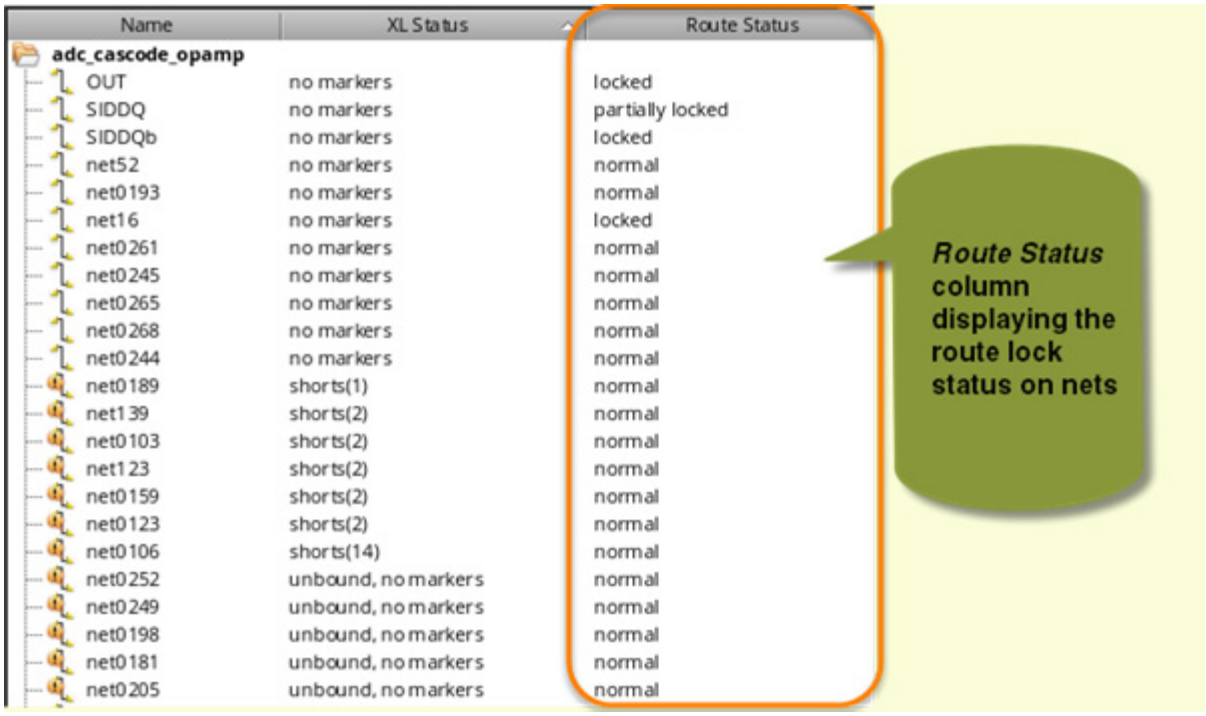
2. To lock the net, select the *Lock* command.

   This locks the position of the selected net on the canvas, making it impossible to move the net to a new position unless the net is unlocked.

The route lock status of a net can be viewed using the *Route Status* column, as shown in the figure below. The *Route Status* column is off by default.

© 2023 Cadence Design Systems, Inc.

Do not confuse the Navigator *Lock Status* column as the one displaying the routing lock status set using the Lock/Unlock commands. The *Lock Status* column displays information about the user and the host currently editing the cellview. For the route lock status of a net, use the *Route Status* column.



To unlock a net or a pin, select the *Unlock* command. Unlocking makes the net (or pin) open for relocation along the canvas.

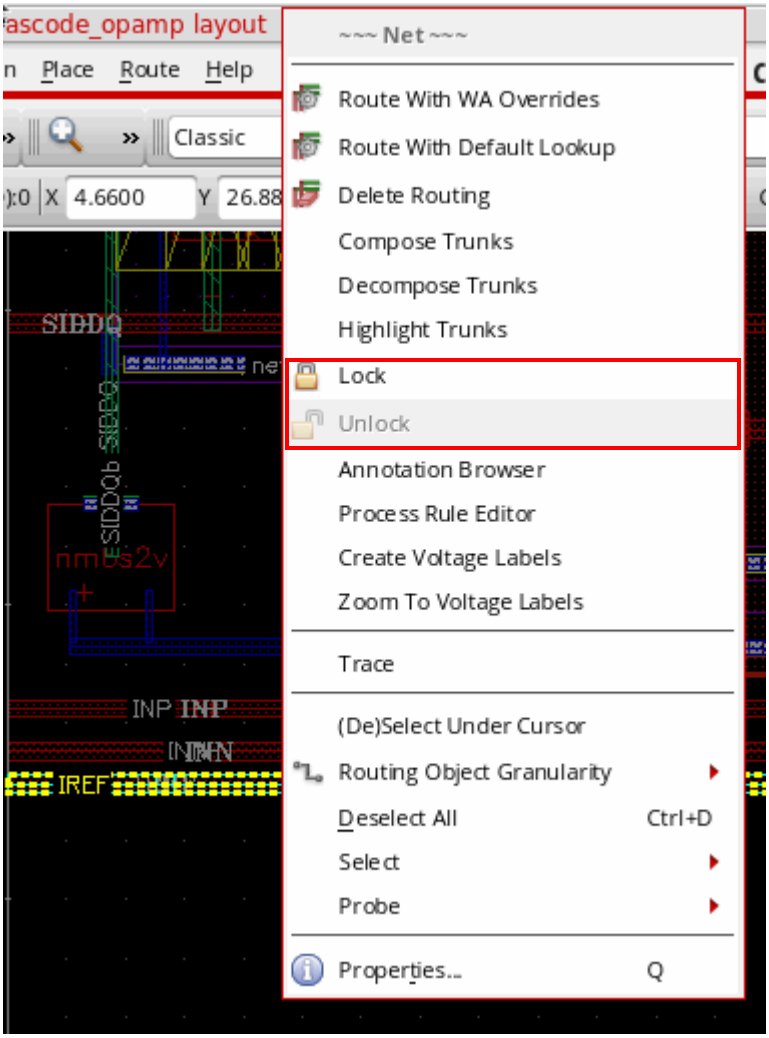
Locking and Unlocking a Net from the Canvas

To lock a net from the shortcut menu available on the Canvas:

1. Right-click a net, shape, or a pin on the canvas.

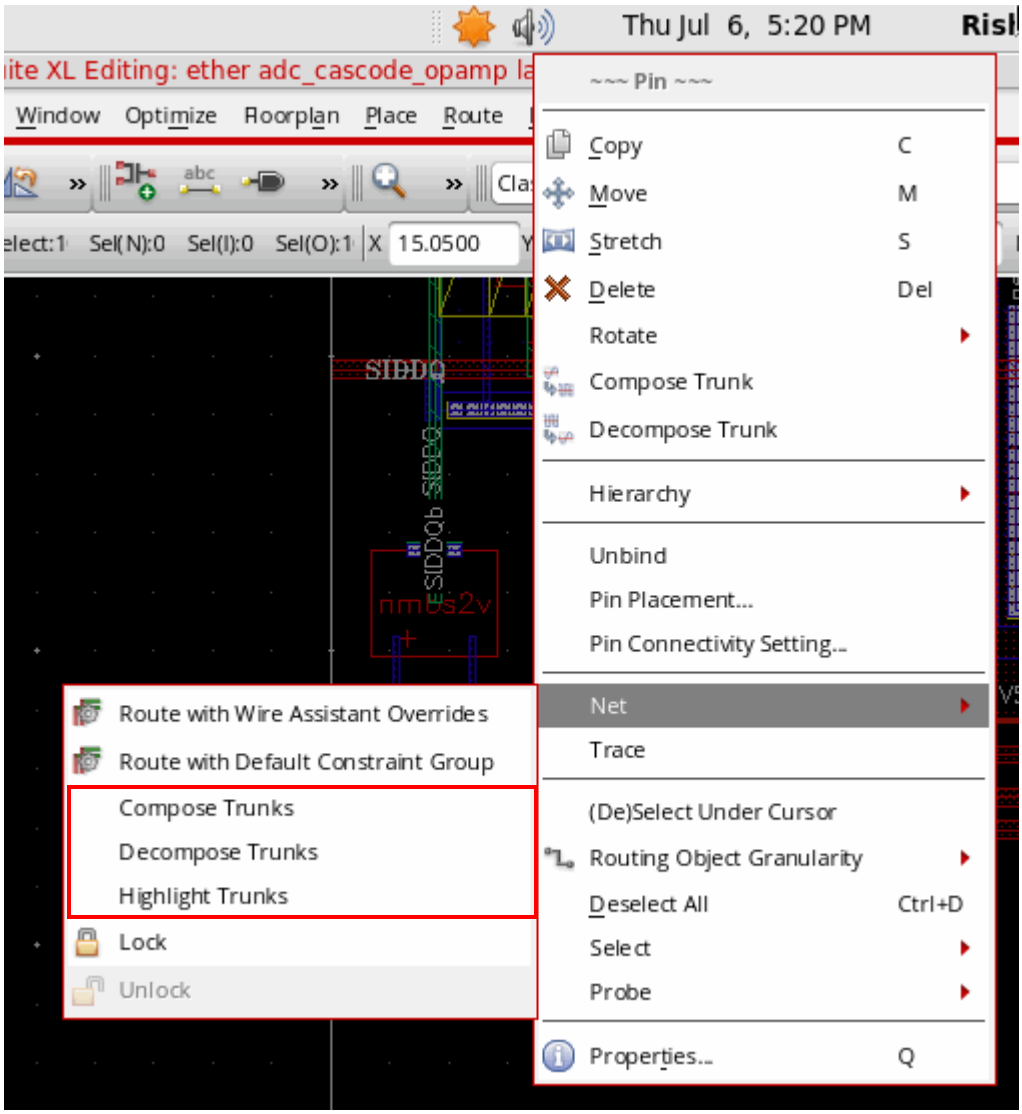
© 2023 Cadence Design Systems, Inc.

The shortcut menu is displayed.



© 2023 Cadence Design Systems, Inc.

If you selected a pin, the *Lock* and *Unlock* commands appear under the **Net** submenu.



2. To lock the net, select the *Lock* command.

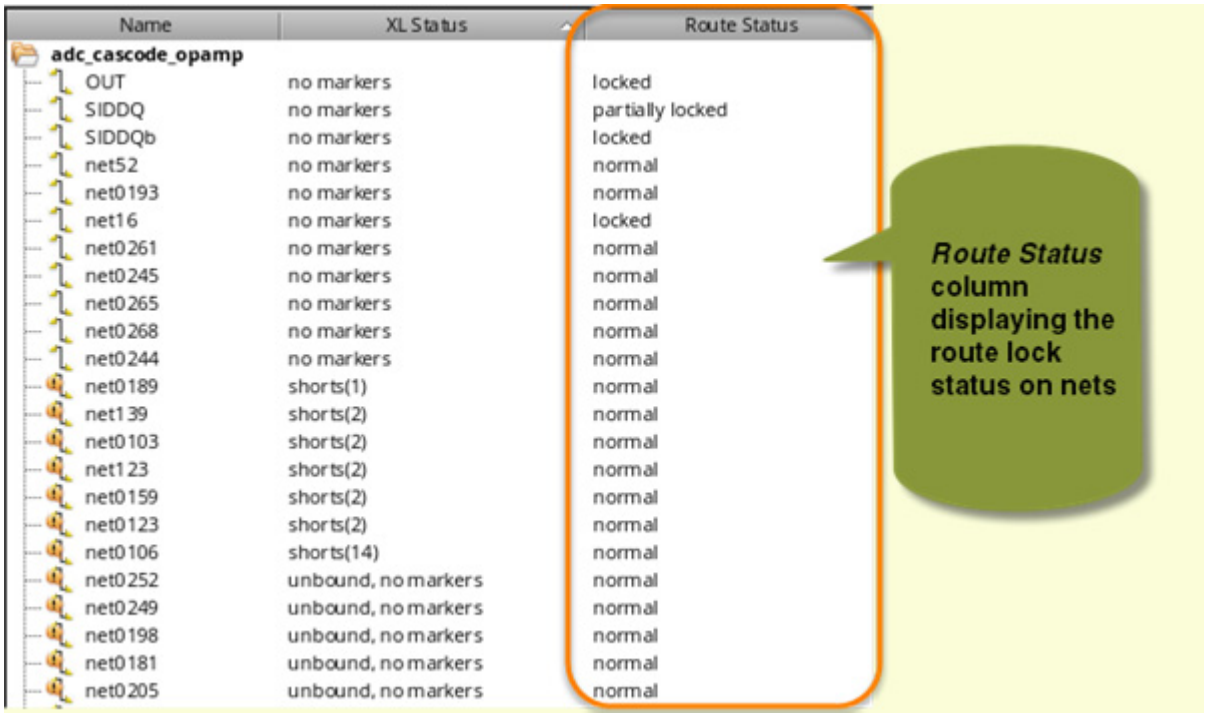
   This locks the position of the selected net on the canvas, making it impossible to move the net to a new position unless the net is unlocked.

The route lock status of a net can be viewed using the *Route Status* column, as shown in the figure below. The *Route Status* column is off by default.

Do not confuse the Navigator *Lock Status* column as the one displaying the routing lock status set using the Lock/Unlock commands. The *Lock Status* column displays information

© 2023 Cadence Design Systems, Inc.

about the user and the host currently editing the cellview. For the route lock status of a net, use the *Route Status* column.



To unlock a net, select the *Unlock* command from the shortcut menu. Unlocking makes the net open for relocation along the canvas.

Related Topics

Locked Constraint

Locking and Unlocking a Net from the Canvas

Editing Net Attributes and Properties

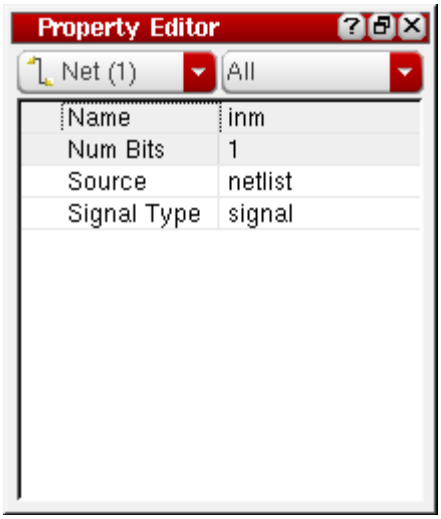
You can create and edit the attributes and properties on existing nets using the Property Editor assistant.

To edit net attributes:

1. In the Navigator assistant, select the net whose properties you want to edit.
2. From the layout window menu bar, choose *Windows – Assistants – Property Editor*.

© 2023 Cadence Design Systems, Inc.

The Property Editor assistant is displayed.



3. To change the signal type for a net:

   a. Click in the value column next to the *Signal Type* label to reveal a drop-down list of all the available signal types.

   b. Choose the signal type you want from the list.

4. To change the net assignment for a selected object, click in the *Net Name* field and type in the new net name for the object.

   When the selected object is a route element (i.e., a via or a pathSeg) the *Net Name* field is relabeled *Route Net Name*. This indicates that changing the net name for that object also changes the net name for the entire route.

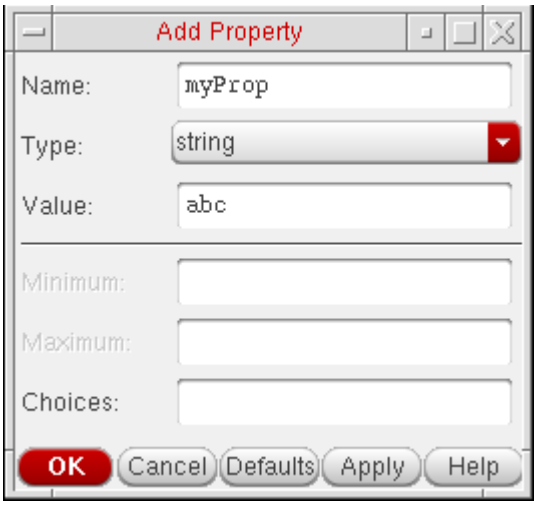
5. To delete a net assignment, remove the name of the net from the *Net Name* field.

6. To add a new property for a selected net;

© 2023 Cadence Design Systems, Inc.

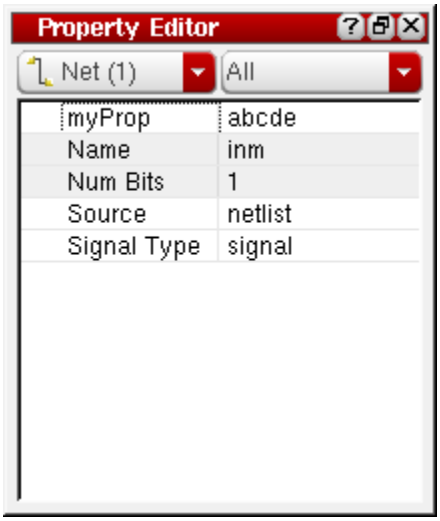
a. Click the right mouse button in the Property Editor assistant and choose *Add property to.*

The Add Property dialog is displayed.



b. Define the new property and click *OK* or *Apply* to add it to the net.

The Property Editor is updated to show the property you just added.



Related Topics

The Property Editor Assistant

© 2023 Cadence Design Systems, Inc.

Design Probing

Probing lets you select an instance, net, or pin in the layout or schematic window to highlight the corresponding element in the other window. This helps you interactively explore the design and check that the selected instance, net, or pin has a corresponding counterpart in the schematic window and where it is placed in the canvas.

- Use the options in the right mouse button menu to specify how nets, instances, and terminals are probed and which layer and colors are used to draw the probes. Also use the right mouse button to remove existing probes.
- Use the options in the XL Probe form to filter the objects that can be selected by clicking in a window and to locate a specific object to probe.
- Use the Highlight Options form to specify how nets, instances, and terminals are probed and which layer and colors are used to draw the probes.

The probes remain even after probing is canceled, meaning that you can use the results of probing as the basis for further operations in the layout, such as moving and stretching connected objects, increasing contact sizes, or adding well ties.

To display flightlines when probing an instance or a net, select the Net granularity using the right mouse button contextual menu or the Selection Options form. Then, choose the *Options - Highlight Options* command to display the Highlight Options form. From the Net section, select the *FlightLines* option. You should now be able to see flightlines on selecting the instance in the canvas.

Related Topics

XL Probe Form

Highlight Options

General Probing Behavior

You can select the object you want either by clicking in the schematic or layout window, or by choosing it from the list in the XL Probe form. For example,

- If you click an instance in the layout, the layout instance and the corresponding schematic instance are both highlighted using the display color specified in the Highlight Options form, and a message like the one below is printed in the Information Panel in the XL Probe form.

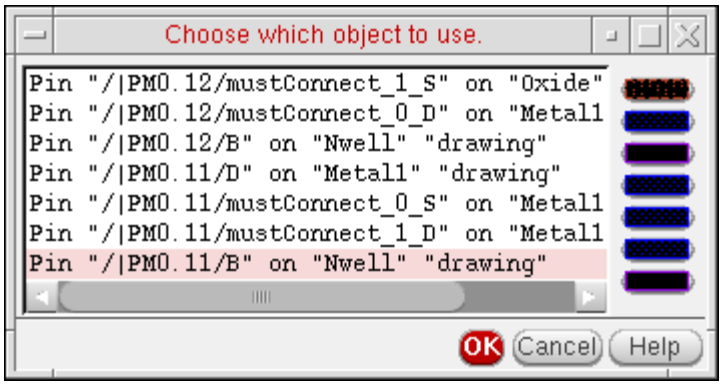
```
inst:(lay)Q12->(sch)Q12
```


© 2023 Cadence Design Systems, Inc.

- If Layout XL cannot find the corresponding schematic object, you see a question mark (?) at the end of the message in the XL Probe form.

  ```
  inst:(lay)Q12->(sch)?
  ```

- If you click a location where there is more than one object, Layout XL highlights the selected objects in the following order: pins; nets; instances.
- If you click a location where there is more than one of the same kind of object, a message window opens asking which one you want to probe.



- The probes remain highlighted even after probing is canceled.

Probing Terminals and Nets in Must-Connect Relationships

The following behavior applies when you probe terminals and nets in must-connect relationships.

- If you probe a terminal in the schematic, all the corresponding terminals (including must-connect terminals) in the layout are cross-probed using the same display color.
- If you probe a net in the schematic, all corresponding nets (including those connected by must-connect terminals) are cross-probed in the layout using the same display color.
- If you probe a terminal in the layout, the corresponding terminal in the schematic is cross-probed using the same display color and the other must-connect terminals in the layout are cross-probed in a different color.
- If you probe a net in the layout, the corresponding net in the schematic is cross-probed using the same display color and the other must-connect nets in the layout are cross-probed in a different color.

© 2023 Cadence Design Systems, Inc.

Probing Components in OTM, MTM, and MTO Relationships

When you probe a component bound by a one-to-many, many-to-many, or many-to-one correspondence, Layout XL highlights all the components in the bound group.

If you probe an external net of a bound group, the corresponding external net in the other window is highlighted. If you probe an internal net of a bound group, the entire bound group is highlighted.

Probing with Multiple Layout Cellviews

If you have multiple layout cellviews open and you select a design element in one part of a cellview pair (a connectivity source and a layout) Layout XL highlights the corresponding element in any other implementations of the other part. For example, if you have several versions of a layout open, selecting `R14` in the schematic highlights `R14` in each of the layouts of that schematic.

However, if you have a schematic and two different layouts open, the *XL Probe* command applies to only the layout from which you selected the command.

Related Topics

XL Probe Form

Highlight Options

Object Probing Using the Right Mouse Button

Dynamic Probing

Design Probing

Object Probing Using the Right Mouse Button

The shortcut menu available via the right mouse button (RMB) provides only relevant probing commands for a selected object. This is a significant usability enhancement that allows faster access to the appropriate commands.

For example, if the selected net does not have a probe added; the RMB menu displays the *Probe – Add* option. After the probe is added, the *Add* option is disabled and the *Remove* option is enabled instead.

The probe options available in the right mouse button shortcut menu are:

© 2023 Cadence Design Systems, Inc.

- **Add**: Adds a probe to a selected object. The color of the probe added is determined based on the color selected on the Highlight Options form.
- **Remove**: Removes the probe from a selected object.
- **Remove All**: Removes all the probes from all the objects in a design. This option also removes the cross probes in the schematic.
- **hilite drawing <0-9>**: If the selected object is not yet probed, selecting a hilite drawing color adds a probe in the selected color. If the selected object has already been probed, selecting a hilite drawing will change the color of the probe to the selected color.

Related Topics

Object Probing Using the Right Mouse Button

Adding a Probe using the Right Mouse Button

Removing Probes using the Right Mouse Button

Dynamic Probing

Design Probing

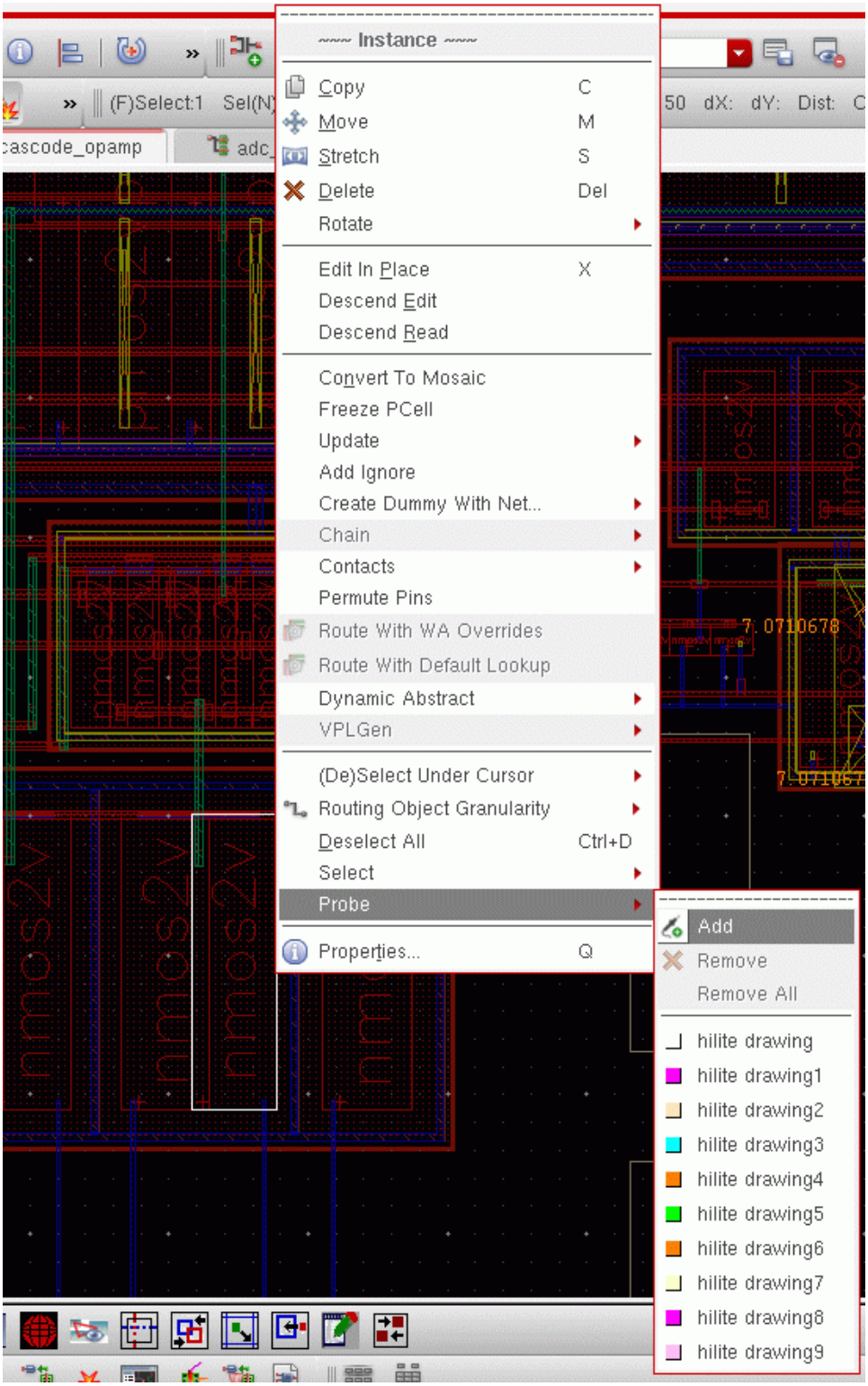
Adding a Probe using the Right Mouse Button

To probe an object by using the Probing commands available in the right-mouse button shortcut command:

1. Select the object in the canvas or the navigator and click the right mouse button.

© 2023 Cadence Design Systems, Inc.

A shortcut menu appears displaying the probing options.



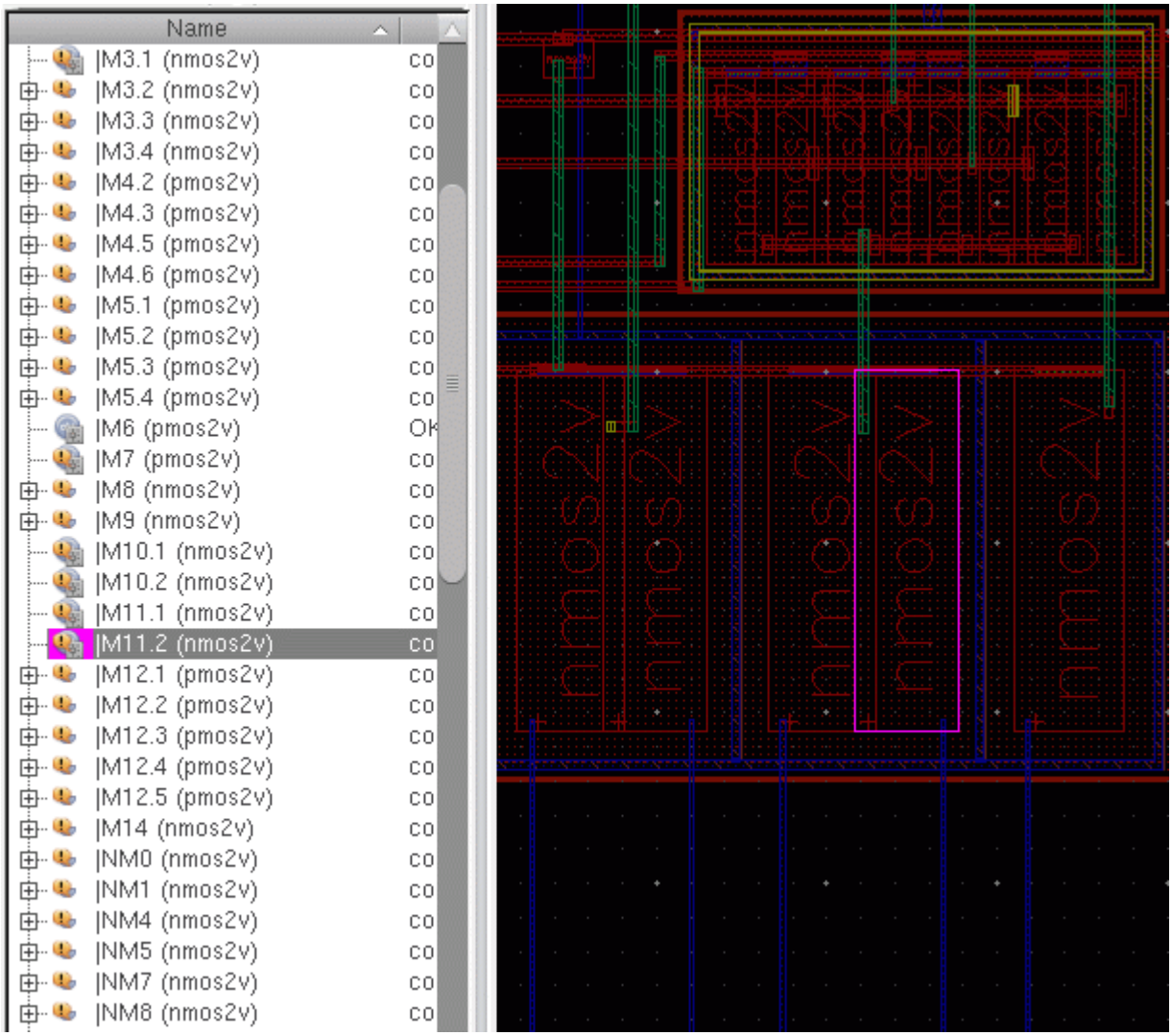
© 2023 Cadence Design Systems, Inc.

2. Click *Add*.

   A colored probe is added to the selected object in the layout view. A similar, corresponding probe is also added in the schematic view.

   To probe an object only in the schematic view, you must select the object in the schematic navigator.

The new probe you added also displays in the Navigator assistant.



The magenta highlight in both the views indicates the probe

The Navigator assistant not only displays new probes added in the canvas, but you can also use it to add new probes to the design. In fact, a probe you add to a higher level in the navigator tree automatically applies to the lower levels also. Therefore, you can use the

© 2023 Cadence Design Systems, Inc.

Navigator assistant for probing objects across the design hierarchies. This implies that probing a net at any level in the hierarchy will probe that net anywhere else it appears in the navigator, up or down the tree.

A probe on a lower instance will probe itself “up” the way. So, if instance I2, lies within instance I1; a probe you place one I2 will automatically propagate “up” to display itself on I1 also.

To probe an object, select the Navigator tree object you want to probe and click the right mouse button. From the options displayed, select the appropriate probing command.

The right mouse button probing options displayed in the Navigator tree are the same as those displayed in the canvas. Depending on the hierarchical level you select, the probe may be applied across the levels in a design. However, only probing an object at the current level of hierarchy will cross probe in the schematic. Probing at any other level of the hierarchy will only probe the layout.

- To add a probe in a specific color, select *Probe - hilite drawing<number>* from the color palette in the right mouse button menu.
- If you want the tool to automatically display subsequent probes in different colors, select the *Options - Highlight* command. Under the *Display Layer* section, select the *Cycle* check box. This ensures that any new probes added take their color from the palette. The first probe added gets the color at the top of the palette. Any subsequent probes added, get the color next in the sequence.
- If you want all the new probes to appear in a specific color, you must uncheck the *Cycle* option and specify the color to be used for the highlight.
- To modify a highlight color of an existing probe, you must first select the object from which the probe originates. Then, select an appropriate color from the palette in the right mouse button menu. To identify the object from which a probe originates, you can view the tooltip information on the probed object.

Related Topics

Object Probing Using the Right Mouse Button

Dynamic Probing

Design Probing

© 2023 Cadence Design Systems, Inc.

Removing Probes using the Right Mouse Button

An object may have two types of probes—the one added directly and the ones inherited from objects at a higher level. For the inherited probes to be deleted from the object, you need to remove the probe from where it originates.

For the direct probe to be removed from an object, you can use the shortcut menu available through the right mouse button.

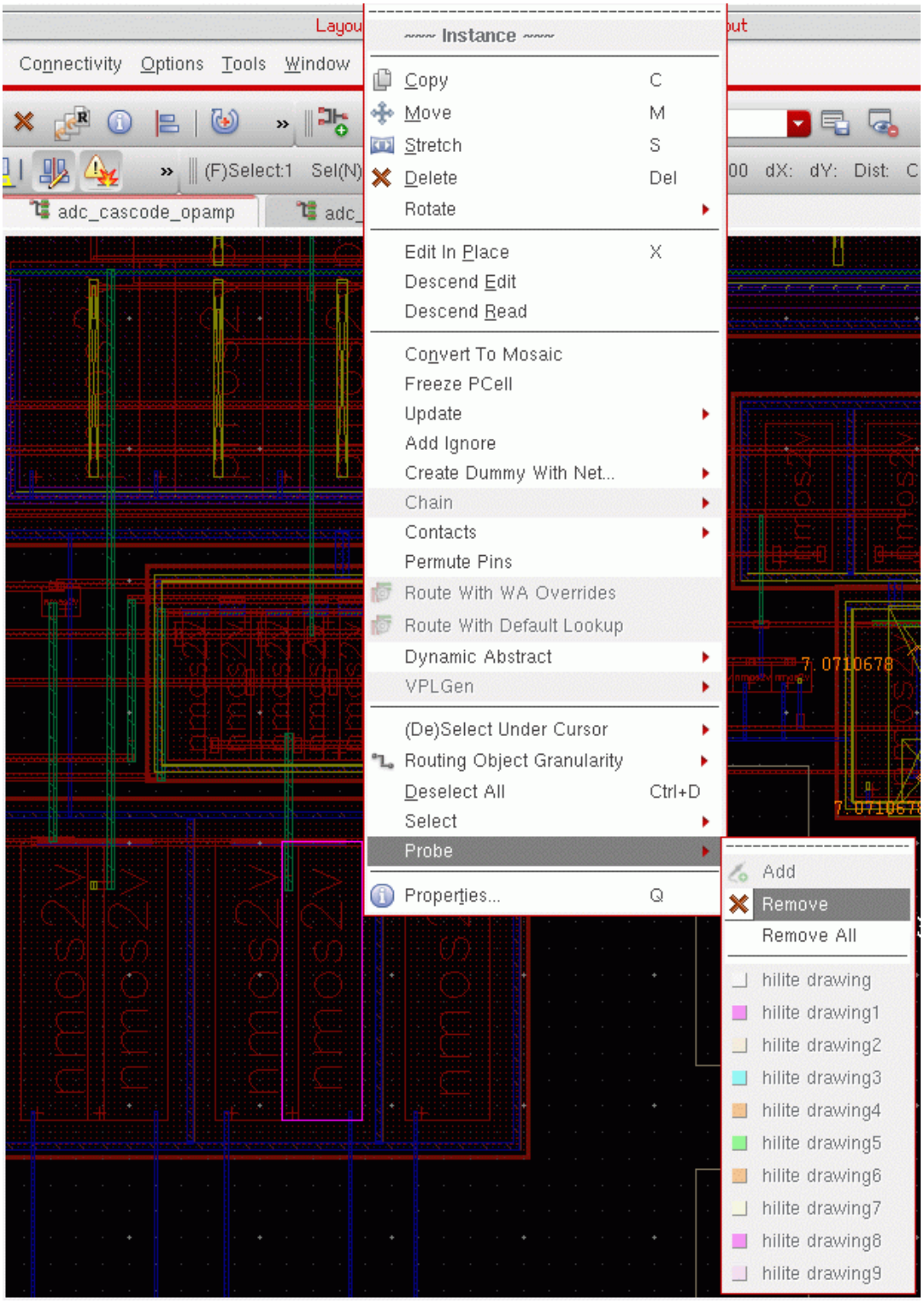
Removing a Probe using the Right Mouse Button

To remove a probe on an object using the right mouse button:

1. Select the object in the layout canvas or the navigator and click the right mouse button.

© 2023 Cadence Design Systems, Inc.

A shortcut menu appears displaying the probing options.



© 2023 Cadence Design Systems, Inc.

2. Click *Remove*.

The probe on the selected object is deleted.

Removing All Probes using the Right Mouse Button

Unlike removing probes from specific objects, if you want to remove all the probes in a design with a single click; you can use the *Removes All* option. This removes all the probes in the design, irrespective of the object selected. In fact, the option can also be invoked using the right mouse button with no object selected.

The *Remove All* option is enabled in the shortcut menu only when a design carries one or more probes. This option allows for a single command to be run on several objects, improving usability and performance. In addition, the same command can be used for removing probes across the schematic and layout views. This not only improves performance but also helps ensure consistency across the two views.

Related Topics

Object Probing Using the Right Mouse Button

Adding a Probe using the Right Mouse Button

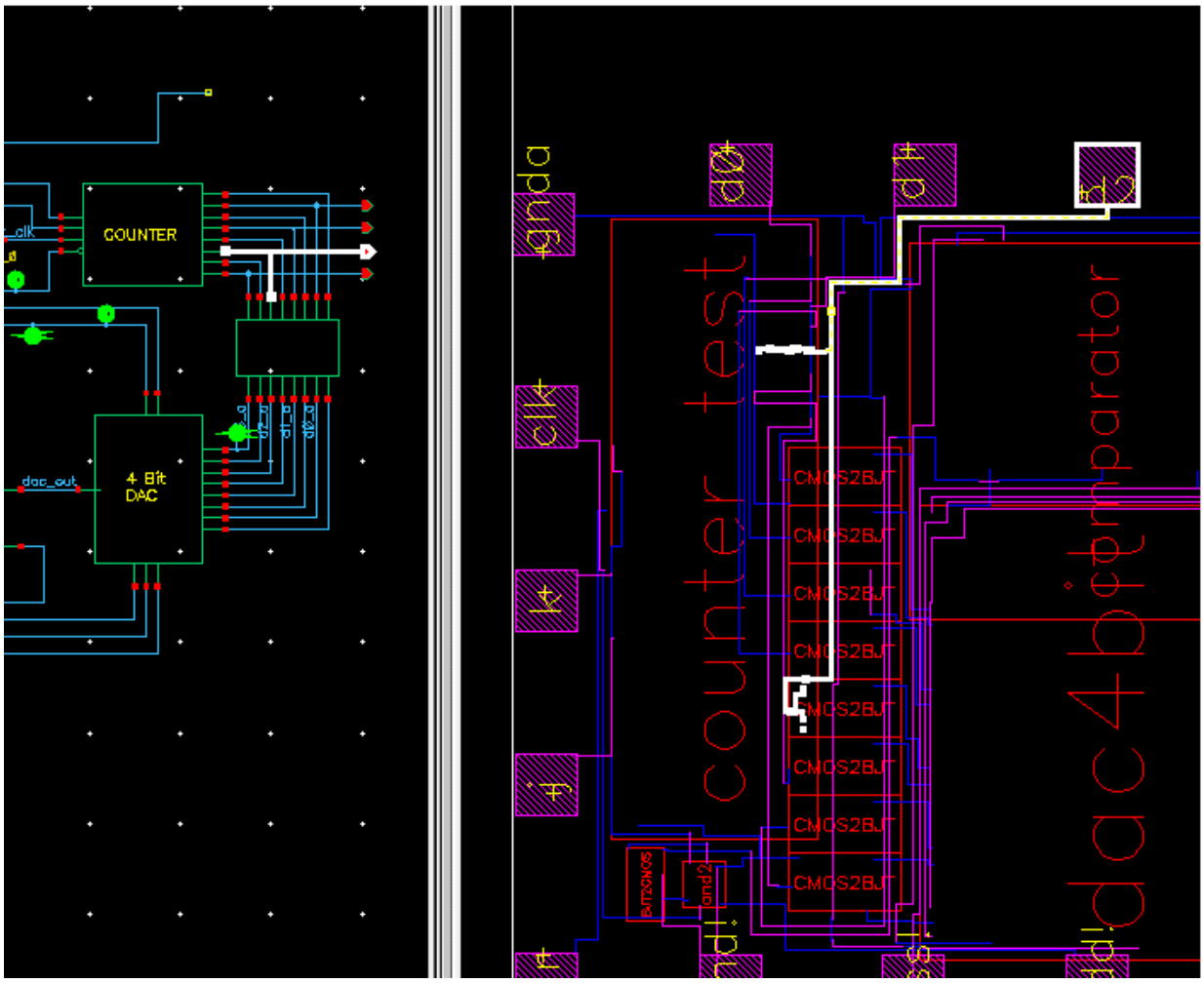
Dynamic Probing

Design Probing

© 2023 Cadence Design Systems, Inc.

Dynamic Probing

The Dynamic Probing feature allows for probing an object on mouse over. For each probe created in the canvas, a corresponding probe in the schematic view is also created. However, as soon as the mouse moves off the object; the probe is removed.



The probe in white is a dynamic probe

The rules for probe highlighting and color selection are the same as those for adding a new probe. For more information, see Adding a Probe using the Right Mouse Button.

To invoke the dynamic probing feature, access *Options – Highlight* and select the *Dynamic Probe* option.

© 2023 Cadence Design Systems, Inc.

For the probes to be dynamically highlighted in the navigator; make sure the *Display Probe* option on the Customize Navigator Filters form is selected. To access the Customize Navigator Filters form, click the ellipsis (**...**) button adjacent to the **Show** drop-down in the Navigator assistant.

Related Topics

The Navigator Assistant

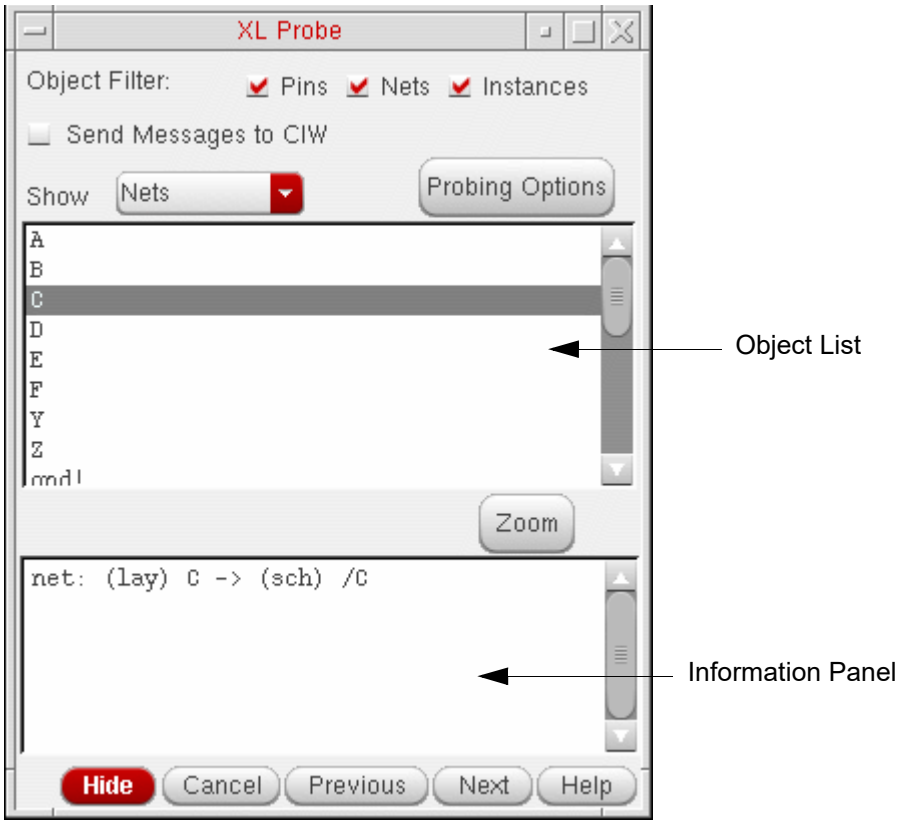
Design Probing

Probing Objects by Clicking in a Window

To probe an object by clicking in either the layout or schematic window,

1. From the layout window menu bar, choose *Connectivity – XL Probe*.

   The XL Probe form is displayed.



© 2023 Cadence Design Systems, Inc.

2. In the *Object Filter* section, specify the types of objects that can be probed when you click in the layout or schematic window.

   For example, if you check only *Pins*, you can create probes only for pin objects by clicking in either window. Clicking on a net or instance has no effect. You can use this option to prevent certain types of objects from being probed in congested areas of your design.

   This option has no effect on probe creation using the object list in the XL Probe form, only on probe creation by clicking in the layout or schematic window.

3. Click the object you want to probe in either the schematic or layout window.

   If you click a pin, Layout XL probes only the `pinFig`s associated with the pin. If you click a net, Layout XL probes all the shapes on the specified net, including routing shapes and `pinFig`s.

   Information on the probed object is printed to the Information Panel in the XL Probe form and the relevant objects are highlighted in the schematic and layout windows.

4. Adjust the probing behavior as required using the options on the form.

   - To display information on the selected device in the CIW, turn on *Send Messages to CIW*.
   - To specify how nets, instances, and terminals are probed and which layer and colors are used to draw the probes, click *Probing Options*.
   - To zoom in on the bounding box of the probed objects, click *Zoom*.

Exiting the Probe Command

To exit the *XL Probe* command:

➡ Click *Cancel* in the XL Probe form or press `Esc`.

Removing Probes

To remove an individual probe from the layout:

➡ Hold down the `Control` key and click the probe to delete.

To remove all probes from the layout and schematic windows, do one of the following.

1. Click an empty space in the layout canvas.
2. Press `Ctrl+l` or run the following SKILL command in the CIW.

© 2023 Cadence Design Systems, Inc.

```
lxProbeRemoveAll()
```

Related Topics

XL Probe Form

Highlight Options

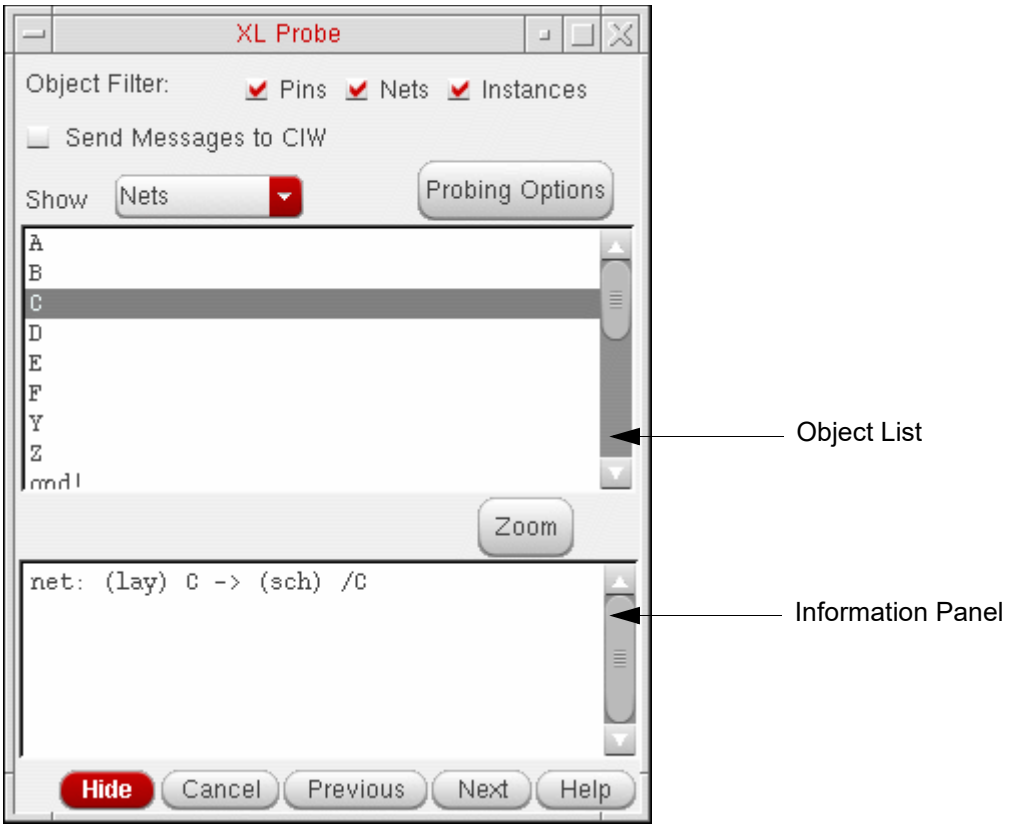
Design Probing

Probing Objects using the XL Probe Form

To probe an object from the XL Probe form,

1. From the layout window menu bar, choose *Connectivity – XL Probe*.

   The XL Probe form is displayed.



2. From the *Show* drop-down, choose the type of layout object you want to probe: *Pins*, *Nets*, *Net Classes*, or *Instances*.

© 2023 Cadence Design Systems, Inc.

The Object List is updated to show only layout objects of the specified type.

The *Object Filter* option has no effect on this list, only on probe creation by clicking in the layout or schematic window.

Here, CMX legacy net classes are listed in the XL Probe form but are not shown under the *Net Class* category in the Constraint Manager assistant.

3. Choose the layout object you want to probe from the list.

   Information on the probed object is printed to the Information Panel in the XL Probe form, the relevant object is highlighted in the layout window, and its schematic counterpart in the schematic window.

4. Adjust the behavior as required using the options on the form.

   - To display information on the selected device in the CIW, turn on *Send Messages to CIW*.
   - To specify how nets, instances, and terminals are probed and which layer and colors are used to draw the probes, click *Probing Options*.
   - To zoom in on the bounding box of the probed objects, click *Zoom*.

Related Topics

XL Probe Form

Highlight Options

Design Probing

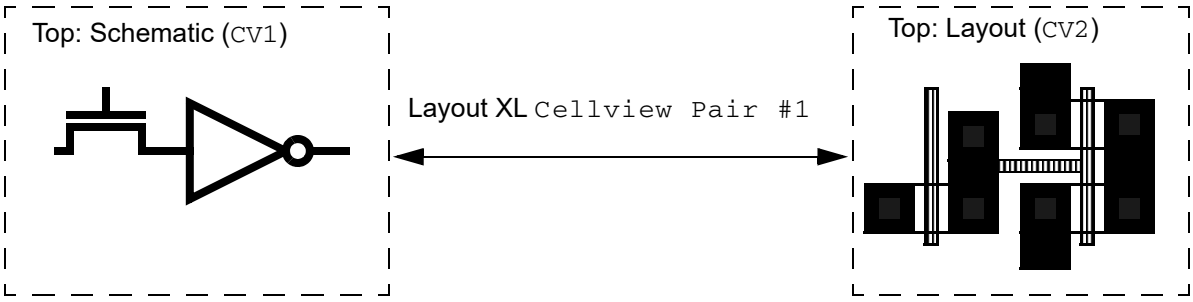
Probing a Hierarchical Design

To probe a hierarchical design,

1. Open a hierarchical design in Layout XL.

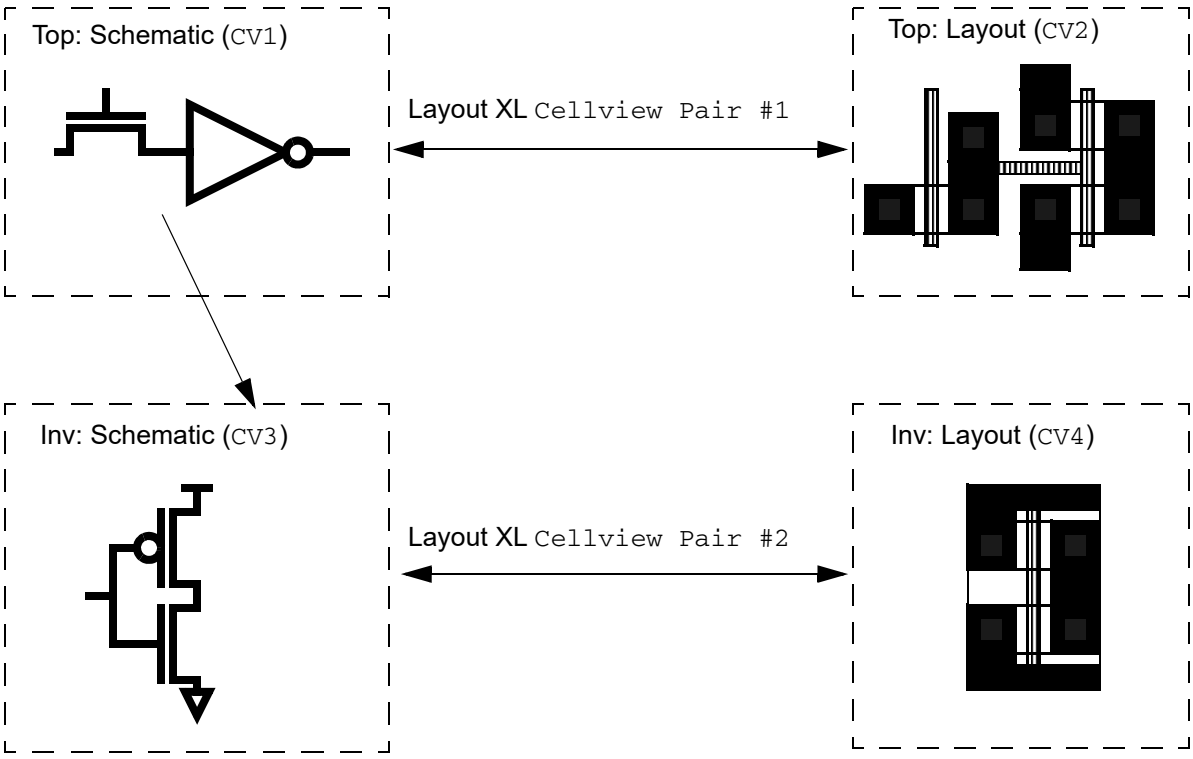
© 2023 Cadence Design Systems, Inc.

In the example below, this is `Cellview Pair #1`.



2. From the layout window menu bar, choose *Connectivity – XL Probe*.
3. Choose the instance to probe; for example, `CV1`, which is represented in the layout as pair of transistors (a flat representation).
4. Choose *Edit – Hierarchy – Edit in Place*.

   The schematic view of the inverter opens (`CV3` in the diagram below).

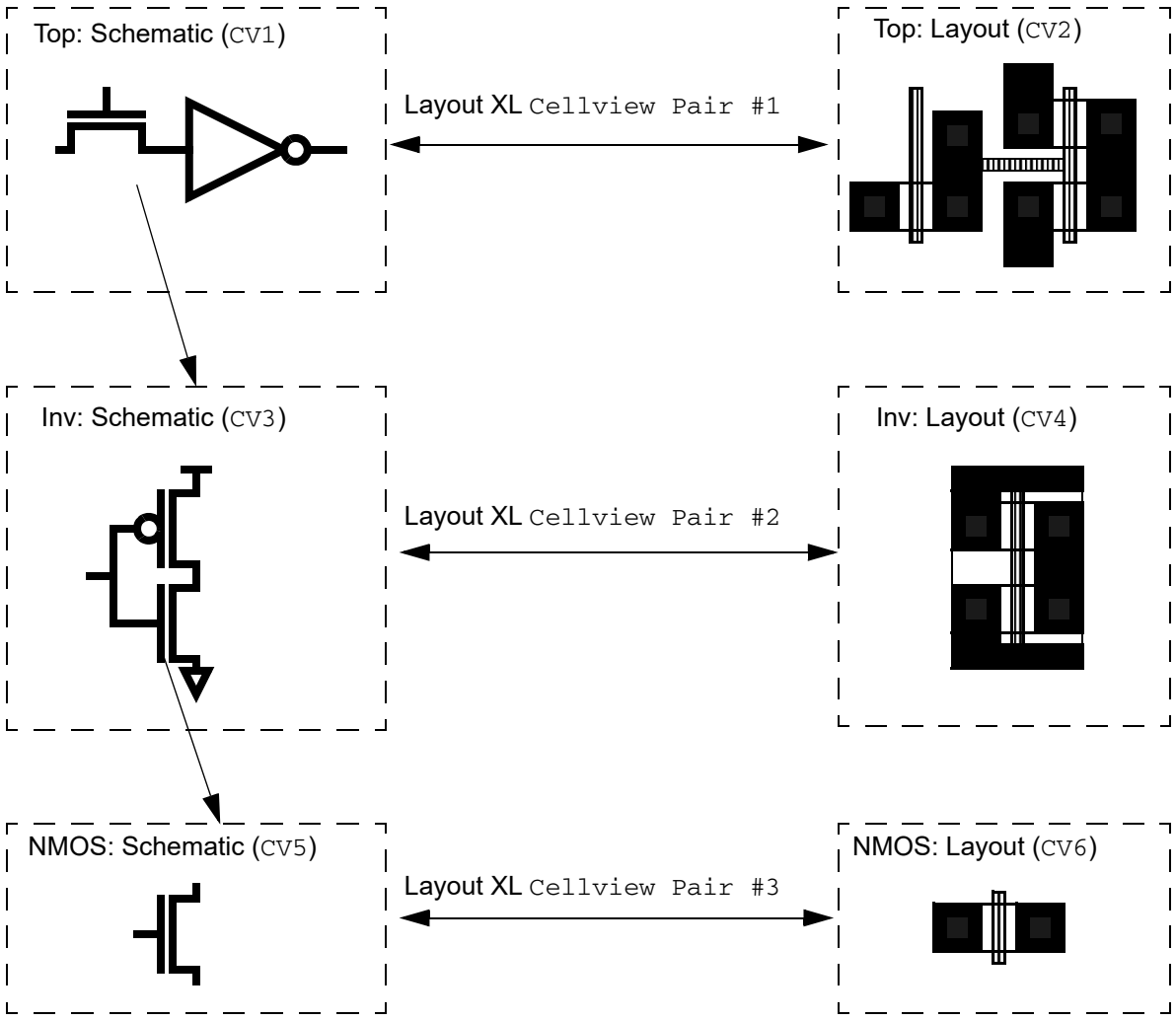


If you probe one of the transistors in this schematic, the corresponding transistor in the layout in the level above is highlighted (`CV2` in the diagram).

© 2023 Cadence Design Systems, Inc.

5. In the inverter schematic (CV3), select one of the two NMOS instances.
6. In the schematic window, choose *Launch – Layout XL* to open the layout view for the inverter in Layout XL. This is CV4 in the diagram above.

   This creates another cellview pair, `Cellview Pair #2`.
7. To descend into the schematic of the inverter, select one of the NMOS instances and from the schematic window choose *Edit – Hierarchy – Edit in Place*.



The schematic of the NMOS opens, CV5 in the diagram.

If you probe the transistor in this schematic, the corresponding transistor in the layout in the level above is highlighted (CV4 in the diagram).

© 2023 Cadence Design Systems, Inc.

8. In the schematic window, choose *Launch – Layout XL* to open the layout of the transistor (`CV6` in the diagram) in Layout XL.

   This creates another cellview pair, `Cellview Pair #3`. You can probe from `NMOS: Schematic` back to `Top: Layout`, `Inv: Layout`, and `NMOS: Layout`.

Related Topics

XL Probe Form

Design Probing

Tracing Nets

Use the *Connectivity – Net Tracer* command to trace the physical and logical connectivity of nets in a design.

Depending on the tracing mode you select, the traces generated by the *Net Tracer* command allow you to explore the following:

- physical connections of a shape, the layer on which the shape exists, and the associated nets (*physical net tracing*)
- logical connectivity of the shapes and the associated nets (*logical net tracing*)
- shapes chased as a net is traced

The true color display capability available in the Net Tracer allows you to easily distinguish the shapes belonging to a trace. In addition, you can halo (highlight) the shapes belonging to a trace. The automatic dimming capability that the Net Tracer supports further improves the visibility of traces and the shapes belonging to a trace.

The Net Tracer reads the layer connectivity information from the same technology file constraint group as used by the Layout XL connectivity extractor. Therefore, if your technology file is correctly set up, no additional setup is required for running the Net Tracer. By default, the constraint group used is `virtuosoDefaultExtractorSetup`. But, you can customize the setup, if required, using the Net Tracer Custom Setup form.

Key features provided by the Net Tracer command:

- Allows single-click trace creation and removal.
  - Click a shape in the canvas to add a trace. To remove an existing trace, click the shape, keeping the `Ctrl` key pressed.

© 2023 Cadence Design Systems, Inc.

  - The *Trace* command is also available in the shortcut menu for traceable objects, such as nets and pins.

- Enables hierarchical shape tracing without any performance degradation.
- Displays traces in the true shape color, making it easy to view the shapes that are being traced. Also supports creating halos in true color.
- Supports dynamic display of the created traces.
- Enables saving the created traces to a cellview, including their locked mask color, and allows loading the saved traces, if required.
- Displays a dynamic preview that highlights the shape being hovered over, and displays the name of the net and the layer on which the trace will be created, if the shape is selected for tracing.
- Dynamically updates the net traces to reflect any changes to the underlying traced shapes, such as during move, stretch, or delete.
- Enables tracing the neighboring shapes within a specified distance.
- Provides a Trace Manager Form form to control the color of the existing traces and the trace visibility.

Related Topics

Net Tracer Custom Setup Form

Net Tracer Toolbar

Adding Net Traces

Removing All Traces

Saving All Visible Traces

Loading Physical Traces

Creating a Step Trace

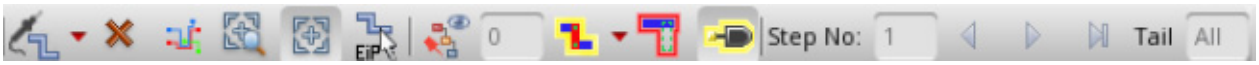
Finding Neighboring Shapes of a Traced Shape

Adding a Trace Using the Trace Manager

© 2023 Cadence Design Systems, Inc.

Net Tracer Toolbar

Use the Net Tracer toolbar buttons to launch the Net Tracer, to manage the generated traces, and to control how traces are displayed in the layout canvas.



Icon	Name	Description
	*Add/Remove trace*	Adds or removes a physical, logical, or step trace. Press F3 after clicking the toolbar button to launch the Net Tracer Options Form.
	*Delete all traces*	Deletes all traces in the design. See Removing All Traces.
	*Show Traces*	Opens the Trace Manager Form.
	*Zoom to fit visible traces*	Zooms to display all the visible traces in the layout canvas.
	*Auto zoom to selected traces*	Automatically zooms into traces that are selected in the Trace Manager.
	*Net Tracer EIP*	Edits in place the instances of the traced figure. Press F3 after clicking the toolbar button to launch the Net Tracer Edit In Place dialog box.

© 2023 Cadence Design Systems, Inc.

Icon	Name	Description
	*Display Depth*	Sets the hierarchy depth up to which the corresponding shapes for a traced net are displayed. By default, *Display Depth* is in sync with the *Display Levels – Stop* value in the Display Options form. ■ When the *Display Depth* toolbar button is clicked and a value specified in the associated text field, the specified value is used. The *Display Depth* drop-down in the Net Tracer Display Options form also reflects the same value. ■ When the *Display Depth* toolbar button is not clicked, the value set in the *Display Levels – Stop* field in the Display Options form is used. The *Display Depth* drop-down in the Net Tracer Display Options form is set to *Same as stop display*. See Net Tracer Display Options Form.
	*Display Mode*	Lets you choose if a trace is displayed: ■ in true color and with a halo ■ only in true color ■ only with a halo See Net Tracer Display Options Form.
	*Display halo on visible shapes only*	When selected, displays a halo only around visible shapes. When deselected, which is the default setting, displays a halo around all the shapes on a trace, even if the shapes are not visible. See Net Tracer Display Options Form.

© 2023 Cadence Design Systems, Inc.

Icon	Name	Description
	*Pins*	When selected, pins have the same halo thickness as the shapes. When deselected, which is the default setting, creates a wider halo around the pin shapes, making it easier to locate the pins in the canvas. See Net Tracer Display Options Form.
	*Current Step*	Displays the current step value of the trace.
	*Tail Value*	Displays the value of the tail of a trace.

Related Topics

Tracing Nets

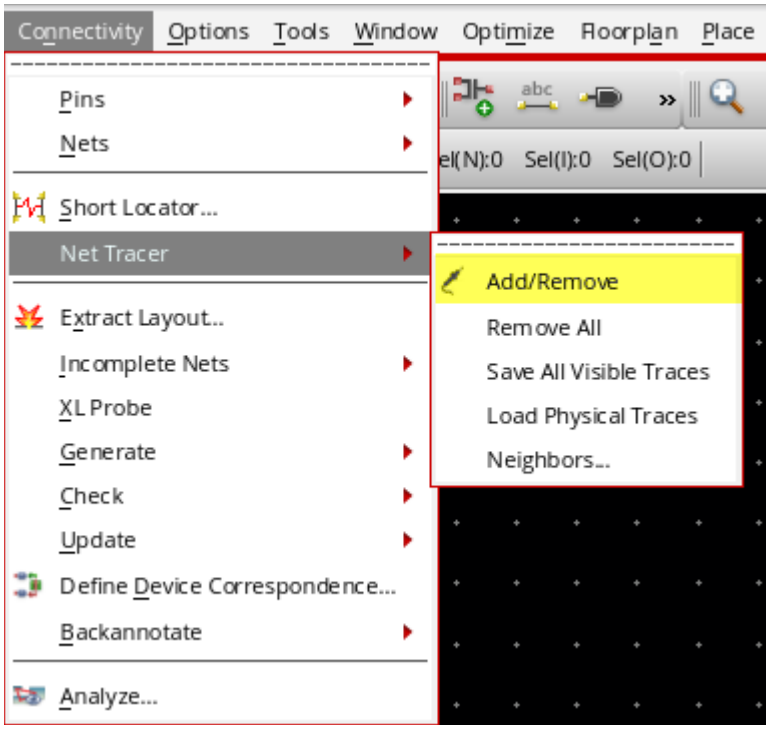
Adding Net Traces

To invoke the Net Tracer to add (or remove) traces for selected nets:

1. Use the *Connectivity – Net Tracer – Add/Remove* command and press `F3`.

© 2023 Cadence Design Systems, Inc.

Alternatively, you can use the Net Tracer Toolbar to launch the Net Tracer.

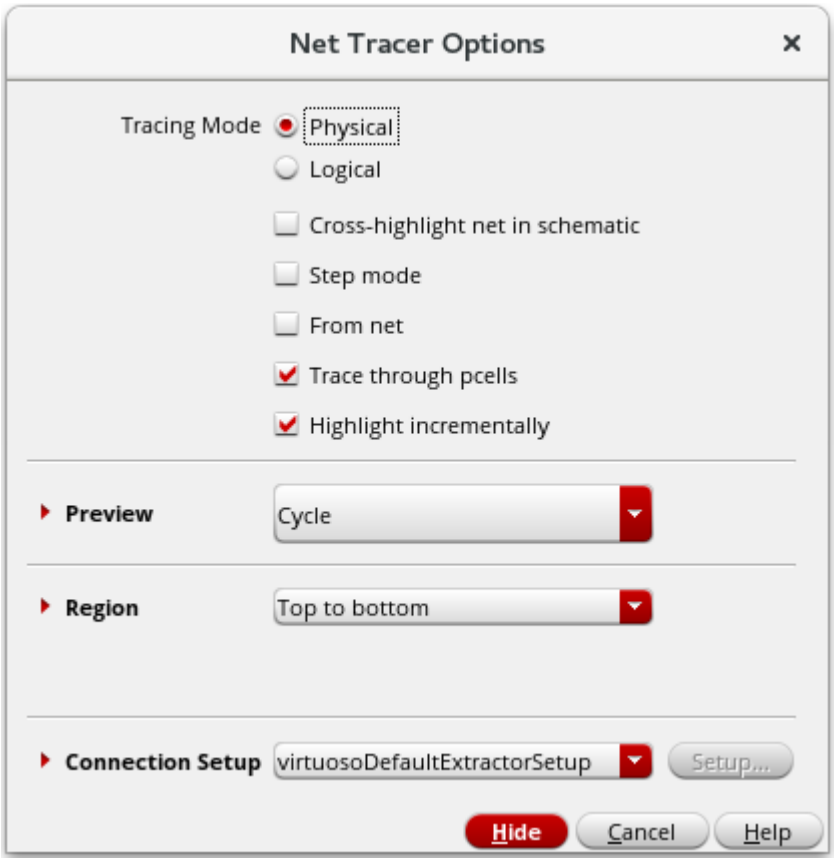


The layout cursor changes to display a pen icon,  , indicating the Net Tracer functionality is enabled.

2. (*Optional*) Press `F3` to open the Net Tracer Options form. Use the form to specify the Layout XL constraint group to use for net tracing and the hierarchy range for which to create the traces. You can also use the form to specify your trace preview preferences and to specify whether to create traces from a net and whether the traces should be created incrementally. The Net Tracer Options form is available through the

© 2023 Cadence Design Systems, Inc.

Connectivity – Net Tracer – Add/Remove command. Alternatively, use the lntHiNetTracer SKILL function.



3. (*Optional*) Choose *Options – Net Tracer* to open the Net Tracer Display Options form. Use the form to specify the required trace display options.
4. Click the shape in the layout canvas for which a trace needs to be created.

   If the shape that you select for creating the trace happens to be at an overlap of several layer purpose-pairs, the *Choose LPP To Trace* dialog box pops up.
5. Select the layer purpose pair to use for creating the trace and click *OK*.

   The shape is traced in the layout canvas, using the tracing and display options selected.
6. To remove an already existing trace for a shape, click the shape in the layout canvas while keeping the `Ctrl` key pressed.

Removing All Traces

To remove all the existing traces in a design:

© 2023 Cadence Design Systems, Inc.

➡ Choose *Connectivity – Net Tracer – Remove All.*

Alternatively:

- Select the *Delete all traces*  toolbar button.
- In the Navigator assistant, right-click the traced net and choose *Trace Net – Remove All* from the shortcut command.

 *Tip*

To remove the trace for a specific shape, click the shape while keeping the `Ctrl` key pressed.

Related Topics

Net Tracer Display Options Form

IntHiNetTracer

Net Tracer Toolbar

Tracing Nets

Saving All Visible Traces

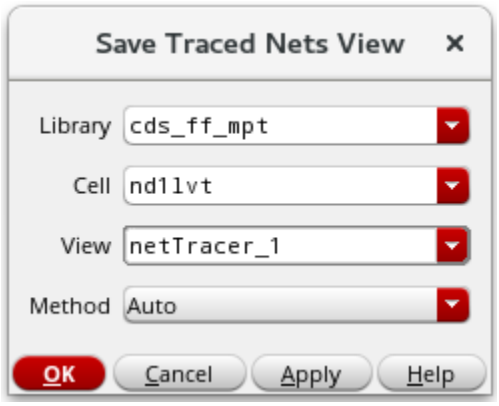
You can save all the traces that are visible in a design to a cellview. You can mark traces as visible by using the Trace Manager form.

To save all the visible traces in your design to a cellview:

1. In the layout window, choose *Connectivity – Net Tracer – Save All Visible Traces.*

© 2023 Cadence Design Systems, Inc.

The Save Traced Nets View form opens.



2. Specify the name of the library, cell, and view in which you want to save the visible traces.
3. Optionally, select the method to be used for saving all visible traces.
4. Click *OK* or *Apply*.
   The visible traces are saved to the specified cellview and can be loaded in another cellview.

Related Topics

Save Traced Nets View Form

Trace Manager Form

Load Traced Nets View Form

netTracerSaveMethod

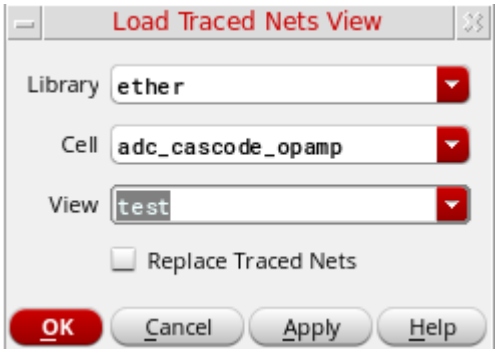
Loading Physical Traces

To load physical traces from a previously saved cellview to your current cellview:

1. Choose *Connectivity – Net Tracer – Load Physical Traces.*

© 2023 Cadence Design Systems, Inc.

The Load Traced Nets View Form opens.



2. Specify the view name to use for loading the visible traces.
3. (*Optional*) Choose *Replace Traced Nets* to replace the displayed traces in the current cellview with the traces being loaded.
4. Click *OK*.

   The previously saved traces from the selected cellview are loaded to the current cellview.

Related Topics

Load Traced Nets View Form

Net Tracer Toolbar

Creating a Step Trace

Tracing Nets

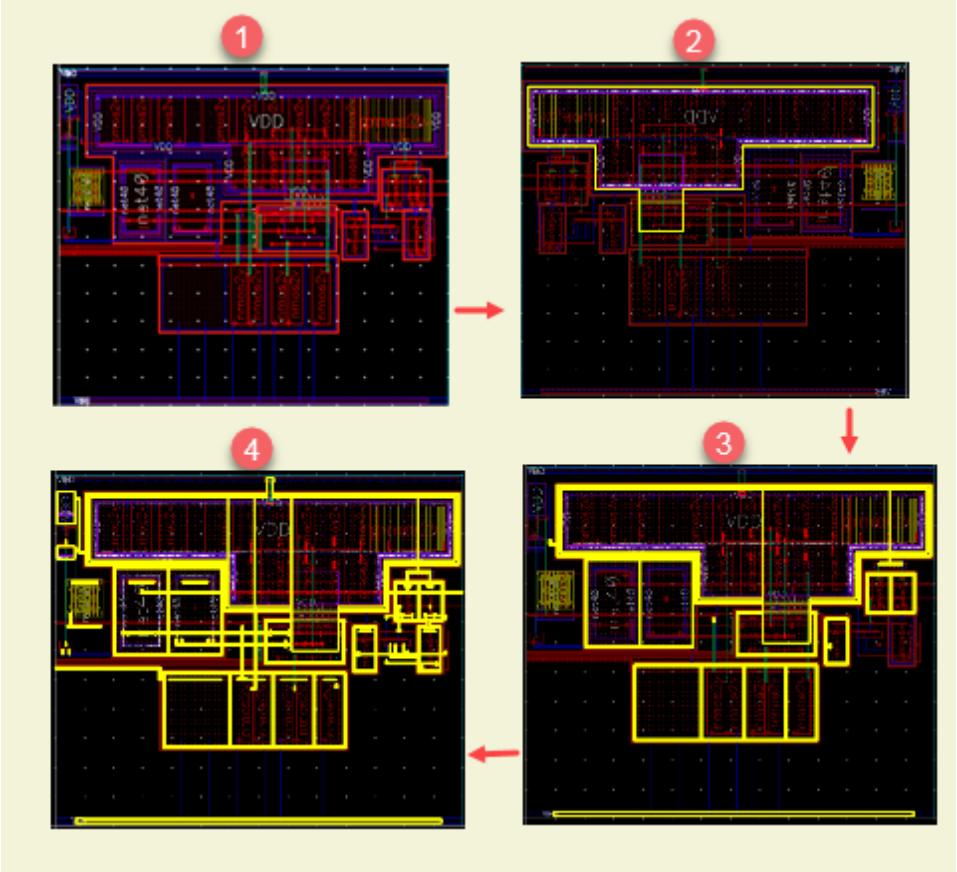
Creating a Step Trace

With the Net Tracer, you can simultaneously create a physical trace for the entire island, or choose to gradually build up the trace by generating the trace for a small set of shapes at a time. When you choose to gradually build up the trace, the trace that you generate is called a Step trace. Step tracing allows you to generate and evaluate the trace step-wise, helping verify that the nets that the trace runs through are well-connected.

The shape that you select for tracing defines the start point for the trace. Depending on the number of shapes to be traced in a design, the number of steps needed to complete a trace varies for each trace.

© 2023 Cadence Design Systems, Inc.

The figure below illustrates how a Step trace progressively chases the shapes in the design, tracing a set of shapes at each step.



The Net Tracer toolbar provides additional controls, as displayed in the figure below, to help you traverse back and forth along a Step trace.



The *Step No* value indicates the current step number and it applies uniformly to all the traces generated. For example, if the *Step No* value is set to `5`, all the traces are currently set to step `5`and the layout canvas displays the trace in accordance with the specified step value. If you are generating two traces with total steps of 15 and 10 each, the trace that is 10-steps long will need only five more steps to reach the finish point. The 15-step long trace, on the other hand, will need 10 more steps to reach the finish.

Use the *Prev Step* and *Next Step* arrow buttons () on the toolbar to go back and forth along the trace. The canvas displays the trace in accordance with the currently selected step value in the *Step No* field.

© 2023 Cadence Design Systems, Inc.

Use the *Continue* arrow button () to display the entire trace in one go. However, unlike the regular trace, the generated trace being a Step trace, you can use the step navigation buttons to traverse back and forth along the step trace.

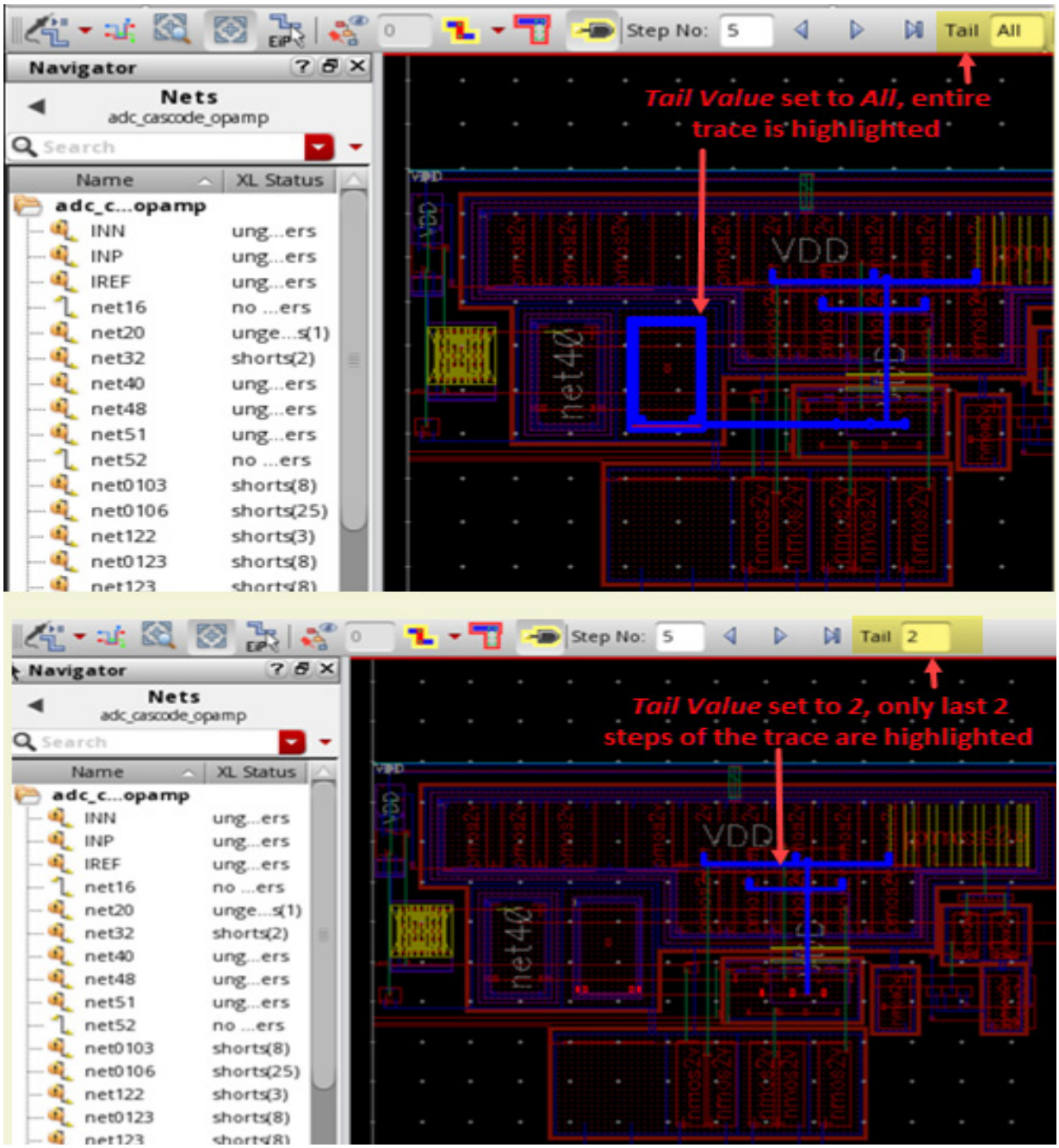
Use the *Tail Value* to choose what part of the generated trace is highlighted on the canvas. Choose *All* to highlight the entire trace. Use a numerical value to specify the number of steps to be highlighted on the canvas, starting from the last step.

Shapes of the step traces are visible according to the values set in the *Tail* and *Current Step* fields.

Preview always behaves as if *Tail* is set to *All*, which means that all the shapes are always visible up to the *Current Step* value.

© 2023 Cadence Design Systems, Inc.

The figure below illustrates how the display of a trace on the canvas varies based on the specified *Tail Value*.



To generate a Step trace:

1. In the Net Tracer toolbar, click the *Add/Remove trace* drop-down arrow and choose *Add/Remove step trace*.

© 2023 Cadence Design Systems, Inc.

2. In the layout canvas, click a shape to start tracing the net.

Depending on the shape that you click and the Step tracing option you select, a trace is generated in the layout covering the set of shapes that are chased in the first traversal. The trace can be progressed by clicking the associated *Prev Step* and *Next Step* buttons.

Related Topics

Load Traced Nets View Form

Net Tracer Toolbar

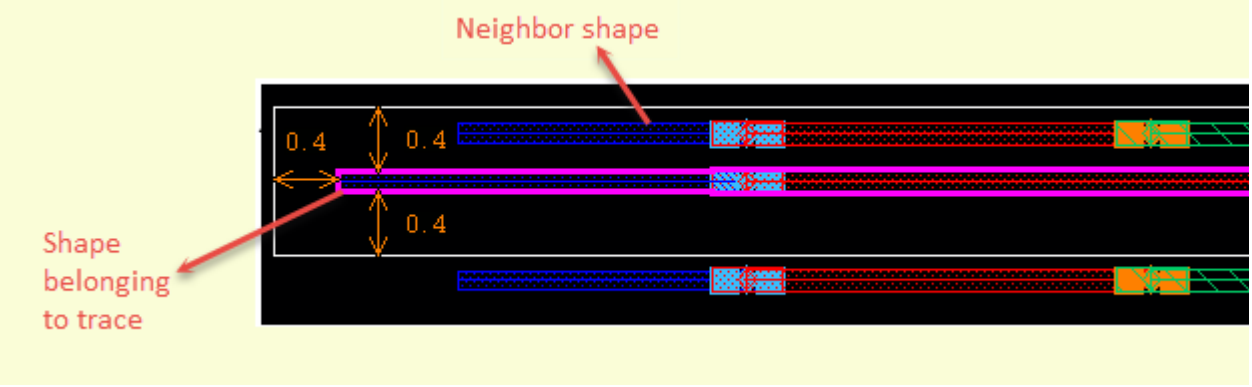
Loading Physical Traces

Tracing Nets

Finding Neighboring Shapes of a Traced Shape

A neighbor is a shape close to a shape belonging to a trace. The Net Tracer lets you find neighboring shapes on the same layer as the traced shape or on any extractable layer. Neighboring shapes need not exist on the same purpose as the traced shape. A neighbor can belong to a different hierarchical level or figGroup.

You can specify the distance from the traced shape within which the Net Tracer can find the neighboring shapes. For example, in the following figure, the shape belonging to a trace has a neighboring shape at `d = 0.4`.

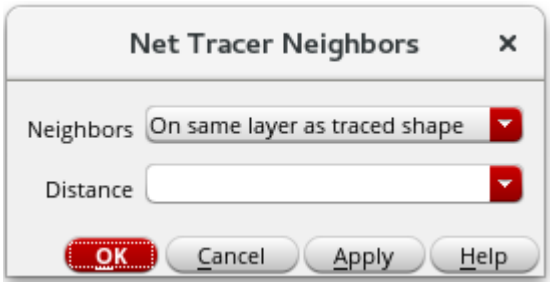


To find the neighboring shapes of a traced shape:

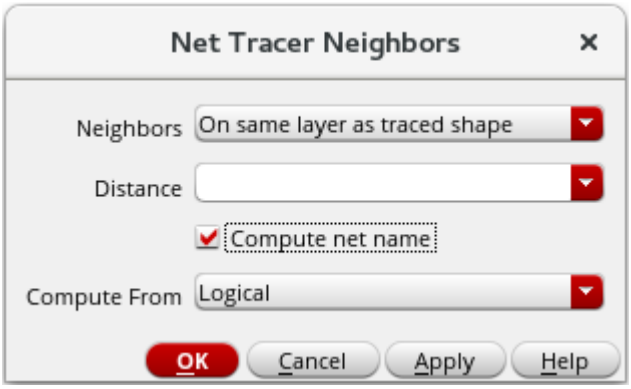
1. In the layout window, choose *Connectivity – Net Tracer – Neighbors.*

© 2023 Cadence Design Systems, Inc.

The Net Tracer Neighbors form opens.



2. From *Neighbors*, select whether you want the Net Tracer to find the neighboring shapes on the same layer as the traced shape or on any extractable layer. You can choose *On same layer as traced shape* or *On any extractable layer*.
3. In *Distance*, specify the distance from the traced shape within which shapes are to be considered as the neighboring shapes of the traced shape.
4. In Layout EXL or higher tier, specify whether the Net Tracer should compute net names for the neighboring shapes.
   a. Select the *Compute net name* check box.
   b. From *Compute From*, select the computation method as *Logical* or *Logical and Physical*.



5. Click *OK*.

   All neighboring shapes within the specified distance from the traced shape are highlighted, depending on whether you specified to find them on the same layer or on any extractable layer.

To delete all traces and their neighboring shapes, you can use the following ways:

- Choose *Connectivity – Net Tracer – Remove All.*

© 2023 Cadence Design Systems, Inc.

- Click the *Remove all traces* icon in the Trace Manager form.

Related Topics

Net Tracer Neighbors Form

netTracerNeighborsOnLayer

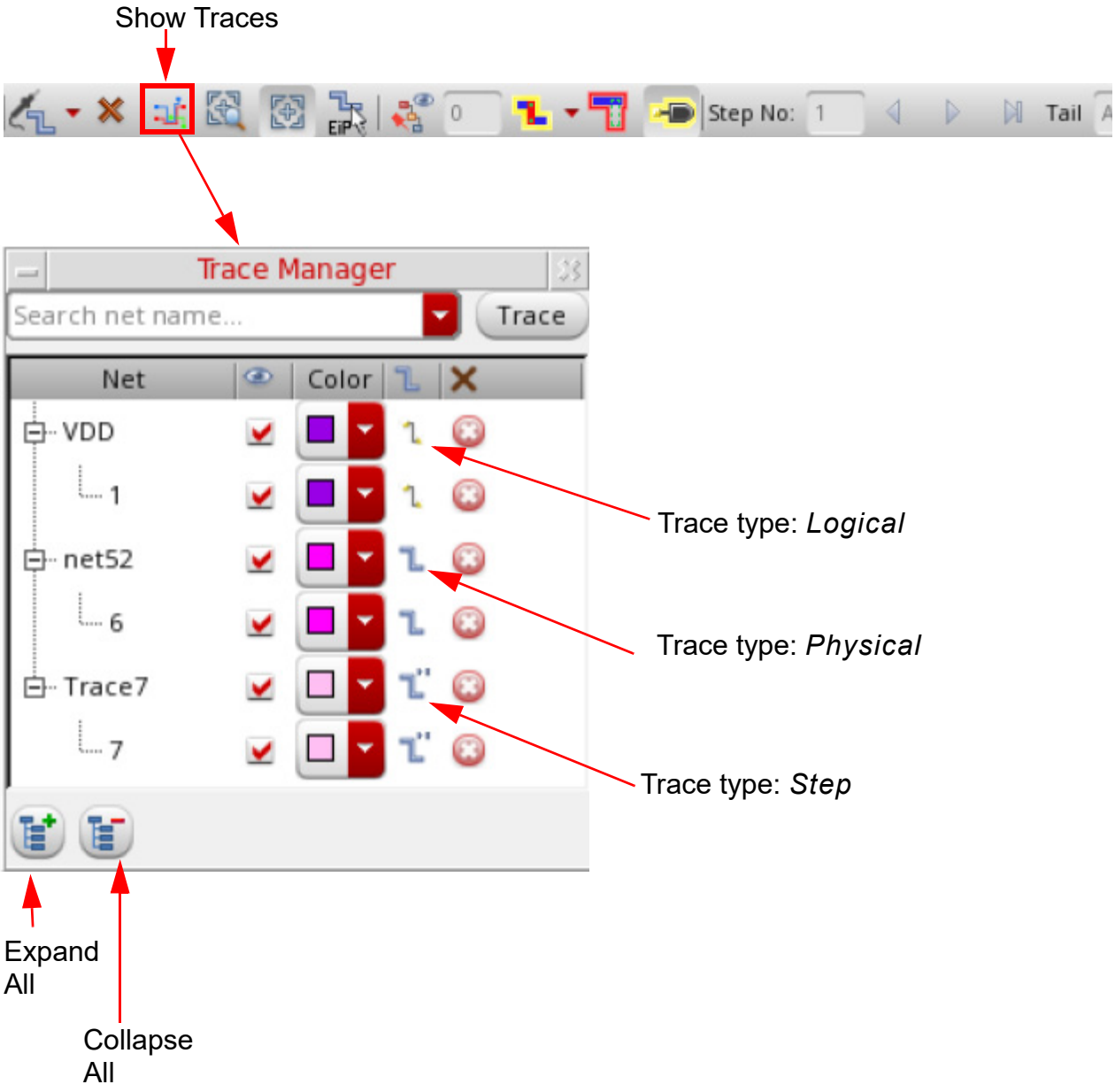
netTracerComputeNeighborNet

netTracerComputeNeighborNetMethod

© 2023 Cadence Design Systems, Inc.

Adding a Trace Using the Trace Manager

The Net Tracer toolbar provides a *Show Traces* option that opens the Trace Manager form. Use the Trace Manager form to create new traces on specific nets and to control the display of the generated traces.



© 2023 Cadence Design Systems, Inc.

To add a new trace using the Trace Manager:

1. In the *Search net name* field, type the name of the net for which the trace needs to be created.

   Alternatively, click the drop-down arrow associated with the field and select an available cellview net to which to add a trace.

2. Click the *Trace* button.

   The Net Tracer creates a trace for the specified net and displays the net name and the associated traced shapes in a tree structure in the Trace Manager window. The type of the trace generated—*physical*, *logical*, or *step*—is indicated by the icon displayed in the *Type* column.

3. (*Optional*) Use the *Visibility* icon adjacent to the generated trace to control whether or not the trace is displayed in the canvas.

   Click the column header to control the visibility of all the generated traces.

4. (*Optional*) Use the *Remove all traces* icon to remove a selected trace. To remove all the traces, click the column header.

Related Topics

Trace Manager Form

Net Tracer Toolbar

Creating a Step Trace

Tracing Nets

Editing In Place a Trace

The Net Tracer enables you to *Edit In Place* a traced object, allowing you to perform an interactive edit of the instance. This functionality enables you to focus only on the traced objects for running the *Edit In Place* instead of performing the operation on all the objects in the cellview.

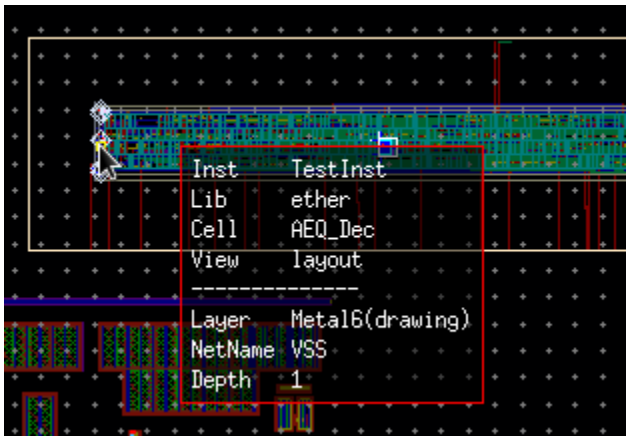
To run the Net Tracer *Edit In Place* command:

1. Add at least one trace to the cellview.
2. Click the *Net Tracer EIP* toolbar button and click the traced object to be edited in place.

© 2023 Cadence Design Systems, Inc.

The traced object is made available for editing in place and its corresponding connectivity reference opened.

Hovering the pointer over the selected object displays an information balloon, as displayed in the figure below.



To prevent the information balloon from opening:

1. Click the *Net Tracer EIP* toolbar button and press F3.
2. The Net Tracer Edit In Place dialog box displays.



3. Deselect the *Show info balloon* check box.

   Alternatively, you can use the `netTracerShowInfoBalloon` environment variable to control the display of the information balloon.

The EIP information balloon is not displayed the next time you hover the mouse on a traced object.

Related Topics

Editing In Place

Net tracer Features (video)

© 2023 Cadence Design Systems, Inc.

Tracing Nets

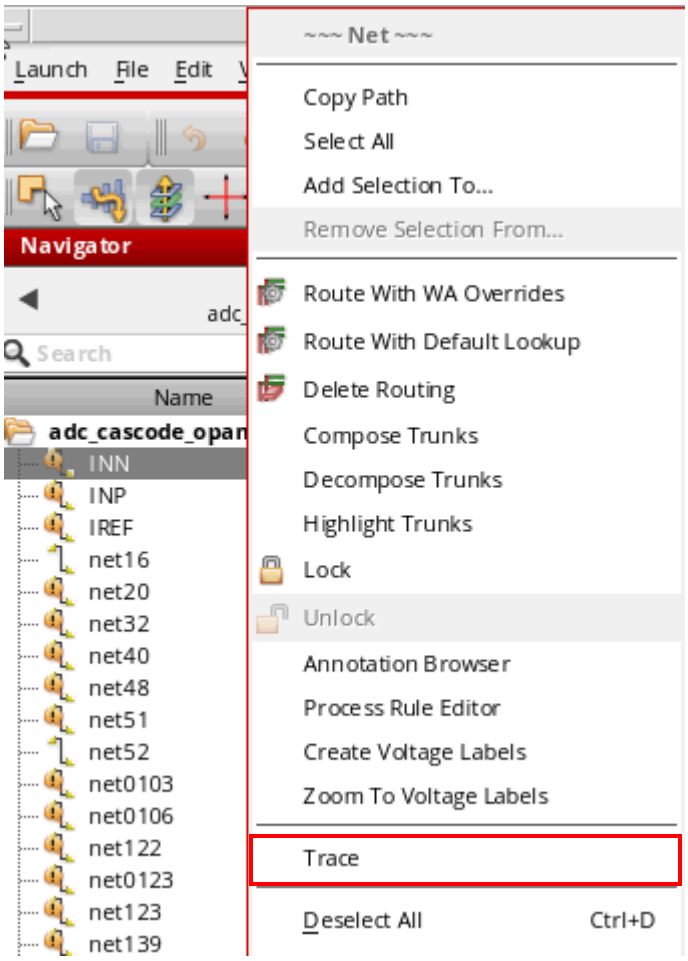
Tracing Nets using Navigator

You can trace the physical or logical connectivity of nets in a design by using the *Trace* command in the shortcut menu of the selected net.

To trace a net by using the *Trace* command in the Navigator assistant shortcut menu:

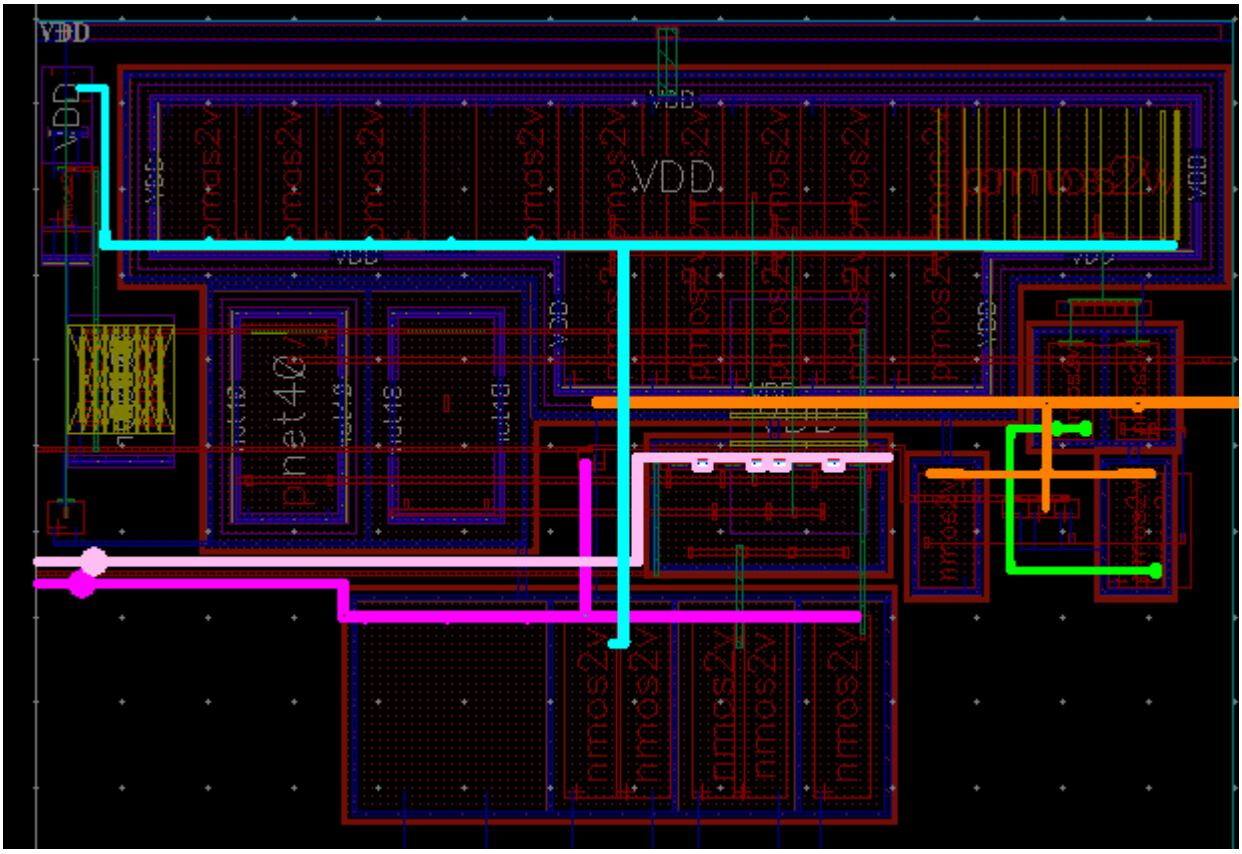
1. Right-click a net in the Navigator assistant and choose either from the shortcut menu—*Add Logical Trace* or *Add Physical Trace*.

   You can select multiple nets to simultaneously trace them.

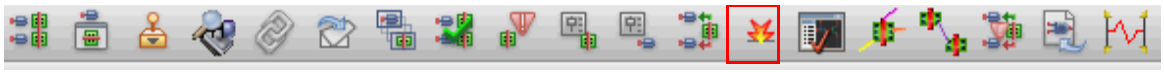


© 2023 Cadence Design Systems, Inc.

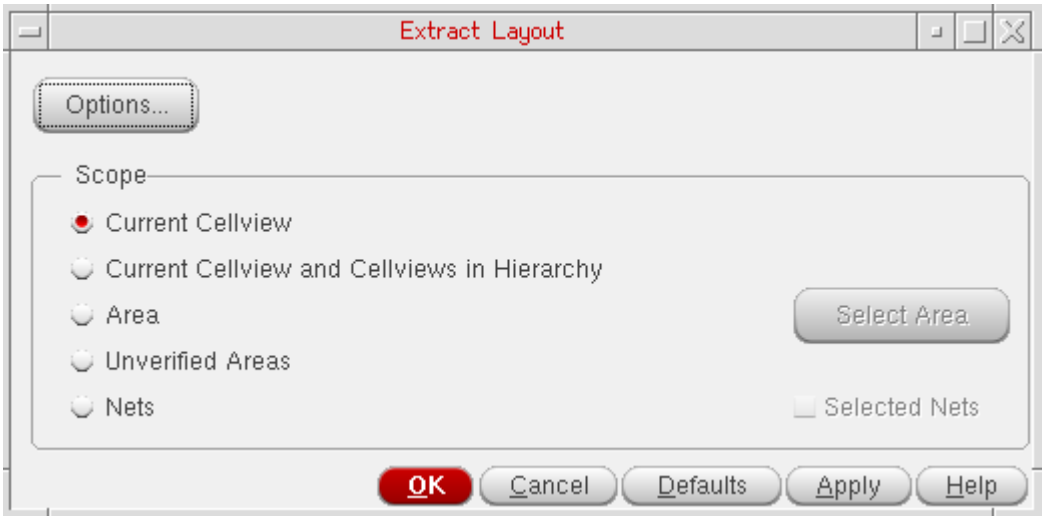
2. The selected net (or nets) are traced in the layout canvas, as shown in the figure below.


VDD
VDD
VDD

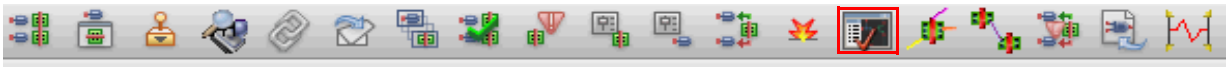
- From the layout window menu bar, choose *Connectivity – Update – Extract Layout*.
- Click the *Extract Layout* button on the Layout XL toolbar.



The *Extract Layout* form displays. Click *OK* to extract the design.



2. Click the *Annotation Browser* button on the Layout XL toolbar.



The *Connectivity* tab of the Annotation Browser is displayed.

The tab label indicates the total number of connectivity violations in the design. These are separated into different categories for *Illegal Layer Overlaps*, *Incomplete Nets*, and *Shorts* in the browser pane.

3. Click the *Set Highlight State* column for the entry whose flight line you want to see.
4. Click the *Set Highlight Color* column to set the color used to draw the corresponding marker in the design window. Choose *cycle* to let Layout XL select the color automatically by cycling through a predefined list.
5. Click the *Set Marker Check State* column to set the *Checked* state of the corresponding marker. When a marker is set to *Checked*, you can use the *Hide Checked Markers* button to hide it in the Annotation Browser without deleting the marker from the design window.

© 2023 Cadence Design Systems, Inc.

Clicking these columns next to the *Incomplete Nets* category shows all the incomplete net flight lines in the design.

Related Topics

Annotation Browser Pane

Annotation Browser Assistant

Checking Shorts, Incomplete Nets, and Illegal Overlaps

Connectivity Extraction Use Model

Locating Shorts Using Short Locator

Locating the Shapes Causing the Short

Locating Short with Large Number of Common Shapes

Locating Shorts Using Short Locator

A short is an overlap between two shapes that belong to different nets. To indicate shorts in the design, the Layout XL connectivity extractor creates short markers.

Although the extractor creates markers that indicate that shorts exist and which are the nets involved, the short markers may not indicate where the error is actually located. Manually locating the erroneous shapes can be difficult and time consuming. The *Short Locator* command can be used to locate the error that is causing a short.

Short Locator finds the pins and labels for the shorted nets and then computes the shortest paths between the shorted nets. The shapes that are common to these shortest paths are called *common shapes*. In most cases, the shape causing the short is one of the common shapes. But, if there are several errors causing a short, the errors may not be on the common shapes.

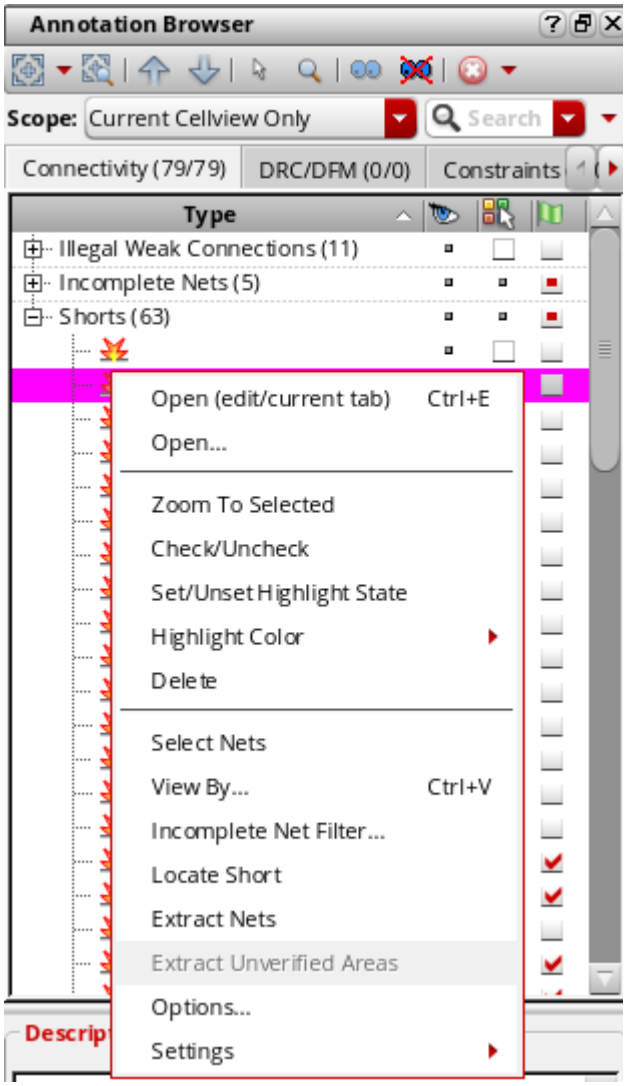
Short Locator can be launched from a short reported in the Annotation Browser assistant. Alternatively, it can be launched from the menu or the toolbar.

The Short Locator functionality requires a VLS GXL license, drawing three tokens.

To invoke Short Locator:

© 2023 Cadence Design Systems, Inc.

1. In the Annotation Browser, right-click a shorted net pair or a short marker and choose *Locate Short*.

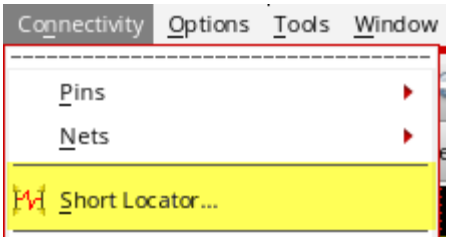


2. Click the *Short Locator* toolbar button.



© 2023 Cadence Design Systems, Inc.

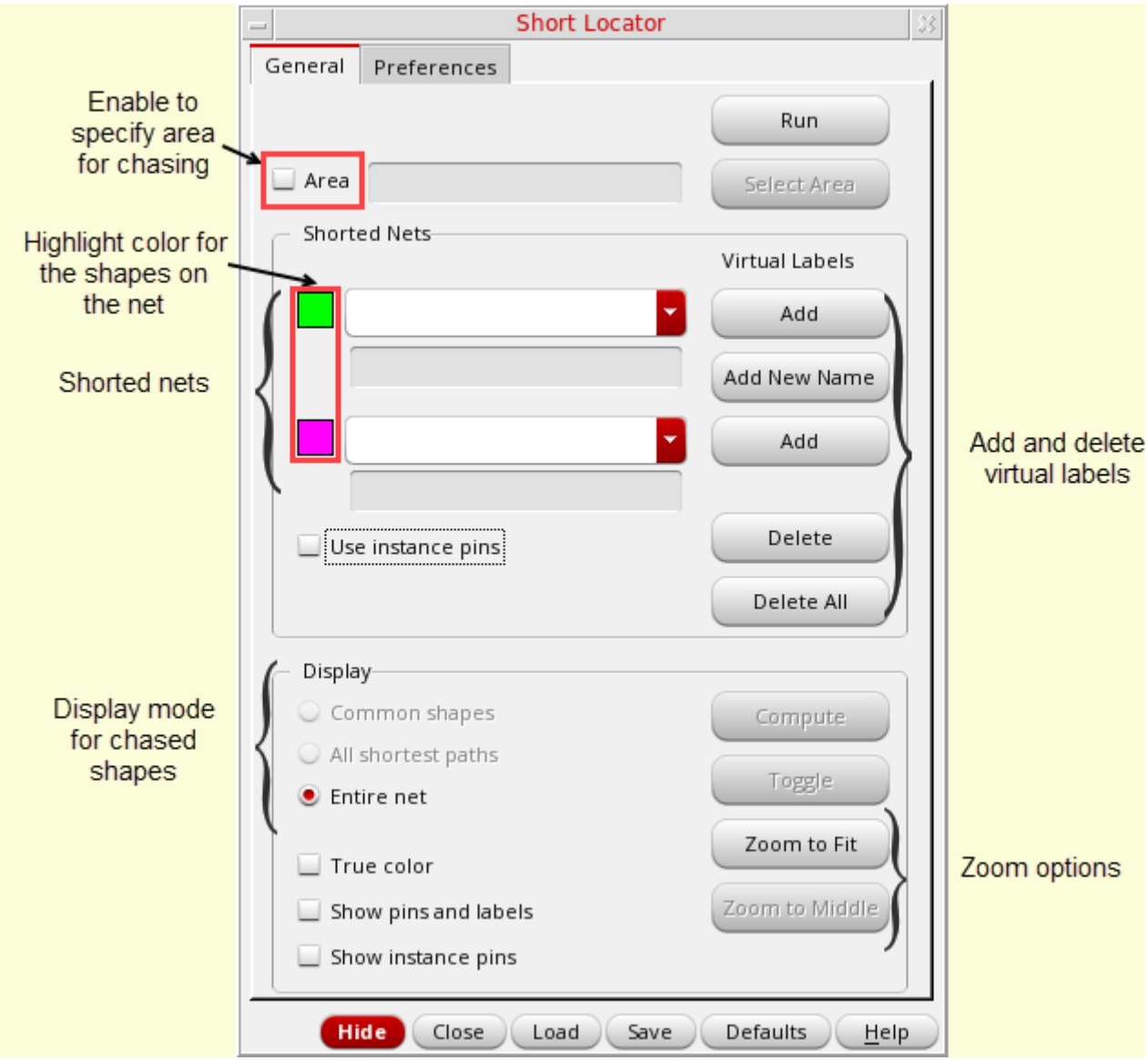
3. Choose the *Connectivity – Short Locator* command.



If a label or shape is selected in the canvas, Short Locator is run on the selected shape. If nothing is selected, the Short Locator form is opened empty. Click the *Run* button and select a shape.

© 2023 Cadence Design Systems, Inc.

The Short Locator form displays, indicating the shorted nets between which the short is located, as displayed in the figure below.



You can use the options in the form to choose the color for the different kind of shapes and specify the information to be displayed about the short. You can also use the form to add a virtual label for one of the shorted nets as you progress in the analysis of the short.

Depending on your design situation, the methodology that you use for locating the shapes causing the short can vary.

© 2023 Cadence Design Systems, Inc.

Related Topics

Locating the Shapes Causing the Short

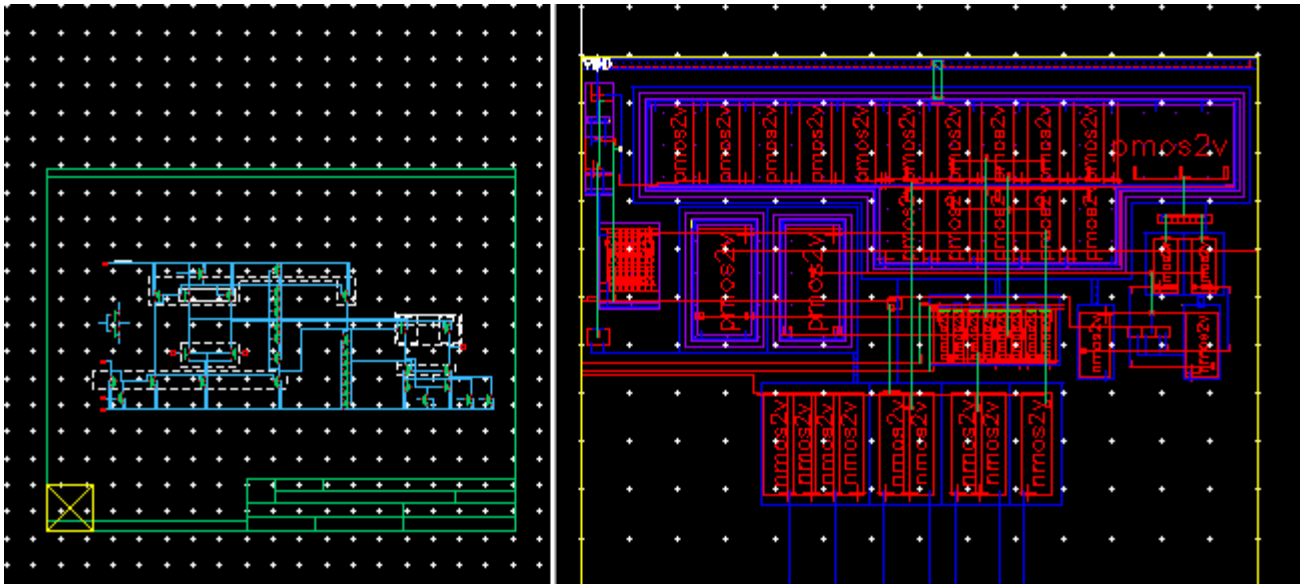
Locating Short with Large Number of Common Shapes

Short Locator Form

Locating the Shapes Causing the Short

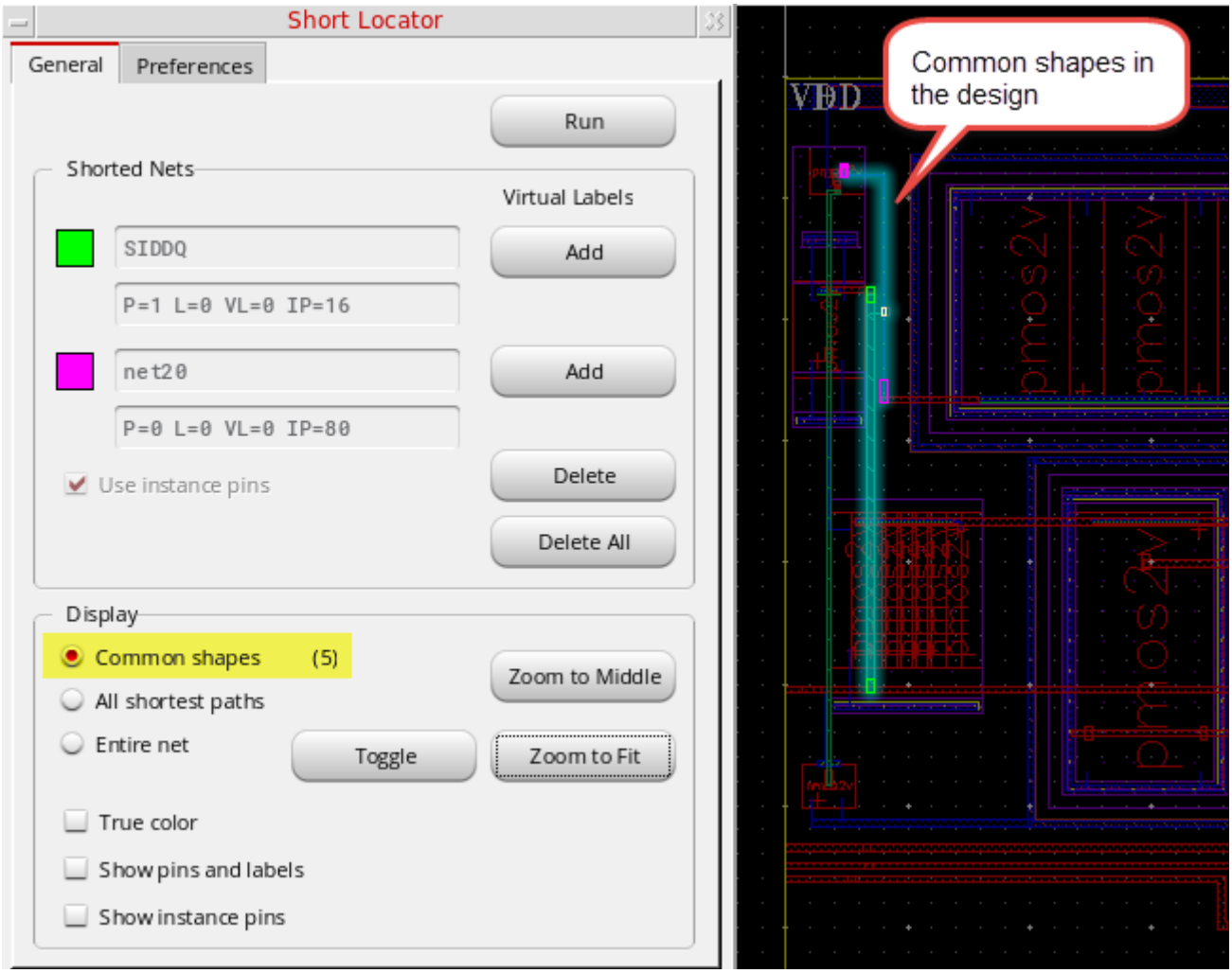
If the number of common shapes is small, you can evaluate the common shapes one by one, beginning the evaluation with a shape that has known connectivity.

Let us consider the design scenario as illustrated in the figure below.



© 2023 Cadence Design Systems, Inc.

Common shapes indicates that the design has a small number of common shapes (5), as shown in the figure below. Therefore, evaluating the shapes one by one to locate the short looks feasible.



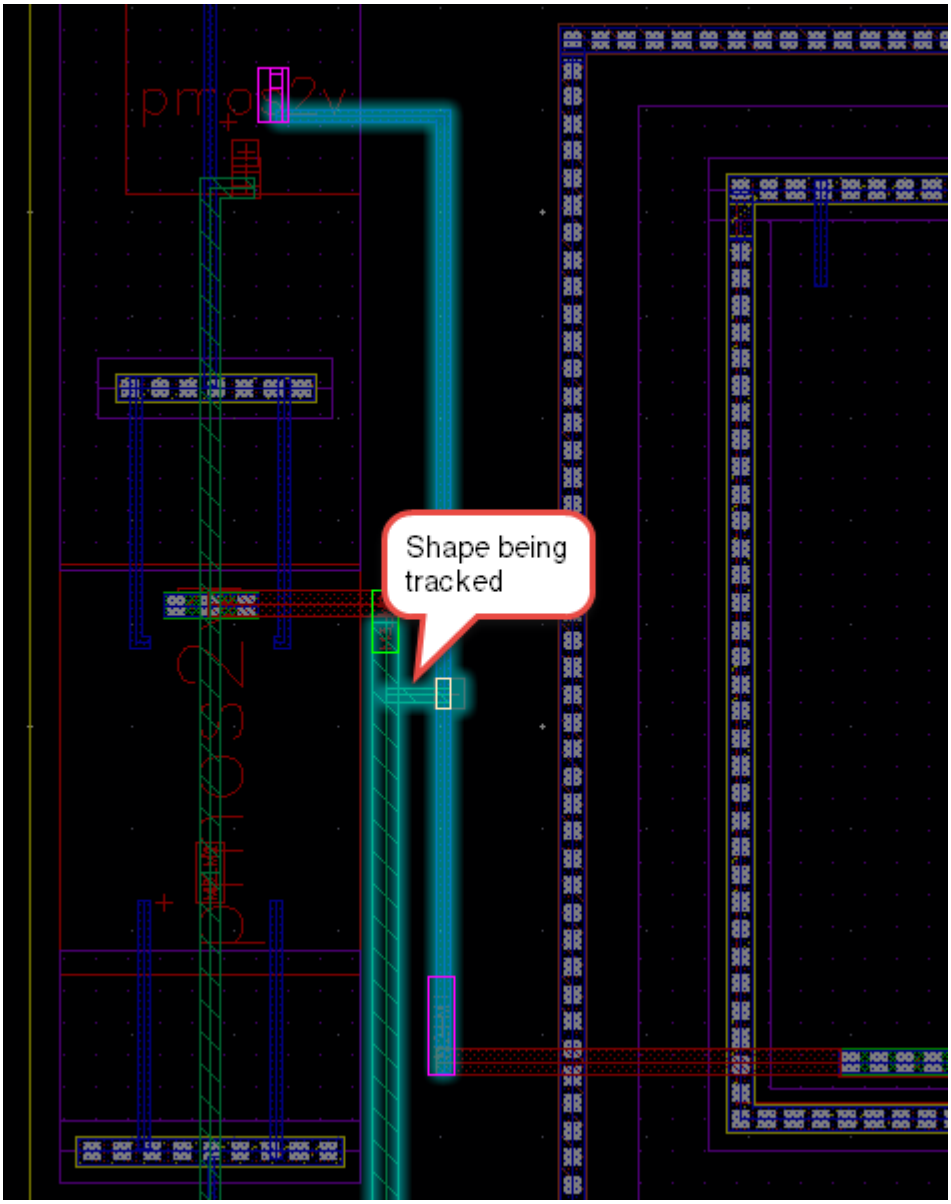
To locate the shape(s) in the design that are causing the short:

1. Because the number of common shapes in this design is small, look for common shapes connected to a known source of connectivity.

   For common shapes not directly connected to a known source of connectivity, look for the T-junction. Follow the T-junction to a known source of connectivity and add a virtual label on the corresponding net to the shape forming the T-junction.

© 2023 Cadence Design Systems, Inc.

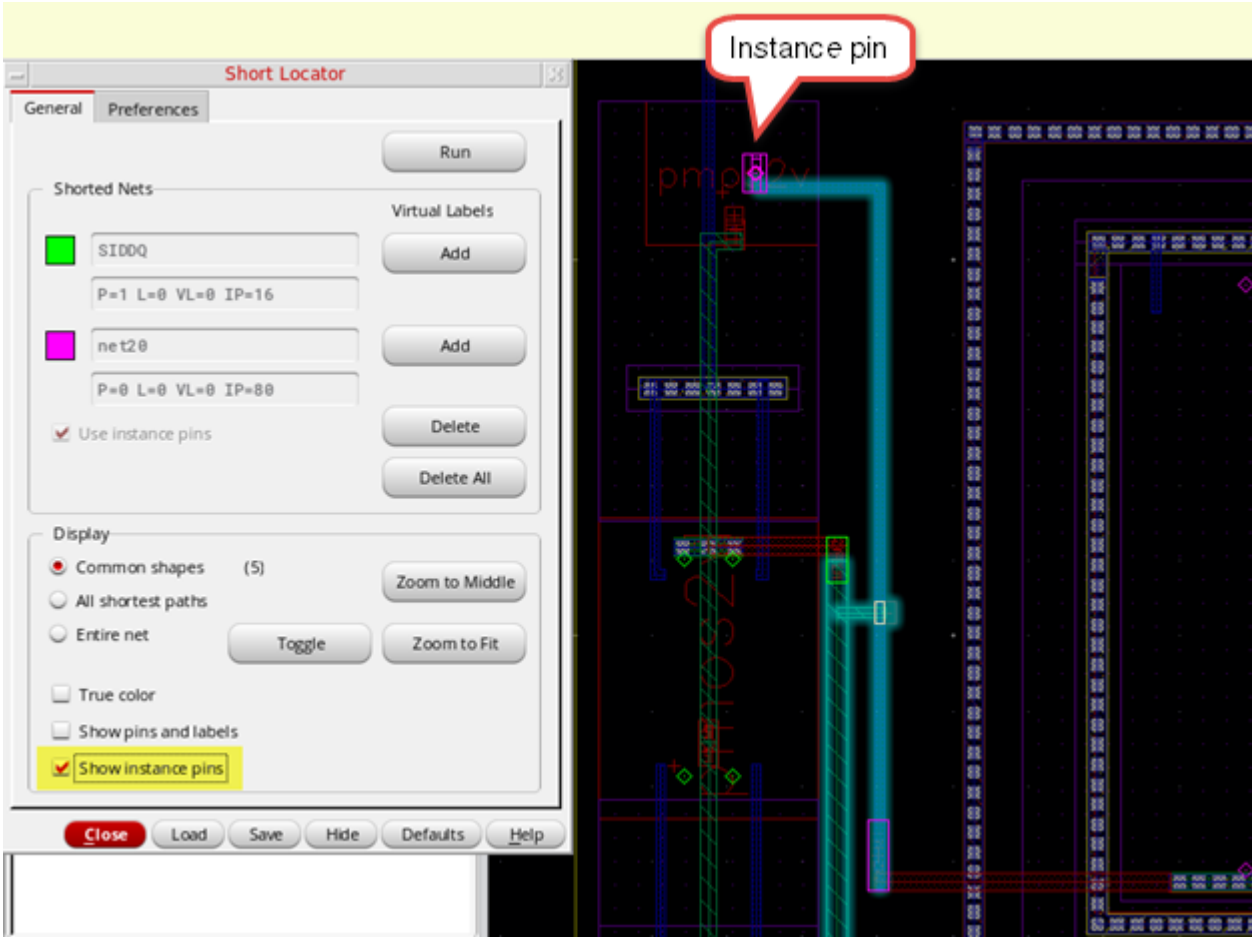
For the sample design, let us track the common shape that is highlighted in the design, till it connects to a known source of connectivity, which is the vertical path.



To view the entire path of the tracked shape, you can set the display mode to *Entire net.* This also highlights the shapes on the net that do not belong to the set of common shapes.

© 2023 Cadence Design Systems, Inc.

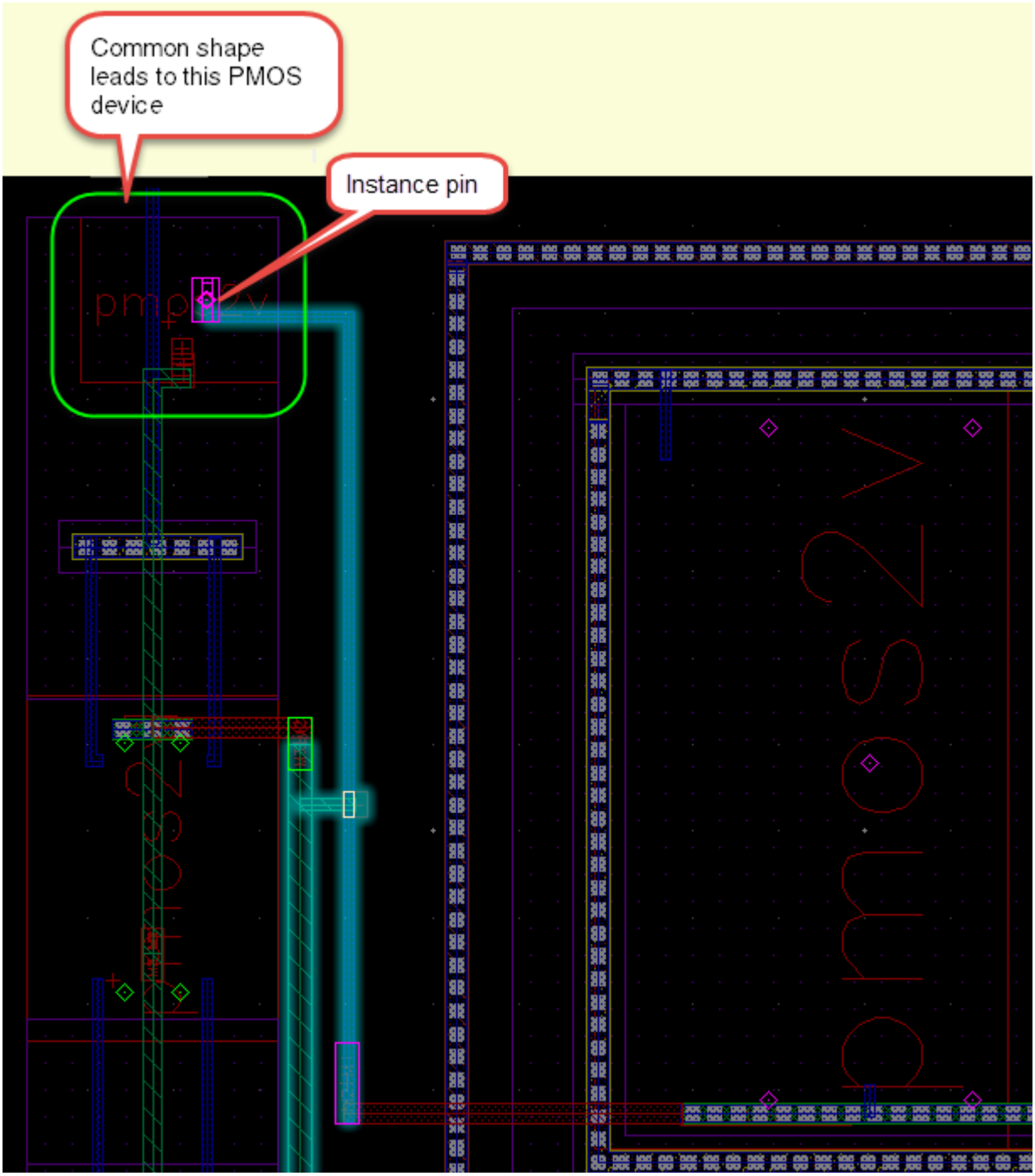
2. To see the instance pin that gives the selected shape its connectivity, select *Show instance pins* in the Short Locator form, as shown in the figure below.



3. Depending on the design, follow the T-junction or the source of known connectivity.

© 2023 Cadence Design Systems, Inc.

For the sample design, the common shape is connected to the terminal of a PMOS device, as displayed in the figure below.



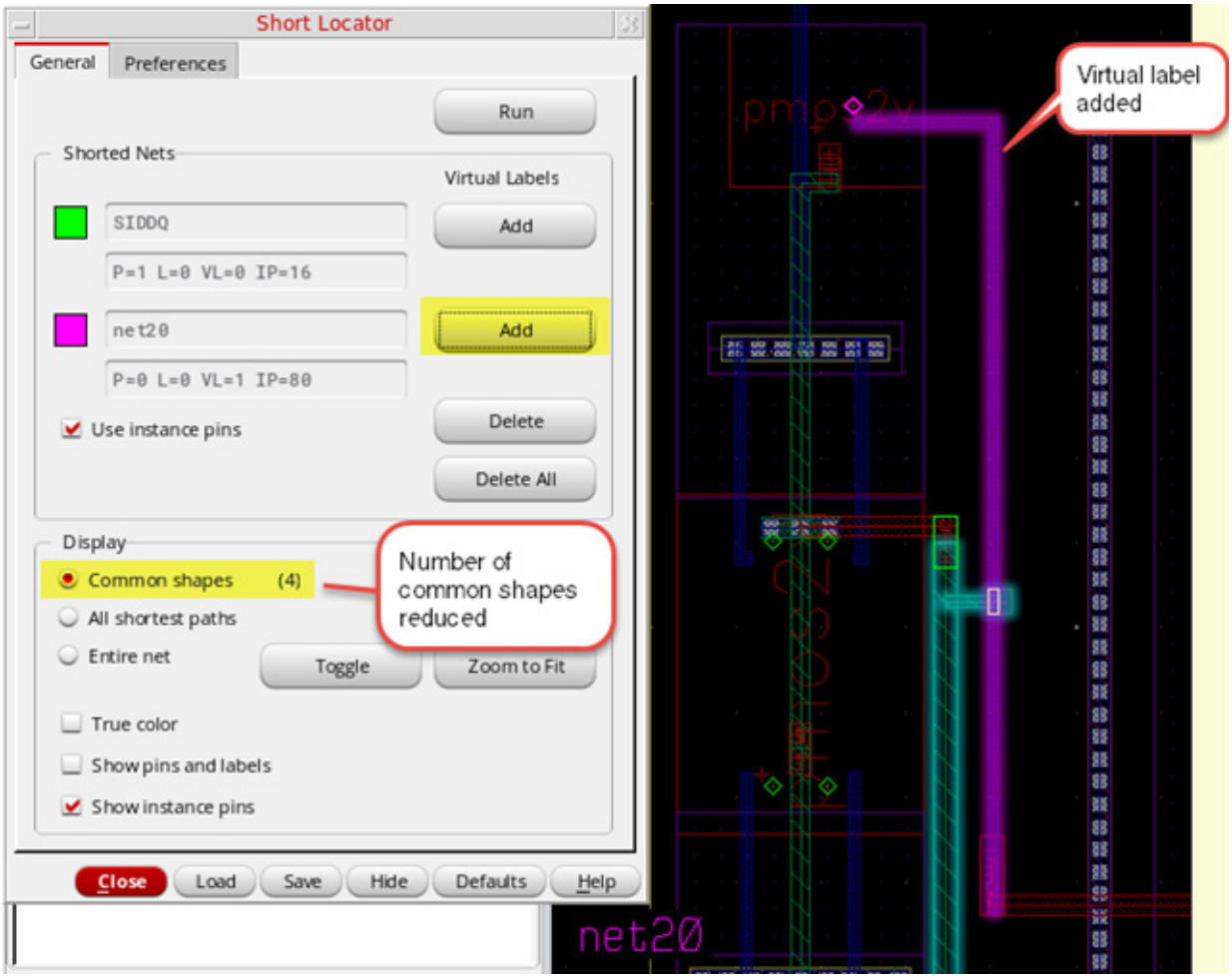
Checking the connectivity of the layout device using the *Check Against Source* command can confirm that the device is connected correctly. Alternatively, you can use

© 2023 Cadence Design Systems, Inc.

the Property Editor assistant to verify the connectivity of the PMOS device. Also, verify that the instance is connected to the same net in the schematic view.

4. After the connectivity of the device is verified, open the Short Locator form and add a virtual label on the net, `net20` for this example, to indicate that the connectivity of the common shape is known and has been verified.

   To add a virtual label to the shape, click the *Add* button for the corresponding net. Then, click the shape in the canvas to which the virtual label needs to be added.



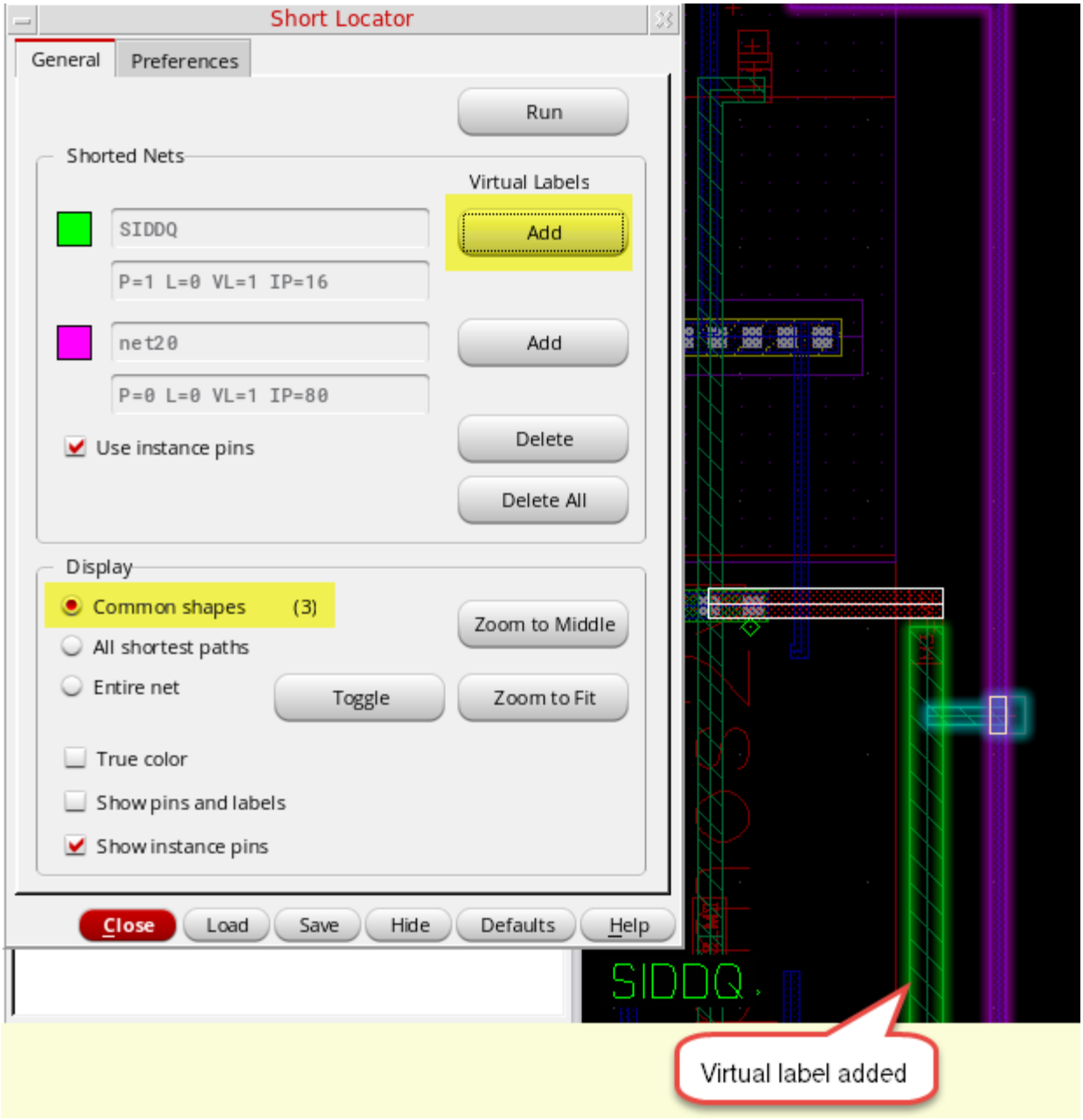
   If the same virtual label can be applied to multiple overlapping shapes, *Choose Figure to Create Virtual Label* dialog box pops up. You can use the dialog box to choose the shape to which the virtual label needs to be added.

   The number of common shapes is reduced due to the addition of the virtual label.

5. Look at the remaining common shapes and aim to zoom on the one in the middle, as shown in the figure below.

© 2023 Cadence Design Systems, Inc.

For the sample design, let us now track the connectivity of another common shape. We find that both the ends of the common shape are connected to `instTerms` of known connectivity. Therefore, we can add a virtual label to the shape, as displayed in the figure below.



Notice that the number of common shapes is reduced further.

6. Now continue to track the remaining common shapes, as described in the steps above. Repeat step 3 to step 5.

© 2023 Cadence Design Systems, Inc.

7. After the common shapes have been brought down to a few, the shape causing the short can be visually identified. To resolve the short, the shape can be moved, stretched, or deleted, as appropriate.

Short Locator automatically updates the form and the highlights as the design is modified.

Related Topics

Short Locator Form

Locating Shorts Using Short Locator

Locating Short with Large Number of Common Shapes

Locating Short with Large Number of Common Shapes

If the number of common shapes is large, evaluating the common shapes one by one may not be feasible. Instead, using the dichotomy approach can be more efficient. In such a design, virtual labels can be repeatedly set to the middle common shape, until the number of common shapes reduces down to the level when it is feasible to manually locate the error causing the short.

When the number of common shapes is large (hundred-thousand shapes), evaluating the shapes one by one is impractical. Therefore, we can begin the short location with adding a virtual label on the canvas at the middle of the common shapes. To do this:

1. In the Short Locator form, click the *Zoom to Middle* button to zoom to the shape at the middle of the common shapes.
2. Next, find the closest connected device, and add a virtual label to the shape, as appropriate.
3. To add a virtual label, click the *Add* button associated with the appropriate net. Then, click the shape in the canvas to which the virtual label needs to be added.

   If the same virtual label can be applied to multiple overlapping shapes, Choose Figure to Create Virtual Label dialog box pops up. Use the dialog box to choose the shape to which the virtual label needs to be added.

   The number of common shapes is reduced due to the addition of the virtual label. For designs that have a large number of common shapes, the addition of a virtual label can reduce the number of common shapes by half.
4. To further reduce the number of common shapes causing shorts, repeat step 1 and step 3 until you arrive at a small number of common shapes that can be manually tracked

© 2023 Cadence Design Systems, Inc.

to identify the error causing the short. The shape can then be stretched, moved, or deleted, to remove the short, as appropriate.

Related Topics

Locating Shorts Using Short Locator

Locating the Shapes Causing the Short

Design Check Against Source

The *Connectivity – Check – Against Source* command lets you check that:

- There are no missing or mismatched cellview masters between the schematic and the layout.
- There are no devices present in the schematic but missing in the layout and vice versa.

  The command does not report missing I/O pins and instances that have a property specified in the `propsUsedToIgnoreObjsForCheck` environment variable.

  See Ignoring Components with no Schematic Counterpart.
- There are no connectivity issues on top-level pins and global nets, mismatched or missing terminals and instance terminals, or unbound nets in the layout.
  - *Check Against Source* does not report the physical-only terminals (`physOnly=t`) created to support an implicit inherited connection, which is defined using a net expression in the schematic.
  - *Check Against Source* does not report any missing pins for terminals that have an `ignore` property set.
- The CDF parameters and cell and instance properties listed for devices in the schematic are associated with equivalent parameters or properties on devices in the layout; and that the values of parameters and properties on the layout and schematic views are consistent.

  You specify how the parameters and properties are compared on the *Parameters* tab of the Connectivity form. The default settings check only the parameter values and ignore missing parameters or properties in either the schematic or the layout. This reduces the number of messages in the Check Against Source report.

© 2023 Cadence Design Systems, Inc.

- The bus terminal information in the layout cellview matches the bus terminal information in the schematic cellview. This behavior is controlled by the `createImplicitBusTerminals` environment variable.
- The min and max voltages on a net are same in the schematic and the layout views.
- Check Against Source does not report any issues found in user-defined bindings created using the Define Device Correspondence form.
- When *Check Against Source* is run, the information about the schematic nets does not include any hierarchical information in the name.

Using CDF Callbacks with Check Against Source

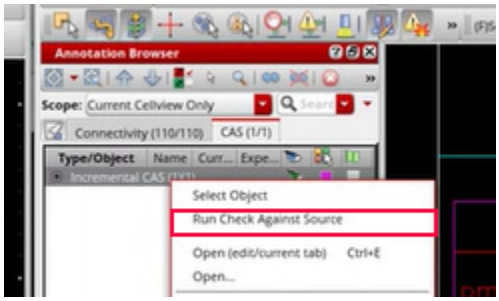
Check Against Source uses the schematic list of parameters, and the layout value of the `width` (for folding), `sParam` (for `sfactor`), and `mfactor`=1 and `sfactor`=1 to verify the list of parameters.

The total `widthsParam` is compared before applying the callback; all other parameters are compared with the corresponding master after applying the callback.

Incremental Check Against Source

You can use the `IncrementalCas` environment variable or click the  button on the *Options* toolbar to enable incremental Check Against Source. When this option is enabled, markers are interactively checked and created as changes occur.

By default,`100` objects are checked. If the limits exceeds, a marker is generated in Annotation Browser to report that the objects exceeding the limit have not been processed. The marker bounding box shows the objects that were not processed. You can run the *Run Check Against Source* command on the marker to check these objects later.



You can use the `incrementalCasLimit` environment variable to change the number of objects to process at one time.

© 2023 Cadence Design Systems, Inc.

Incremental Check Against Source does not run on non-modified objects, such as objects that are copied to the design.

Related Topics

Connectivity Form

Define Device Correspondence Form

createImplicitBusTerminals

incrementalCas

incrementalCasLimit

propsUsedToIgnoreObjsForCheck

Checking a Layout Against a Schematic

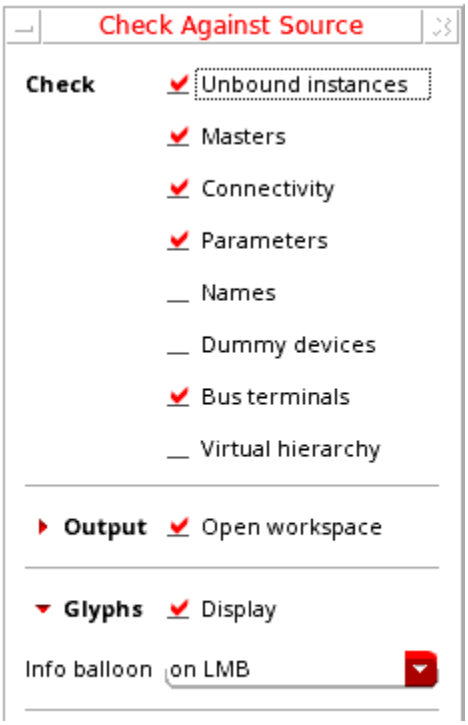
To check the components in your layout view against the schematic:

1. From the layout window menu bar, choose *Connectivity – Check – Against Source*.

   The *Check Against Source* command is also available through the *Check Against Source* icon () in the Layout XL toolbar.

© 2023 Cadence Design Systems, Inc.

The Check Against Source form is displayed.

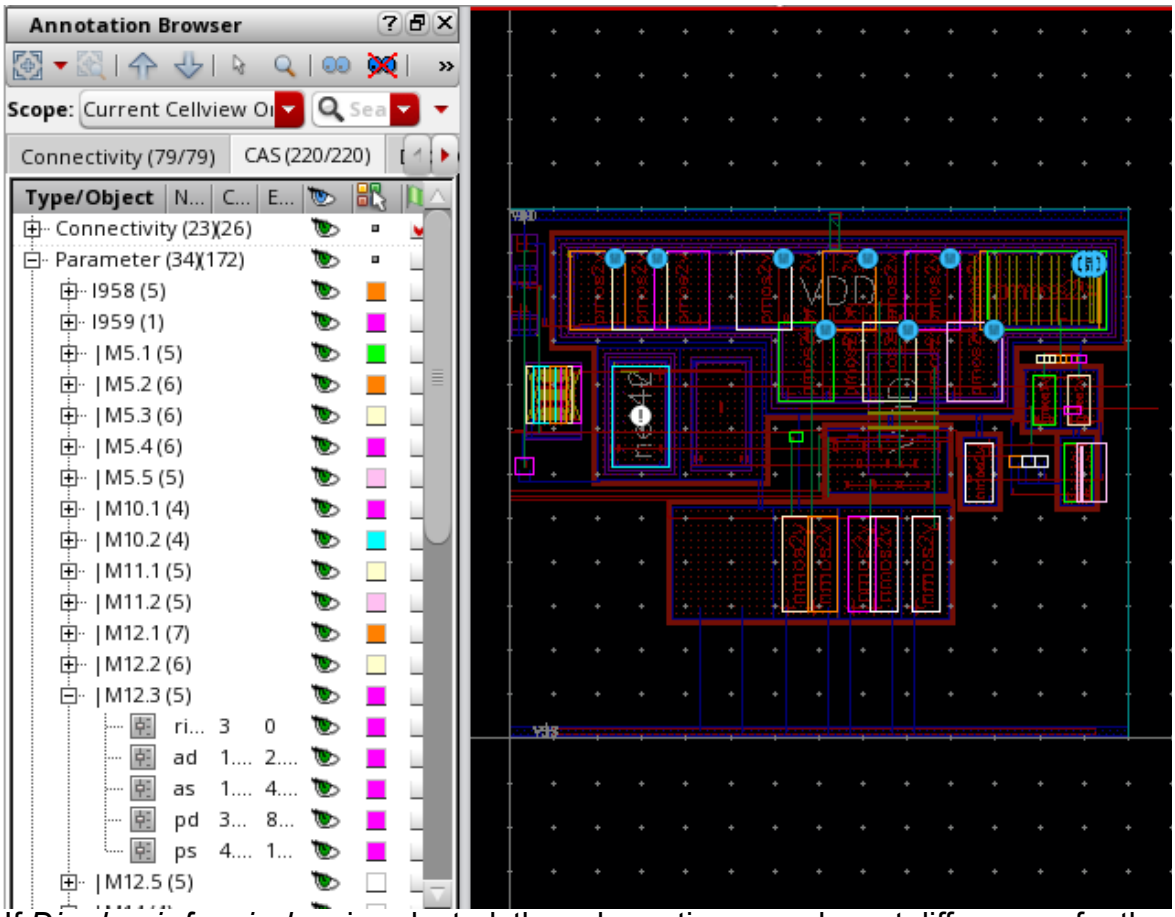


2. In the *Check* group box, choose the differences you want to report.
3. In the *Output* group box, choose whether you want to open the CAS workspace or the Info window, or both, and specify whether you want to overwrite the log file from a previous CAS run, or append the results to the log file.
4. Specify the name of the log file in which the report of the CAS run can be logged.
5. Click *OK* to run the check.

   The schematic is extracted and the CAS report is generated and displayed based on the *Output* options that you selected.

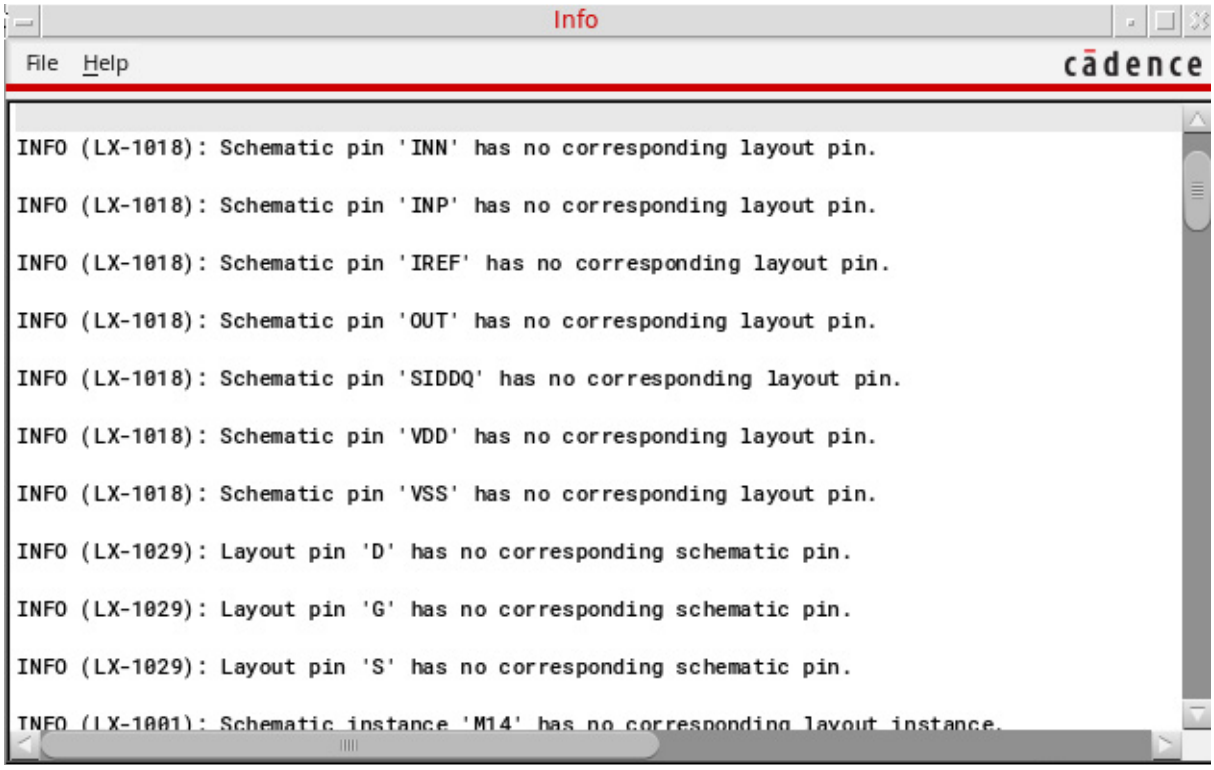
© 2023 Cadence Design Systems, Inc.

- If *Open workspace* is selected, Layout XL opens the *CAS* workspace to display the generated markers in the Annotation Browser *CAS* tab.



If *Display info window* is selected, the schematic versus layout differences for the selected checks are reported in an information window.

© 2023 Cadence Design Systems, Inc.



To save the report to an ASCII file, use the *File – Save As* command in the Info window.

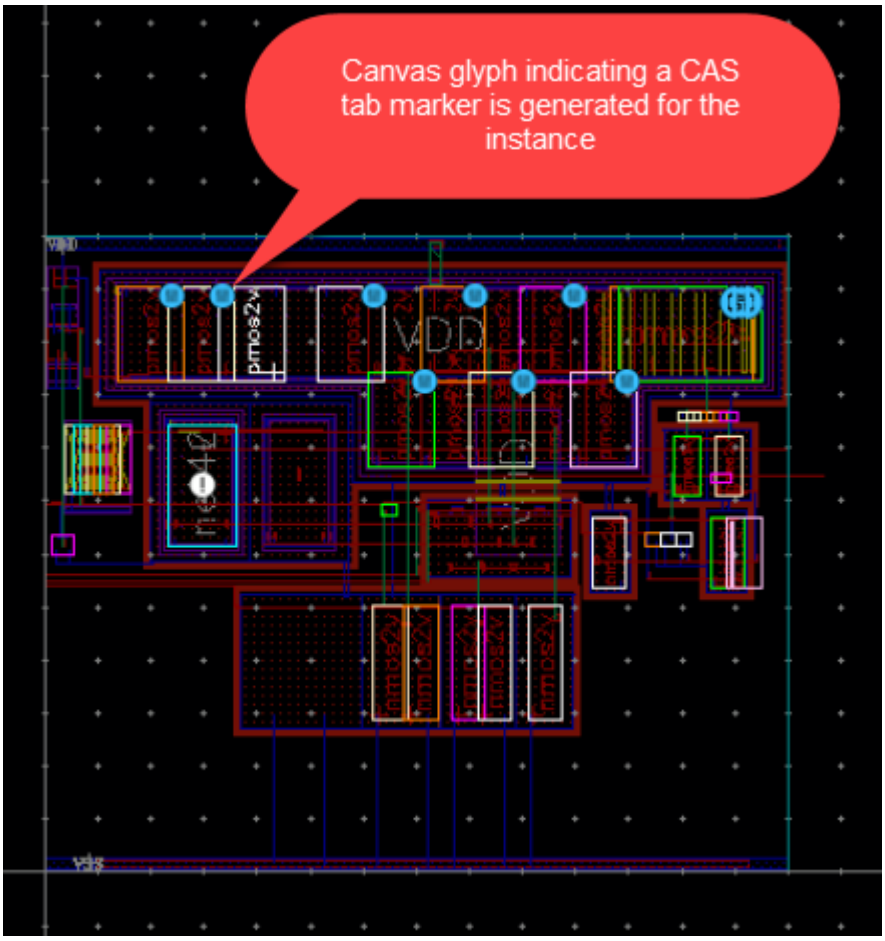
- If *Overwrite log file* is selected, the specified log file is cleared of all the information from the last CAS run and updated with the information from the new run.
- In the CAS workspace, each layout instance that has a marker generated in the CAS tab of the Annotation Browser also displays a canvas glyph that provides a visual cue to the type of mismatch it has with the source. See Canvas Glyphs to Find CAS Tab Markers.

Canvas Glyphs to Find CAS Tab Markers

In addition to generating unique icons in Annotation Browser to make it easy for the CAS tab markers to be spotted, Layout XL creates unique canvas glyphs to highlight the associated instance on the layout canvas. Click each generated glyph or hover the mouse pointer over

© 2023 Cadence Design Systems, Inc.

the glyph to see the associated information balloon, detailing the kind of mismatch and the current and expected values for the layout object.

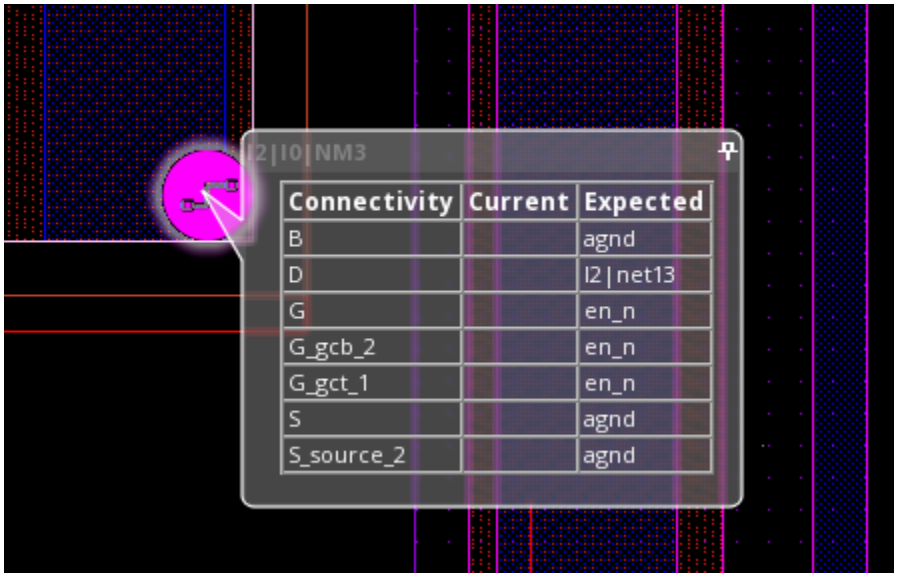


Depending on the type of marker generated for the instance, the canvas glyph representation on the canvas varies. Here are a few examples:

© 2023 Cadence Design Systems, Inc.

Example 1: *Connectivity mismatch glyph*

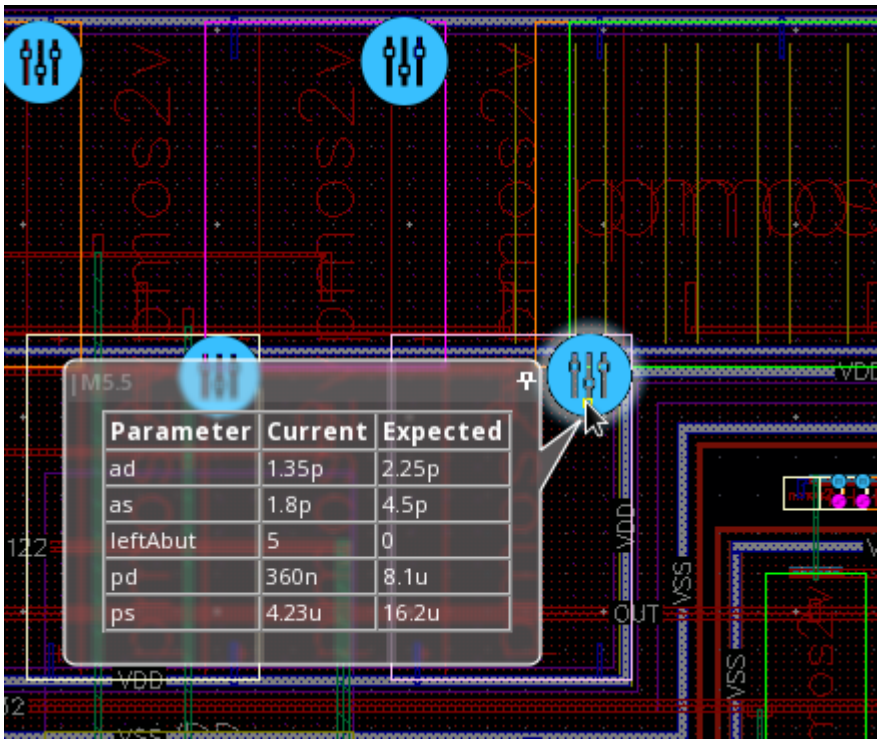
This glyph indicates connectivity mismatch on top-level pins and global nets, mismatched or missing terminals and instance terminals, and unbound nets in the layout. For more information about the mismatch, see the Check Against Source form.



© 2023 Cadence Design Systems, Inc.

Example 2: *Parameter mismatch glyph*

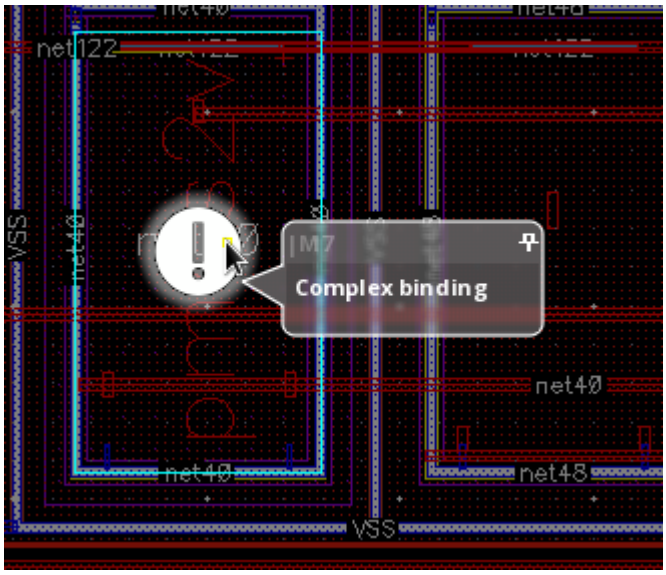
This glyph indicates issues with CDF parameters (and properties) set on the schematic and the layout views. For more information about the mismatch, see the Check Against Source form.



© 2023 Cadence Design Systems, Inc.

Example 3: *Binding mismatch glyph*

This glyph indicates layout instances that are not bound to the schematic instances, and the schematic instances that are not yet generated in the layout. For more information about the mismatch, see the Check Against Source form.



To help analyze the markers on the canvas and to help editing the layout, Layout XL enables you to pin the information balloons associated with the markers. This allows for the information related to specific markers to be available on the canvas while you explore the markers for other related objects.

Related Topics

Checking a Layout Against a Schematic

Pinning a CAS Glyph Information Balloon

Pinning a CAS Glyph Information Balloon

To pin a CAS glyph information balloon:

1. Click the required CAS glyph generated on the layout canvas.

   The related information balloon appears, displaying a *Pin* icon ( )on the top right corner.
2. Click the *Pin* icon to replace it with the *Delete* icon ( ).

© 2023 Cadence Design Systems, Inc.

The information balloon is now ready to be moved around on the layout canvas and placed appropriately.

3. Drag the information balloon and drop it at an appropriate location on the layout canvas.

   Analyze the generated markers and resolve them using appropriate commands from the shortcut menu. When the markers are resolved and the balloon information is no longer needed on the layout canvas, delete the pinned information balloon.

4. Click the *Delete* icon () on the information balloon to delete the pinned balloon.

Related Topics

Fixing CAS Tab Markers

Checking a Layout Against a Schematic

Pinning a CAS Glyph Information Balloon

CAS TAB Marker Categories

The table below lists the various marker categories under which the schematic versus layout differences are reported in the CAS tab. The table also displays the icon that identifies each marker category, and the commands available in the shortcut menu to resolve each marker category.

Marker Category	Icon	To resolve, choose...
Bus Terminals		*Update All Bus Terminals*
Connectivity		*Update Connectivity*
Master		*Update Master*
		Update Physical Bindings
Net Expression		*Update Terminal Net Expressions*

© 2023 Cadence Design Systems, Inc.

Parameter		*Update Selected Layout Parameters* *Update Selected Schematic Parameters* *Update All Layout Parameters* *Update All Schematic Parameters* When the allowSelectedParamUpdate environment variable is set to `nil`, the following commands are not displayed in the shortcut menu of the reported instance: ■ *Update Selected Layout Parameters* ■ *Update Selected Schematic Parameters*
Pin Direction		*Update Pin Direction*
Power Domain		*Update Net Power Domain*
Signal Type		*Update Net Signal Type*
Unbound		*Bind* *Ignore Instance* *Delete Object*
Unchecked		- NA-
Ungenerated		*Bind* *Generate* *Create Pins From Labels* The *Create Pins From Labels* command allows creating any missing layout pins using the existing pin labels in the layout.

© 2023 Cadence Design Systems, Inc.

Voltage		*Backannotate Net MinV/MaxV* A mismatch in min and max voltage values is reported using a Pin glyph, which also reports mismatches in pin direction, signal type, and power domain.
Name		*Update Names*
Dummy Devices		*Backannotate Dummies*

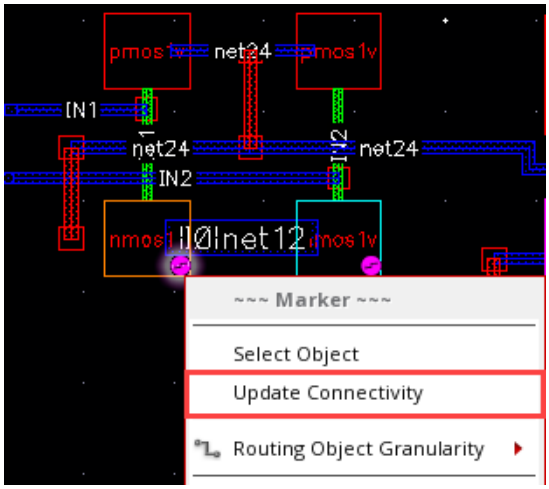
Fixing CAS Mismatches

➡ To interactively fix CAS mismatches, right-click the marker entry in the Annotation Browser CAS tab or the associated glyph in the layout canvas and choose the appropriate *Update* command from the shortcut menu:

- *Update Connectivity* to fix a glyph or marker that highlights connectivity mismatch issues.
- *Update Layout Parameters* to update the layout instance and resolve parameter mismatch issues.
- *Update Schematic Parameters* to update the schematic instance and resolve parameter mismatch issues.
- *Update Master* to update the instance master to match the physical binding in the physical configuration view.
- *Update Physical Binding* to update the physical binding to match the instance master in the layout view.
- *Update Virtual Hierarchy* to resolve mismatches between the virtual hierarchy in the layout and schematic views.
- *Update Net Min/Max Voltage* to fix a mismatch in minimum and maximum voltage values.
- *Update Name* to fix a mismatch between the schematic and the layout instance and net names.

© 2023 Cadence Design Systems, Inc.

Depending on the command you choose, the marker will be resolved, or you will be guided to the next appropriate form, command, or action.



Fixing CAS Tab Markers

To interactively fix the CAS tab markers:

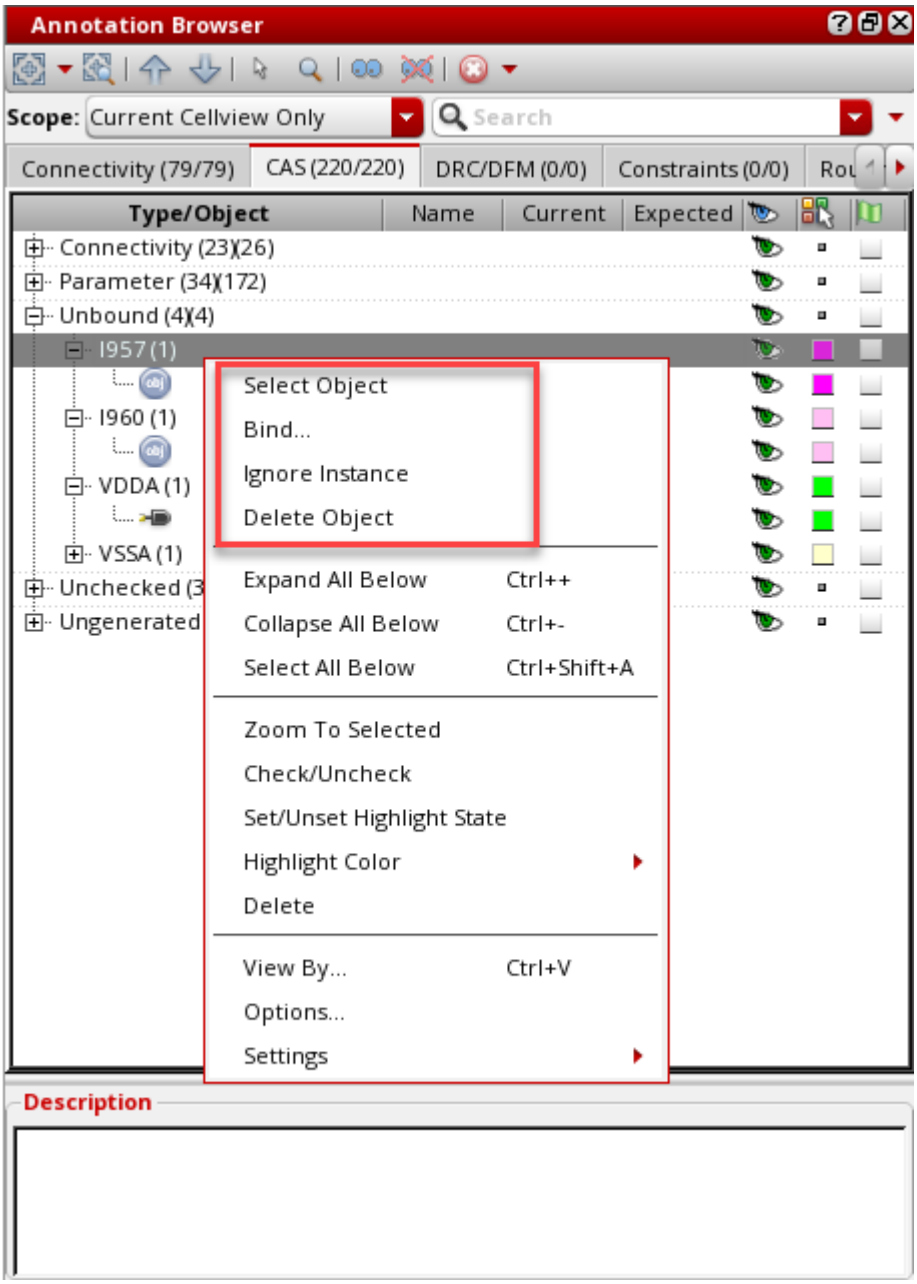
➡ Right-click a marker in the Annotation Browser or the layout canvas and choose the appropriate command that is available in the shortcut menu for the marker.

For example, if a marker represents an instance as *Unbound*, you can choose from the following commands in the shortcut menu to resolve the marker:

- *Bind*
- *Ignore Instance*

© 2023 Cadence Design Systems, Inc.

- *Delete Object*



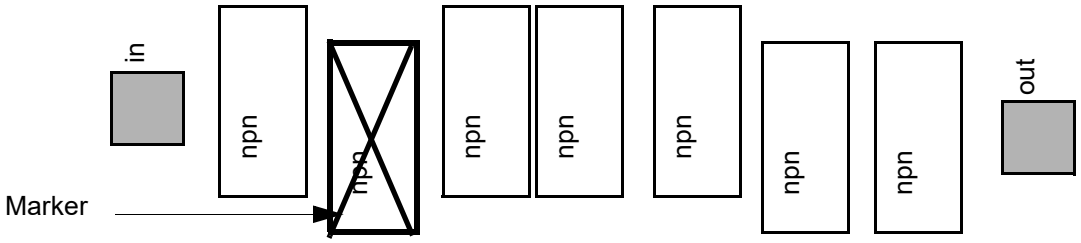
Depending on the command you choose from the shortcut menu for the marker, the marker will either be resolved, or you will be guided to the next appropriate form, command, or action.

To simultaneously resolve multiple markers within a marker category, right-click the marker category and choose the appropriate command from the shortcut menu.

© 2023 Cadence Design Systems, Inc.

Changes in the Layout Window

If a device that is not in the schematic is present in the layout, it appears with a blinking marker in the layout canvas.



If a device parameter in the layout is different from the corresponding parameter in the schematic, a text window appears and lists the device names and properties (unless the device is ignored).

The command also reports shapes that are on redundant or unmatched nets which no longer exist in the schematic, but it does not put markers on these shapes in the layout canvas. You can check the shapes and delete them manually as required.

Related Topics

Check Against Source Form

Annotation Browser Tabs

Annotation Browser Assistant

Suppressing the Check Against Source Form

➡ To run *Check Against Source* without seeing the Check Against Source form, do the following.

Type the following command in the CIW:

```
envSetVal("layoutXL" "disableCASOptionsPopUp" 'boolean "t")
```

To see the Check Against Source form the next time you run the command:

➤ Type the following command in the CIW.

```
envSetVal("layoutXL" "disableCASOptionsPopUp" 'boolean "nil")
```


© 2023 Cadence Design Systems, Inc.

Related Topics

disableCASOptionsPopUp

Ignoring Components with no Schematic Counterpart

If your layout contains components (for example, template cells or dummy devices) with no schematic counterpart, you can have *Check Against Source* ignore them using the *Properties used to ignore objects during check option* on the Layout XL Parameters Options form. To do this:

1. Add a user-defined boolean property (for example, `ignoreDummyDevice` set to `t`) to the layout instance master, or selectively to each layout instance, to be ignored.

   See Setting Layout XL Properties.

   If you add the property to the master, each instance of that master that you create will be ignored by *Check Against Source*.

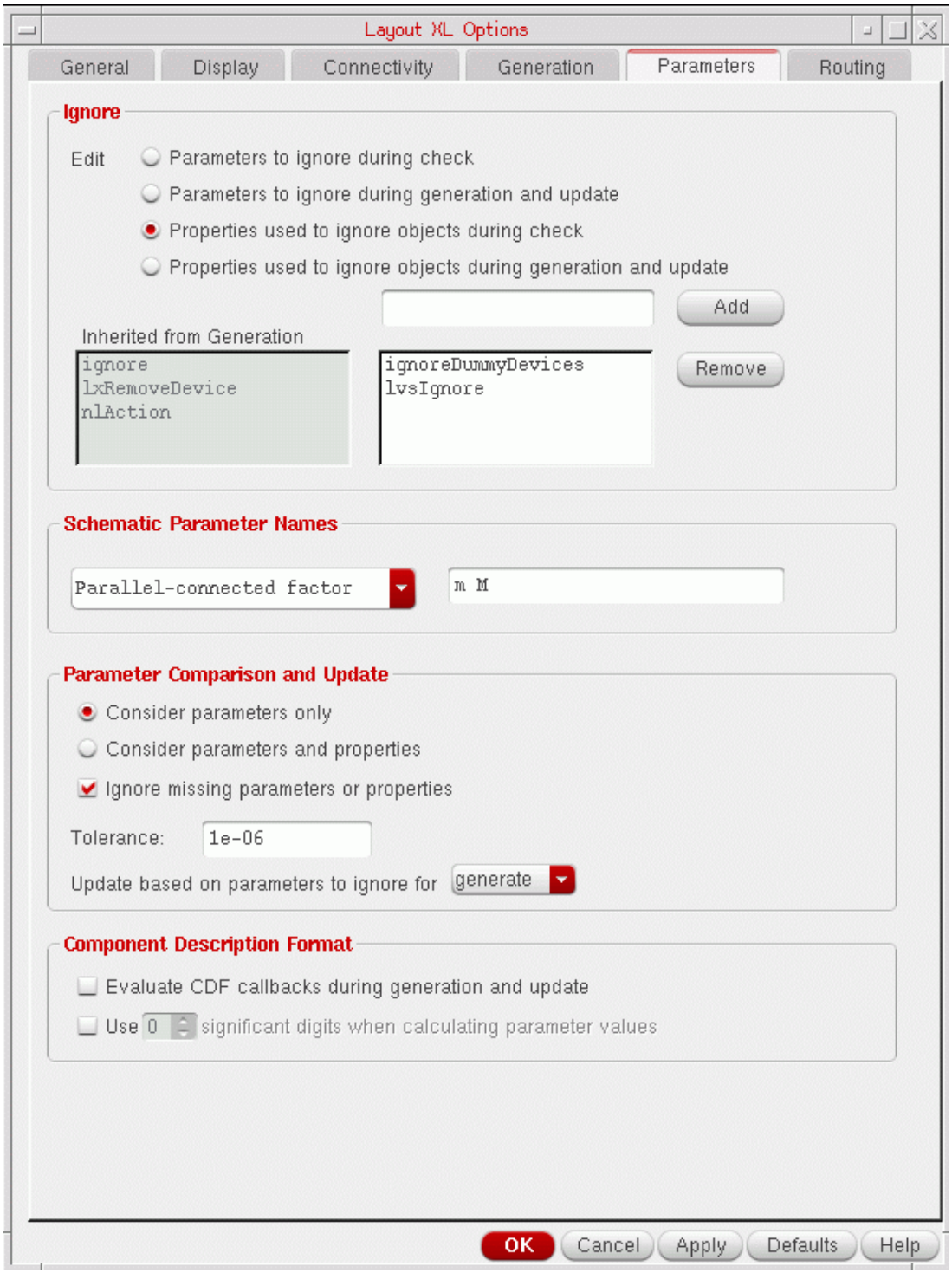
2. From the layout window menu bar, choose *Options – Connectivity* and choose the *Parameters* tab on the Connectivity form.

3. Check the radio button marked *Properties used to ignore objects during check*.

   Environment Variable: propsUsedToIgnoreObjsForCheck

© 2023 Cadence Design Systems, Inc.

4. Type in the property name string (for example, `ignoreDummyDevices`) in the field next to the *Add* button, and click *Add* to add the property to the list of ignored properties.



5. Click *OK* in the Connectivity form.

© 2023 Cadence Design Systems, Inc.

The next time you run *Check Against Source*, cellviews with the specified property will not be reported as mismatches.

Connectivity is traced as normal; that is, the pins on the dummy device are propagated to the top level and assume correct connectivity.

Related Topics

ignoreTermsWithNonRoutingLabels

Check Against Source Form

Connectivity Form

Annotation Browser Assistant

Design Check for Layout XL Compliance

To check if your design fulfills the compatibility criteria that allow it to fully leverage the connectivity-driven features of Virtuoso Layout Suite XL, run the *Connectivity – Check – XL Compliance* command.

Alternatively, you can call the lxCheck SKILL function to verify XL-compliance of a layout. To check each unique layout master in the layout hierarchy, use the lxHierCheck SKILL function.

The XL Compliance check evaluates the design for device correspondence with the schematic and reports information about ungenerated and unbound devices, if any. The report is intended to help you resolve any XL-compliance issues beforehand so that you can take full advantage of the numerous connectivity-driven capabilities provided by Layout XL for optimal layout generation.

The XL Compliance report is issued in CIW and you can choose to also display the report in HTML, if the xlComplianceHtmlOpen environment variable is set in accordance.

You can use the following environment variables to control the display of messages issued after the compliance check.

- checkCompType
- checkPermute
- checkRemoveDevice
- ignoreTermsWithNonRoutingLabels

© 2023 Cadence Design Systems, Inc.

You can also choose to save the logs generated during an XL-compliance check. Use the xlComplianceLogs and xlComplianceLogDir environment variables to control if the generated logs are saved and where.

During an XL-compliance check, the layout is checked for:

- Any labels or text displays in the layout that match the name of an unbound schematic terminal, indicating a possible missing pin in the layout.

  Use the *Tools – Create Pins From Labels* command to generate the pins from existing labels, or manually add the pins using the *Create – Pin* command.

- Possible missing permute rules. In addition to checking the layout, the schematic is also checked for any possible missing permute rules.

  - For a device to be considered for a missing permute rule check, the device must have either:

    - Two terminals and an optional bulk terminal, which is usually identified by the name `b` or `B`
    - Component type set to `PMOS` or `NMOS`
    - Three terminals, `S`, `D`, `G` (or `s`, `d`, `g`) and an optional bulk terminal
    - Three terminals and must contain the substring "`mos`" (case insensitive) in the cell name

  If the check determines that a permute rule is needed, add it in the CDF. For information on how to add a permute rule, see permuteRule Property.

  If a permute rule is not needed, add a fixed permute rule to avoid the issue.

- Missing `lxRemoveDevice` property.

  - If there is an unbound schematic instance that has two terminals and no layout master exists for this instance, this indicates a possible missing `lxRemoveDevice` property on the layout instance or in the physConfig.

  See lxRemoveDevice Property.

- Possible missing component type.

  - Checks if the master has terminals named `S`, `D`, and `G`. If not, checks if the cell name contains the string "`mos`" (case insensitive) and if the instance has 3 or 4 terminals.

  If the check determines that the component type is missing, use the Configure Physical Hierarchy window to add the component type.

© 2023 Cadence Design Systems, Inc.

See Defining a Design-Level Component Type. Alternatively, you can update the LAM file to add the component type.

See LAM File Component Type Section.

- Pins for layout master.
  - Checks if each terminal has a pin for a layout master.

  Use the *Create – Pins* command to manually add the missing pins.

- Presence of mosaics.
  - Checks if the design includes any mosaics, which can bring some design limitations such as preventing auto abutment, auto permute, and auto mirror.

  To convert mosaics instances into regular instances, use the *Convert To Instance* command available in the shortcut menu of a mosaic. Alternatively, you can right-click the mosaic and select the *Convert To Instance* command from the shortcut menu.

- Shorts if the devices are extracted through at level 32.
  - If the `?checkDevices` argument is set to `t`, all the masters in the layout are checked for any shorts if the devices are extracted through at level 32.

    See lxCheckLib

  To prevent the shorts, define stop layers in the technology

Related Topics

validLayers – Stop Layers Specification

Troubleshooting XL Compliance

validLayers – Stop Layers Specification

© 2023 Cadence Design Systems, Inc.

Viewing XL Status

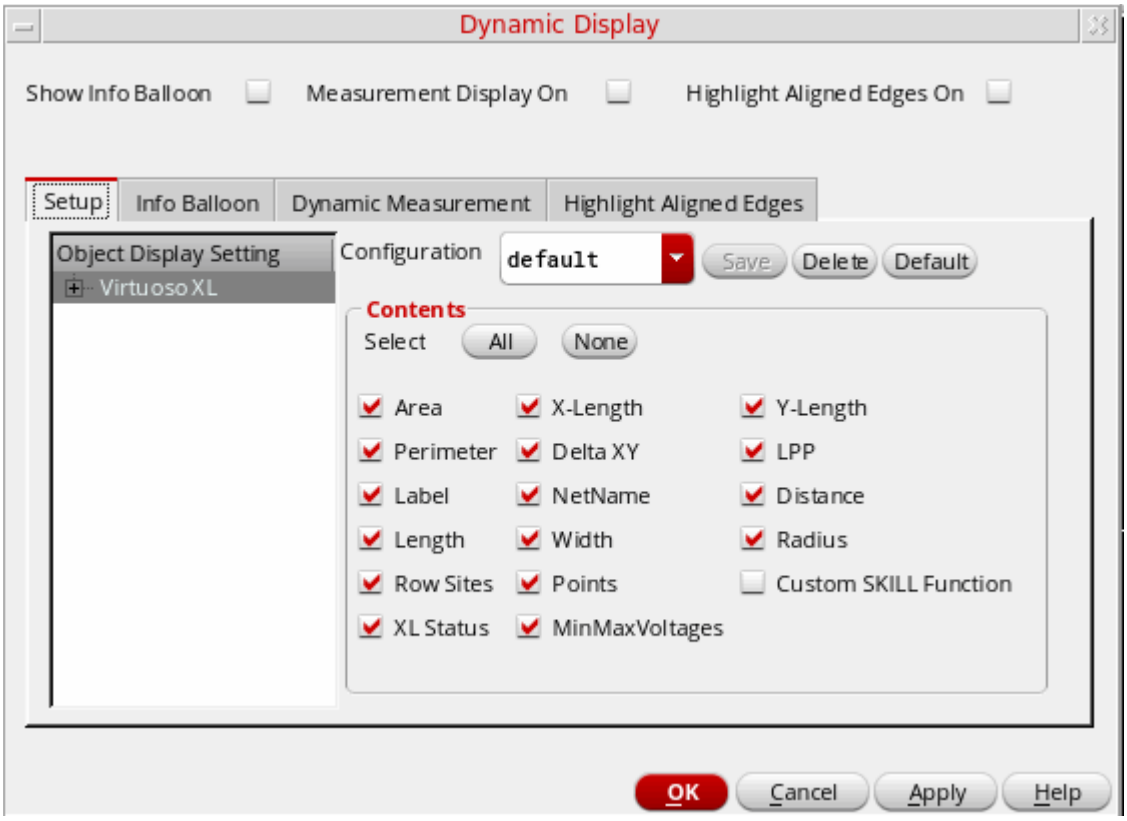
You can look up the XL Status information on the canvas by pointing over the object for which you want to look up the information.

To view the XL Status information on the canvas:

1. Select the **Options – Dynamic Display** command.

   The Dynamic Display form appears.

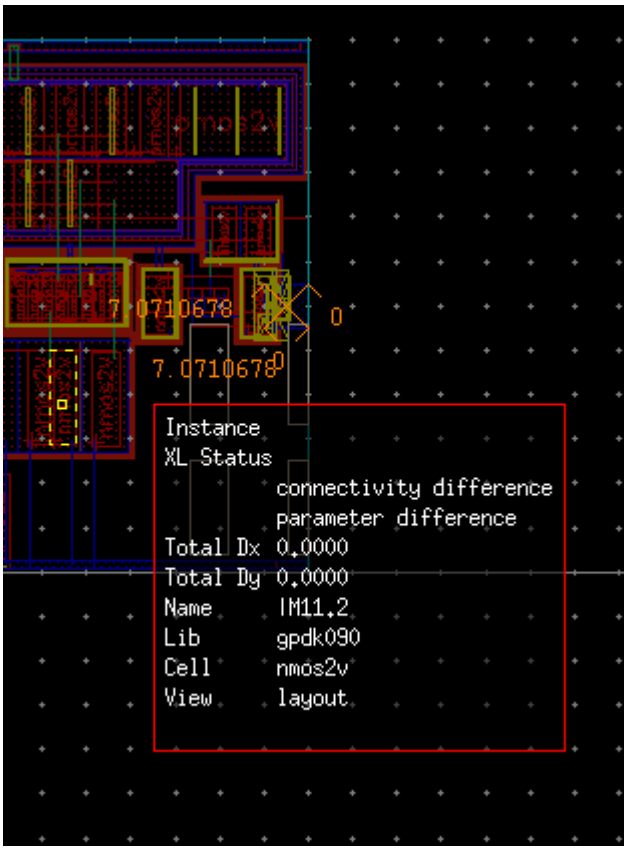
2. Select the **Show Info Balloon** check box.
3. In the Setup tab, select the **XL Status** check box.



4. Click *OK*.

© 2023 Cadence Design Systems, Inc.

Pointing over a design object pops up the associated information balloon displaying the XL status information, as shown in the figure below.



The advantage of using the balloon information to view the XL Status is that you can get the required information straight on the canvas without having to pull up the Navigator assistant.

The XL Status information is displayed on the canvas only for instances, shapes, and pins. However, if a shape is attached to a net, the balloon also displays information about that net.

XL Status of a design provides up-to-date information about the XL-compliance of the design. The information, such as whether or not an instance is bound or has any opens or shorts, is dynamically updated and can be accessed via the Navigator assistant or through the information balloon available in the canvas.

You can also view the XL Status information in the Navigator assistant using one of the following:

- Navigator — XL Status Column
- Navigator — Attention Icon

© 2023 Cadence Design Systems, Inc.

The various XL Status values possible and their description is given below. Irrespective of the method you use to view XL Status, the same information is available for each object at a given point of time.

XL Status Value	Description
shorts	layout nets with connectivity shorts markers
opens	layout nets with connectivity open markers including must-connect-pins
parameter difference	layout instance parameters different from the schematic counterpart
master difference	layout instance has a different master from the schematic counterpart
connectivity difference	layout instance connectivity different from the schematic counterpart
placed outside boundary	instance placed outside place and route boundary
complex binding limited check and no update	Instances are included in a complex binding between the schematic and layout. Limited information about these instances is reported and parameter values will not be updated automatically.
unbound	instance exists in layout but not bound to a layout instance
ungenerated	instance exists in schematic but not bound to a layout instance
ungenerated instance, no markers	net has no opens or shorts; therefore, no markers are generated; net is connected to an ungenerated instance
ungenerated instance, opens	net is connected to an ungenerated instance; net has opens
ungenerated instance, shorts	net is connected to an ungenerated instance; net has shorts
OK	object is XL-compliant
OK (ignore)	object is XL-compliant and has an `ignore` property set

© 2023 Cadence Design Systems, Inc.

OK (ignore for check)	object is XL-compliant and is `ignored for check`
(user binding)	instance bound manually using Define Device Correspondence
no markers	no discrepancies reported during last extraction run

- For instances and pins with no associated problems, the *XL Status* displays as *OK*.
- For mosaic instances that are in a one-to-one correspondence with the schematic mfactored instance, the *XL Status* reports an *OK*. If the correspondence is not one-to-one, the *XL Status* reports a *complex binding*.
- For nets with no opens or shorts reported during the last extraction run, the *XL Status* displays as *no markers*. However, if the design has been edited since the last extraction run, the *no markers* status may no longer be valid.
- For instances and terminals that are bound by using the Define Device Correspondence Form form, the *XL Status* displays as *OK* (user binding).

Navigator — XL Status Column

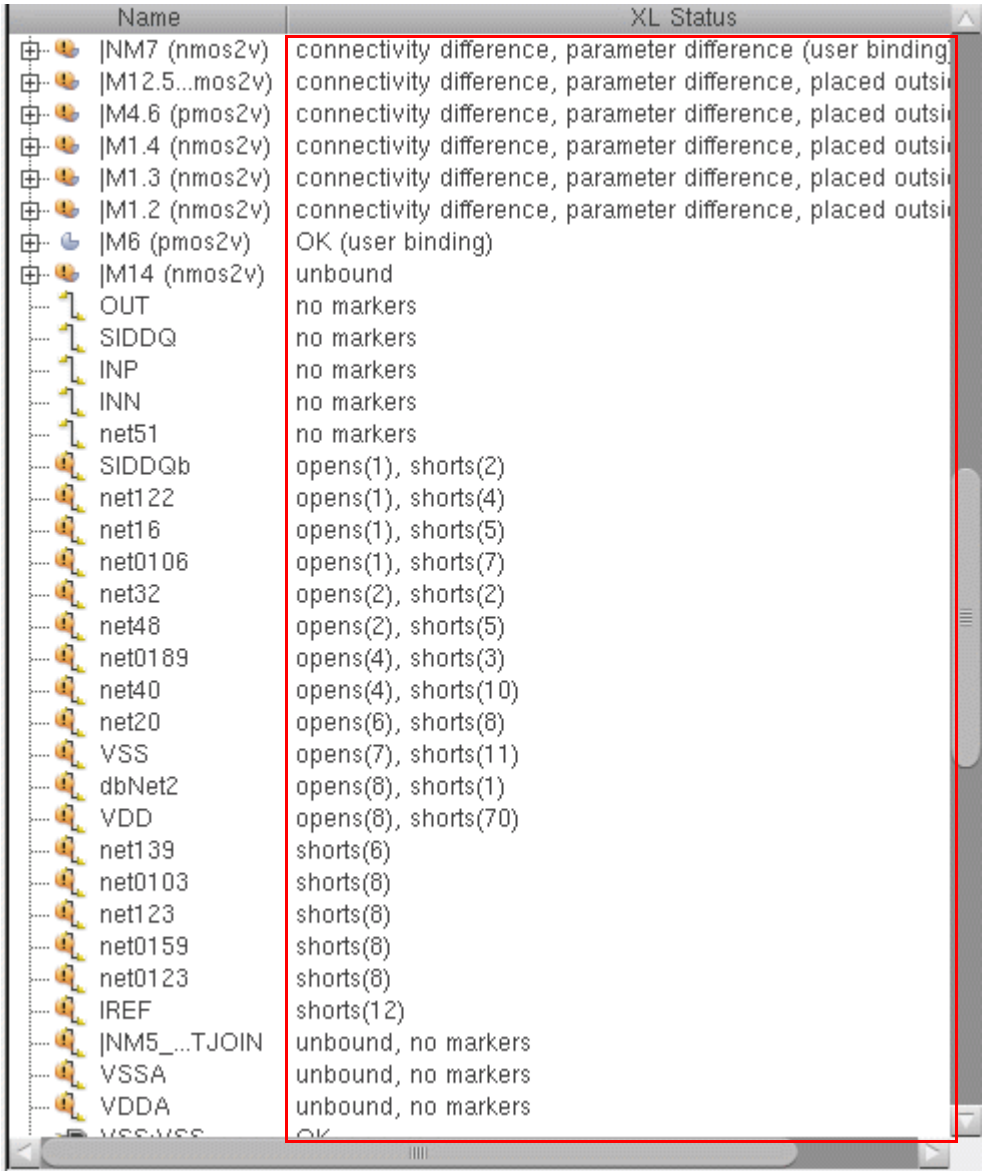
The advantage of using the *XL Status* column in the Navigator assistant is that you get to view the XL status for all your design objects in one go. The column displays XL status information for each instance, pin, and net in the design and also cross-selects the object in the canvas.

The *XL Status* column is displayed in the Navigator assistant by default if the xlStatus environment variable is set to `true`.

You can use the Search feature to filter the information based on same status or sort the objects by XL Status by clicking the *XL Status* column header. This feature is extremely useful when resolving a particular type of design issue, such as resolving unbound instances. By default, the information in the Navigator assistant is sorted based on name.

© 2023 Cadence Design Systems, Inc.

You can also directly select an object from the Navigator and apply appropriate commands from the context menu to resolve the schematic and layout discrepancy being reported for the object in the XL Status column.

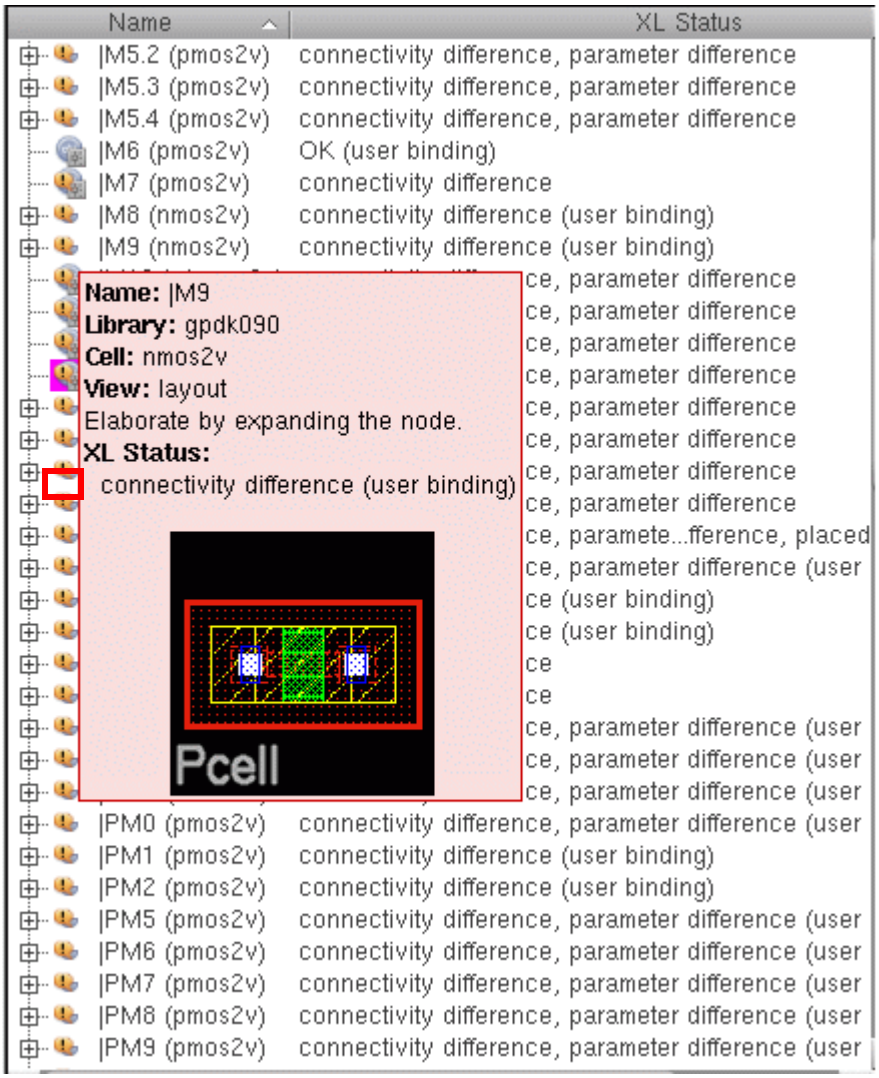


Navigator — Attention Icon

You can view the XL Status information by pointing towards the Attention icon ( ) displayed adjacent to the instance name. The icon is displayed only for those instances that have a design issue requiring your intervention.

© 2023 Cadence Design Systems, Inc.

Based on the type of the issue, you are required to take appropriate corrective action, after which the Attention icon disappears.



For example, an instance displaying an Attention icon may show the XL status in the information balloon as *placed outside boundary (ignored)*. For the status to be converted to *OK* and the Attention icon to be removed, you must move the instance inside the PR Boundary.

Related Topics

The Navigator Assistant

Rapid Adoption Kit (*Virtuoso Navigator Assistant*)

© 2023 Cadence Design Systems, Inc.

Selecting and Deselecting Nets in Navigator

Design Check for Manufacturability

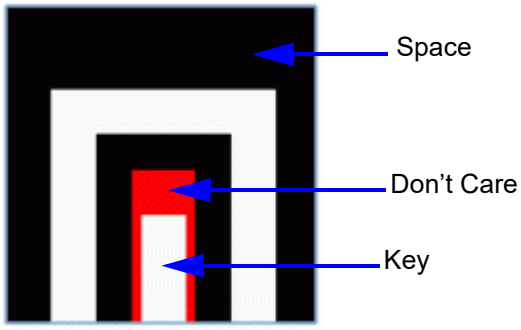
To check manufacturability of a design for optimum device yield and performance, the Virtuoso Studio custom design environment is integrated with various verification and fixing flows. You can use these flows to evaluate a layout design against specific schematic parameters to detect any violations and fix them interactively. The advantage of using these flows is that any design violations that can impact the device performance can be detected and fixed well before the design is sent for manufacturing.

Match and Fix Flow uses a *layout pattern* and an associated fixing *rule* that is defined in the *rule deck* to determine any inconsistencies that need to be fixed before the design is manufactured.

Match and Fix Flow

The *Match and Fix Flow* uses a *layout pattern* and an associated fixing *rule* that is defined in the *rule deck* to determine any inconsistencies that need to be fixed before the design is manufactured.

A *layout pattern* can be described as a layout area that is defined using one or many layers, as shown in the figure below:



The “space” area is inferred as the region within the bounding box that is not designated as “key” or “don’t care”.

A region of the target search layout is reported as “matching” the pattern if the following conditions are fulfilled:

1. The “key” area is fully covered by the layout geometry

© 2023 Cadence Design Systems, Inc.

2. The “space” area is fully uncovered

The list of layout patterns to be used for matching and the list of associated fixing rules are contained within a *rule deck*.

Therefore, a *rule deck* comprises the following:

- The list of layout patterns (such as stream file, layer definitions, and filtering rules)
- The list of Match and Fix rules (such as pattern, layers, and fixing rules)

The format for writing a rule deck is XML and the syntax is as shown below:

```
<?xml version="1.0" ?>
<MatchAndFixRules version="0.1">
<!-- Pattern definitions section -->
<!-- Match and fix rule definitions section -->
</MatchAndFixRules>
```

Important

Cadence recommends that you consult with your foundry for availability of a compatible rule deck to run the Match and Fix flow.

Pattern Definition

A single-layer layout pattern can be described in the rule deck as shown below:

```
0 <!-- Single layer Pattern -->
1 <Pattern name="single_pattern1" version="v2">
2 <Source type="oasis">
3 <File name="pattern/ single_pattern1.oas" />
4 <BBoxPurpose>5000</BBoxPurpose>
5 <Purposes type="key">10</Purposes> <!-- Key Purpose -->
6 <Purposes type="dc">50</Purposes> <!-- Don’t care Purpose -->
7 </Source>
8 <Description>
9 Necking Pattern 1
10 </Description>
11 <Severity>1</Severity>
12 <FilteringRule type="CatenaDRC" name="minSpaceRule">
13 <DRCRuleValue type="UserUnits">90 nm</DRCRuleValue>
14 </FilteringRule>
15 </Pattern>
```


© 2023 Cadence Design Systems, Inc.

The description of each line in the pattern definition is given in the table below.

Line #	Description
1	Pattern name (mandatory). Engine version (optional; default is v2).
2	Pattern source type (mandatory). Pattern source can be either “oasis” or “gdsii”.
3	Source file name (mandatory).
4	Bounding box purpose (mandatory).
5 – 6	Purpose mapping for “key” and “don’t care” areas. ■ “Key” purpose is mandatory ■ “Don’t care” purpose is optional
8 – 10	Detailed description (optional).
11	Severity of the pattern (optional).
12 – 14	Filtering rule (optional).

The *Filtering rule* is used by the Match and Fix flow to filter out and return only matching regions that also fail the given rule. Currently, the following filtering rules are supported:

- *Min space rule*

```
<FilteringRule type="CatenaDRC" name="minSpaceRule">
<DRCRuleValue type="UserUnits">90 nm</DRCRuleValue>
</FilteringRule>
```

- *Min notch rule*

```
<FilteringRule type="CatenaDRC" name="oaMinNotchSpacing">
<DRCRuleValue type="UserUnits">90 nm</DRCRuleValue>
```

Let us now consider a sample pattern definition in the rule deck for a multi-layer layout pattern.

```
1 <!–Multi Layer Pattern  -->
2 <Pattern name="multi_pattern2" version="v2">
3 <Source type="gdsii">
4 <File name="pattern/ multi_pattern2.gds2" />
5 <BBoxPurpose>5000</BBoxPurpose>
6 <PatternLayer name="custom_M02">
7 <Purposes type="key">20</Purposes>
```


© 2023 Cadence Design Systems, Inc.

```
8 <Purposes type="dc">50</Purposes>
9 </PatternLayer>
10 <PatternLayer name="custom_M03">
11 <Purposes type="key">30</Purposes>
12 </PatternLayer>
13 </Source>
14 <Description>
Multi layer pattern 1
15 </Description>
16 <Severity>1</Severity>
17 <FilteringRule type="CatenaDRC" name="minSpaceRule">
18 <DRCRuleValue type="UserUnits">90 nm</DRCRuleValue>
19 </FilteringRule>
20 </Pattern>
```

Notice that the lines 6-9 and 10-13 illustrate how the individual layers of a two-layer pattern can be specified.

Related Topics

Match and Fix Rule Definition

Match and Fix Rule Definition

In addition to carrying a layout pattern definition, the rule deck carries associated Match and Fix rule definitions. If the target search layout is not found to match the defined pattern, the layout pattern is fixed according to the associated fixing rules defined in the rule deck.

A single match rule is often associated with several fixing rules. If a match rule does not have an associated fixing rule, the fixing engine uses the default fixing flow.

A sample Match and Fix rule definition for a single-layer layout pattern is given below:

```
1 <MatchAndFixRule name="Match&Fix Rule 1 on M2">
2 <MatchRule>
3 <Pattern name=" single_pattern1" />
4 <Layer>M2</Layer>
5 </MatchRule>
6 <FixRule name="moveEdges">
7 <Move unit="nm">194 156 194 208 WEST 130</Move>
8 </FixRule>
9 <FixRule name="fixFilteringRule" />
```


© 2023 Cadence Design Systems, Inc.

```
10 <FixRule name="decongest">
11 <Option name="ripupEngine">detail</Option>
12 </FixRule>
13 </MatchAndFixRule
```

The description of each line in the Match and Fix rule definition is given in the table below.

Line #	Description
1	Match and fix rule name (mandatory).
2- 5	Match rule description (mandatory).
3	Name of the pattern to match (mandatory).
4	Target layout layer (or layers) on which to search for the pattern (mandatory).
6 - 12	Fixing rule description (optional)
6 – 8	First fixing rule (name is a mandatory parameter).
7	Parameter for the first fixing rule (specification of an edge to be moved).
9	Second fixing rule.
10 - 12	Third fixing rule.
11	Option for the third fixing rule.

Currently, Virtuoso supports the following fixing strategies:

- *fixFilteringRule*

  Apply internal heuristics, such as moving the shapes, to fix the pattern by using the filtering rule. To be usable, either the pattern or the match rule should have a filtering rule.

- *decongest*

  Use the rerouting engine to decongest the area where the pattern has been found.

- *moveEdges*

  Use the move engine to move one or several edges.

Let us now consider a Match and Fix rule definition for a multi-layer layout pattern as given below:

```
1 <MatchAndFixRule name="Multi-layer rule on M2-M3">
2 <MatchRule>
```


© 2023 Cadence Design Systems, Inc.

```
3 <Pattern name=" multi_pattern2" />
4 <PatternLayerMap name="custom_M02">M2</PatternLayerMap>
5 <PatternLayerMap name="custom_M03">M3</PatternLayerMap>
6 </MatchRule>
7 </MatchAndFixRule>
```

The individual layers specified in the pattern are mapped to search layers in the target layout. Lines 4 and 5 aim at mapping the pattern layers, `custom_M02` and `custom_M03` to the layout layers `M2` and `M3`.

To run the Match and Fix flow, choose *Optimize – Match and Fix Flow*.

Related Topics

Match And Fix Form

Match and Fix Flow

Batch Checker

The batch checker offers different types of checking.

- A shape-based checker that verifies the shapes against process rules specified in the technology file, design, and object level.
- A connectivity checker that verifies the design for shorts, loops and dangles.
- A constraints checker that verifies the specialty routing constraints such as Diff Pair and Symmetry.
- A placement checker that verifies the placement constraints.
- A routability checker that verifies the routing problems in the design.
- A Fabrication checker that verifies the antenna violations in the design.
- A Width Spacing Pattern (WSP) checker that verifies the shapes for color, width, and/or wireType conformance to the tracks in the active WSP.

Related Topics

Running the Batch Checker

Checking Routability Using Batch Checker

© 2023 Cadence Design Systems, Inc.

Checking Process Rules Using Batch Checker

Checking Connectivity Using Batch Checker

Checking Specialty Routing Using Batch Checker

Checking Placement Using Batch Checker

Checking Fabrication Using Batch Checker

Checking WSP | SP Active Using Batch Checker

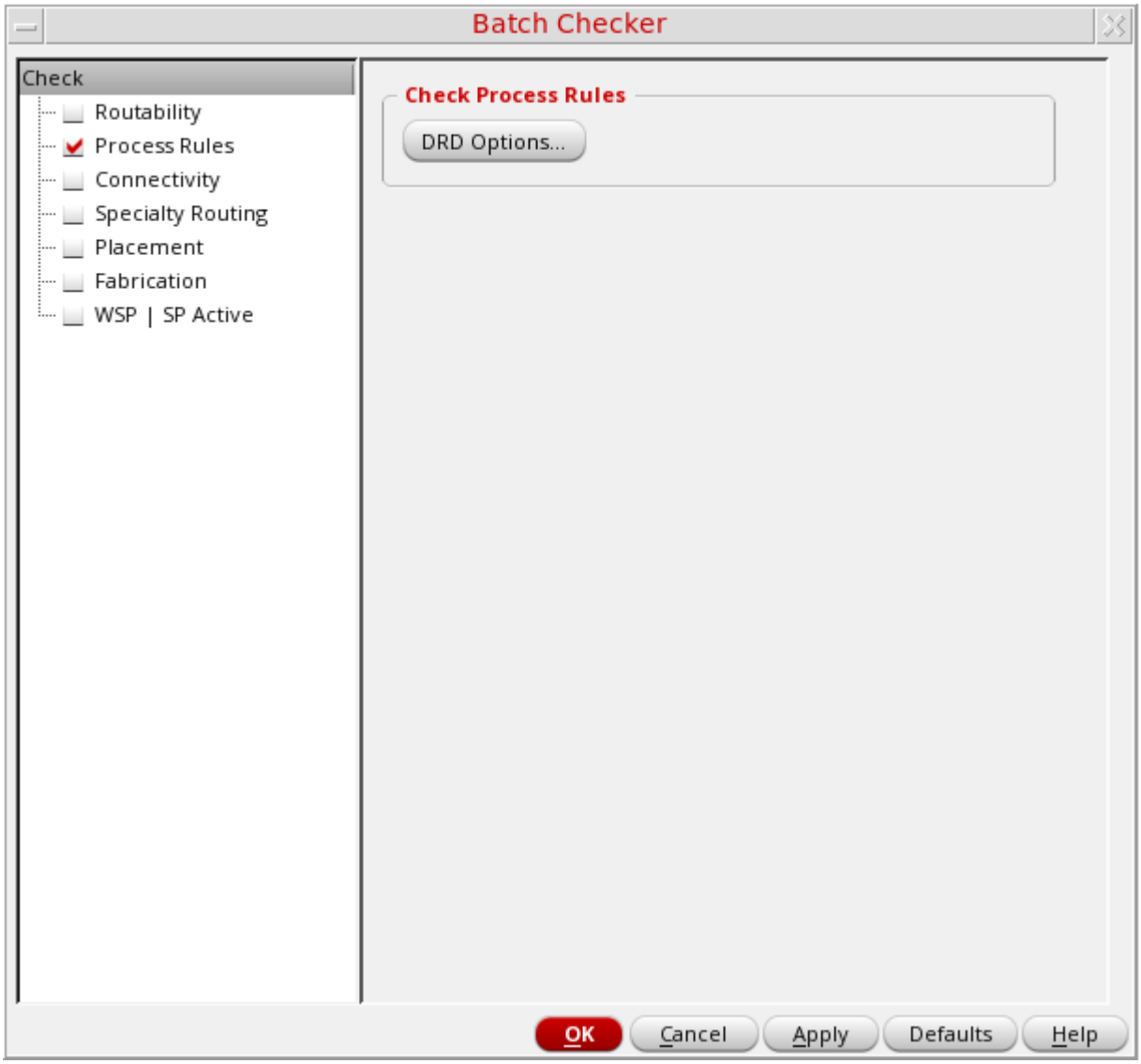
Running the Batch Checker

To run the *Batch Checker*:

1. Choose Verify – Design.

© 2023 Cadence Design Systems, Inc.

The Batch Checker form appears, as shown in the following figure.



There is a tree widget on the left panel for the checks that can be performed using this form.

The *Verify – Design* command is disabled when a cellview is opened in read-only mode. This is because when a cellview is in read-only mode, DRD is unable to write into the cellview and hence cannot create markers in it.

2. Choose a check from the list of checks in the left panel.

   The options for the check are shown in the right panel.

© 2023 Cadence Design Systems, Inc.

   a. Specify the options for the check. Valid options are: *Routability, Process Rules, Connectivity, Specialty Routing, Placement, Fabrication,* and *WSP | SP Active.*

3. Repeat step 2 until all the checks that you want to run are enabled with the desired options.
4. Click *OK* or *Apply* to run the checks.

   For information about using the Annotation Browser to inspect the markers that are created by these checks, see Finding Violations Using Annotation Browser.

Related Topics

Batch Checker

Checking Routability Using Batch Checker

Checking Process Rules Using Batch Checker

Checking Connectivity Using Batch Checker

Checking Specialty Routing Using Batch Checker

Checking Placement Using Batch Checker

Checking Fabrication Using Batch Checker

Checking WSP | SP Active Using Batch Checker

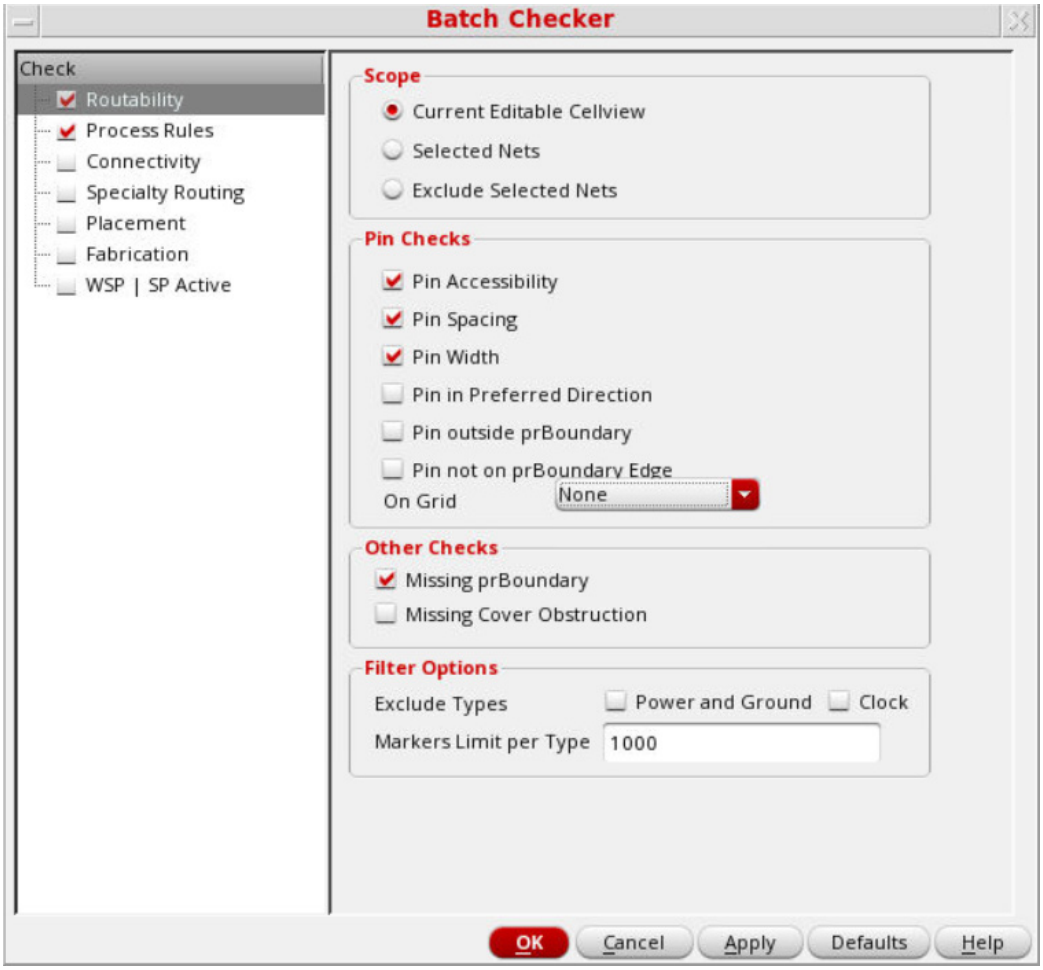
Checking Routability Using Batch Checker

Routability checks let you identify the problems with your data before the router is run. By independently using routing check, you can check the width, spacing, and accessibility of pins, and other issues that may affect routing. Because it checks the data against foundry rules, user constraints, and Wire Assistant overrides, it is important to set up the routability check options in the Wire Assistant before running Routability Check.

Routability checks are designed primarily for the *Chip Assembly* routing mode. Therefore, if you set the routing mode to *Chip Assembly* in the Wire Assistant, the whole suite of routing checks is available. However, if you set the routing mode to *ASIC* or *Device Style* mode, only pin access checks are available.

© 2023 Cadence Design Systems, Inc.

1. To perform routability checks on a design, select the *Routability* option. The routability checks are displayed on the right panel of the form, as shown in the following figure.



2. In the *Scope* group box, select

   - *Current Editable Cellview*

     Checks the current cellview. This option is selected by default.

   - *Selected Net*

     Checks only the nets that you currently have selected.

   - *Exclude Selected Nets*

     Checks all the nets except those selected.

For the other routability checks, such as pin checks, missing prBoundary, cover obstructions, and filter options, see Routability Check.

© 2023 Cadence Design Systems, Inc.

Related Topics

Batch Checker

Checking Process Rules Using Batch Checker

Checking Connectivity Using Batch Checker

Checking Specialty Routing Using Batch Checker

Checking Placement Using Batch Checker

Checking Fabrication Using Batch Checker

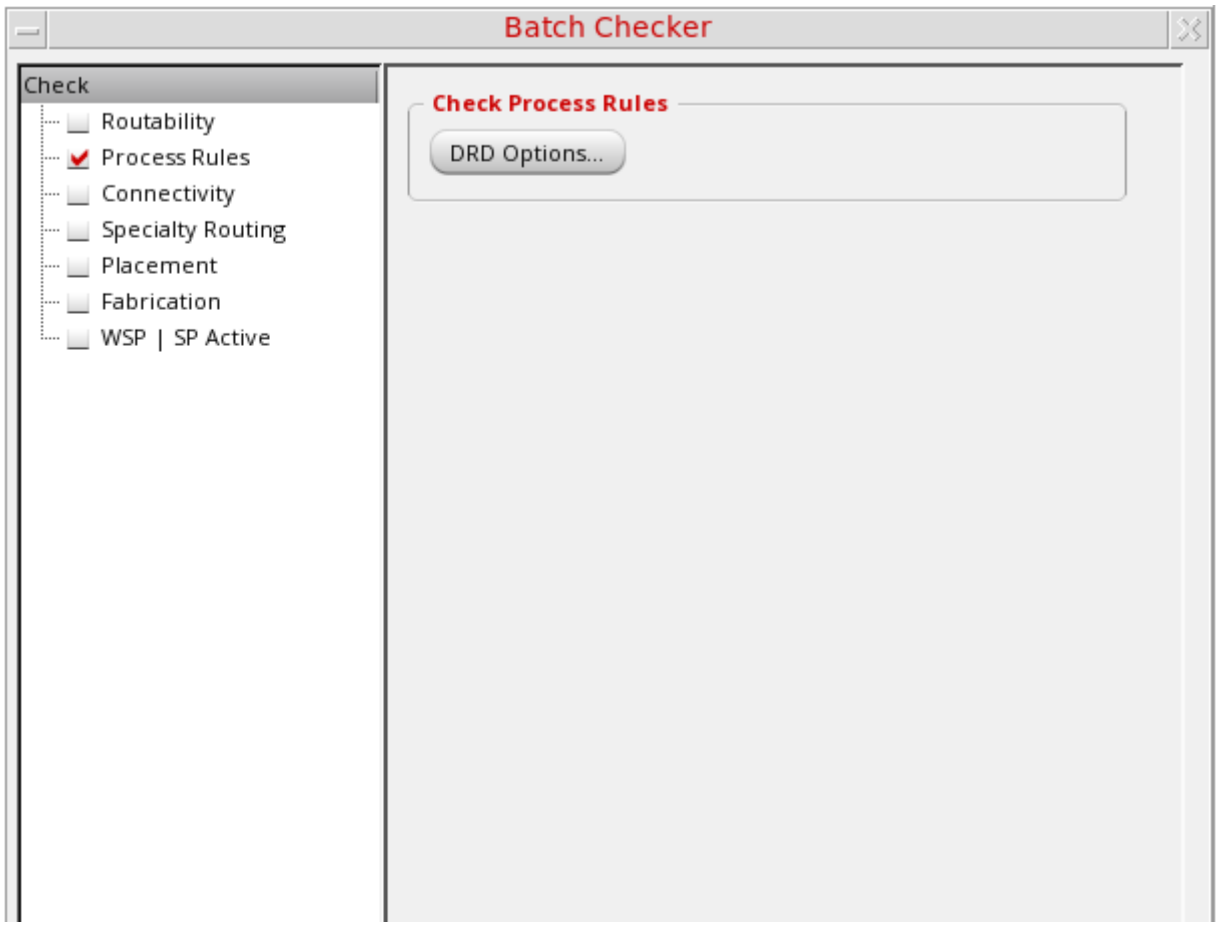
Checking WSP | SP Active Using Batch Checker

Checking Process Rules Using Batch Checker

Process rule checking offers a set of options that allow you to specify the portions of the design or the types of rules to verify.

© 2023 Cadence Design Systems, Inc.

The *Process Rules* check is selected by default when the *Batch Checker* form is opened.



To perform process rule checking:

➡ Click the *DRD Options* button in the Check Process Rules group box.

The DRD Options form opens showing the *Batch* page, in which you can set the various Batch mode options. For information, see Batch Check Mode in the *Virtuoso Design Rule Driven Editing User Guide*.

Related Topics

Batch Checker

Checking Routability Using Batch Checker

Checking Connectivity Using Batch Checker

© 2023 Cadence Design Systems, Inc.

Checking Specialty Routing Using Batch Checker

Checking Placement Using Batch Checker

Checking Fabrication Using Batch Checker

Checking WSP | SP Active Using Batch Checker

Checking Connectivity Using Batch Checker

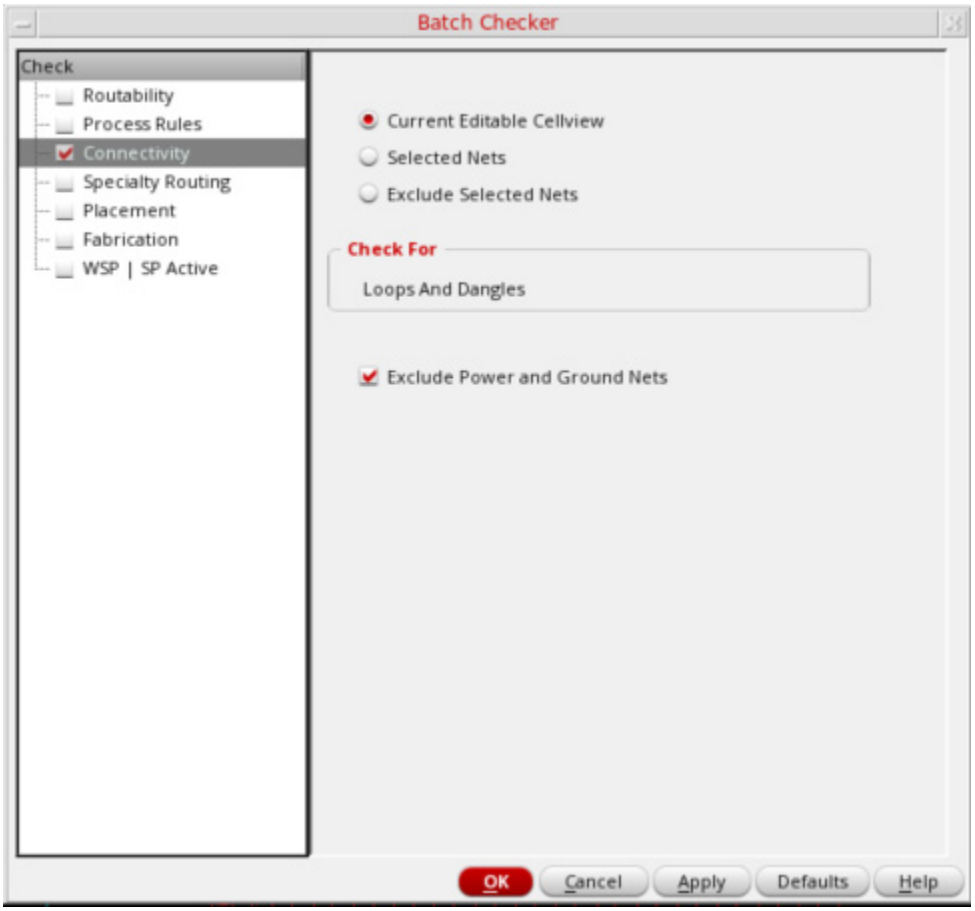
The connectivity verification can be limited to specific set of violation checking. You can check for the following:

- Current cellview
- A selected set of nets
- Exclude selected nets
- Loops and dangling wires
- Exclude power and ground nets

To perform connectivity check on a design:

© 2023 Cadence Design Systems, Inc.

1. Select the *Connectivity* option. The connectivity checks are displayed in the right panel of the form as shown in the following figure.



2. From the *Scope* group box, select:

   - *Current Editable Cellview*

     Checks the current cellview.

   - *Selected Net*

     Checks only the nets that you currently have selected.

   - *Exclude Selected Nets*

     Checks all the nets except those selected.

3. The *Check For* group box checks for the dangling wires and wires with loops.

4. The *Filter Options* group box allows you to choose whether or not to check cellview for shorts and opens that involve power and ground nets.

© 2023 Cadence Design Systems, Inc.

Related Topics

Batch Checker

Checking Routability Using Batch Checker

Checking Connectivity Using Batch Checker

Checking Specialty Routing Using Batch Checker

Checking Placement Using Batch Checker

Checking Fabrication Using Batch Checker

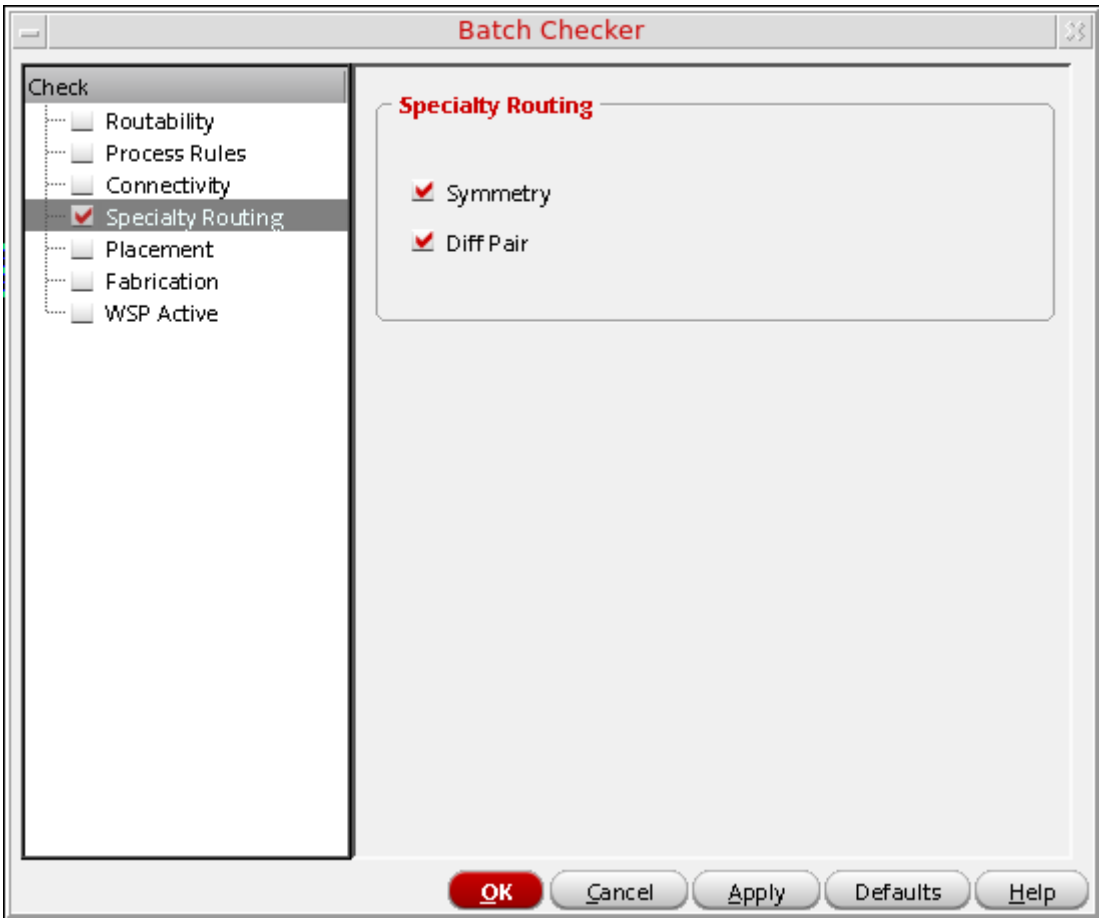
Checking WSP | SP Active Using Batch Checker

Checking Specialty Routing Using Batch Checker

Geometric shapes (e.g. paths, pathSegs/vias not in a route) are not checked. A warning will be issued when checker detects geometric shapes.

© 2023 Cadence Design Systems, Inc.

1. To perform special routing checks on a design, select the *Specialty Routing* option. The special routing checks are displayed in the right panel of the form as shown in the following figure.



2. In the *Specialty Routing* group box, select the type of constraint checking to be performed.

   - *Symmetry* checks whether a symmetry axis is defined and whether the position of the axis is correctly defined. If the axis is not considered well defined, a marker will be generated at the location of the axis. Only routed portions of the symmetric nets are checked and only shapes on metal and poly layers are checked.
   - *Diff Pair* checks the percentage of unpaired lengths for each net in of a diff pair.

Related Topics

Batch Checker

© 2023 Cadence Design Systems, Inc.

Checking Routability Using Batch Checker

Checking Process Rules Using Batch Checker

Checking Connectivity Using Batch Checker

Checking Placement Using Batch Checker

Checking Fabrication Using Batch Checker

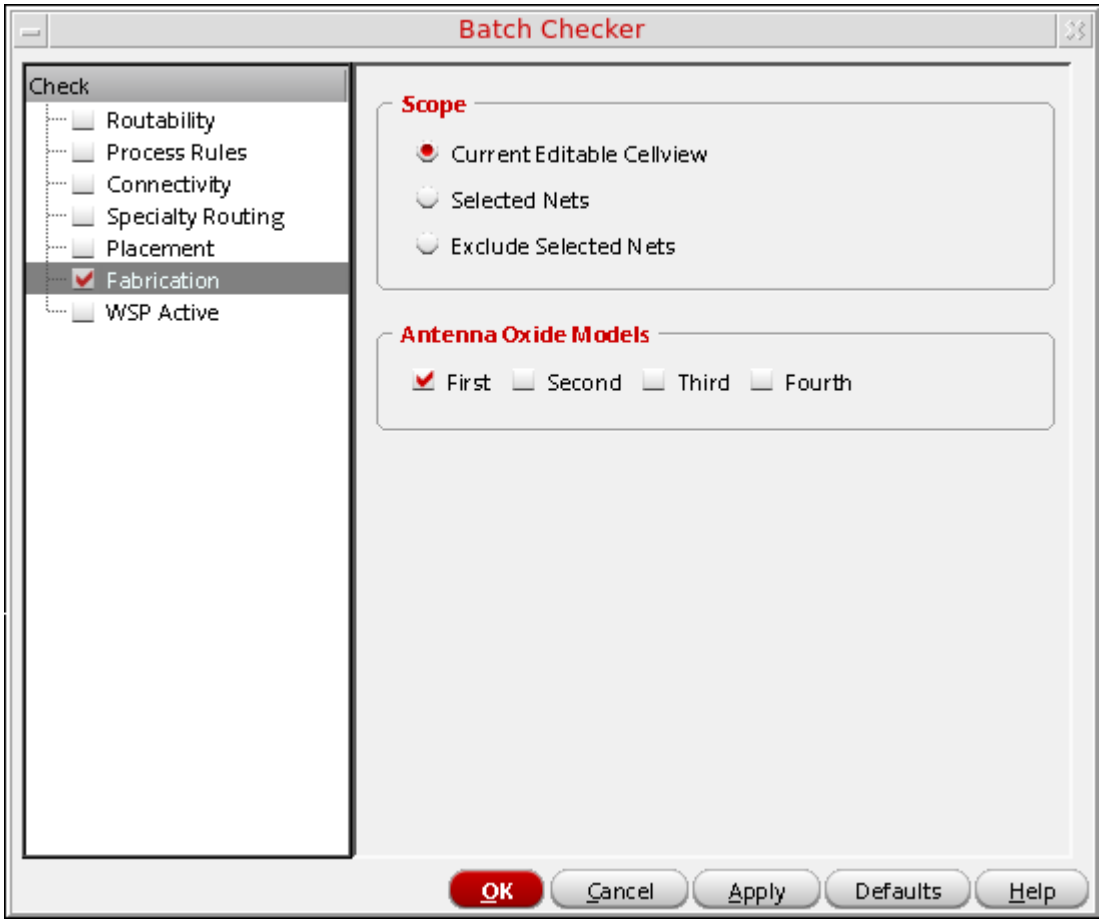
Checking WSP | SP Active Using Batch Checker

Checking Fabrication Using Batch Checker

This check verifies the process antenna violations for the entire design, selected nets, or except selected nets.

© 2023 Cadence Design Systems, Inc.

1. To perform fabrication checks on a design, select the *Fabrication* option. The fabrication checks are displayed in the right panel of the form as shown in the following figure.



2. From the *Scope* group box, select:

   - *Current Editable Cellview*

     Checks the current cellview.

   - *Selected Net*

     Checks only the nets that you currently have selected.

   - *Exclude Selected Nets*

     Checks all the nets except those selected.

3. From the *Antenna Oxide Models* group box, select which oxide model(s) to use: *First*, *Second*, *Third*, *Fourth*. By default, the *First* oxide model is selected.

© 2023 Cadence Design Systems, Inc.

Related Topics

Batch Checker

Checking Routability Using Batch Checker

Checking Process Rules Using Batch Checker

Checking Connectivity Using Batch Checker

Checking Specialty Routing Using Batch Checker

Checking Placement Using Batch Checker

Checking WSP | SP Active Using Batch Checker

Checking WSP | SP Active Using Batch Checker

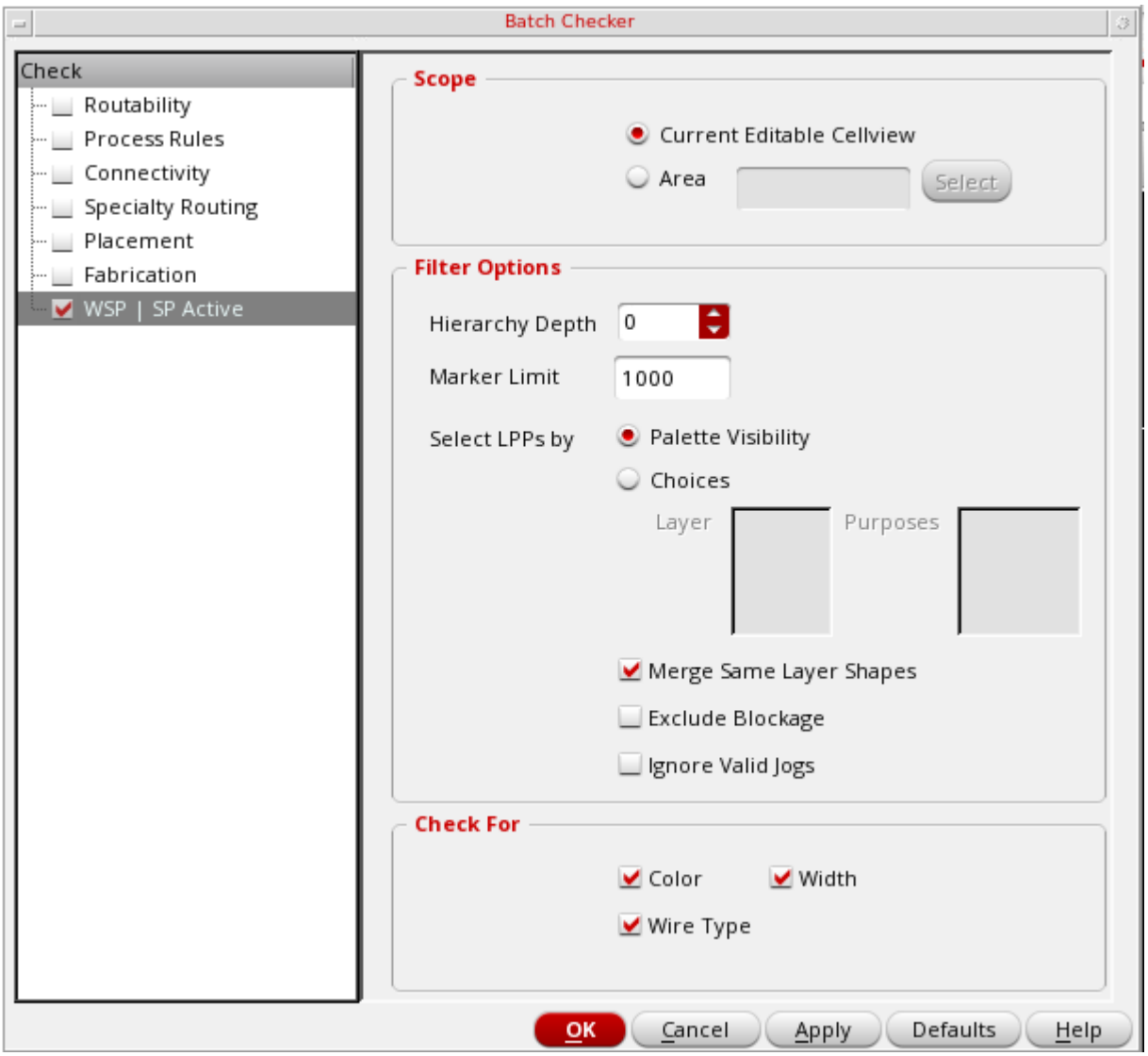
Checks shapes for width, color, and wireType conformance as compared to the active width spacing pattern (WSP). The active WSP can be for the global grid or a pattern region.

To check shapes for WSP conformance:

1. Choose *WSP* | *SP Active* in the *Check* panel.

© 2023 Cadence Design Systems, Inc.

The *WSP | SP Active* options are displayed on the right panel of the form, as shown in the following figure:



2. In the *Scope* group box, choose one of the following:

   - *Current Editable Cellview*

     Checks all the shapes in the current editable cellview.

   - *Area*

     Checks all the shapes in the given region. Specify the region by doing one of the following:

© 2023 Cadence Design Systems, Inc.

- Specify the lower-left and upper-right coordinates for the region in this format `((xlower ylower) (xupper yupper))`
- Click *Select* and draw the region in the layout canvas.

3. In the *Filter Options* group box, choose the following:

- *Hierarchy Depth*

  Choose the hierarchy depth up to which the cellview will be checked.

- *Marker Limit*

  Specify the maximum number of violation markers. This is a cumulative count for all the layers and purposes to be checked, rather than per layer or purpose. If more than one layer or purpose is specified, the count can be reached before all those layers and purposes are checked. The default is 1000. For information on viewing and managing these markers, see Finding Violations Using Annotation Browser.

- *Select LPPs by*

  Choose one of the following option to select LPPs.

  - *Palette Visibility*

    Checks for the LPPs that have the visibility option set to `true` in Palette. For example, If `Metal2` and `Metal3` drawing LPPs are visible in the Palette and the other LPPs are invisible, WSP Active checks are performed on `Metal2` and `Metal3` layers only.

    This is the default option. *Layer* and *Purposes* fields are disabled when *Palette Visibility* is selected.

  - *Choices*

    Enables you choose a single layer with multiple purposes for WASP active checking.

  - *Layer*

    Choose *All* or the layer for the shapes to be checked from the drop-down list of valid layers.

  Layers with `spdef` and `wsspdef` are also displayed in the *Layer* field.

  - *Purposes*

    Choose one or more purposes for the shapes to be checked from the list of valid purposes. Notice that purposes with an `annotation` parent purpose are excluded from the list.

© 2023 Cadence Design Systems, Inc.

If only layer blockage object exists, the *Purposes* field is inactive.

- *Merge Same Layer Shapes*

  Before checking, merges same-color shapes and uncolored shapes that are connected and on the same layer. If multiple purposes are selected, merging applies to the shapes on those purposes.

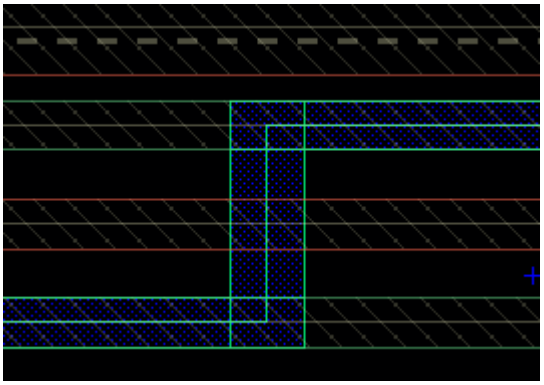
- *Exclude Blockages*

  Before checking, exclude blockages if you do not want to include blockages while running the batch checker.

- *Ignore Valid Jogs*

  Before checking, ignores jogs that are defined in the `allowedWidthRanges` constraint for the layer in the jog direction. A job is a short connection between two tracks. Jogs are also referred to as non-preferred direction (NPD) pathSegs.

  In the figure below, the vertical segment of layer `M1` is not on a WSP track. It is a jog.



  The `allowedWidthRanges` constraint for layer `M1`, is defined as follows:

```
( allowedWidthRanges  "M1" 'horizontal (0.048 0.062) 'ref "M1.W.1h"
'description "M1 width horz" )
( allowedWidthRanges  "M1" vertical (0.032 0.048 ">= 0.062") 'ref "M1.W.1"
'description "M1 width" )
```

  If the vertical `M1` pathseg or jog of width `0.048` or `0.062` is added to the design, it is ignored and no error marker is generated.

  Environment variable: ignoreValidJogs

4. In the *Check For* group box, choose the following:

   - *Color*

     Checks the shapes for conformance to the track colors of the active WSP.

© 2023 Cadence Design Systems, Inc.

- *WireType*

  Checks the shapes for conformance to the wireTypes of the active WSP.

- *Width*

  Checks the shapes for conformance to the track widths of the active WSP (if enabled), or the centerline alignment (if disabled) with the active WSP.

Related Topics

Batch Checker

Checking Routability Using Batch Checker

Checking Process Rules Using Batch Checker

Checking Connectivity Using Batch Checker

Checking Specialty Routing Using Batch Checker

Checking Placement Using Batch Checker

Checking Fabrication Using Batch Checker

Finding Violations Using Annotation Browser

The Annotation Browser assistant is used for viewing and managing violation markers in your design.

To view and manage violation markers:

1. Choose *Window – Assistants – Annotation Browser* or click the *Annotation Browser* icon.
2. Click the tab in the Annotation Browser assistant for the markers that you want to view.

Check	Annotation Browser tab	Categories
Routability	Routing	
Process Rule	DRC/DFM	
Connectivity	Connectivity	
Specialty Routing	Constraints	

© 2023 Cadence Design Systems, Inc.

Placement	Constraints	
Fabrication		
WSP | *SP Active*	DRC/DFM	drdEdit–`widthSpacingPattern`, grouped by layer

Related Topics

Running the Batch Checker

Annotation Browser Assistant

© 2023 Cadence Design Systems, Inc.

9

Layout XL Design Data Updates

This topic summarizes how you can use the Virtuoso® Layout Suite XL layout editor (Layout XL) to make design changes and update your design as you progress.

- Components and Nets Updates - Automatically update your layout to take account of instances, pins, and connectivity that you have changed in the schematic.
- Layout Parameters Updates - Update the parameter values of devices in the layout to match the values in the schematic.
- Layout Parameters Updates - Update the parameter values of devices in the layout to match the values in the schematic.
- Schematic Parameter Updates - Update the parameter values of devices in the schematic to match the values in the layout.
- Device Correspondence Updates - For each schematic and layout pair, the correspondence between the components in the views is determined by the Layout XL binder engine.
- Updating the Connectivity Reference - Update the source schematic for the layout view to change the physical configuration, which specifies how the layout is generated from the schematic.
- Binding Updates - Increase Layout XL-compliance by improving the bindings, if you have unbound instances between the schematic and layout views due to any of the following reasons
- Updating All Physical Bindings - Update the CPH master of an instance to the layout master if bindings have been created between layout and schematic instances that have different masters.

© 2023 Cadence Design Systems, Inc.

Components and Nets Updates

Use the *Update Components And Nets* command to automatically update your layout to take account of instances, pins, and connectivity that you have changed in the schematic.

In contrast to *Generate All From Source*, which deletes all the existing components in the layout view and regenerates everything from scratch, *Update Components And Nets* updates only the components that have changed. It

- Adds new instances and pins.
- Removes old instances (including any unbound vector and mosaic instances) and pins, along with any empty nets left as a result of the removal.
- Updates instance masters to match those in the schematic.
- Updates the instance connectivity to match the connectivity of the schematic instance to which it is bound.
- Updates the names of instances, terminals, and nets to match them with the corresponding schematic instance, terminal, or net to which they are bound.
- Updates the net signal types that have been modified to match the schematic net signal type.Updates the layout parameters and constraints.
- Removes any non-matching global terminals on the net, if a matching terminal already exists. Such non-matching global terminals may result if an original global terminal is modified, for instance, due to a name change in an associated cell.
- Adds missing split-fingered devices in the layout, if the environment variables `bindStrict` and `bindNewSplitParam` are set to `t`.
- (Virtuoso Photonics Option) When using the Virtuoso Photonics Solution with the `Virtuoso_Photonics_Option` license checked out, the *Update Components And Nets* command updates the default pin values such as *Width*, *Radius*, and *Input Angle*.

Related Topics

bindStrict

bindNewSplitParam

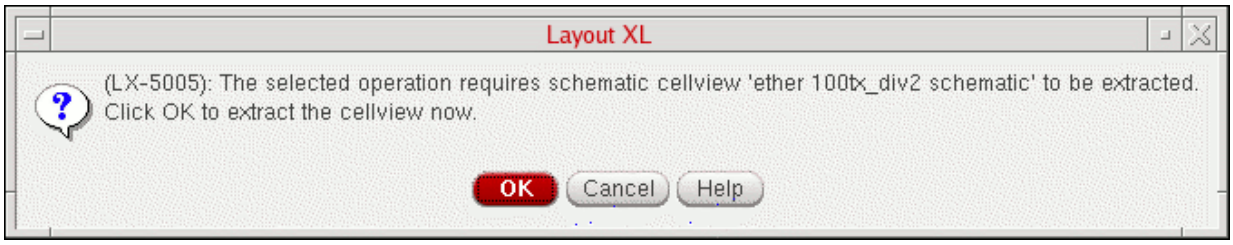
© 2023 Cadence Design Systems, Inc.

Updating Components and Nets

To update the components and nets in a layout view:

1. From the layout window menu bar, choose *Connectivity – Update – Components And Nets*.

   If the schematic has been changed since it was last extracted, you see a dialog box indicating that it needs to be re-extracted.

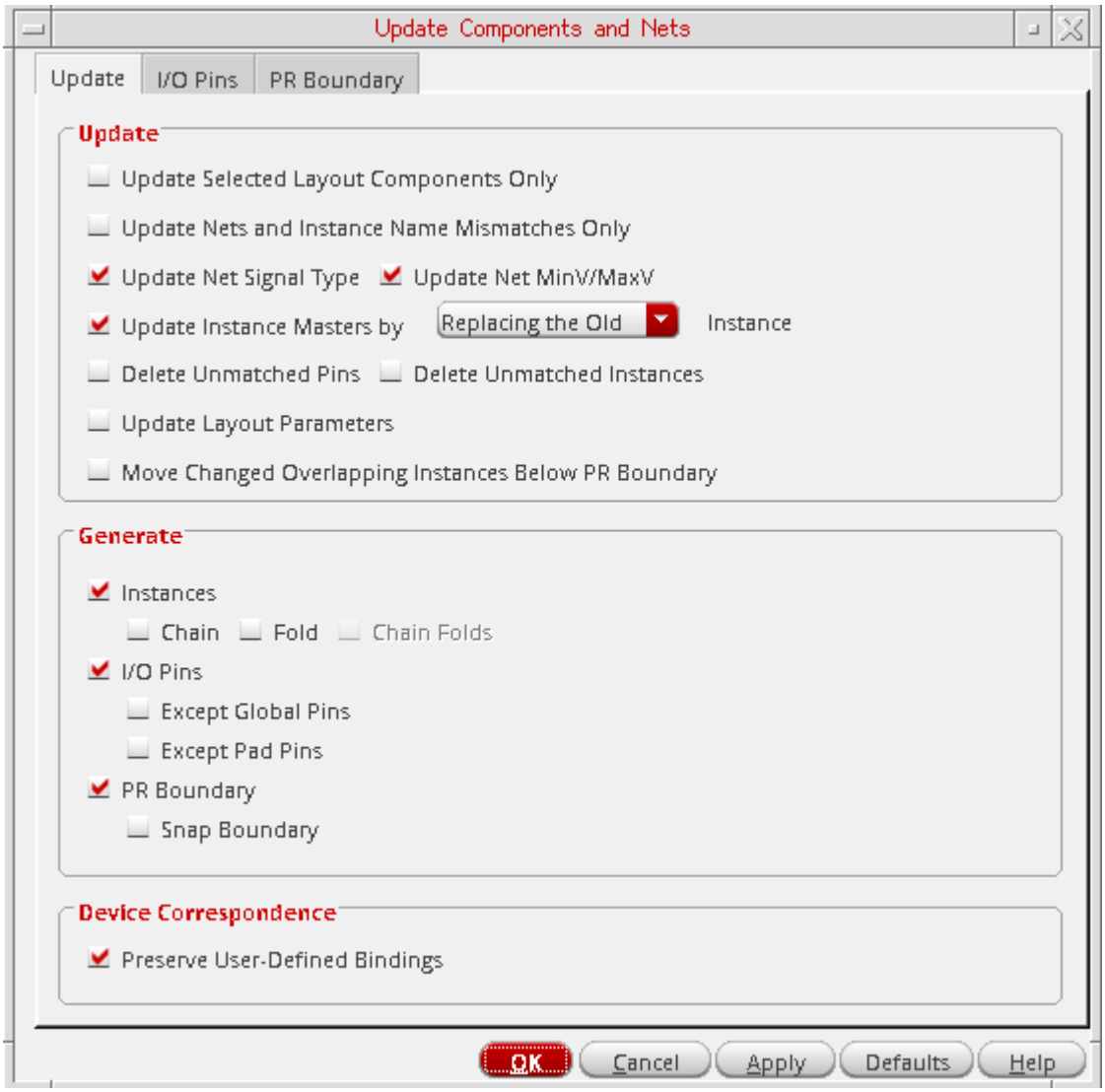


2. Click *OK* in the message box to extract the schematic and all its reference libraries. (The extracted cellviews are not automatically saved.)

   To specify a different extraction behavior, click *Cancel* in the message box and choose the *Check – Hierarchy* command from the schematic window menu bar.

© 2023 Cadence Design Systems, Inc.

When the schematic is extracted, the Update Components and Nets form is displayed.



The form remembers values set previously in the current Virtuoso session. When you open it for the first time, it shows the default values set in your `.cdsenv` file.

3. Set the options on the form as needed.

   See:

   a. Updating Selected Layout Components

   b. Updating Nets and Instance Name Mismatches Only

   c. Specifying the Components to be Updated

© 2023 Cadence Design Systems, Inc.

d. I/O Pins and PR Boundary Updates

4. Click *OK* in the Update Components and Nets Form form.

   Layout XL updates the layout design as specified in the form. For detailed information on these settings, see the sections listed in step 3.

   If either the *Update Instance Masters by Creating a New* or the *Instances* option is selected, clicking *Apply* creates any ungenerated schematic instances or instances with a master mismatch in the layout. To prevent any new layout instances with the same name as that of an existing layout instance creating a conflict, the new layout instance is renamed with the suffix `_1`, for example `I5` to `I5_1`. Clicking *Apply* a second time or clicking *OK*, confirms that the names mismatch and require updating. The new layout instance is renamed by removing the suffix, `I5_1` to `I5`, and the existing layout instance is renamed by adding the suffix `_old`, for example `I5` to `I5_old`.

Limitations of Update Components and Nets

Update Components And Nets does not correct any connectivity errors introduced while defining a many-to-many device binding using the *Define Device Correspondence* command. These errors are reported by *Check Against Source* and must be fixed manually using the Define Device Correspondence form.

Troubleshooting Update Components and Nets

Any problems encountered during the update are reported in the CIW and in a Layout XL Info text window. Use Checking Against Source to get more information on the problems encountered and how to resolve them.

Updating Components and Nets Using SKILL

You can use the SKILL functions to run *Update Components And Nets* using either of the following methods.

- The lxUpdateComponentsAndNets SKILL function.
- A combination of the following functions, which let you specify what is generated and updated directly in SKILL.

  lxUpdateComponentsAndNetsStart

  lxGetAvailablePinLPPs

  lxGetPinNets

© 2023 Cadence Design Systems, Inc.

lxSetBoundaryOptions

lxSetAreaEstimationOptions

lxSetGenerateOptions

lxSetNetPinSpecs

lxSetUpdateOptions

lxUpdateComponentsAndNetsFinish

Related Topics

Checking a Design Hierarchy

Update Components and Nets Form

Design Check Against Source

Define Device Correspondence Form

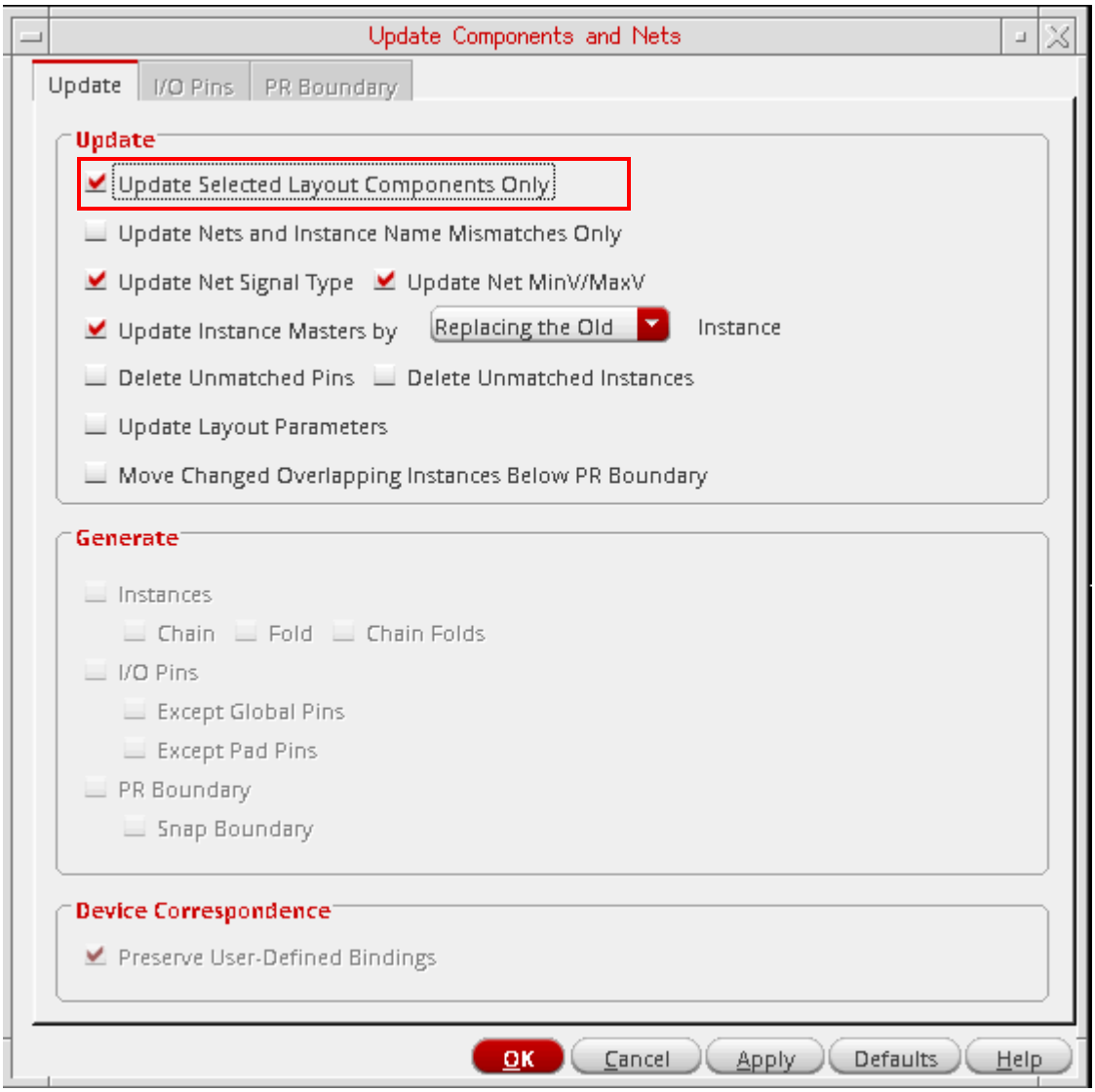
Updating Selected Layout Components

To update only selected instances and pins:

1. Select the *Update* of the Update Components and Nets form.

© 2023 Cadence Design Systems, Inc.

2. In the *Update* group box, select the *Update Selected Layout Components Only* check box.



The options to generate missing components, and consequently the *I/O Pins* and *PR Boundary* tabs, are turned off and grayed out.

Preserve User-Defined Bindings is switched on and grayed out.

3. To update only the connectivity information for the selected instances and pins, select the *Update Nets and Instance Name Mismatches Only* check box and proceed to step 9. If any pins have got swapped due to a schematic ECO, run the *Update* with *Update Nets and Instance Name Mismatches Only* selected to ensure that the

© 2023 Cadence Design Systems, Inc.

original pin positions are retained in the layout. To reflect the pin swaps in the layout, run the *Update* with *Update Nets and Instance Name Mismatches Only* deselected.

4. To update the signal types assigned to nets from the schematic view and the min and max voltages on the schematic nets, select the *Update Net Signal Type* check box.
5. To update instances with incorrect masters, select the *Update Instance Masters* check box and specify how an incorrect instance is handled:
   - By default, it is removed and replaced by an instance of the correct master in the same location.
   - When set to *Creating a New*, the command puts a marker on the instance with the incorrect master and renames it `name_old`. It then creates a new instance with the correct master and places it below the PR boundary.

     Because you are creating a new instance, you can optionally enable *Chain, Fold*, or *Chain Folds* in this mode. If the `disableFolding` environment variable is set and the folding threshold for the component type is set to `0`, folding is disabled.
6. To delete layout pins and instances that are not present in the schematic, select the *Delete Unmatched Pins* and *Delete Unmatched Instances* check boxes. When you delete unmatched pins, redundant nets and terminals are also deleted.
7. To update parameters and parameter values in the layout instances to match those on their schematic counterparts, select the *Update Layout Parameters* check box.
8. To automatically place instances below the PR boundary if they overlap other instances during the update, select the *Move Changed Overlapping Instances Below PR Boundary* check box. The overlapping instances are the ones that have changed in size, and if placed at their original location, may overlap other unchanged instances.
9. Click *OK* to update the selected instances and pins.

Related Topics

Update Components and Nets Form

Updating Nets and Instance Name Mismatches Only

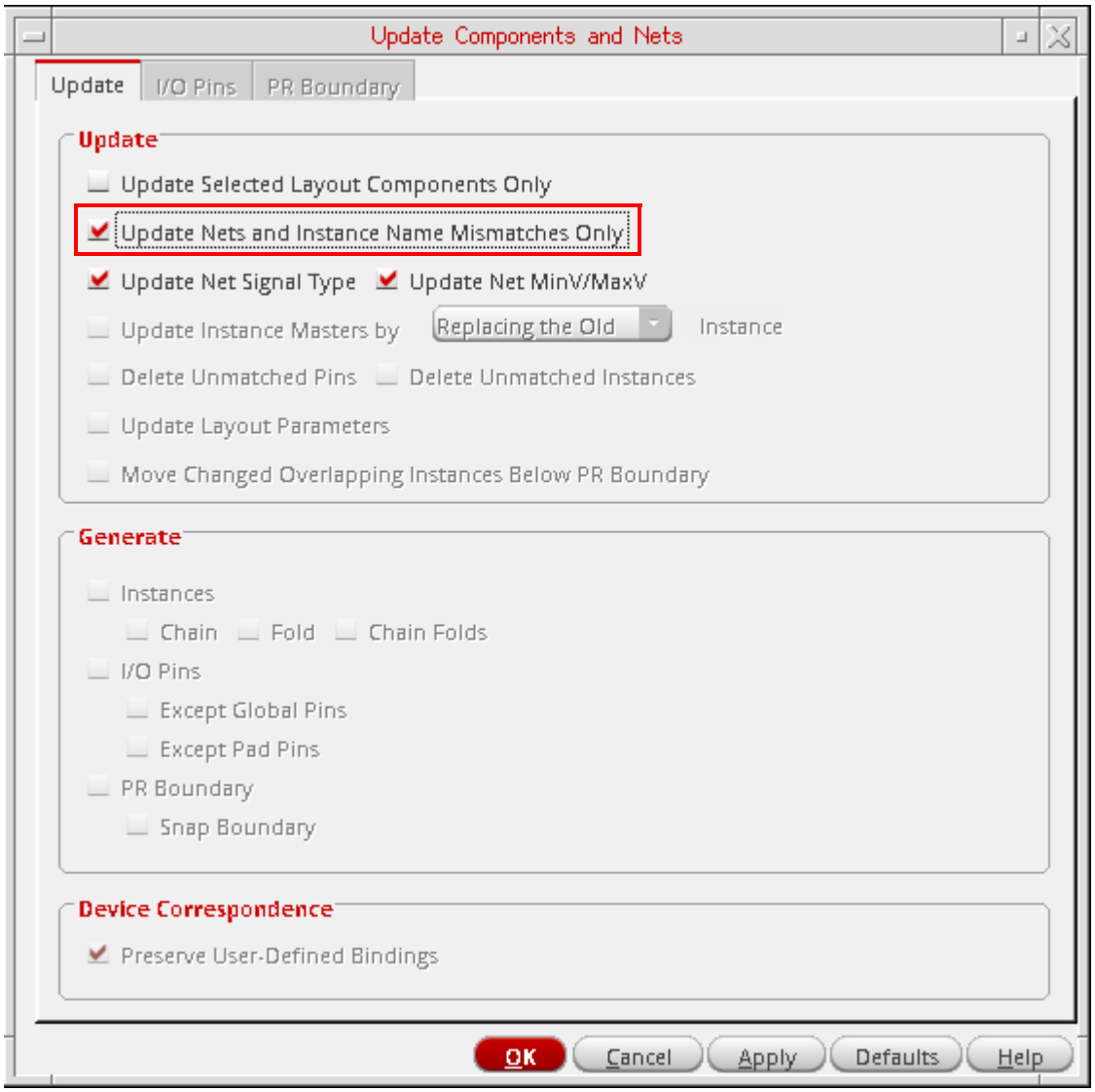
disableFolding

© 2023 Cadence Design Systems, Inc.

Updating Nets and Instance Name Mismatches Only

To update only the net assignments and instance and net names in your layout design without changing any other components:

1. Select the *Update* of the Update Components and Nets form.
2. In the *Update* group box, select the *Update Nets and Instance Name Mismatches Only* check box.



Preserve User-Defined Bindings is selected but inactivated, ensuring that these bindings are maintained. All other options on the form are inactive, except *Update Selected Layout Components Only* and *Update Net Signal Type*.

© 2023 Cadence Design Systems, Inc.

3. To update only selected instance and pins, select the *Update Selected Layout Components Only* check box.
4. To update the signal types assigned to nets and the min and max voltages on the schematic nets, select the *Update Net Signal Type* check box.
5. Click *OK*.

   Layout XL updates net assignments and instance, terminal, and net names to match those in the schematic. If any pins have got swapped due to a schematic ECO, running the *Update* with *Update Nets and Instance Name Mismatches Only* selected ensures that the original pin positions are retained in the layout. To reflect the pin swaps in the layout, run the *Update* with *Update Nets and Instance Name Mismatches Only* deselected.

Related Topics

Update Components and Nets Form

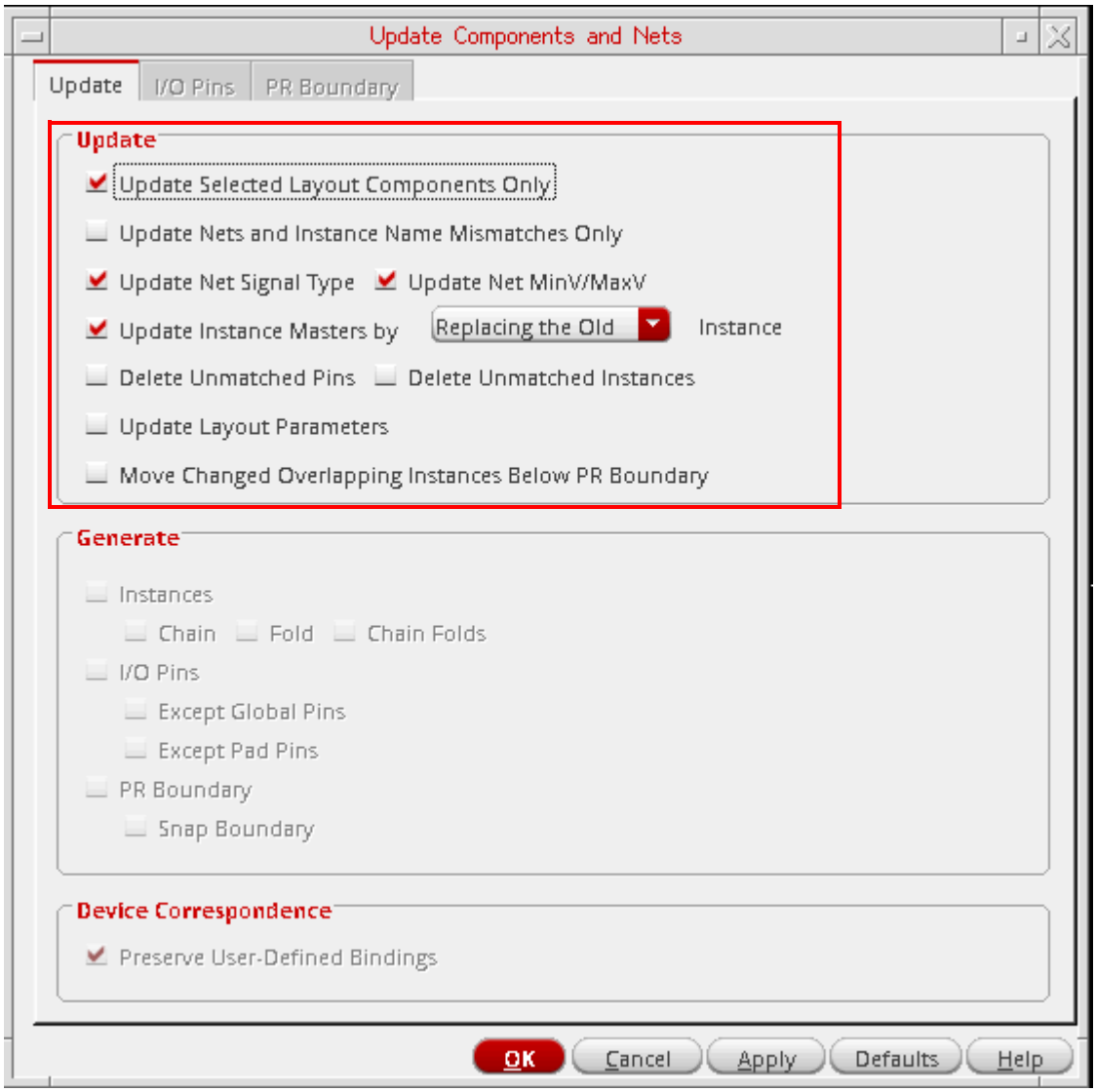
Specifying the Components to be Updated

When updating the whole design, you can specify that missing components are generated and existing components are regenerated. To do this:

1. Select the *Update* of the Update Components and Nets form.

© 2023 Cadence Design Systems, Inc.

2. Select the appropriate options in the *Update* group box to specify how the layout components that need to be changed or removed should be handled.



   a. To update only selected instances and pins, see Updating Selected Layout Components.

   b. To update only the connectivity information for the selected instances and pins, see Updating Nets and Instance Name Mismatches Only.

   c. To update the signal types assigned to nets from the schematic view, select the *Update Net Signal Type* option.

   d. To update instances with incorrect masters, select *Update Instance Masters* and specify how the incorrect instance should be handled.

© 2023 Cadence Design Systems, Inc.

- By default, it is removed and replaced by an instance of the correct master in the same location.
- When set to *Creating a New*, the command puts a marker on the instance with the incorrect master and renames it *name*_old. It then creates a new instance with the correct master and places it below the PR boundary.

  Because you are creating a new instance, you can optionally enable *Chain, Fold,* and *Chain Folds* in this mode. If the disableFolding environment variable is set and the folding threshold for the component type is set to 0, folding is disabled.

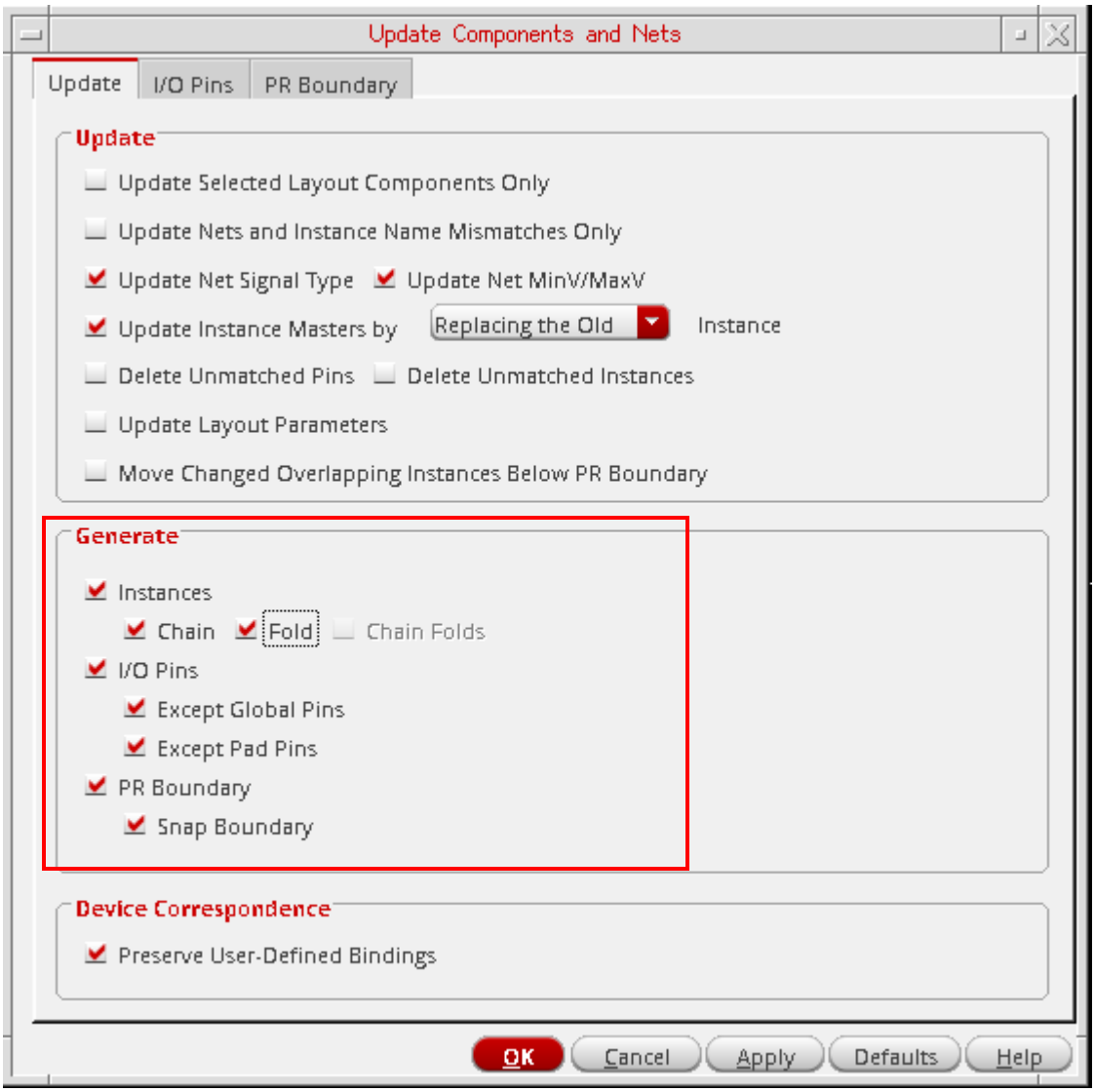
e. To delete layout pins and instances that are not present in the schematic, select the *Delete Unmatched Pins* and *Delete Unmatched Instances* options. When you delete unmatched pins, redundant nets and terminals are deleted at the same time.

f. To update parameters and parameter values in the layout instances to match those on their schematic counterparts, select the *Update Layout Parameters* option.

g. To automatically place instances below the PR boundary if they overlap other instances during the update, select the *Move Changed Overlapping Instances Below PR Boundary* option. The overlapping instances are the ones that have changed in size, and if placed at their original location, may overlap other unchanged instances.

© 2023 Cadence Design Systems, Inc.

3. Use the options in the *Generate* group box to regenerate instances, pins, and PR and snap boundaries.



   a. Select the *Instances*, *I/O Pins*, and *PR Boundary* options as required.

      Layout XL puts any missing instances and pins below the design boundary and updates their connectivity. Pins are automatically snapped to the placement grid.

   b. To chain any new transistors, select the *Chain* check box.

      New chains are not attached to existing chains. Devices that were chained prior to running the command are not moved from their existing positions; missing devices are added in the correct position in the chain and not under the place and route boundary.

© 2023 Cadence Design Systems, Inc.

c. To divide new transistors in the design into two or more folds, select the *Fold* check box.

If the `disableFolding` environment variable is set and the folding threshold for the component type is set to `0`, folding is disabled.

d. To chain the individual folds of a transistor, select the *Chain Folds* check box.

The *Chain Folds* check box is active only when the *Chain* check box is deselected and the *Fold* check box is selected. If the `disableFolding` environment variable is set and the folding threshold for the component type is set to `0`, folding is disabled.

If the *Chain* and *Fold* check boxes are deselected, and you have deleted all the folds of a folded transistor, *Update Components And Nets* generates a single device for that transistor. If you have deleted only some folds of a folded device, Layout XL creates markers only if the folds in a numbered sequence are missing. Otherwise, Layout XL does nothing about the missing folds of folded transistors.

All devices to be chained or folded must belong to a component type which has *Component class* set to *PMOS* or *NMOS* and values set for the *Active layer*, *Width parameter*, and *Folding threshold* parameters.

See Component Types Mode in CPH.

a. To avoid generating layout pins for global nets in the schematic, select the *Except Global Pins* check box.

b. To avoid generating layout pins for schematic pins that are connected to I/O pads, select the *Except Pad Pins* check box.

c. To generate a snap boundary, select the *Snap Boundary* check box.

4. To preserve any user-defined binding of devices between the schematic and the layout, select the *Preserve User-Defined Bindings*.

This option preserves user-defined one-to-one, many-to-many, many-to-one, and one-to-many device correspondence defined in the Define Device Correspondence form. It does not report missing devices or shapes within a bound group.

See Device Correspondence Updates.

Related Topics

Update Components and Nets Form

Define Device Correspondence Form

© 2023 Cadence Design Systems, Inc.

disableFolding

I/O Pins and PR Boundary Updates

You specify the pins to be generated on the **I/O Pins** of the Update Components and Nets form. The options and use model are the same as for the *I/O Pins* tab on the generate Layout form.

Layout XL generates any missing pins, places them below the design boundary, and updates their connectivity appropriately. If a layout instance includes a terminal but no pin for a global layout net (because, for example, you have deleted the pin), the pin is regenerated. All pins are automatically snapped to the placement grid.

Pin permutation is not preserved during the update. If any component in the design needs to be updated, the command resets the permutation status of all the cells to that originally stored in the schematic.

Physical-only terminals (those with the `physOnly` property set to `t`) created to support an implicit inherited connection defined using a net expression in the schematic are not updated. Physical-only terminals are deleted only if they have no connections.

You specify how the place and route boundary is updated on the **PR Boundary** of the Update Components and Nets form. The controls and use model are the same as for the *I/O Pins* tab on the Generate Layout form.

If you uncheck the *PR Boundary* option on the form, the existing place and route boundary is retained and is not automatically resized to take into account the updated design.

Related Topics

Update Components and Nets Form

Specifying the Place and Route Boundary During Layout Generation

Updating Layout Constraints

To update constraints from schematic to layout, do **one** of the following:

- From the layout window menu bar, choose *Connectivity – Update – Layout Constraints*.

© 2023 Cadence Design Systems, Inc.

➡ In the Constraint Manager toolbar, click the *Update Layout Constraints* button.



Constraints in the schematic are transferred to the top-level layout view. Constraints that have been created in the schematic but not yet saved are also transferred.

Any instances that have a parameter mismatch between the schematic and the layout views, will not have their parameters updated. To update the parameters of such instances, you may need to run *Update Layout Parameters*, *Update Schematic Parameters*, or *Update Components And Nets* with the parameter update option selected.

Layout Constraints Updates after Folding

If you fold instances in the layout view using one of the methods listed below, you must use the *Update Layout Constraints* command to ensure that all the transferred constraints have the correct folded members.

- *Generate Selected From Source*
- *Generate Folded Devices*
- Placement Planning incremental layout generation

The command does not update any constraints you created in the layout. These constraints might have incorrect folded members.

Related Topics

Constraint Manager Assistant

© 2023 Cadence Design Systems, Inc.

Layout Parameters Updates

Use the *Update Layout Parameters* command to update the parameter values of devices in the layout to match the values in the schematic. The software checks the parameter values on the specified devices in the schematic against the values in the layout and updates the layout parameters when it finds differences (unless you have set the `lvsIgnore` or `ignore` property on a device).

The updates are reported in a Layout XL Info window, along with information about any layout devices that contain properties that do not exist on their schematic counterparts. If updating parameters results in a change in the layout, Layout XL updates the connectivity and indicates any problems by drawing markers in the layout window.

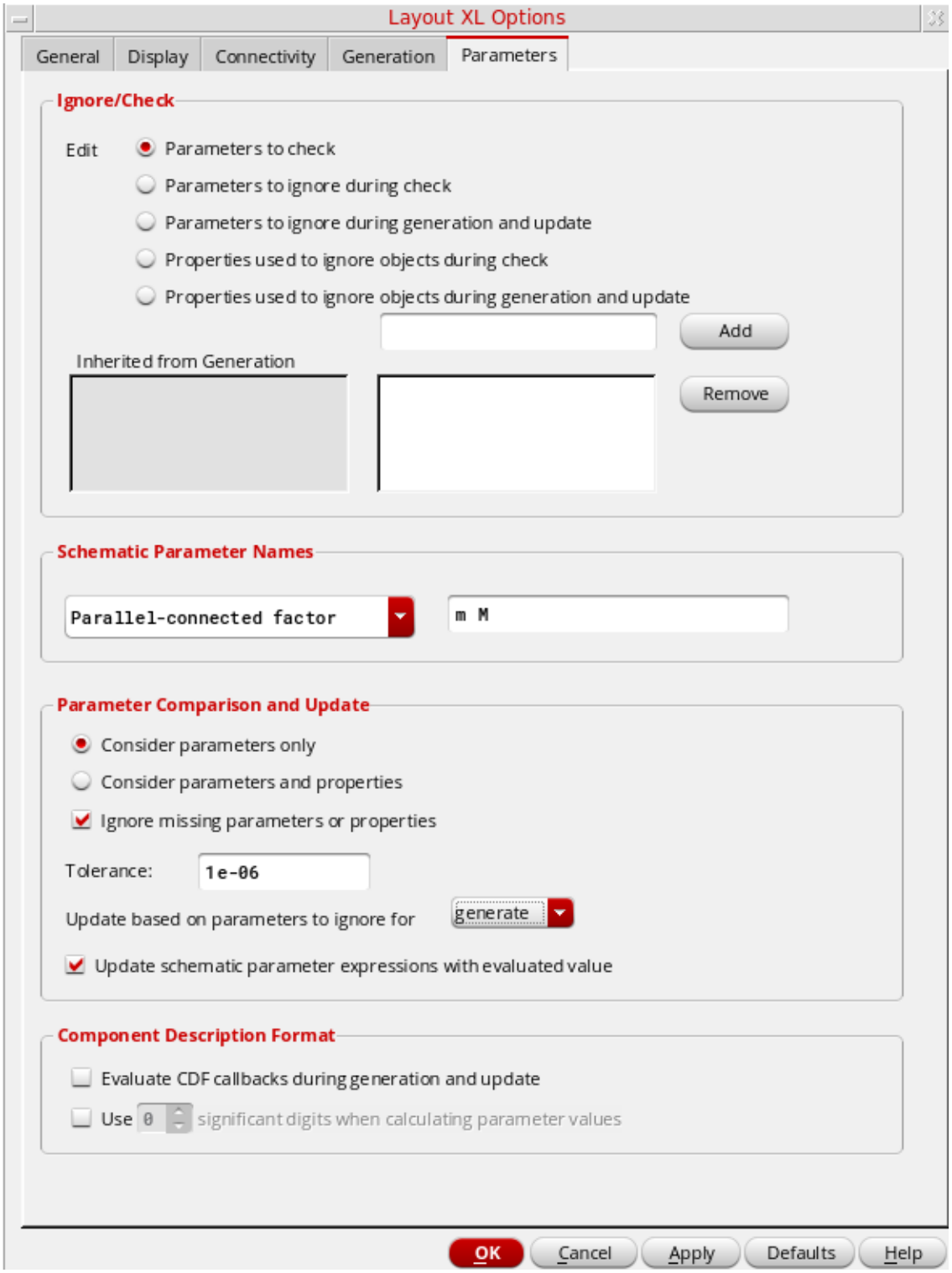
If you are using CDF callbacks, the command first applies the callback and then applies all the differing parameters on the schematic device.

To see a report on what would be updated without making any changes to your design, run *Check Against Source* before you run *Update Layout Parameters*.

You specify how the parameters and properties are updated on the Parameters of the Connectivity form.

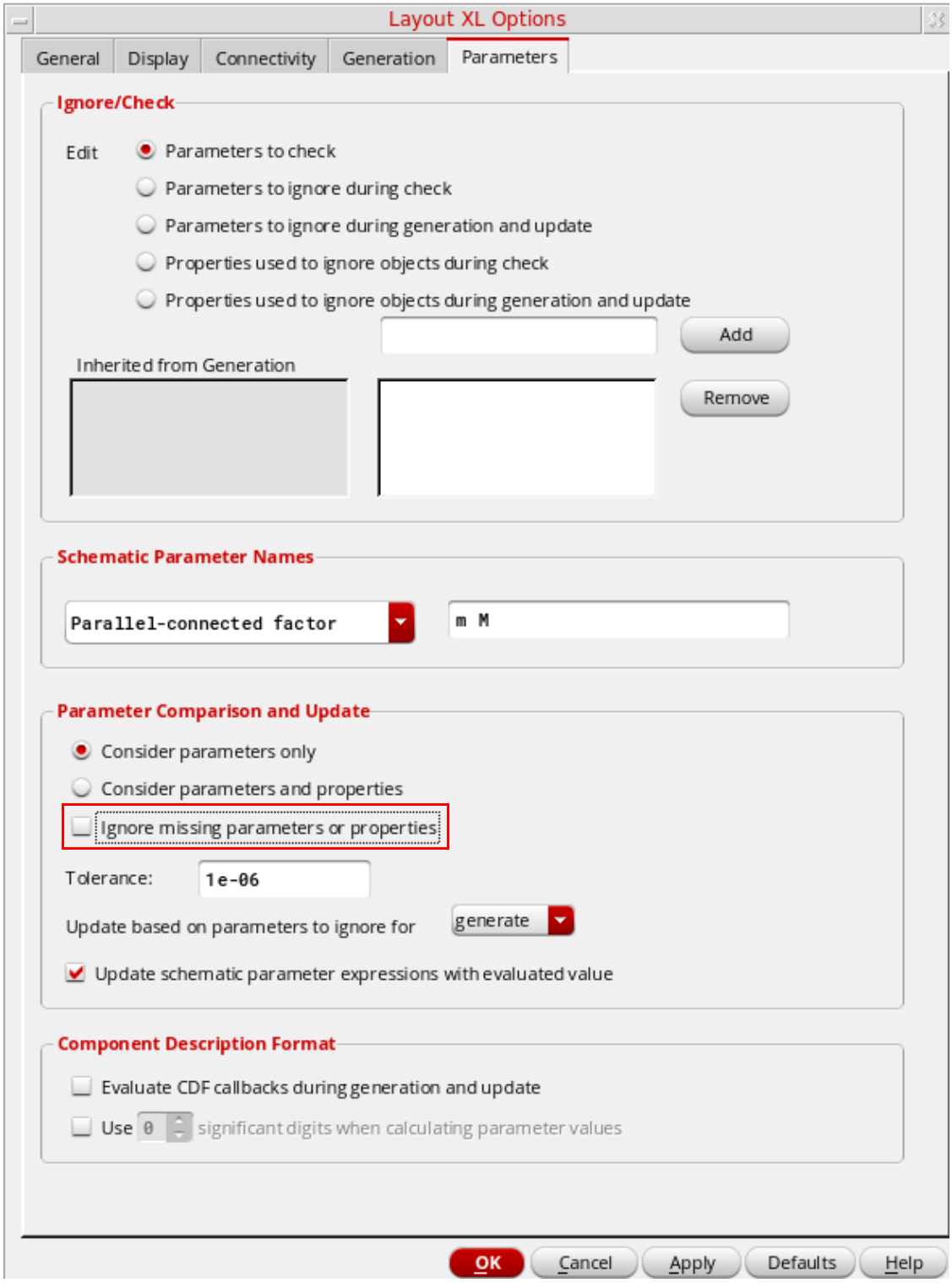
© 2023 Cadence Design Systems, Inc.

By default, the software updates layout parameter values only and ignores parameters that are present in the schematic but missing from the layout.



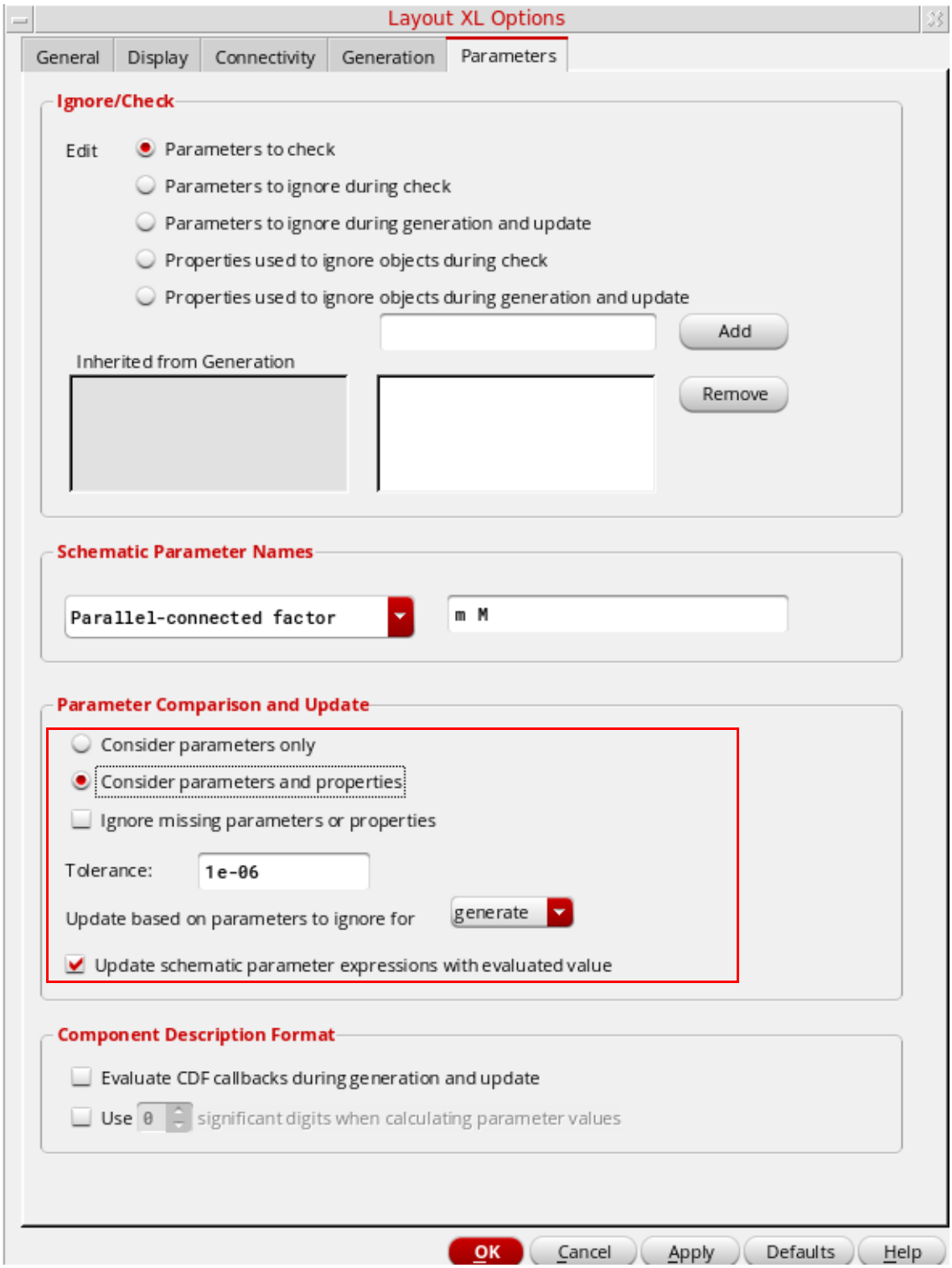
© 2023 Cadence Design Systems, Inc.

- To propagate any missing parameters from the schematic to the layout, switch off *Ignore missing parameters or properties*.



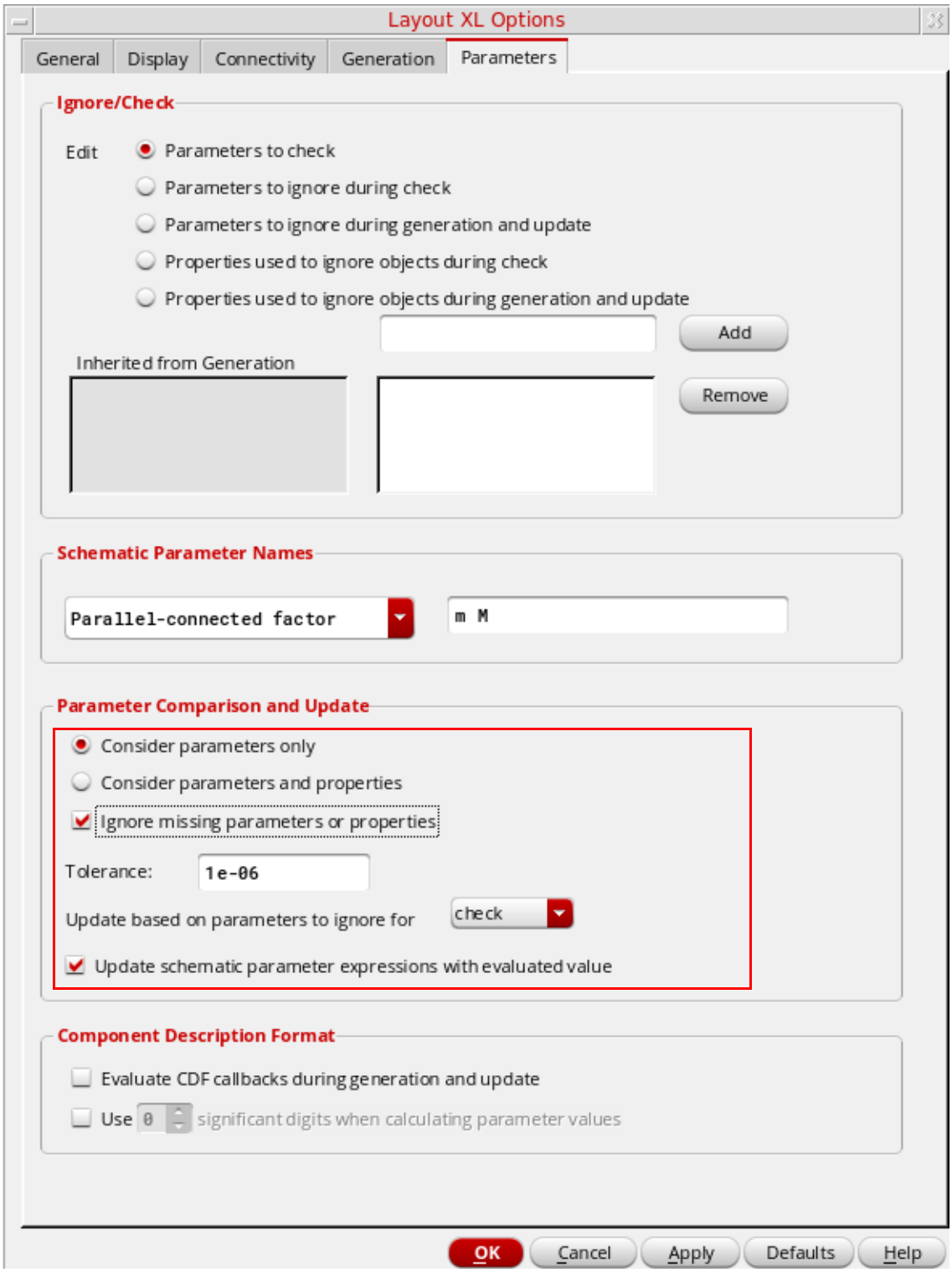
© 2023 Cadence Design Systems, Inc.

- To update parameters and properties, including missing properties, set the options as shown below.



© 2023 Cadence Design Systems, Inc.

- To update parameters and properties, ignoring those that have not been reported during *Check Against Source*, set the options as shown below.



© 2023 Cadence Design Systems, Inc.

Limitations

Update Layout Parameters does not:

- Delete any parameters or properties from the layout view.
- Update the parameters of complex devices defined using the `lxCombination` property, because this might involve deleting and creating instances in the layout view. Use *Update Components And Nets* instead.
- Update split transistors if the sizes of the resulting transistors are different, because such transistors may be the result of manual splitting. If the layout figure selected for update is a transistor that has been folded, Layout XL displays a message that says that it is not updating the parameters.
- Copy `netSet` properties, which are used to specify the new value of a global signal when you are using inherited connections to assign more than one value to a global net.

Related Topics

Connectivity Form

lvsIgnore Property

ignore Property

Checking a Layout Against a Schematic

Updating Layout Parameter Values in Preselection Mode

To update parameter values for devices already selected,

1. In the layout canvas, select the devices you want to update.
2. From the layout window menu bar, choose *Connectivity – Update – Layout Parameters*.

   If you are using CDF callbacks, *Update Layout Parameters* applies the callback first and then updates the parameters on the selected layout devices to match the values of the corresponding devices in the schematic.
3. Press `Esc` to exit the command.

© 2023 Cadence Design Systems, Inc.

Related Topics

Connectivity Form

Updating Layout Parameter Values in Postselection Mode

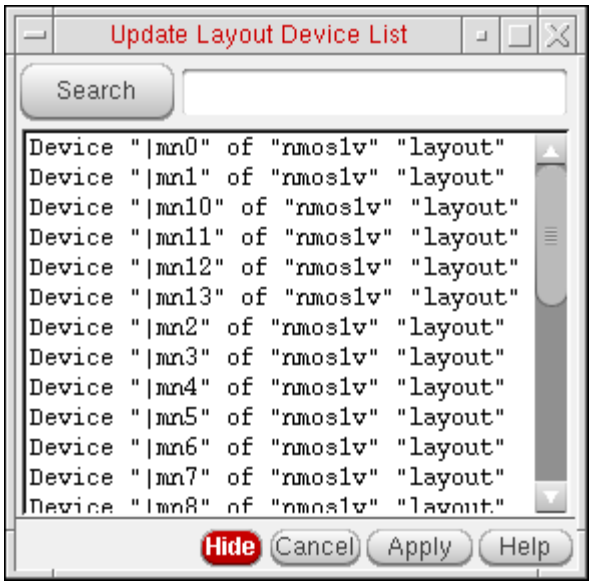
To update parameter values in postselection mode,

1. From the layout window menu bar, choose *Connectivity – Update – Layout Parameters*.

   The CIW and the layout window prompt you to select the devices whose parameter values you want to update.

2. To see a list of the layout devices in the design, move the cursor into the layout canvas and press `F3`.

   The DFM Settings form is displayed.



3. Select a device in the form, in the layout canvas, or in the schematic window.

   Layout XL highlights the device in all the three locations.

4. Click *Apply* in the form or move the cursor into the layout canvas and press `Return`.

   If you are using CDF callbacks, *Update Layout Parameters* applies the callback first and then updates the parameters on the selected layout devices to match the values of the corresponding devices in the schematic.

© 2023 Cadence Design Systems, Inc.

5. Press `Esc` to exit the command.

Related Topics

Connectivity Form

Updating Parameter Values for Folded Devices

To update parameter values for folded devices:

➤ Choose the *Update width parameter method* option on the *Generation* tab of the Connectivity form before you run *Update Layout Parameters*.



Tip

Check the size of folded devices manually to avoid overriding the specified size.

Related Topics

updateParamChangeNumFolds

Excluding a Specific Parameter from the Update

To prevent a specific layout parameter from being updated by the *Update Layout Parameters* command:

1. From the layout window menu bar, choose *Options – Connectivity* and click the *Parameters* tab.
2. Select *Parameters to ignore during generation and update*, type the name of the parameter you want to ignore in the field provided, and click *Add*.

   The parameter name is added to the list and will not be updated when you run *Update Layout Parameters*.
3. From the layout window menu bar, choose *Connectivity – Update – Layout Parameters*.

   The layout parameters are updated to match the values in the schematic; however, the parameter you specified to be ignored is not updated.

Alternatively, you can update the LAM file to specify the parameters to be ignored.

© 2023 Cadence Design Systems, Inc.

Related Topics

Connectivity Form

LAM File Logical Section

Schematic Parameter Updates

Use the *Update Schematic Parameters* command to update the parameter values of devices in the schematic to match the values in the layout.

To use the *Update Schematic Parameters* command, you must have the schematic window open in the edit mode. The layout window, on the other hand, could be open in the read mode.

The software checks the values of the parameters of the specified devices in the schematic against the values in the layout and updates the schematic parameters when it finds differences (unless you have set the `lvsIgnore` or `ignore` property on a device).

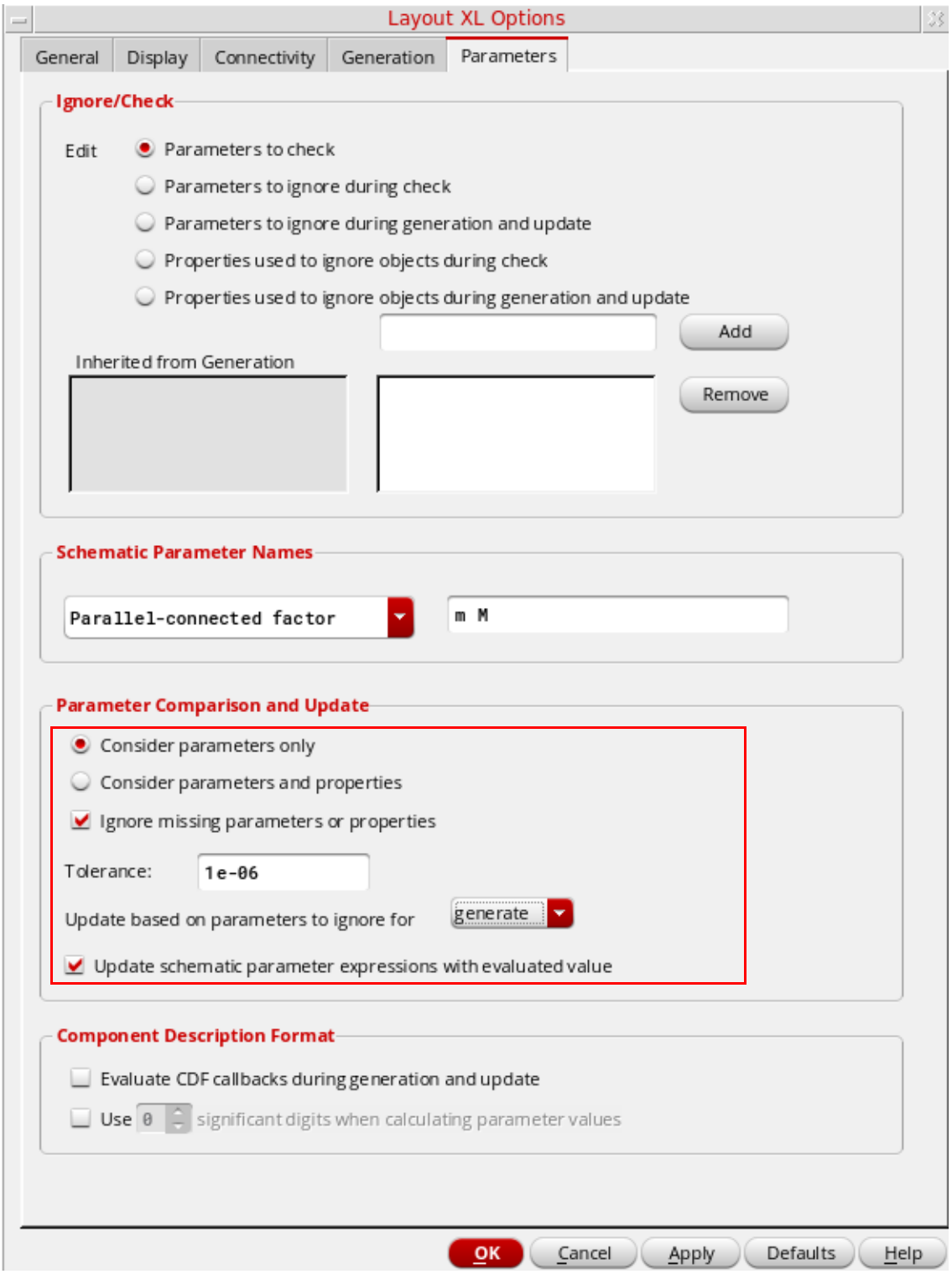
The updates are reported in a Layout XL Info window. If you are using CDF callbacks, the command applies the callback first then applies all differing parameters on the layout device.

To see a report on what would be updated without making any changes to your design, run *Check Against Source* before you run *Update Schematic Parameters*.

Use the LAM file to specify how the parameters and properties are updated. Alternatively, specify this on the *Parameters* of the Connectivity form.

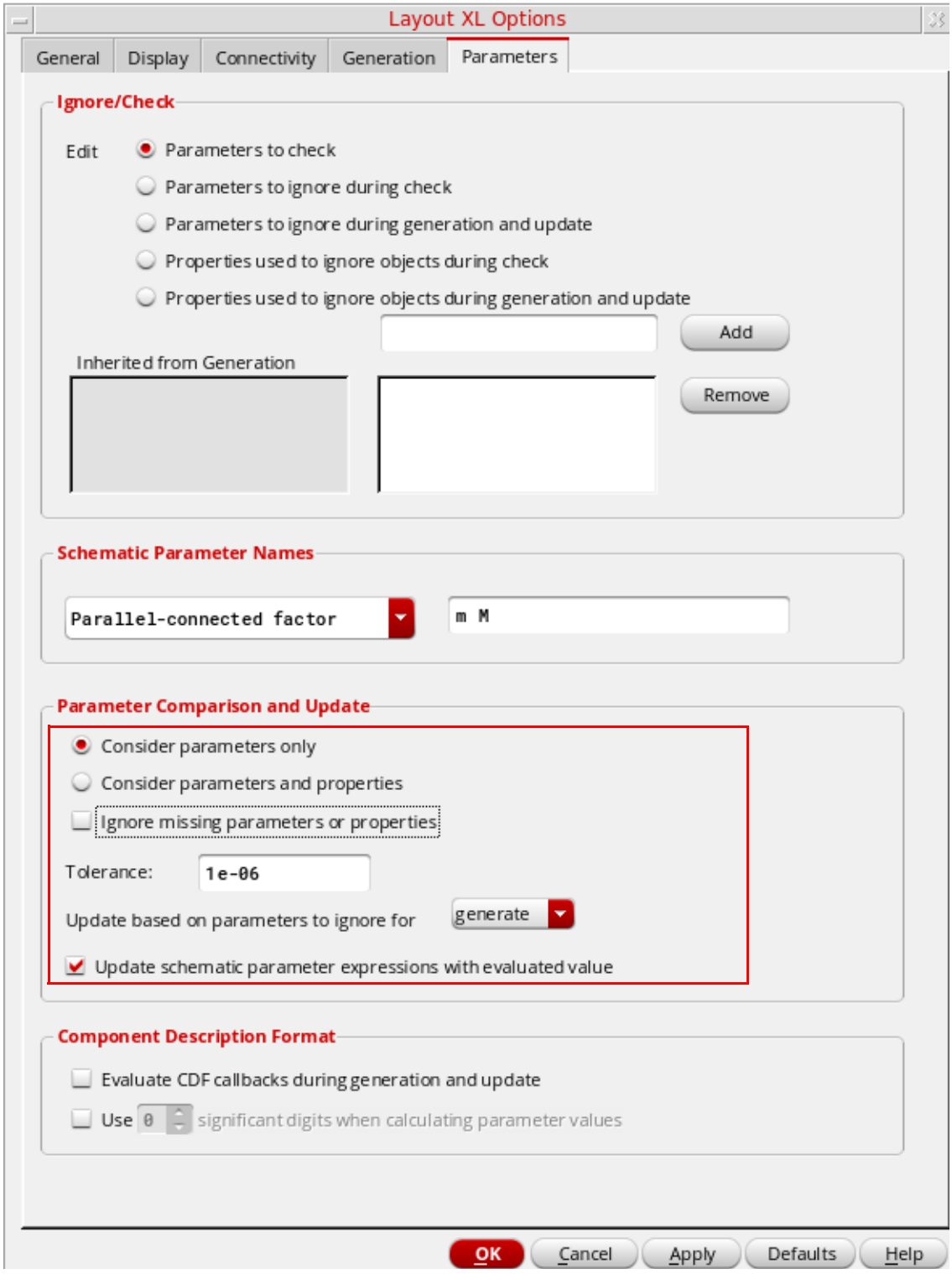
© 2023 Cadence Design Systems, Inc.

By default, the software updates schematic parameter values only and ignores parameters that are present in the layout but missing from the schematic.



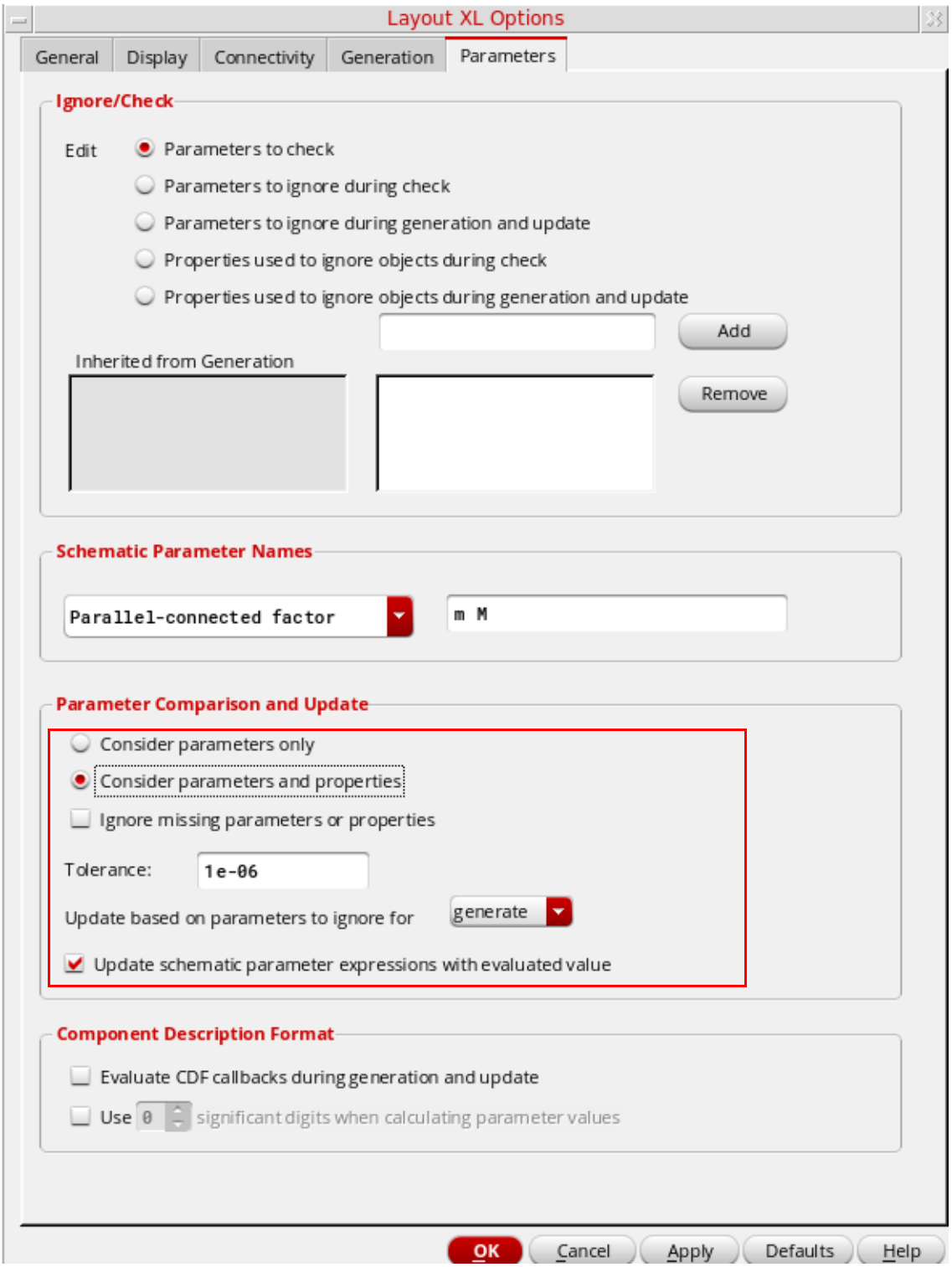
© 2023 Cadence Design Systems, Inc.

To propagate any missing parameters from the layout to the schematic, switch off *Ignore missing parameters or properties*.



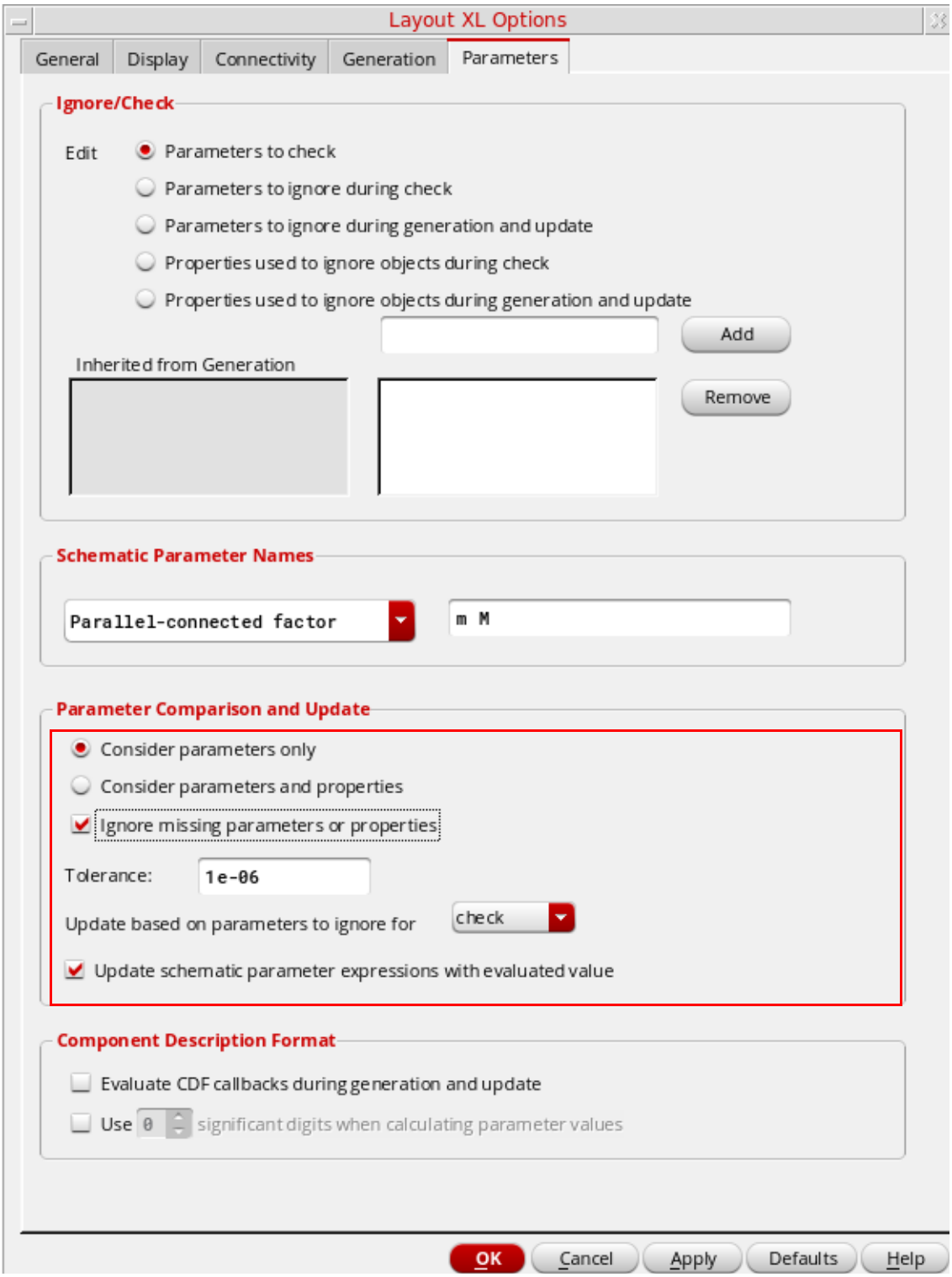
© 2023 Cadence Design Systems, Inc.

To update parameters and properties, including missing properties, set the options as shown below.



© 2023 Cadence Design Systems, Inc.

To update parameters and properties, ignoring those that have not been reported during *Check Against Source*, set the options as shown below.



© 2023 Cadence Design Systems, Inc.

Related Topics

lvsIgnore Property

ignore Property

Connectivity Form

Checking a Layout Against a Schematic

Ignore Parameters and Terminals

Updating Schematic Parameter Values in Preselection Mode

To update the schematic parameters of devices already selected:

1. In the schematic or layout window, select the devices you want to update.
2. From the layout window menu bar, choose *Connectivity – Update – Schematic Parameters*.

   If you are using CDF callbacks, Layout XL applies the callback first and then updates the parameters on the selected schematic devices to match the values of the corresponding devices in the layout (unless you have set the `lvsIgnore` or `ignore` property on a device).

   An Information window reports any changes that are made.
3. Press `Esc` to exit the command.

Related Topics

lvsIgnore Property

ignore Property

Updating Schematic Parameter Values in Postselection Mode

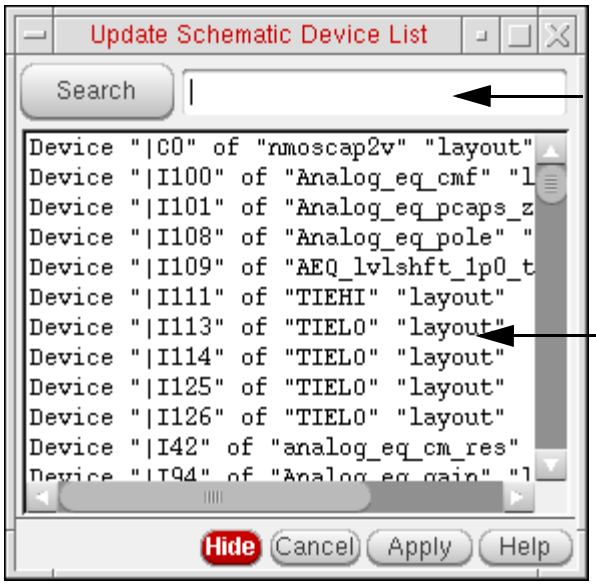
1. From the layout window menu bar, choose *Connectivity – Update – Schematic Parameters*.

   The CIW and layout window prompt you to select the devices whose parameters you want to update.

© 2023 Cadence Design Systems, Inc.

2. To see a list of the layout devices in design, move the cursor to the layout canvas and press `F3`.

   The DFM Settings form is displayed.



3. Select a device in the form, in the layout window, or in the schematic window.

   Layout XL highlights the device in all three locations.

4. Click *Apply* in the form or move the cursor into the layout canvas and press `Return`.

   If you are using CDF callbacks, Layout XL applies the callback first and then updates the parameters on the selected schematic devices to match the values of the corresponding devices in the layout (unless you have set the `lvsIgnore` or `ignore` property on a device).

   An Information window reports any changes that are made.

5. Press `Esc` to exit the command.

Related Topics

lvsIgnore Property

ignore Property

© 2023 Cadence Design Systems, Inc.

Device Correspondence Updates

For each schematic and layout pair, the correspondence between the components in the views is determined by the Layout XL binder engine.

■ Initial Binding

When Layout XL is started, the binder binds by respecting the existing bindings.This means only those bindings that were previously saved are retained when a new Layout XL session is started.

When components are generated in the layout, the system propagates the connectivity to the layout view and sets the correspondence between the generated instances and their schematic counterparts. This correspondence is reported in the CIW, as shown below, when you start Layout XL.

```
INFO (BND-1003): Binder initialization finished:

                bound       unbound     ungenerated
  terminals         7             2               0
  nets             29            10               0
  instances        60             0              23
```

The report displayed above indicates some unbound terminals and nets in the design, suggesting the correspondence is not perfect. In such cases, an additional warning message is issued in CIW, as shown in the figure below, suggesting possible methods for improving the bindings.

```
*WARNING* (BND-2005): Layout instances and terminals do not match the source.
Use 'XL Status' in the Navigator or run 'Connectivity - Check - Against Source' for more details.
If the bindings are correct, run the 'Update from Source' commands,
otherwise run 'Connectivity - Update - Binding'.
```

■ Incremental Binding

Device correspondence and connectivity information are updated incrementally by the system, whenever required, for example, when instances are created in the design. However, for incremental binding to occur, the layout device to be bound must have a single, unique connectivity match in the schematic with the same master.

Instances created in the layout by running *Generate All From Source*, *Generate Selected From Source*, or *Generate Clones* commands continue to be fully bound to the schematic.

Because the incremental binder is physical-connectivity-aware, instances connected by shapes that are not assigned to a net are recognized as interconnected. Therefore, when

© 2023 Cadence Design Systems, Inc.

incrementally binding such instances, Layout XL automatically creates and assigns nets to the instance terminals, if required.

You can also choose to switch off the incremental binder, if required. To do this, you must set the `bindIncr` environment variable to `nil`. By doing so, any unbound layout instances or pins will remain unbound within the Layout XL session. On the other hand, if you set `bindIncrConn` to `nil`, any unbound layout instances or pins will be bound by name even when the devices do not have exactly the same master or have different connectivity.

When you run the schematic editor *Check and Save* command during a Layout XL session, the binder is reinitialized to take account of any changes in the schematic connectivity. This can lead to the creation of new bindings between schematic and layout components.

■ Ignoring Instances for Binding

If you have any unbound instances that you have edited and want to retain as unbound even after running incremental binding, you can set an `ignore` property on them. To do this right-click the instance and choose the *Add Ignore* shortcut command.

The selected instance will have an *ignore* property set due to which the instance will be ignored for binding. However, any existing bound instances will continue to get updated, if required, during incremental binding.

Alternatively, you can set `bindIncrAddIgnore` to `t`. This will set an ignore property on any unbound instances that have been edited and would otherwise be considered for binding during an incremental binder run.

After the ignore property is set, the instances become unavailable for all binding—initial and incremental. However, any unbound instances that have not been edited, will not have the ignore property set. Therefore, these instances will continue to be available for binding during the next binding run.

To remove the `ignore` property from an unbound instance to make it available for binding, right-click the instance and select the *Remove Ignore* shortcut command.

Related Topic

Working with the Binder to stay XL-compliant Rapid Adoption Kit

bindIncr

bindIncrConn

bindIncrAddIgnore

© 2023 Cadence Design Systems, Inc.

User-Defined Device Correspondence

If you need to change the system-generated device correspondence manually, you can do so by using the *Define Device Correspondence* command available in the *Connectivity* menu. User-defined device correspondence is persistent between Layout XL sessions, and can be preserved during *Generate All From Source* and *Update Components And Nets* by selecting the *Preserve User-Defined Bindings* option in the Generate Layout and Update Components and Nets forms, respectively.

The *Define Device Correspondence* command and, where required, the Assign layout instance terminals form correctly propagate connectivity to the layout implementation. There is no need to run the *Update Components And Nets* command after defining the device correspondence. To update the connectivity of devices that are already correctly bound, you need to only run the *Update Components And Nets* command.

Check Against Source will report issues found only in user-defined one-to-one bindings that are created by using the Define Device Correspondence form. The command will not report any issues in one-to-many, many-to-one, or many-to-many bindings.

Related Topics

Generate Layout Form

Update Components and Nets Form

Assign Layout Instance Terminals Form

Define Device Correspondence Form

Dynamic Updates to the Define Device Correspondence

The Define Device Correspondence form shows the current correspondence between instances and terminals in the schematic and layout views. The form is dynamically updated to reflect the changes you make.

Note the following:

- For performance reasons, the form does not update dynamically if you select more than 100 objects from the design canvas. To define device correspondence for a large number of objects, select them in the form itself.
- The *All Terminals* and *Unbound Terminals* filters show the terminals in the design, not the pin figures that typically represent them in the layout window.

© 2023 Cadence Design Systems, Inc.

- If you delete a pin figure in the layout window, the corresponding terminal is removed from the *Define Device Correspondence* form only if the net connected to this terminal has no other connections. If the net has other connections, the terminal is not removed.

  Similarly, if you delete an instance, the terminals connected to the nets of the instance terminal are deleted only if they are no longer connected in the design.

The procedures that follow describe how to manually define bindings between instances by using the *Define Device Correspondence* form. These procedures apply equally for terminals as well.

You can also use the *Bind* (or *Unbind)* command available in the shortcut menu of the selected instance, mosaic, or terminal to bind (or unbind) it.

See Binding Using the Bind Command in the Shortcut Menu.

Manually-defined bindings or user-defined bindings can be quickly identified using the dark blue icon () against their name in the *Define Device Correspondence* form.

Handling Transparent Layout Hierarchy

Define Device Correspondence supports transparent layout hierarchy. What this means is that the binder can bind to the instances and nets within a transparent instance master.

When descended into a transparent layout hierarchy:

- The layout tree of the Define Device Correspondence form updates to display only the instances that are within context of the transparent instance that you have descended into.
- The schematic tree of the Define Device Correspondence form updates to display only the unbound instances and the bound instances that are within context of the transparent instance that you have descended into.

The options under the *Update Layout Instance* group box on the Define Device Correspondence form are disabled when descended into the transparent layout hierarchy.

See Transparent Layout Hierarchy Generation.

Related Topics

Define Device Correspondence Form

© 2023 Cadence Design Systems, Inc.

Defining a One-to-One Device Correspondence

Defining a One-to-Many Device Correspondence

Defining a Many-to-One Device Correspondence

Defining a Many-to-Many Device Correspondence

Assigning Layout Instance Terminals to Nets

Deleting a Device Correspondence

Working with the Binder to stay XL-compliant Rapid Adoption Kit

Defining a One-to-One Device Correspondence

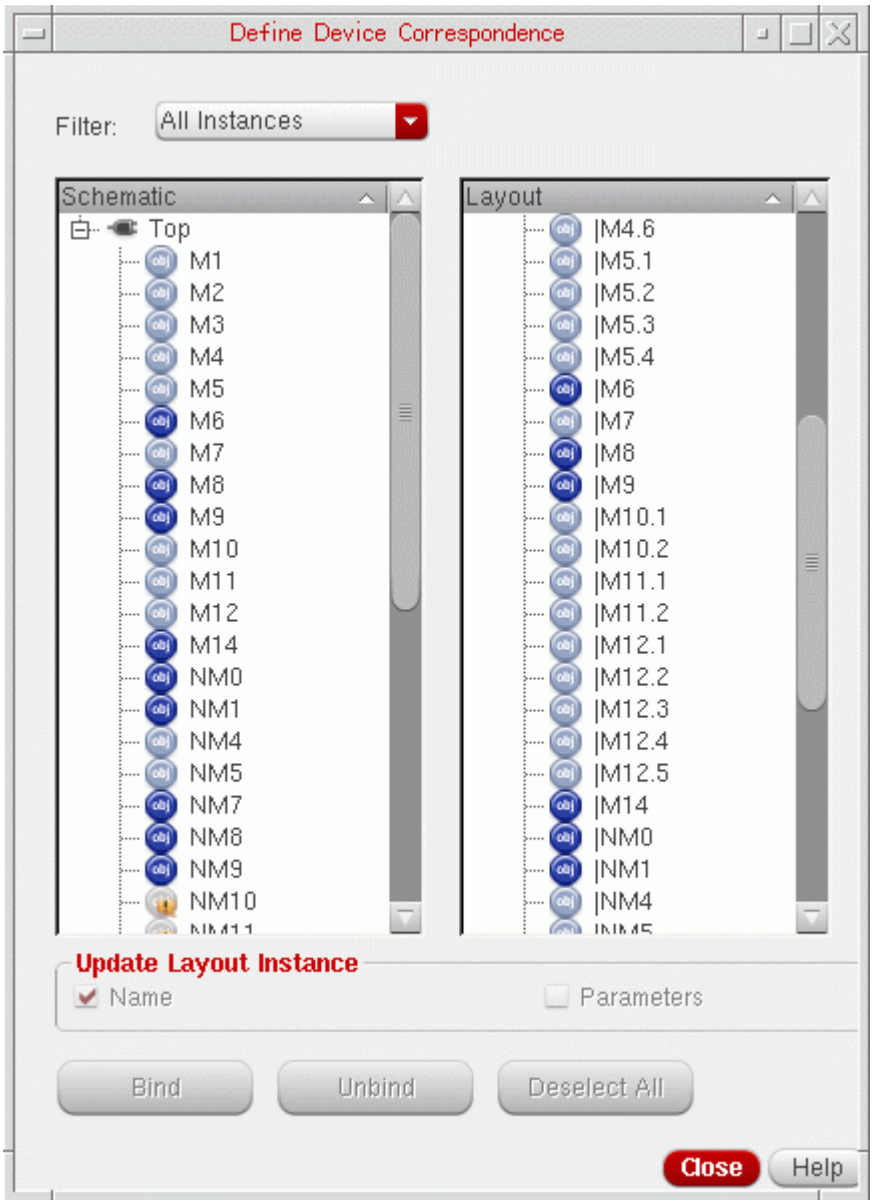
To create a one-to-one device correspondence, you must select only a single schematic device and a single layout device at a time and click *Bind*. If you select multiple schematic and layout devices and click *Bind*, Layout XL creates a single many-to-many correspondence and NOT multiple one-to-one bindings.

To define a correspondence between a single schematic instance (or terminal) and a single layout instance (or terminal),

1. From the layout window menu bar, choose *Connectivity – Define Device Correspondence*.

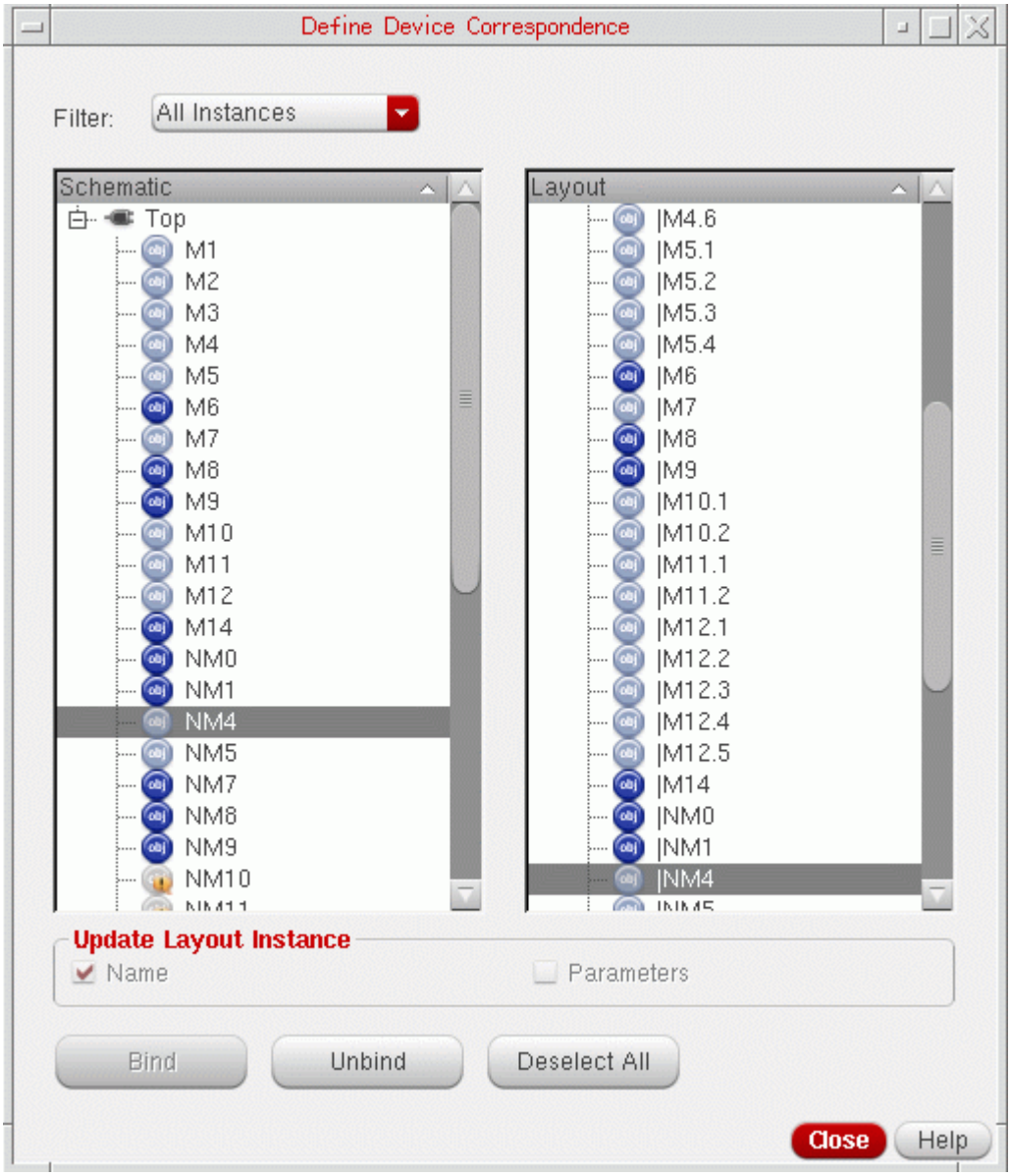
© 2023 Cadence Design Systems, Inc.

The Define Device Correspondence form appears.



© 2023 Cadence Design Systems, Inc.

2. Select an unbound instance from the *Schematic* tree and an unbound instance from the *Layout* tree.



You can only bind the unbound instances. To only see the unbound instances, choose *Unbound Instances* from the *Filter* cyclic field. To see only the unbound terminals, choose *Unbound Terminals*.

3. (Optional) Use the two *Update Layout Instance* options at the bottom of the form to update the name and parameter values of the layout instance to match those in the schematic instance.

© 2023 Cadence Design Systems, Inc.

4. Click *Bind* to create a correspondence between the selected schematic instance and the selected layout instance.

   - If the instance masters match, the correspondence is updated and the connectivity from the schematic is propagated to the layout.
   - If the instance masters do not match, the Assign layout instance terminals form is displayed. This form lets you assign layout nets to the unbound instance terminals in the selected layout instance.

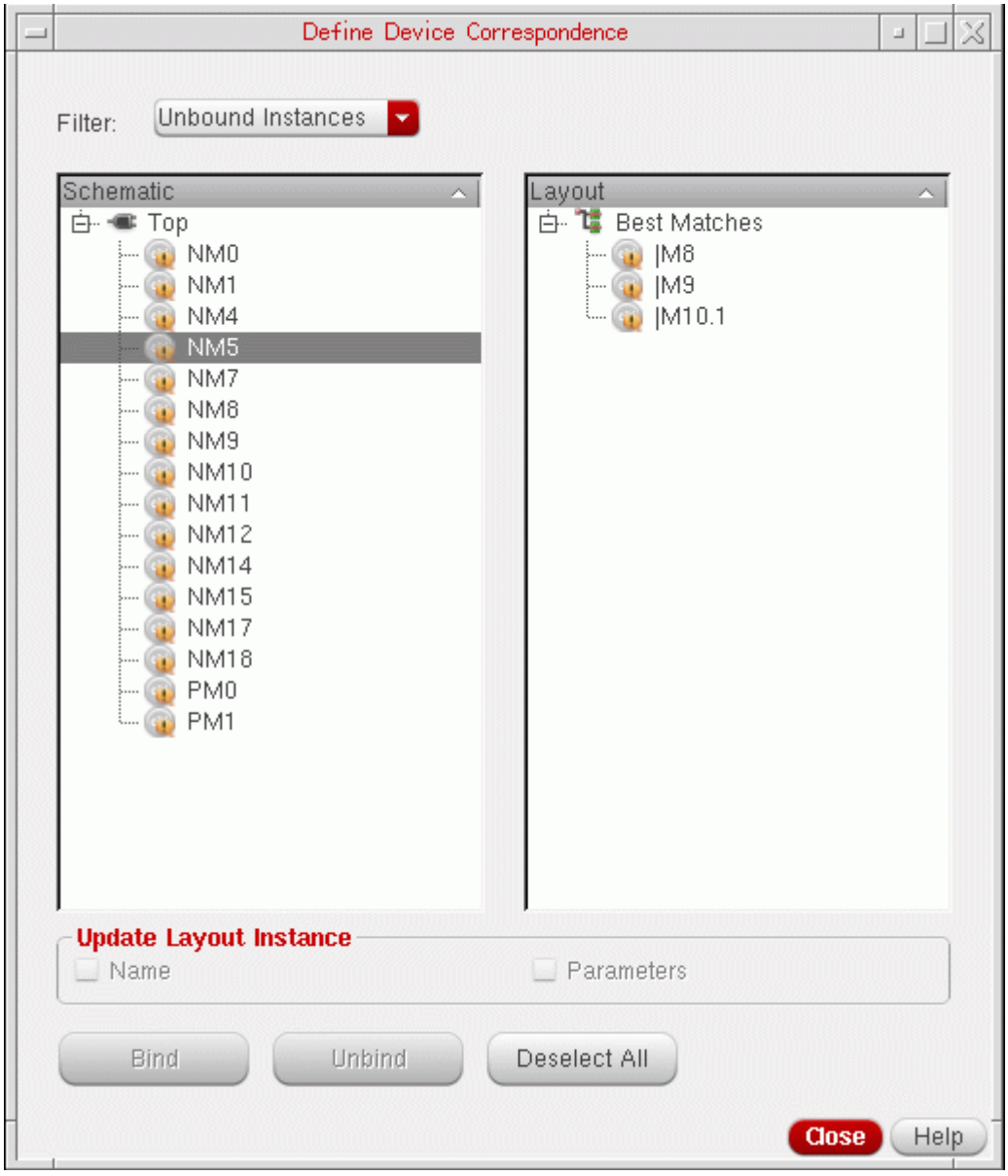
Showing Best Matches

Use the *Show Best Matches* command to display only the best matches—the instances that have the correct master—for an unbound schematic or a layout instance. To list the best matches for an unbound schematic or a layout instance:

➡ Double-click an unbound instance from the *Schematic* or *Layout* tree.

© 2023 Cadence Design Systems, Inc.

Alternatively, you can right-click the unbound *Schematic* or *Layout* instance for which you want to display the best matches and choose the *Show Best Matches command* from the menu.



The corresponding best matches for the selected instance are displayed in the other column. For example, if you selected an unbound *Schematic* instance, such as `NM5` in the figure above, its corresponding best matches are displayed in the Layout tree.

© 2023 Cadence Design Systems, Inc.

You can then select the appropriate best match for defining a one-to-one device correspondence or select appropriate best matches for defining a one-to-many device correspondence.

Related Topics

Define Device Correspondence Form

Assign Layout Instance Terminals Form

Assigning Layout Instance Terminals to Nets

Defining a One-to-One Device Correspondence

Defining a One-to-Many Device Correspondence

Defining a One-to-Many Device Correspondence

To define a correspondence between a single schematic instance and multiple layout instances:

1. From the layout window menu bar, choose *Connectivity – Define Device Correspondence*.

   The Define Device Correspondence form appears.

2. Select an unbound instance from the *Schematic* tree and the unbound instances from the *Layout* tree to which the selected schematic instance needs to be bound.

   You can only bind unbound devices. To only see the unbound instances, choose *Unbound Instances* from the *Filter* cyclic field; to see only the unbound terminals, choose *Unbound Terminals*.

3. (Optional) Use the two *Update Layout Instance* options at the bottom of the form to update the name and parameter values of the layout instance to match those in the schematic instance.

4. Click *Bind* to create a correspondence between the selected schematic instance and the selected layout instances.

   Layout XL binds the instances, but needs more information to propagate the connectivity correctly. Therefore, the Assign layout instance terminals Form form is displayed, which you can use to assign layout nets to the unbound instance terminals in the selected layout instance. For more information, see Assigning Layout Instance Terminals to Nets.

© 2023 Cadence Design Systems, Inc.

Related Topics

Define Device Correspondence Form

Assign Layout Instance Terminals Form

Assigning Layout Instance Terminals to Nets

Defining a Many-to-One Device Correspondence

To define a correspondence between multiple schematic instances and a single layout instance (for example, binding two schematic transistors to a single inverter in the layout):

1. From the layout window menu bar, choose *Connectivity – Define Device Correspon*dence.

   The Define Device Correspondence form appears.

2. Select the unbound instances from the *Schematic* tree and the unbound instance from the *Layout* tree to which the selected schematic instances need to be bound.

   You can only bind the unbound instances. To only see the unbound instances, choose *Unbound Instances* from the *Filter* cyclic field; to see only the unbound terminals, choose *Unbound Terminals*.

3. Click *Bind* to create a correspondence between the selected schematic instances and the selected layout instance.

   Layout XL binds the instances, but needs more information to propagate the connectivity correctly. Therefore, the Assign layout instance terminals form is displayed, which you can use to assign layout nets to the unbound instance terminals in the selected layout instance. For more information, see Assigning Layout Instance Terminals to Nets.

4. When you have finished assigning the layout instance terminals, click *Close* to close the form.

   The schematic and layout correspondence is updated and the connectivity you defined is propagated to the layout.

Related Topics

Define Device Correspondence Form

Assign Layout Instance Terminals Form

© 2023 Cadence Design Systems, Inc.

Assigning Layout Instance Terminals to Nets

Defining a Many-to-Many Device Correspondence



Connectivity errors introduced while defining a many-to-many device correspondence are reported by Check Against Source but cannot be corrected automatically by Update Components And Nets. This is because it is impossible to determine the correct terminal binding in a many-to-many relationship.

To define a correspondence between multiple schematic instances and multiple layout instances:

1. From the layout window menu bar, choose *Connectivity – Define Device Correspondence*.

   The Define Device Correspondence form appears.

2. Select the instances to be bound from the *Schematic* and the *Layout* trees.

   You can only bind unbound devices. To only see the unbound instances, choose *Unbound Instances* from the *Filter* cyclic field; to see only the unbound terminals, choose *Unbound Terminals*.

3. Click *Bind* to create a correspondence between the selected schematic instances and the selected layout instances.

   Layout XL binds the instances, but needs more information to propagate the connectivity correctly. Therefore, the Assign layout instance terminals form is displayed, which you can use to assign layout nets to the unbound instance terminals in the selected layout instance. For more information, see Assigning Layout Instance Terminals to Nets.

Related Topics

Define Device Correspondence Form

Assign Layout Instance Terminals Form

Assigning Layout Instance Terminals to Nets

© 2023 Cadence Design Systems, Inc.

Assigning Layout Instance Terminals to Nets

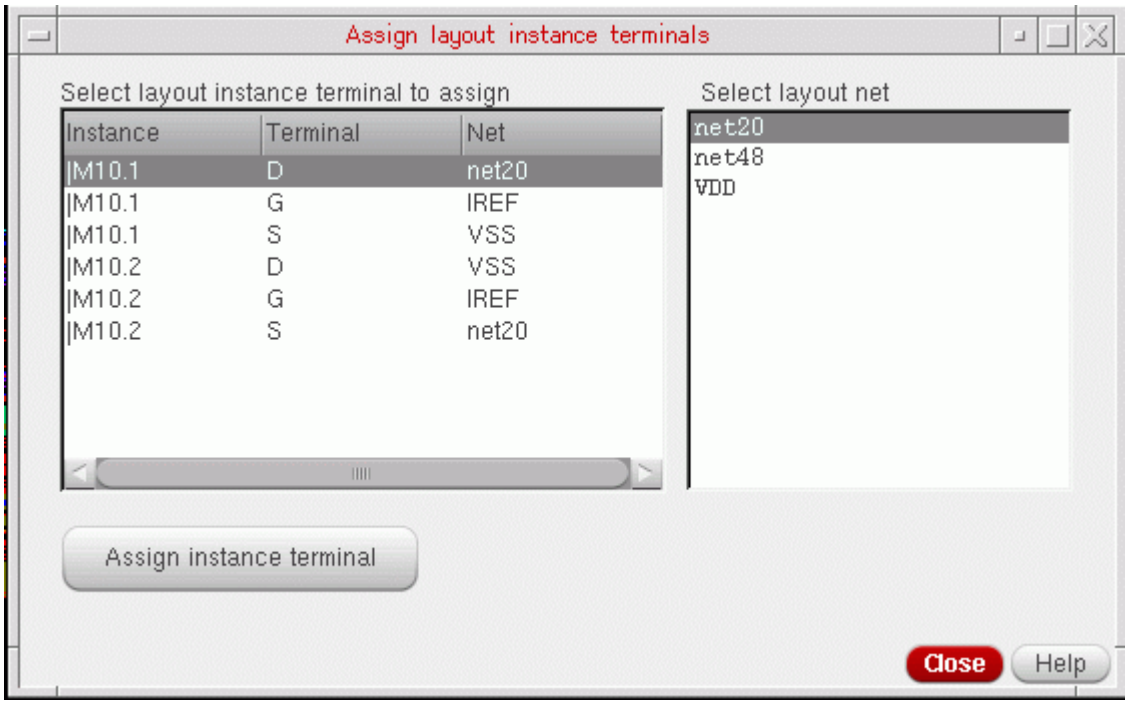
When creating a one-to-one device correspondence between schematic and layout instances with matching masters, *Define Device Correspondence* is able to assign layout instance terminals to the correct nets automatically. However, in some cases, the command is unable to assign all the instance terminals automatically; for example,

- If the instance masters in a one-to-one correspondence do not match.
- If you are creating a one-to-many, many-to-one, or many-to-many device correspondence.

In these cases, the Assign layout instance terminals form is displayed. You can use this form to manually assign layout nets to the unbound instance terminals in the selected layout instances.

To assign layout instance terminals:

1. In the *Assign layout instance terminals* form, select a layout net from the list on the right and one or more instance terminals to which it is to be assigned from the list on the left.



2. Click *Assign instance terminal*.

   The instance terminals are assigned to the net and the updated assignment is instantly reflected in the *Assign instance terminal* form.

© 2023 Cadence Design Systems, Inc.

3. Click *Close* to dismiss the form.

   The device correspondence is updated and the connectivity you defined is propagated to the layout.

If you close the form without manually assigning all the terminals, the connections of the remaining terminals are not modified.

After you have assigned an instance terminal to a net, you can select the instance terminal again and assign it to a different net, if required. However, once assigned to a net, you cannot unassign the instance terminal and disconnect it.

Related Topics

Assign Layout Instance Terminals Form

Deleting a Device Correspondence

To delete a schematic to layout device correspondence:

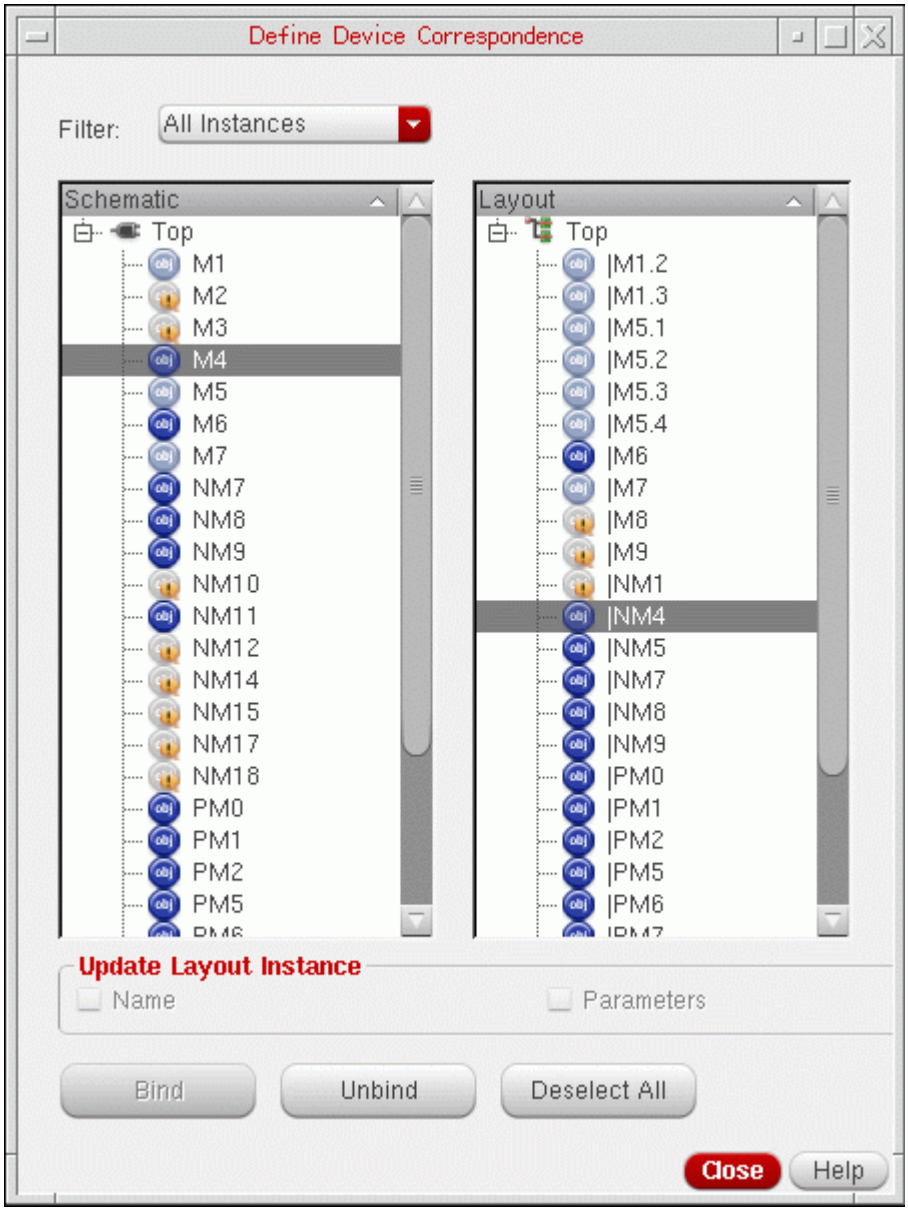
1. From the layout window menu bar, choose *Connectivity – Define Device Correspondence*.

   The Define Device Correspondence form appears.

2. Choose either *All Instances* or *All Terminals* from the *Filter* cyclic field, depending on the type of object that you want to unbind.

© 2023 Cadence Design Systems, Inc.

3. From the *Schematic* or *Layout* tree, select the schematic or the layout device that you want to unbind.



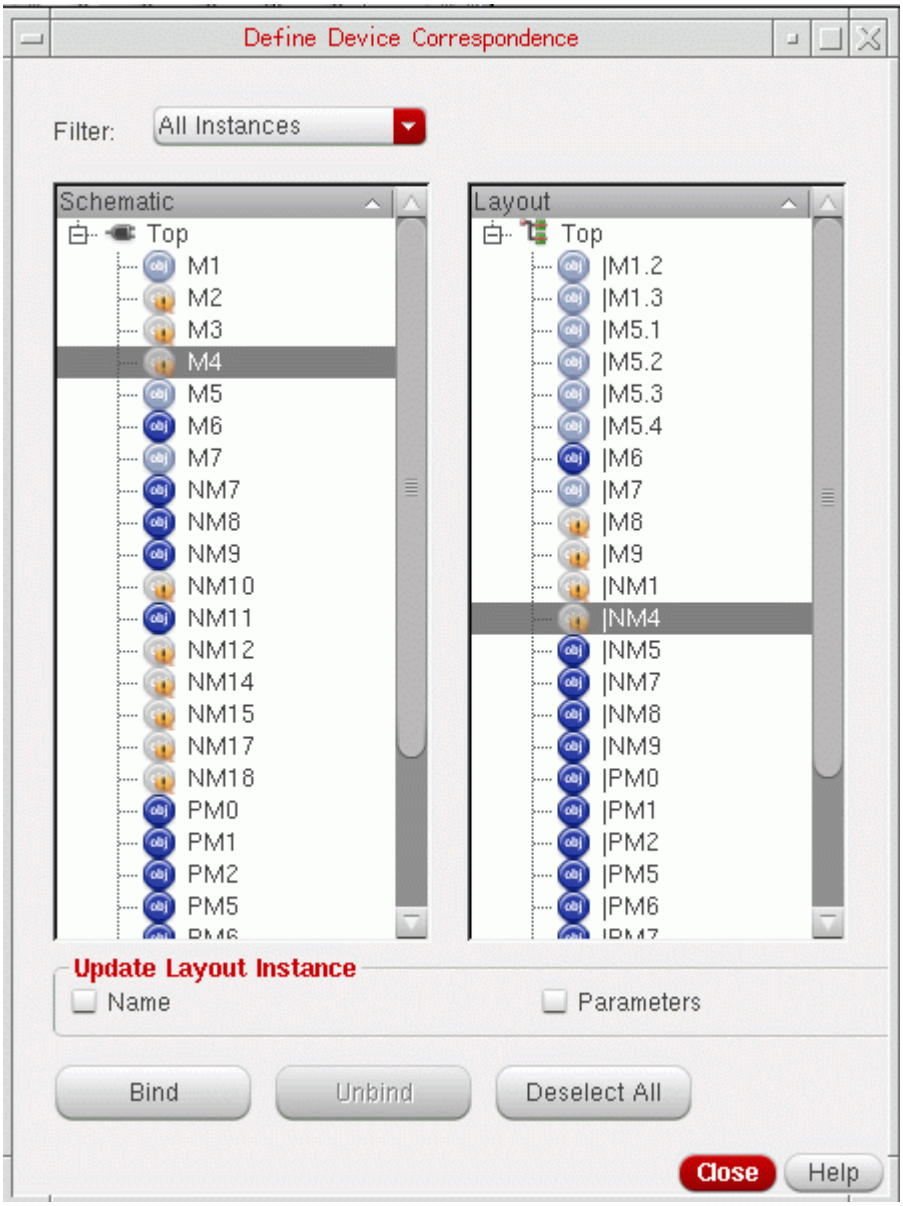
The single device or the devices to which the selected device is bound are selected in the other tree.

4. Click *Unbind*.

All the devices that formed the selected device correspondence are unbound—have their instance terminal connectivity removed—and the binding has been updated. The

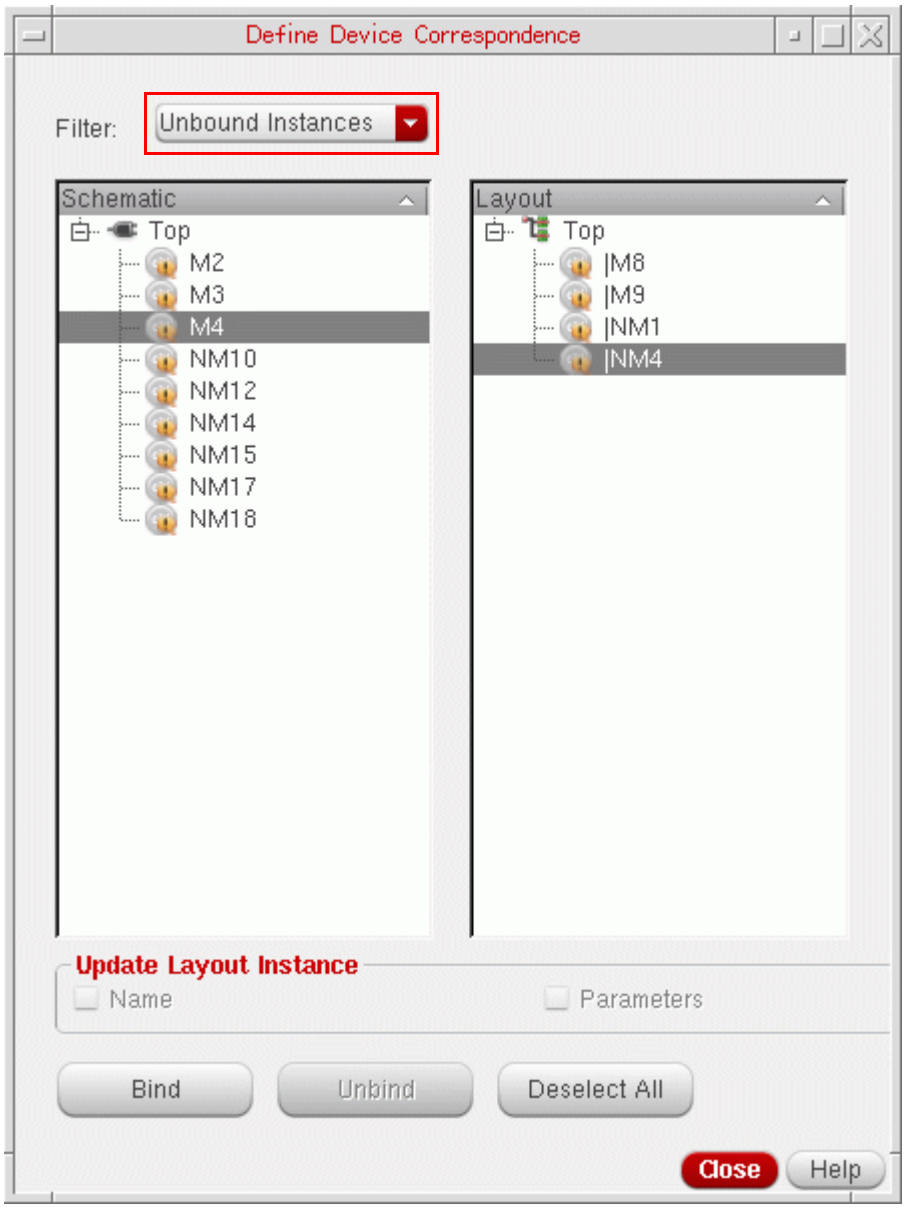
© 2023 Cadence Design Systems, Inc.

unbound devices display an orange icon ( ) next to their name, as shown in the figure below.



© 2023 Cadence Design Systems, Inc.

Also, the devices that you have just unbound now appear in the *Unbound Instances* list.



Related Topics

Define Device Correspondence Form

© 2023 Cadence Design Systems, Inc.

Binding Using the Bind Command in the Shortcut Menu

To manually bind an instance, mosaic, or terminal:

1. In the layout navigator or canvas, select the layout instance to be bound.

   You can also use this procedure to bind selected mosaics and terminals.

2. In the schematic navigator assistant or canvas, select the instance to which the selected layout instance needs to be bound.
3. In the layout canvas, right-click the selected instance and choose *Bind* from the shortcut menu.
4. The selected schematic and layout instances are bound and the *XL Status* column of the layout navigator assistant displays the binding status, as appropriate.

   - If the environment variable `bindUpdateName` is set to `t`, the name of the layout instance is automatically updated after binding to match the name of the corresponding schematic instance.
   - After binding, the logical (or schematic instance) connectivity is automatically propagated to the layout instance. But, if there is a terminal mismatch between the schematic and layout instances, the Assign layout instance terminals form is displayed so that layout nets can be selected and assigned manually to instance terminals, as appropriate.
   - If the manual binding results in complex binding between the selected schematic and layout instances, a warning message is displayed in the CIW to indicate the same.

To unbind a layout instance, right-click the instance in the layout navigator or canvas and choose *Unbind* from the shortcut menu.

Related Topics

bindUpdateName

Define Device Correspondence Form

Assign Layout Instance Terminals Form

Assigning Layout Instance Terminals to Nets

Working with the Binder to stay XL-compliant Rapid Adoption Kit

© 2023 Cadence Design Systems, Inc.

Define Device Correspondence SKILL Function

The following public SKILL functions let you set and access device correspondence information in your design. For more information, see Layout XL SKILL Functions in the *Virtuoso Layout Suite SKILL Reference*.

To get device correspondence information, use

- bndGetBoundObjects
- bndGetSiblingBoundObjects

To add, set, and replace device correspondence information for instances only, use

- bndAddInstsBindingByName
- bndRemoveInstBindingByName
- bndReplaceInstsBindingByName
- bndSetInstsBindingByName

To add, set, and replace device correspondence information for terminals only, use

- bndRemoveTermBindingByName
- bndReplaceTermsBindingByName
- bndSetTermsBindingByName

To add, set, and replace device correspondence information for either terminals, shapes or instances, use

- bndAddObjectsBinding
- bndRemoveObjectBinding
- bndReplaceObjectsBinding
- bndSetObjectsBinding

© 2023 Cadence Design Systems, Inc.

Updating the Connectivity Reference

Use the *Update Connectivity Reference* command to update the source schematic for the layout view to change the physical configuration, which specifies how the layout is generated from the schematic.

Updating the Schematic View

In Layout XL, the default schematic is always the schematic view with the same cell name as the layout view. To update your layout using a different schematic, you need to update the connectivity reference.

1. Choose *Connectivity – Update – Connectivity Reference*.

   The Update Connectivity Reference form is displayed.

2. Check the *Use schematic view* option and do one of the following:

   - Type the *Library*, *Cell*, and *View* names of the schematic in the fields provided. You can type in the name of a different library. The default view name is `schematic`.
   - Use *Sel by Cursor* and click an open schematic window.
   - Use the *Browse* button to browse your file system for the schematic you want.

3. Select *Update layout cellviews hierarchically* to update the library setting for any lower-level cellviews in a hierarchical cellview.

4. Click *OK*.

   Layout XL sets the connectivity reference to the schematic you selected.

Updating the Physical Configuration View

To change the physical configuration in use for a particular schematic,

1. Choose *Connectivity – Update – Connectivity Reference*.

   The Update Connectivity Reference form is displayed.

2. Check the *Use configuration view* option and do one of the following.

   - Type the *Library*, *Cell*, and *View* names of the view in the fields provided. You can type in the name of a different library. The default view name is `physConfig`.
   - Use the *Browse* button to browse your file system for the view you want.

© 2023 Cadence Design Systems, Inc.

3. Click *OK*.

   Layout XL sets the physical configuration to the view you specified.

Related Topics

Update Connectivity Reference Form

Binding Updates

Use the *Connectivity – Update – Binding command* (or `lxUpdateBinding` SKILL function) to increase Layout XL-compliance by improving the bindings, if you have unbound instances between the schematic and layout views due to any of the following reasons:

- The layout view was created outside Virtuoso Layout Suite XL. In this case, the logical and physical connectivity of the design may be in conflict. In addition, the number of bound instances and the number of shorts may not be as expected.
- The design is known to be LVS-clean, yet there are unbound instances and shorts in the design.
- A hierarchy mismatch exists between the schematic and the layout views, making it impossible to achieve a one-to-one device correspondence.
- The layout contains different-connectivity mosaics. The Update Binding command flattens these mosaic instances for the binder to process them. Alternatively, you can manually flatten the mosaic instances before the binder can process them.
  - If you have same-connectivity mosaics in your design, these can be handled by the binder like other layout instances. Therefore, you do not need to run the Update Binding command to flatten these mosaic instances.
  - If you do not want your mosaic instances to be flattened during the *Update Binding* run, you must set an `ignore` property on these instances.
- The layout contains route cells that were not defined in the schematic. The *Update Binding* command enables you to ignore such layout instances that are identified as route cells. In addition, the *Update Binding* command sets an `extractToLevel` property value of `31` to such ignored route cells, even if the route cells already have an `extractToLevel` property set. Alternatively, you can manually select these layout instances and add an `ignore` property on them. If the route cells exist within a transparent layout hierarchy, the `ignore` property cannot be added during an *Update Binding* run. To add the `ignore` property on instances within a transparent hierarchy requires editing the transparent instance master. The *Update Binding* command does

© 2023 Cadence Design Systems, Inc.

not support such edits that require modifying the layout cellviews other than at the top level.

If the layout has been created using the Layout XL connectivity-aware commands, the layout view is correct by construction and the layout instances are automatically bound to the schematic. In this case, running the *Update Binding* command is not required. For layouts that are not Layout XL-compliant or do not have the required connectivity information, the *Update Binding* command improves the binding between the schematic and the layout by setting options that control how the connectivity, the manual bindings, and the design hierarchy are processed.

The improvement in bindings and the desired Layout XL-compliance may not be achieved with a single *Update Binding* run. The command may need to be run iteratively to achieve the desired compliance, with each iteration modifying the command based on the improvement suggestions in the *Update Binding* report from the previous run.

To control which issues are reported during an *Update Binding* run, set the following environment variables, as appropriate:

- bindReportDummies
- bindReportHierarchy
- bindReportTerminalMismatch
- bindReportUnconnInstTerms
- ignoreTermsWithNonRoutingLabels



The Update Binding command cannot be undone.

Related Topics

lxUpdateBinding

LVS-clean

physical connectivity

route cells

bindIgnoreRouteCells

Layout XL-compliant

© 2023 Cadence Design Systems, Inc.

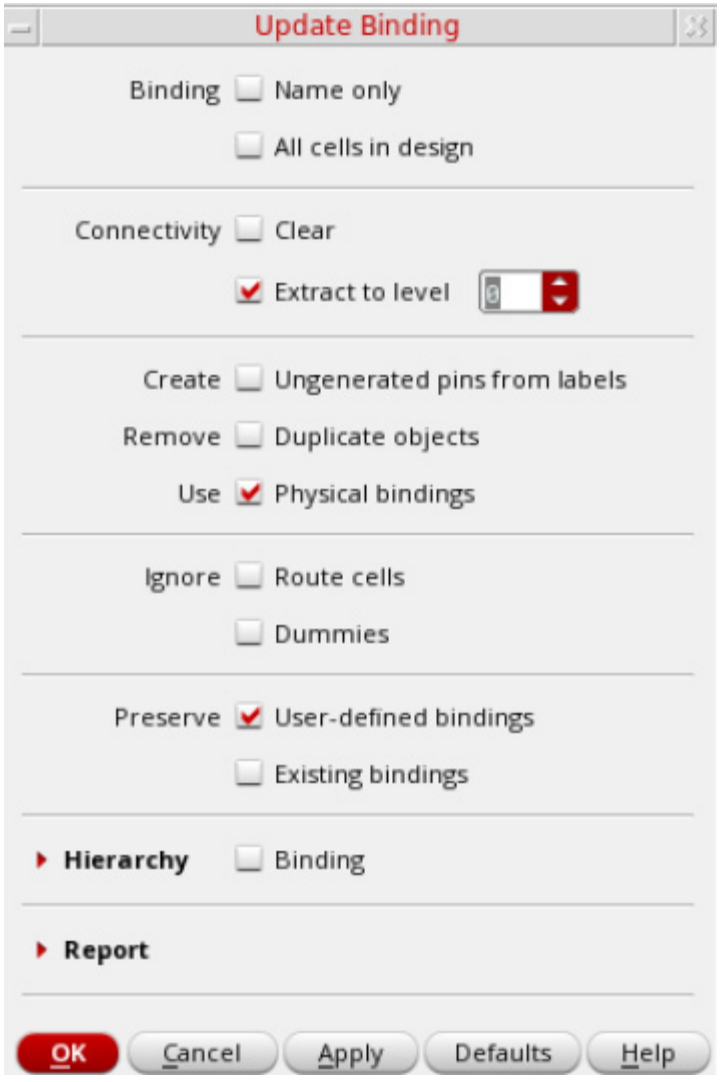
Transparent Layout Hierarchy Generation

Update Binding Report

Mosaic Instances Generation

Update Binding Flow

Before you run the *Update Binding* command to initialize the *Update Binding* flow, a dialog box confirming whether you want to proceed displays, if you have modified the layout or physConfig in the current session.



© 2023 Cadence Design Systems, Inc.

When the *Update Binding* command is called, by default, the binder re-runs in the connectivity-driven mode, discarding any existing bindings other than those set to be preserved using one of the *Preserve* options. If all the user-defined and other existing bindings were to be preserved, the binder would only bind any unbound instances.

When binding at the current level, the binder binds the schematic and layout devices only at the current level of hierarchy. *Current* level binding is best suited for designs that have matching hierarchies. A design that has matching hierarchies but was not bound correctly using *Name* binding after a Schematic ECO can benefit from *Current* level binding. It is recommended to run *Current* level binding in this situation, prior to running the *Update Components And Nets* command.

A design does not need to be LVS clean to derive better results from *Current* level binding. Although, largely LVS-clean designs and designs that have mosaics with same connectivity are a good starting point for *Current* level binding.

Hierarchical binding, on the other hand, by default, binds the schematic and layout instances across the hierarchy, based on the physical bindings defined in the current physConfig view. This type of binding is recommended for LVS-clean designs, because the design is flattened and the physical connectivity extracted.



Tip

A design that does not have any routing can result in bad bindings, if bound hierarchically.

To avoid binding hierarchically by using the bindings defined in the current physConfig view, which is the default binding mechanism for hierarchical binding, deselect *Use – Physical bindings*.

When performing a hierarchical binding run, you can stop the binder from binding beyond a particular schematic or layout library cellview by setting the `bindSourceStop` or `bindLayoutStop` environment variable. By doing so, the binder considers the specified schematic (source) or layout cellview as a leaf-level instance and stops looking beyond the instance for any devices to bind.

If hierarchical binding is run with the *Flatten layout instances* option selected, *Update Binding* flattens the instances, if required, and creates leaf-level bindings. Since the leaf-level bindings have a one-to-one correspondence, full ECO capability is available for the design.

With the *Flatten Layout Instances* option selected, you can choose to retain the flattened instances as free objects, or choose to create a *group* or a *synchronized family*. By default, the flattened instances form a synchronized family.

© 2023 Cadence Design Systems, Inc.

For example, if you flatten an instance, `I1`, all the shapes that the instance flattens into form a figure group or a synchronized family. Therefore, for each flattened instance, a corresponding figure group or a synchronized family will be created. If hierarchical binding is run with the *Flatten layout instances* option deselected, the *Update Binding* command preserves the layout hierarchy and creates either complex bindings or transparent instances to resolve hierarchy mismatches between the schematic and the layout. Whether hierarchical *Update Binding* will generate complex bindings or transparent instances depends on the *Create* option you select. Due to the creation of complex bindings and transparent instances, the Engineering Change Order (ECO) capability of the design is restricted.

By default, the *Update Binding* command binds based on connectivity—both logical and physical. Logical connectivity refers to the connectivity information that is derived directly from the schematic. So, if the logical connectivity information in the layout is incorrect, it can lead to incorrect bindings. To overcome such situations, the *Update Binding* command provides the *Clear* and *Extract to Level* options. You can select these options to clear or remove any incorrect logical connectivity, re-extract the layout up to a desirable hierarchical level, and then bind based only on physical connectivity.

However, the caveat with re-extracting through the hierarchy is that the extractor can extract through devices, causing shorts. To prevent this, you may need to define `stopLayers` in your technology file to indicate the layers that are being used for “Stopping” or “Cutting” the diffusion layers. The extractor will then ignore these layers for extraction and any false violations will not be reported. For more information about `stopLayers` and how you can set these up in your technology file, see validLayers – Stop Layers Specification

The *Update Binding* command also provides you the options to preserve any bindings that exist by using the Define Device Correspondence form. Conversely, you can also choose to delete all the existing device correspondences and create new bindings.

Contrary to using the hierarchical binder that flattens the schematic (physConfig) and layout hierarchies to the leaf level to determine the best bindings, which may not be at the leaf level, you can either specify a binding file that is derived from a PVS LVS run, or specify a PVS file that has the leaf-level binding information.

The binding file, `updateBinding.bnd`, is automatically generated by the binder if the instance cross-reference file (`.ixf`) and the extracted netlist file (`.net` or `.spi`) generated from the PVS run are specified.

However, if the generated binding file is empty, *Update Binding* cannot proceed.

- The instance cross-reference file is created as a `.ixf` file and the extracted netlist file is created as a `.net` file in the `svdb` directory of the PVS run directory, if you run PVS with the *Create Quantus Input Data* option selected.

© 2023 Cadence Design Systems, Inc.

- If you run PVS in LVS mode with the `-qrc_data` argument specified, the extracted netlist gets created in the form of a `.spi` file in the same directory as the `.ixf file`. The `.spi` file provides a normalized output for *Quantus*. Therefore, it is not the preferred input for generating the binding file required by *Update Binding*. But, a `.spi` file can be used if a `.net` file cannot be generated for some reason.

  Cadence recommends that you set `lvs_keep_data` to `yes` in your `lvs.pvl` rule file so that the extracted netlist output file is a *.net* file.

To ensure better binding, the *Input Scale* value set on the Update Binding form must be the same as that used for the PVS LVS run. If the scale at which PVS is run is different from the layout scale, you can use the PVS rule deck to determine the PVS input_scale. By default, the PVS scale is set to `0`, which means the `dbuPerUu` for the layout is used.

See bindPVSScale.

If you are using the same *Input Scale* for Update Binding as used for the PVS LVS run, but your binding results are not as expected, it is likely that the layout was either updated since the last PVS LVS run, or PVS LVS was run with a different precision setting. Ensure that the layout is up-to-date with the PVS run.

For more information about running the PVS LVS and for generating the required `.ixf`, `.net`, and `.spi` files, see the *Cadence Physical Verification User Guide*.

Here is an example illustrating the format of a binding file:

```
I I1/N0 |I2/MN0

I I2    M3(0,0)

I I3    M3(1,0)
```

where,

`I:` Instance

`I1/N0:` Schematic hierarchy path

`|I2/MN0:` Corresponding layout hierarchy path

`M3(1,0):` Corresponding mosaic row/column

Irrespective of the method used—automatic hierarchy match or binding file—when the leaf-level binding information is available, the layout is extracted to the specified depth to update the connectivity, and the binder sets the correspondence between the updated layout instances and their schematic counterparts.

© 2023 Cadence Design Systems, Inc.

If the *Ignore – Route cells* option is selected and the extractor identifies any route cells—cells that do not contain any devices—the *Update Binding* command adds an `ignore` property to the instances of “route-only” cellviews to avoid binding to them. Any instances that have stop layers defined are not considered for the `ignore` property addition, as these instances are recognized as “devices” and, therefore, must be considered for binding during the *Update Binding* run.

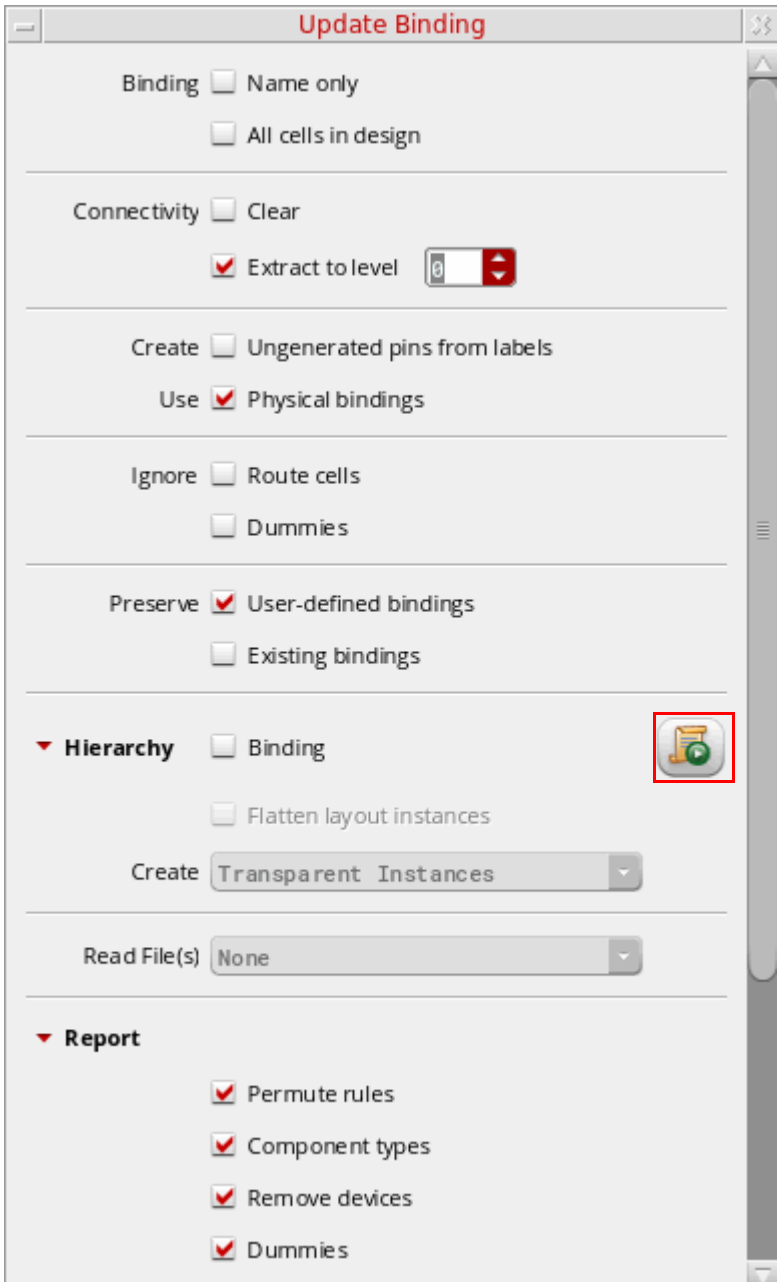
During the update, the physConfig also gets updated to set the physical bindings and a force descend ensures that the masters of the devices being bound match, and that the layout instances are being bound to the correct level in the schematic. If the physConfig gets modified during the update, the CPH window automatically opens after the update is complete to enable you to save or discard the edits.

Pre-populating the Update Binding Form Using SKILL

You can pre-populate the *Update Binding* form if you define a SKILL procedure using the `lxPreUpdateBinding` function that can control the environment variables associated with

© 2023 Cadence Design Systems, Inc.

the form. If the procedure is defined before opening the *Update Binding* form, a button is added to the *Read File* group box, as displayed in the figure below.



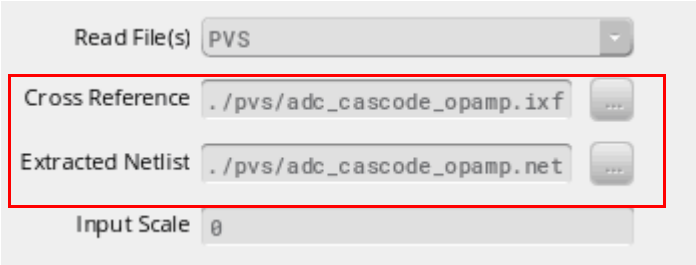
Click this button to call the user-defined SKILL procedure and reset the form based on the environment variable values defined in the procedure.

© 2023 Cadence Design Systems, Inc.

For example, instead of manually specifying the instance cross reference and netlist files to generate the binding file, you can define an `lxPreUpdateBinding` procedure, such as the one below, to pre-populate the form.

```
procedure(lxPreUpdateBinding(cv)
   let((base)
       base = strcat("./pvs/" cv~>cellName)
       envSetVal("layoutXL" "bindCrossRefFile" 'string strcat(base ".ixf"))
       envSetVal("layoutXL" "bindExtractedNetlistFile" 'string strcat(base
".net"))
   )
)
```

When the above procedure is called, the *Update Binding* form opens pre-populated with data from a PVS LVS run.



Related Topics

bindSourceStop

bindLayoutStop

lxUpdateBinding

validLayers – Stop Layers Specification.

Ignoring Route Cells

Binding Updates

Transparent Layout Hierarchy Generation

Device Correspondence Updates

Define Device Correspondence Form

Update Binding Form

© 2023 Cadence Design Systems, Inc.

Issues Faced when Setting up the PDK to Improve Layout XL Compliance

Update Binding Report

To confirm that the binder is initialized, Layout XL issues the following message.

```
INFO (BND-1003): Binder initialized for layout cellview 'gsclib090/CLKINVX1/layout' and source cellview
'gsclib090/CLKINVX1/schematic':

                bound       unbound     ungenerated
  terminals         4             0               0
  nets              4             0               0
  instances         0             0               2
```

Based on the Update Binding form options that you select (or ignore) before you run the command, the CIW reports the start of the command in the form of a report. The *Update Binding* report details the options that are being used (or ignored) for the update. In addition,

© 2023 Cadence Design Systems, Inc.

the report lists the checks that are being reported (or ignored for reporting), as shown in the sample report below.

```
Options
    Binding                                Connectivity
    Clear connectivity                     nil
    Extract to level                       0
    Create ungenerated pins from labels    nil
    Remove duplicate objects               nil
    Use physical bindings                  t
    Ignore route cells                     nil
    Ignore dummies                         nil
    Preserve user bindings                 t
    Preserve existing bindings             nil
    Hierarchical binding                   nil
    Report
        Permute rules                      t
        Component types                    t
        Remove devices                     t
        Dummies                            t
        Route cells                        t
        Hierarchy binding                  t
        Unconnected instances              t
        Unmatched terminals                t
        Cellviews with no terminals        t
        Terminals with no pins             t
        Duplicate objects                  t
        Reduction                          nil
```





The report also lists the cellviews that are being updated, as shown below.

```
ether adc_cascode_opamp layout
  ether adc_cascode_opamp physConfig
  ether adc_cascode_opamp schematic
```


© 2023 Cadence Design Systems, Inc.

When running Update Binding for all cells in the design, the report also presents a parameter difference summary for the various cells, as shown below.

```
Parameter difference summary
  Lib      Cell    Parameter     Differences
  gpdk090  nmos2v  ad                    12
  gpdk090  nmos2v  as                    12
  gpdk090  nmos2v  connectGates           2
  gpdk090  nmos2v  leftAbut               5
  gpdk090  nmos2v  pd                    12
  gpdk090  nmos2v  ps                    12
  gpdk090  nmos2v  rightAbut              5
  gpdk090  pmos2v  ad                     8
  gpdk090  pmos2v  as                     8
  gpdk090  pmos2v  connectGates           2
  gpdk090  pmos2v  leftAbut               4
  gpdk090  pmos2v  pd                     8
  gpdk090  pmos2v  ps                     8
  gpdk090  pmos2v  rightAbut              2
  Total                                 100
```

Information about the binding status of the updated cellviews and the Cell mapping table is also displayed, as shown in the sample report below.

```
   INFOS:
     INFO (BND-1105): Layout instances with unconnected terminals:
                                 connected  unconnected  terminal  examples of unconnected
nstances
          gpdk090  nmos2v  layout         21            3  S         I957, I959, I960
          gpdk090  nmos2v  layout         22            2  D         I957, I960
          gpdk090  nmos2v  layout         22            2  G         I957, I960
          Unconnected instance terminals can result in wrong bindings or unbound instances.
          If these instance terminals should be connected, check that the extraction stop level
          is deep enough to extract through route cells and off-pin connections.

     INFO (BND-1107): Cell mapping
                                 instances  permute  component  split   terminals
                                            rule     type       params  mapped
          gpdk090  nmos2v  symbol         29  (p D S)  NMOS       w 1     B D G S
       -> gpdk090  nmos2v  layout         24  (p D S)  NMOS       w 1     D G S

          gpdk090  pmos2v  symbol         27  (p D S)  PMOS       w 1     B D G S
       -> gpdk090  pmos2v  layout         27  (p D S)  PMOS       w 1     B D G S

     Unmapped
          gpdk090  nmos2v  symbol          4  (p D S)  NMOS       w 1     B D G S
```

The Cell mapping table provides the following information:

- Permute rule definition on the source and the layout cells being mapped.
- Component type of the mapped cell on each side.

© 2023 Cadence Design Systems, Inc.

- Split parameters enabled for each cell.
- Terminal names being mapped for each cell.

The sample Cell mapping table displayed above also reports an unmapped source cell. In addition, the table can report issues such as an extra or missing terminal for mapped cells.

The table below lists some issues that are likely to be reported in the Cell mapping section of the *Update Binding* report. Also listed against each issue is a suggested fix.

Cell Mapping Issue	Suggested Fix
Unmapped source or layout cells	Set the Configure Physical Hierarchy physical binding
Extra or missing terminals for mapped cells	Set the Configure Physical Hierarchy terminal attributes 'Ignore for generation', 'Ignore for check' or 'Name mapping'
Cells that may be missing permute rules	Add a permute rule, for example, '(p D S)', to the Edit CDF Simulation Information section of auLvs
Cells missing component type	Add a component type to the library LAM files or by using Configure Physical Hierarchy

Any warning messages are displayed in the WARNINGS section, as shown below.

```
  WARNINGS:
    WARNING (BND-2110): Layout instance with multiple fingers and different nets on must-connec
stance terminals may have shorts.
        Examples of these instances are: '|M7'.

    WARNING (LX-2149): Following cellview terminals have no pins:
        ether  adc_cascode_opamp  layout  VDDA, VSSA


    WARNING (LX-2153): Layout has 1 complex binding which cannot be fully checked or updated f
irce.
```


© 2023 Cadence Design Systems, Inc.

The percentage compliance data for instances, nets, and terminals, and the total compliance percentage is displayed in the Summary section, as shown below.

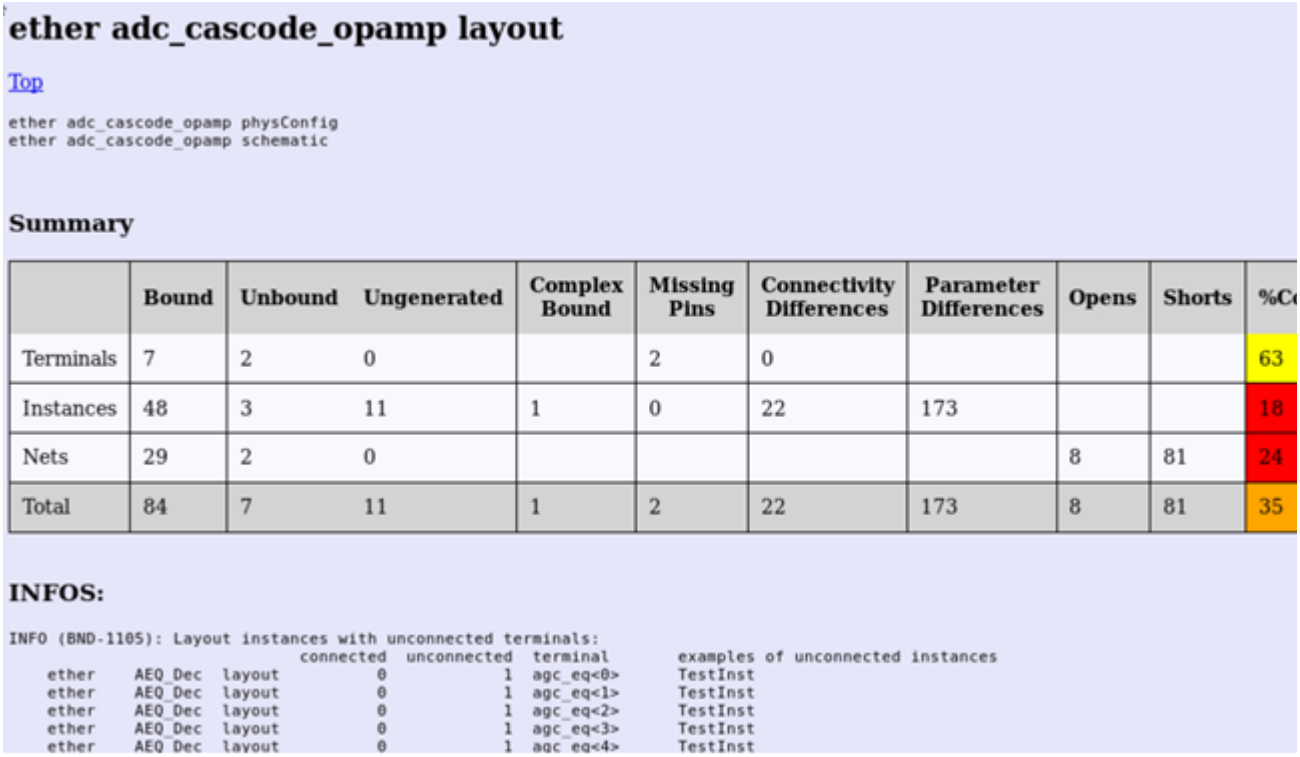
```
Summary
             Bound  Unbound  Ungenerated  Missing Pins  Connectivity Differences  Parameter Differences  Opens  Shorts  %Compliance
  Terminals      7        2            0             3                         1                                                53
  Instances     37        2           23             0                        22                     97                          20
  Nets          36        2            8                                                               80     171              12
  Total         80        6           31             3                        23                     97     80     171      28
```

If *Update Binding* does not bind all the instances, for example, when the source has got modified, or the layout has missing instances, the CIW reports the issue and prompts that the binding could be improved by performing the next iterative run. The sample report displayed is as shown below:

```
   WARNING (LX-2097): Layout instances and terminals do not match the source.
       Use 'XL Status' in the Navigator or run 'Connectivity - Check - Against Source' for more
etails.
       If the bindings are correct, run the 'Update from Source' commands.
       Otherwise, make the changes suggested in the warning messages above, or
       change the options on the form using the tooltips as a guide, and then
       re-run 'Connectivity - Update - Binding'.
```


© 2023 Cadence Design Systems, Inc.

In addition to displaying the *Update Binding* report in the CIW, Layout XL displays the Update Binding report in an HTML window, as shown in the figure below. You can choose to hide this report by appropriately setting the `xlComplianceHtmlOpen` environment variable.



ether adc_cascode_opamp layout

Top

ether adc_cascode_opamp physConfig
ether adc_cascode_opamp schematic

Summary

	Bound	Unbound	Ungenerated	Complex Bound	Missing Pins	Connectivity Differences	Parameter Differences	Opens	Shorts	%C
Terminals	7	2	0		2	0				63
Instances	48	3	11	1	0	22	173			18
Nets	29	2	0					8	81	24
Total	84	7	11	1	2	22	173	8	81	35

INFOS:

INFO (BND-1105): Layout instances with unconnected terminals:

			connected	unconnected	terminal	examples of unconnected instances
ether	AEQ_Dec	layout	0	1	agc_eq<0>	TestInst
ether	AEQ_Dec	layout	0	1	agc_eq<1>	TestInst
ether	AEQ_Dec	layout	0	1	agc_eq<2>	TestInst
ether	AEQ_Dec	layout	0	1	agc_eq<3>	TestInst
ether	AEQ_Dec	layout	0	1	agc_eq<4>	TestInst

Having the Update Binding report available in the HTML format brings the following advantages:

- Allows the *Update Binding* report to be saved on disk.
- Allows a keyword search on the report, which makes it easy to locate the messages related to a specific instance, terminal, or a net.

For information on resolving some of the common issues reported during an *Update Binding* run, see Issues Faced when Setting up the PDK to Improve Layout XL Compliance. This section provides information not only about the various common issues encountered during an *Update Binding* run, but also provides detailed instructions on how you can set up a PDK to counter these issues at the global level, and improve the Layout XL compliance of a design.

Related Topics

Update Binding Form

xlComplianceHtmlOpen

© 2023 Cadence Design Systems, Inc.

Issues Faced when Setting up the PDK to Improve Layout XL Compliance

Working with the Binder to stay XL-compliant Rapid Adoption Kit

Updating All Physical Bindings

Use the *Connectivity – Update – All Physical Bindings* command to update the CPH master of an instance to the layout master if bindings have been created between layout and schematic instances that have different masters.

The *Update All Physical Bindings* command is particularly useful to remove any master differences between the layout and the schematic and provides a quick and simple method for updating the CPH master without opening the CPH. After the command is run, the CPH master for all the instances in the cellview gets updated to the correct layout master.

If the CPH window is already open, it will be brought to the front and you can choose to save or discard the CPH master update, as appropriate. However, if the CPH window is not open, any changes made to the CPH instance master will automatically get saved in the physConfig. You can also choose to turn off the automatic saving feature by setting the autoMirrorFilter environment variable to `nil`.

Instead of updating the CPH master for all the instances in the cellview, you can choose to update the master only for a selected set. To do this, select the instances in the layout canvas or the Navigator and choose the *Update – Physical Bindings* shortcut command. The command, which is available only through the shortcut menu (not available in the main menu), updates the CPH master for the selected set of instances to synchronize it with the layout master.

Related Topics

autoMirrorFilter

Binding Updates

© 2023 Cadence Design Systems, Inc.

© 2023 Cadence Design Systems, Inc.

10

Troubleshoot Layout XL Issues

You might face some unexpected results while using the Virtuoso® Layout Suite XL (Layout XL). You can use the following methods to troubleshoot some of these unexpected problems.

- Troubleshoot Startup Issues
- Troubleshoot Configure Physical Hierarchy
- Troubleshoot Constraint Migration
- Troubleshoot Parameter Evaluation
- Troubleshoot Layout Generation Problems
- Troubleshoot Cloning
- Troubleshoot Chaining
- Troubleshoot Editing
- Troubleshoot Connectivity Extraction
- Troubleshoot Binding Update Issues
- Troubleshoot Layout XL Compliance
- Troubleshoot Performance

© 2023 Cadence Design Systems, Inc.

Troubleshoot Startup Issues

You can face following issues while starting Layout XL:

- Invalid Markers from Previous Software Versions
- Options Form Does Not Appear at Startup
- Unable to Restart Layout XL after Opening Assura LVS or DRC

Invalid Markers from Previous Software Versions

If you are opening layouts you developed using previous versions of Virtuoso software, you can clean up the layout canvas and get rid of invalid markers.

1. From the layout window menu bar, choose *Connectivity – Update – Components And Nets*.

   Layout XL resynchronizes the layout connectivity with the schematic connectivity.
2. From the layout window menu bar, choose *Verify – Markers – Delete All*.

   The markers disappear.
3. Choose *File – Save*.
4. From the layout window menu bar, choose *Window – Close*.

   The layout window closes.
5. From the schematic window, choose *Window – Close*.
6. The schematic window closes.
7. From the Command Interpreter Window (CIW), choose *File – Open*.

   The Open File form is displayed.
8. From the *Library Name* cyclic field, choose the library name of your design.
9. In the *Cell Name* field, type or select from the *Cell Names* list the cell name of your design.
10. From the *View Name* cyclic field, choose the view name of your design.
11. Click *OK*.

    The layout window appears, displaying the updated layout configuration.
12. From the layout window menu bar, choose *Launch – Layout XL*.

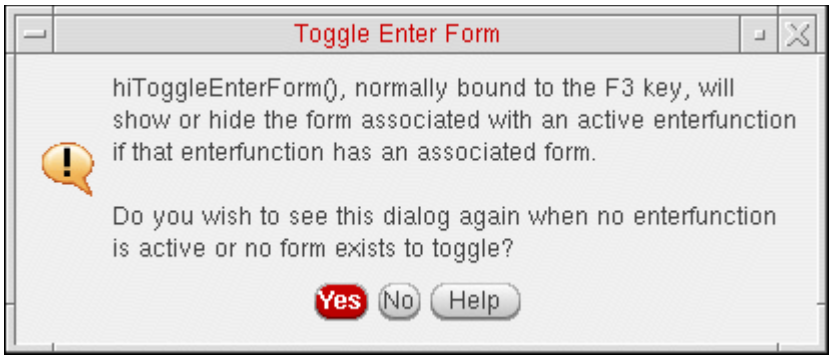
© 2023 Cadence Design Systems, Inc.

The Palette assistant and the schematic view appear.

If you still have problems, check that your technology is correctly defined.

Options Form Does Not Appear at Startup

When you press `F3` to see an options form listing possible choices for a command and the command does not have an options form, the message box shown below appears:



If you want the message box to appear every time you press `F3` for a command that has no options form, click *Yes*.

If you do not want the message box to appear every time you press `F3` for a command that has no options form, click *No*.

Unable to Restart Layout XL after Opening Assura LVS or DRC

In a session, if you close Layout XL and start Assura and then open Layout XL again, it is possible that you may not be able to launch Layout XL again until the library list is updated. To update the library list, set the shell environment variable, `CPH_UPDATE_LIBLIST`.

Related Topics

Technology File Requirements for Layout XL

© 2023 Cadence Design Systems, Inc.

Troubleshoot Configure Physical Hierarchy

You can face following issues while configuring physical hierarchy:

- Schematic Properties Not Present in physConfig on page 956
- NLP Expressions Cannot be Converted to the New Schema on page 956
- Global or Inherited Nets Beyond Physical Leaf Cells are not Detected on page 957

Schematic Properties Not Present in physConfig

A schematic instance contained in a cell must have its master translated first in order for the properties to be converted correctly. The `dfIIoa20222` translator attempts to ensure that this always happens, however, it is not guaranteed. If the properties for a particular schematic instance do not appear in the physical configuration view, Cadence recommends that you translate your data to OpenAccess 2.2 first and then perform the conversion to use the IC 6.1 Layout XL schema as a separate step.

NLP Expressions Cannot be Converted to the New Schema

The conversion commands cannot convert the following schematic properties if they are defined as an NLP expression on a cell.

- `lxRemoveDevice`
- `lxRounding`
- `lxStopList`
- `lxUseCell`
- `lxViewList`

When the system encounters this situation it issues the following message.

```
Cannot convert the 'oldPropName' property with value 'nlpExpr' on instance
'instName' in logical cellview 'libName/cellName/viewName' because NLP
expressions are not supported. Use the Configure Physical Hierarchy window to
specify explicitly the newOptionName either on a per cell or a per instance
basis.
```

`newOptionName` is the name of the relevant field in the Configure Physical Hierarchy window's *Cells* or *Instances* table or the name of the relevant option field in the Attributes

© 2023 Cadence Design Systems, Inc.

pane. The mapping between *oldPropName* and *newOptionName* in the message is as follows.

oldPropName	newOptionName
lxRemoveDevice	*Remove device* option
lxRounding	*Rounding* option
lxStopList	*Inherited Stop List*
lxUseCell	*Physical Cell*
lxViewList	*Inherited View List*

Global or Inherited Nets Beyond Physical Leaf Cells are not Detected

In older releases, Configure Physical Hierarchy always elaborated the entire logical design hierarchy beyond the nodes which mapped to physical leaf cells. Now, the software stops the elaboration when it reaches a logical node that maps to a physical view with one of the view names specified in the *Physical stop view list* field.

This lets you use the physical stop view list to limit the logical elaboration for large hierarchical designs that would otherwise take a long time to open or might not open at all in Layout XL. However, it also means that global or inherited nets below the leaf instance in the schematic are not considered during the elaboration.

If your design relies on such nets, Cadence recommends that you switch off this environment and re-elaborate your design. To do this:

1. In the CIW, type

   ```
   envSetVal("layoutXL" "cphStopLogicalElabAtPhysLeaf" 'boolean nil)
   ```

2. Force a full re-elaboration by changing the *Logical switch view list* in the *Global Bindings* pane in the Configure Physical Hierarchy window.

Related Topics

Physical Hierarchy Configuration

© 2023 Cadence Design Systems, Inc.

Troubleshoot Constraint Migration

Issue with CMX Legacy Net Classes

CMX legacy net classes are listed in the XL Probe form but are not shown under the *Net Class* category in the Constraint Manager assistant. This does not impact the individual constraints migrated.

Related Topics

Constraint Manager Assistant

Troubleshoot Parameter Evaluation

You can face following issues during parameter evaluation:

- Netlisting Mode Issues
- Evaluating CDF Callbacks by Default

Netlisting Mode Issues

To ensure that CDF parameters are always evaluated correctly, make sure that the `CDS_Netlisting_Mode` shell environment variable is set to `Analog`.

To set `CDS_Netlisting_Mode`:

➤ Type the following commands in the CIW:

```
setShellEnvVar("CDS_Netlisting_Mode=Analog")
cdsSetNetlistMode()
```

To check the mode currently set:

➤ Type the following command in the CIW:

```
cdsGetNetlistMode()
```

If you have the `CDS_Netlisting_Mode` shell environment variable set to `Digital`, Layout XL will not evaluate the CDF parameters by default. But, you can set the dbSearchCDF SKILL function to `t` to cause CDF to be searched for parameter values.

© 2023 Cadence Design Systems, Inc.

Evaluating CDF Callbacks by Default

To ensure that SKILL callbacks defined on CDF parameters (for example, `lxCombination`) are evaluated by default, you must set the lxEvalCDFCallbacks environment variable to `t`.

`lxEvalCDFCallbacks` causes all SKILL callbacks defined on CDF parameters to be evaluated by default when you run the following Layout XL commands. Callbacks defined on buttons are never evaluated.

- *Generate All From Source*
- *Generate Folded Devices*
- *Update Components And Nets*
- *Update Layout Parameters*
- *Update Schematic Parameters*
- *Back Annotate (Dummy backannotation)*

The callbacks are evaluated in the order in which they are listed in the CDF, so it is important that you set the appropriate callback to the appropriate place in the list.

See Defining Parameters.

Enabling this environment variable can adversely impact performance, particularly on large designs with several callbacks.

Related Topics

Customizing the Simulation Environment

lxEvalCDFCallbacks

Issues when pPar and Callbacks are Used in a Design when Generating a Layout

Dangers of CDF Callbacks

Issues when pPar and Callbacks are Used in a Design when Generating a Layout

© 2023 Cadence Design Systems, Inc.

Troubleshoot Layout Generation Problems

You can face following issues during layout generation:

- Issues when pPar and Callbacks are Used in a Design when Generating a Layout
- Issues when Generating Selected Components From Source
- Cannot see Generate Selected From Source Command in Hierarchical Designs

Issues when pPar and Callbacks are Used in a Design when Generating a Layout

If a design using `pPar()` and callbacks is generating an incorrect Pcell layout, it is likely that one of the following requirements is not being met:

1. Shell environment variable `CDS_Netlisting_Mode` set to `Analog`

2. CDF parameter `parseAsCEL` set to `yes`

But, if a design is generating an incorrect layout inspite of both the conditions above met, then, it might be an underlying issue with the use of `pPar()` with callbacks that is preventing the layout from generating correctly. Cadence recommends avoiding the using of `pPar()` with callbacks. It is recommended that derived parameters be computed at the Pcell level. But, this may need a change in the PDK, which may not always be feasible. In such cases, manually triggering the callbacks by setting the lxEvalCDFCallbacks environment variable to `t` can help.

Issues when Generating Selected Components From Source

If the *Generate Selected From Source* command fails, check that:

- The instance is not already placed in the layout.

  If it is already placed, you cannot use *Generate Selected From Source* to generate it again.

- All the required masters and reference views are present, including
  - The default layout master for the specified schematic component.
  - Any layout master specified by the `useCell` attribute for the schematic instance or master.
  - The required layout view as derived from the `stopList`.

© 2023 Cadence Design Systems, Inc.

Cannot see Generate Selected From Source Command in Hierarchical Designs

If you push into a hierarchical schematic view beyond the layout stop point, you cannot use the *Generate Selected From Source* command to generate lower-level instances inside that top-level cell.

For example, assume there is a layout view derived from the specified `stopList` corresponding to a `cell1` instantiated in the schematic. If you descend into an instance of `cell1` and try to generate the lower-level instances, the command fails.

Implicit Inherited Connection Terminals to Local Nets Not Created

Generate Selected From Source does not create pins for implicit inherited connections to local nets (i.e., wires with `netSet` properties that do not have a `!` in the name).

Related Topics

lxEvalCDFCallbacks

Layout Generation

Troubleshoot Cloning

If a target clone cannot be found, the software displays an appropriate CIW message, such as the one below:

```
INFO (LX-1353): There are no target structures matching the selected clone
source.
INFO (LX-1371): Either all the schematic components have already been generated
in the layout view, or the criteria you specified in the Generate Clones form
are too restrictive. Change the Search In mode to Layout or change the Search
With criteria to make it less restrictive.
```

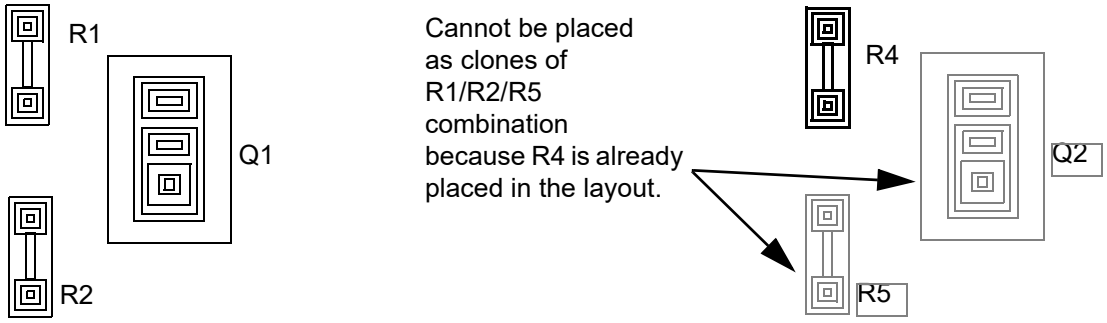
A list of possible reasons for not being able to find clones and how you can resolve the issue preventing the clone generation are described below.

- Structure Already Exists in the Layout
- Connectivity Structure is Different
- Master Cells are Different
- Parameters or Properties are Different
- Automatic Parameter Update Causes Different Submasters (Message LX-2149)

© 2023 Cadence Design Systems, Inc.

Structure Already Exists in the Layout

The structure or part of the structure that you selected as the target already exists in the layout. For example, here `R1`, `R2`, and `Q1` in the layout were the source components for clone search in *Schematic* and `R4` (which is already implemented), `R5`, and `Q2` were the target components.



In the example above, the clones cannot be found because `R4` already exists in the layout. To overcome the issue, delete `R4` from the layout and then search for matching targets. Else, generate `Q2` and `R5` in the layout and search for clones in the *Layout* instead of searching in the *Schematic*.

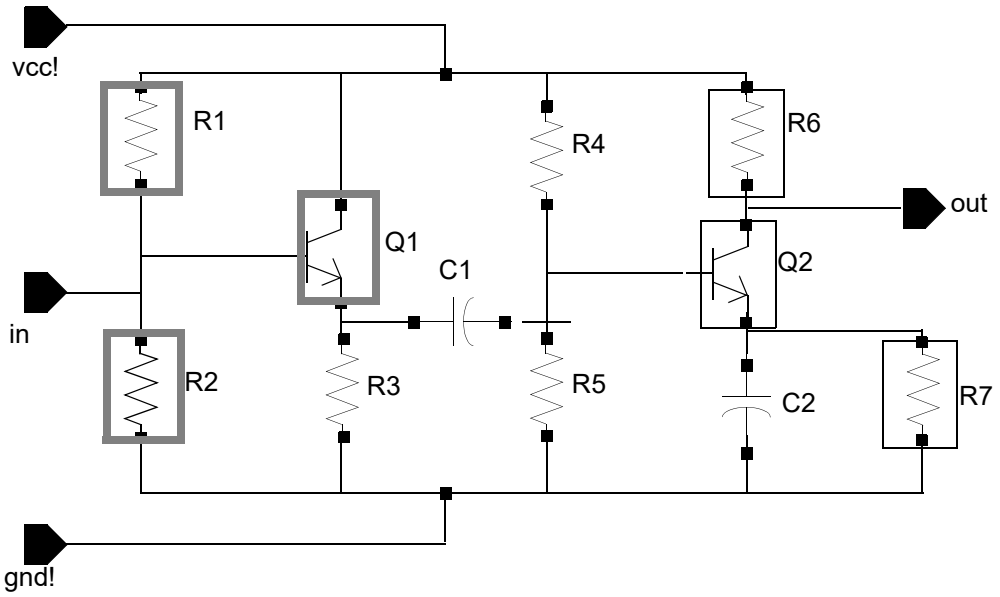
Connectivity Structure is Different

If the connectivity of the source and target structures is not the same, clones cannot be found.

For example, if you choose `R1`, `R2`, and `Q1` as the source components, `R6`, `R7`, and `Q2` cannot be found as clone target components because the clone source and clone target components are connected differently.

© 2023 Cadence Design Systems, Inc.

To overcome the issue, leave the *Exact Connectivity* check box deselected in the Generate Clones form. This allows matching targets with the same set of instances but different connectivity to be found during the clone search.



Master Cells are Different

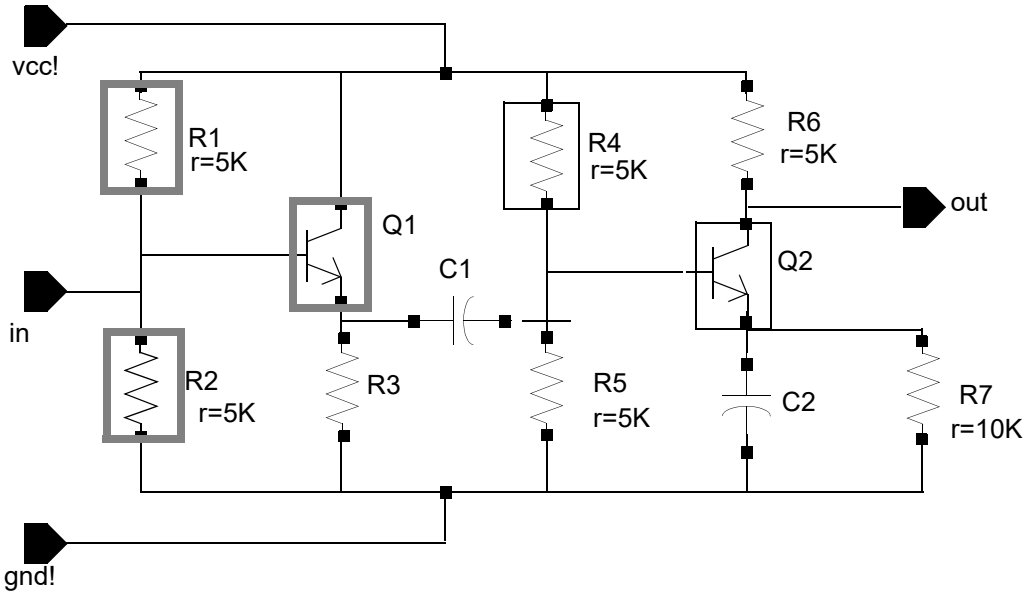
If the master cells of the source and the target instances are not the same, matching target clones cannot be found.

More Source Components are Selected than Target Components

More source components are selected than target components. For example, if you select `R1`, `R2`, and `Q1` as source components and run a search on *Selected Set Only* with `R4` and `Q2` selected in the schematic, no clones can be found.

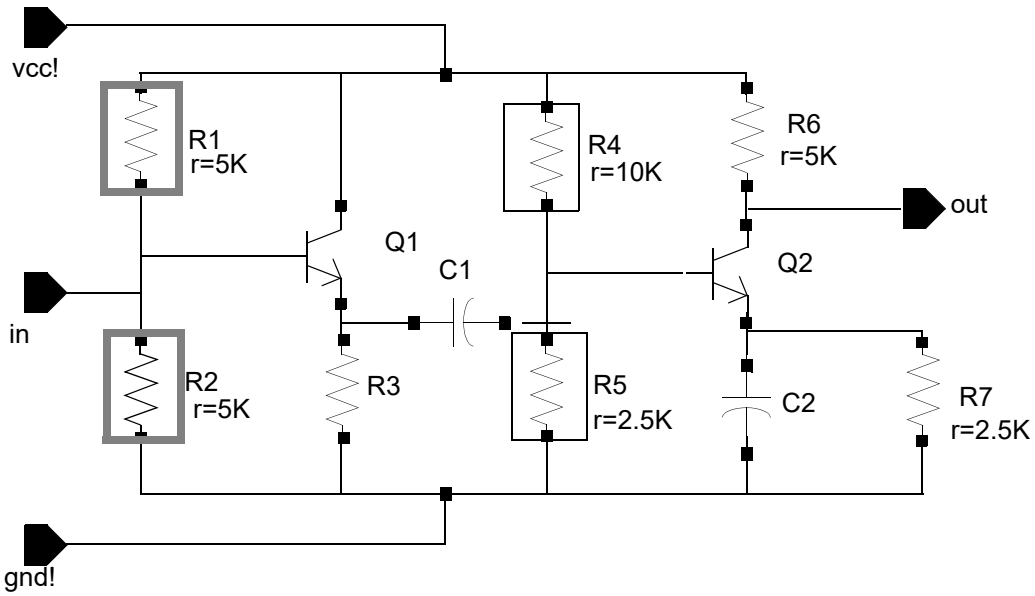
© 2023 Cadence Design Systems, Inc.

To resolve the issue, when searching on *Selected Set Only*, ensure that the selected set in the schematic contains at least the same number of components as the clone source.



Parameters or Properties are Different

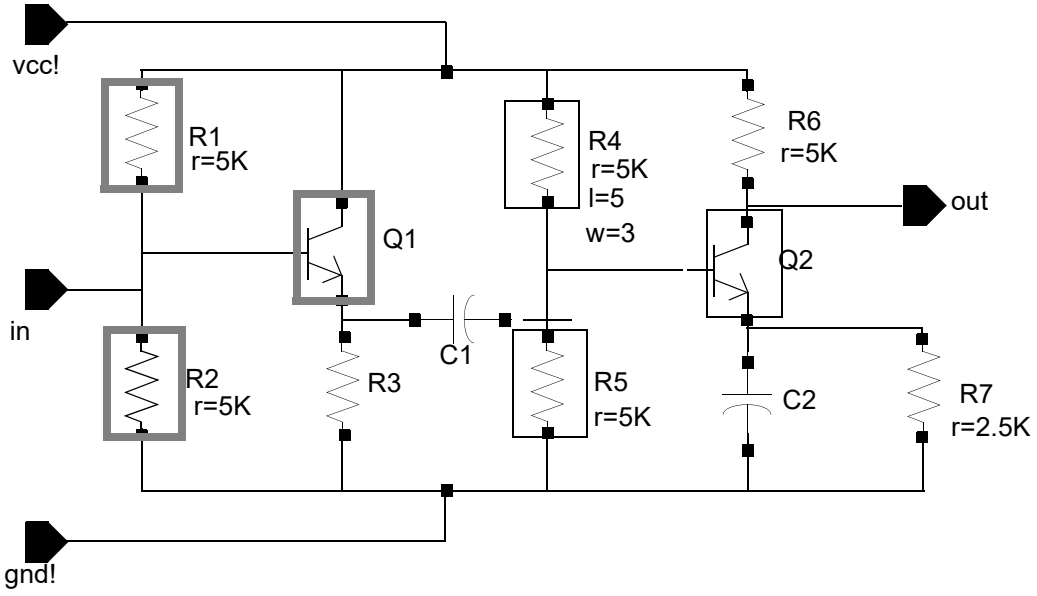
The parameters or properties of the source and target instances are not the same. For example, you could not use `R1` and `R2` in this diagram to clone `R4` and `R5` because their resistance values are different.



© 2023 Cadence Design Systems, Inc.

The target instance must have all properties with values the same as those on the source instance. In some cases, the target instance might have some additional properties as well.

In the example below, you can use `R1`, `R2`, and `Q1` as source components to clone `R4`, `R5`, and `Q2` because the targets have the same properties and values as the sources, even though one of the targets has additional properties.



To ignore the CDF parameters while searching for the clone target, you can use either:

- **Configure Physical Hierarchy** – In the *Attributes* section, add the CDF parameters to ignore in the *Ignore for generation* field.
- **Connectivity form** – In the *Parameters* tab, select *Parameters to ignore during generation and update* and add the CDF parameters to be ignored.

To ignore properties while searching for the clone target; in the *Parameters* tab of the **Connectivity form**, select *Properties used to ignore objects during generation and update* and add the property names to be ignored.

Automatic Parameter Update Causes Different Submasters (Message LX-2149)

When you search for clones with *Exact Parameter Match* turned off, Layout XL also lists targets with components that do not have same parameter values. When you generate a clone from one of these targets, the software automatically updates the parameters and parameter values in the generated clone to match those defined on its schematic counterparts.

© 2023 Cadence Design Systems, Inc.

This automatic parameter update can lead to the creation of different submasters for the cloned instances. This in turn can cause problems when you try to generate synchronous clones and with existing abutments in the design.

You can avoid this situation by running the *Update Layout Parameters* command before selecting the clone source and then *Find Matching Targets* with *Exact Parameters* match turned on. If the different submasters are causing problems with abutment, add the abutment properties to the list of properties to be ignored in the Parameters of the Connectivity form.

To avoid the issue, ensure that all layout-only parameters, such as abutment parameters in this case, are properly ignored by Layout XL.

To do this:

1. Choose *Launch – Configure Physical Hierarchy* window.
2. In the *Cells* tab, select the cell to be ignored.
3. Select the *Attributes – Parameters* tab.
4. In the *Ignore for generation* field, specify the abutment parameter names that need to be ignored for generation.
5. Save and close the window.

The specified abutment parameters will be ignored by Layout XL commands such as *Generate All From Source*, *Update Layout Parameters*, *Generate Clones*, and *Check Against Source.*

Because *Check Against Source* will ignore the specified abutment parameters, no mismatch between the layout and schematic parameters, if any, is reported.

The *Generate Clones* command will be able to generate synchronous clones because it will not require the schematic and layout abutment parameters to be the same.

Related Topics

Generate Clones Form

Clone Generation

© 2023 Cadence Design Systems, Inc.

Troubleshoot Chaining

Device Alignment Changes with Change in Folding Threshold or Chaining

If you find that the alignment that you specified before placing an instance in the layout changes after you change the transistor folding threshold, this is expected behavior.

Device alignment is not preserved if you change the folding threshold because when the folding threshold is changed, the devices are recreated with default alignment. Therefore, if you want the newly folded devices to follow the alignment that you had specified before changing the folding threshold, you will need to specify the alignment again.

Likewise, if you were to switch chaining ON or OFF, the alignment that you had specified will be ignored as the drag set is recreated after every chaining or folding update. For the newly created devices to follow the alignment specified earlier, the alignment will need to be reapplied.

Related Topics

Generation

Connectivity Form

Troubleshoot Editing

You can face following issues while editing in Layout XL:

- Components Move Slowly
- Generate Layout Form Does Not Keep Values from the Last Entry
- Parameters Not Updated
- Schematic Not Editable
- Warning to Update Your Design Appears at Startup

Components Move Slowly

If you find that the *Move* command works too slowly, especially when you have many iterated instances, turn off the display of incomplete net flight lines or display only nets that are relevant to the current task.

© 2023 Cadence Design Systems, Inc.

Generate Layout Form Does Not Keep Values from the Last Entry

The Generate Layout form displays default values each time it opens, it does not carry over values from the previous time the form was displayed.

To use predefined settings, set the appropriate environment variables in your `.cdsenv` file before you start Layout XL. Alternatively, you can load predefined cellview information from another OpenAccess cellview. For more information, see on page 382.

Parameters Not Updated

If the parameters are not updated the way you expect, check for the presence of the lxParamsToIgnore Property on components of your design.

Schematic Not Editable

If you cannot make changes to the schematic,

- Check that the schematic is editable. (If it is, the word “Editing” is displayed in the window banner; if it is not, the word “Reading” is displayed.)

  If it is not editable, make it so using the *File – Make Editable* command.

- Check with your system administrator that you have a license for the Virtuoso Schematic Editor (product 302).

Warning to Update Your Design Appears at Startup

When you open a design in Layout XL and an Info window appears, it means that the schematic has been changed after the last time the layout was changed, so the layout does not represent the latest version of the design.

You can check the layout against the changed schematic using the layout window *Check Against Source* command and then update the layout view using either the *Update Components And Nets* or *Define Device Correspondence* commands as needed.

Related Topics

Layout Editing

Generate Layout Form

Design Data Checks

© 2023 Cadence Design Systems, Inc.

Troubleshoot Connectivity Extraction

You can face following issues during connectivity extraction in Layout XL:

- Connections Not Made
- Markers for Nonexistent Overlaps and Shorts Appear
- Path Ends Not Accepted
- Placement and Routing Do Not Run
- Layout XL Does Not Recognize Physical Vias
- Moving Software Executables To a New Location
- Unmatched Instance Terminals are Automatically Removed

Layout XL Commands Requiring Schematic Extraction

Connections Not Made

If the software does not accept an electrical connection you make, make sure that the following information is set in the technology file. For more information, see Technology File Requirements for Layout XL and in particular Connectivity Rules.

- Connectivity information in the appropriate constraint group, for example `virtuosoDefaultExtractorSetup`.
  - The `validLayers` specifies all the layers you want to extract through.
- Layer information in the `layerRules` section.
  - The `functions` section, which specifies the layer function and assigns a mask number to each layer.
  - The `equivalentLayers` section defines layers that you want to be considered electrically equivalent.

Also check that the layer properties were set when you made the connection.

Markers for Nonexistent Overlaps and Shorts Appear

If there are markers for nonexistent overlaps and shorts in the layout, make sure that the appropriate layer was active when the path was created.

© 2023 Cadence Design Systems, Inc.

If this does not fix the problem, make sure that the layer information and connectivity information are set correctly in the technology file. For more information, see Technology File Requirements for Layout XL

Path Ends Not Accepted

If the software rejects the final segment of a path or wire connection and displays the following message.

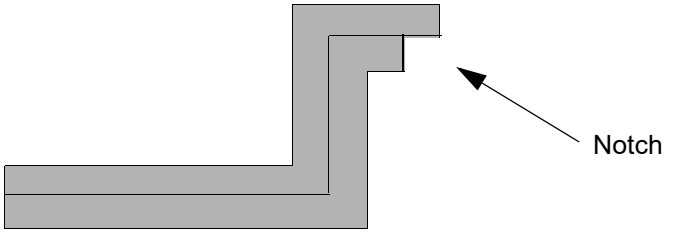
```
First or last segment of created path has length less than or equal to half the path width
```

➤ Press `Return` to end a path instead of double clicking the mouse (do not double click to end a path).

or

➤ From the layout window menu bar, choose *Options – Display* and set the *Snap Mode* for *Create* commands to something other than *L90XFirst* or *L90YFirst*.

When you use the *L90XFirst* or *L90YFirst* snap modes, the double click often makes a notch in the path, which triggers the error message and can cause errors in mask layout.



Placement and Routing Do Not Run

Layout XL does not let you place or route the elements of a design unless you have defined which layers of your layout design are conducting layers.

If you try to place or route your design before defining the conducting layers, you get an error message.

Define the layers you want to be conducting layers in the `viaLayers` constraint in the technology file.

See Technology File Requirements for Layout XL.

© 2023 Cadence Design Systems, Inc.

Layout XL Does Not Recognize Physical Vias

If you have an existing design in which Layout XL does not recognize physical vias, you need to add a property with name `function` and value `via` to the master cellview of the via or to the component description format (CDF) for via layout.

See Editing and Defining Properties.

Moving Software Executables To a New Location

If you move software executables to a different location other than specified by manufacturing, the custom placer function may take longer to start because it is searching for the executable.

Unmatched Instance Terminals are Automatically Removed

When loading designs that have been edited outside Layout XL, the connectivity extractor silently cleans up any instance terminals that have no associated terminal in the cell master. This can mean that you are asked to save your design, even though nothing has visibly changed.

Layout XL Commands Requiring Schematic Extraction

Certain Layout XL commands, such as *Update Components And Nets*, *Pick From Schematic*, *Generate All From Source*, check that the schematic connectivity is up-to-date before proceeding. If the schematic has been changed since it was last extracted, you see a message indicating that it needs to be re-extracted.

If you see this message every time you run one of these commands, it is likely that there are errors in the schematic that have prevented the previous extraction from completing successfully.

If this is the case, you need to run extraction again, fix any errors highlighted, and save the schematic. The next time you run a Layout XL command, you will not be prompted to re-extract the schematic.

Related Topics

Technology File Requirements for Layout XL

Connectivity Extraction in Layout XL

© 2023 Cadence Design Systems, Inc.

Troubleshoot Layout XL Compliance

You can face following issues related to Layout XL Compliance:

- Issues Faced when Setting up the PDK to Improve Layout XL Compliance
- Resolving Terminal Mismatches between Schematic and Layout
- Issues when Editing the CDF to Add Permute Rules

Issues Faced when Setting up the PDK to Improve Layout XL Compliance

The troubleshooting information in this section is targeted at CAD team or library owners who can implement the recommended set up at the PDK level so that the benefits of these global updates can be experienced by all the layout engineers using the library.

Prerequisites

- A layout with pins and cells or Pcells that can be bound to schematic counterparts.
- An LVS-clean layout that can use physical connectivity to establish the required schematic versus layout bindings.

The table below lists some likely layout issues that a CAD team member or a Library owner can resolve at the global level:

Layout issue	Impact	How to resolve...
Missing permute rule	Shorts reported in the layout during *Update Binding* or during a Layout XL-compliance check	Edit the Component Description Format (CDF) to add the missing permute rule. For information on what permute rules are and why it is important to set these correctly to create a layout that is Layout XL compliant, see Issues when Editing the CDF to Add Permute Rules.

© 2023 Cadence Design Systems, Inc.

Layout issue	Impact	How to resolve...
Stop layers not defined	Shorts reported in the layout during extraction of shapes at the level below the specified extraction level	Edit the technology file to define stop layers. For information on what stop layers are and why it is important to set these correctly to create a layout that is Layout XL compliant, see Issues when Editing the Technology File to Add Stop Layers.
Missing component types Terminal mismatches between schematic and layout	Layout XL compliance not as expected	Edit the Library Attributes Mapping (LAM) file to add missing component types and resolve terminal mismatches between the schematic and the layout. For information on editing the LAM file, see Library and Attributes Mapping File Syntax.

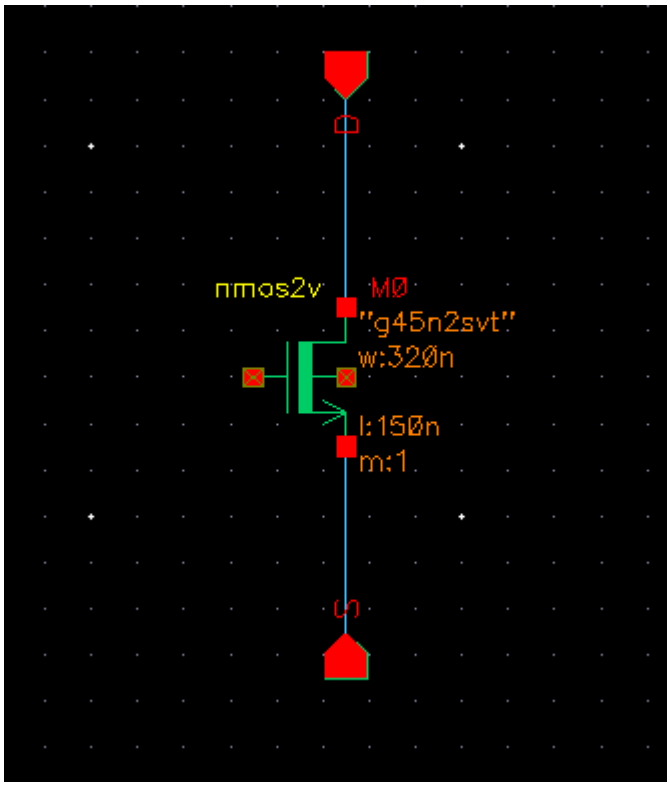
Issues when Editing the CDF to Add Permute Rules

Layout XL being connectivity-aware, it expects a net to be connected to the correct terminal. Else, a short is reported.

If you bring a layout created outside of Layout XL into Layout XL, it is likely that the layout will have its `S` and `D` (or the `n` and `p` terminals) connected interchangeably. This can cause shorts to be reported during an *Update Binding* run as the connectivity will be considered as incorrect in Layout XL. To resolve this, you can create permute rules for any devices that are likely to be connected in this manner.

© 2023 Cadence Design Systems, Inc.

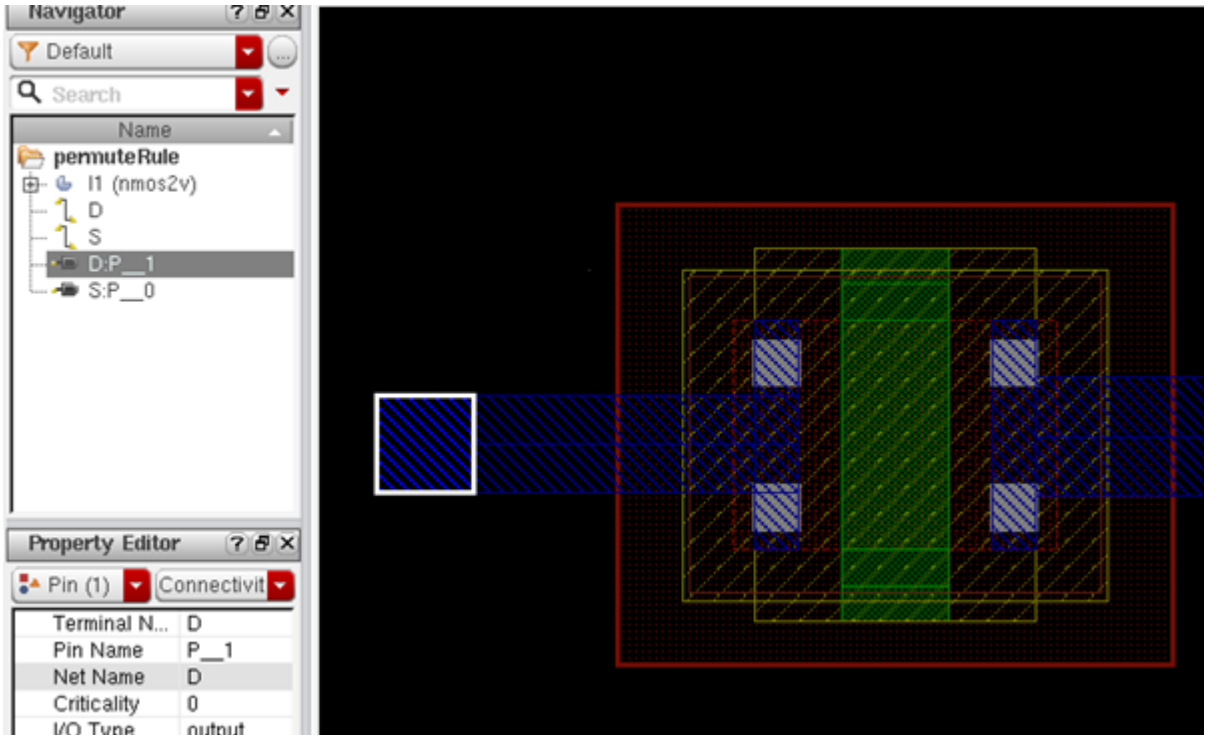
Consider the following example. Here a schematic with a single `nmos2v` device has its net `D` connected to terminal `D` and net `S` connected to terminal `S`, as shown in the figure below.



Let us assume that the layout was created outside Layout XL in a non-connectivity aware environment. Because the layout device is symmetrical and the logical connectivity from the

© 2023 Cadence Design Systems, Inc.

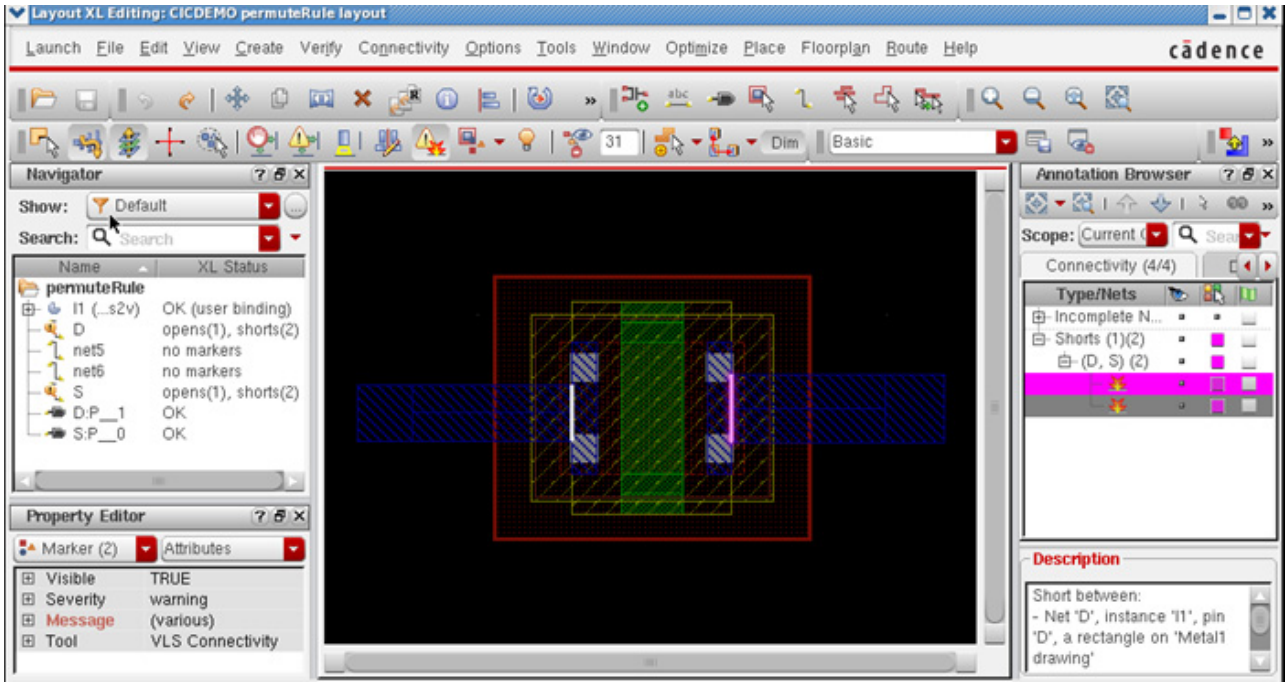
schematic was not considered, you could connect the nets S and D to either side of the device.



If we open the layout in Layout XL, the device connectivity will be considered. If the layout net D is connected to the S terminal, the physical connectivity (routing) is found to be different

© 2023 Cadence Design Systems, Inc.

from the logical connectivity in the schematic. This results in opens and shorts being reported because the terminals are incorrectly connected, as shown in the figure below.



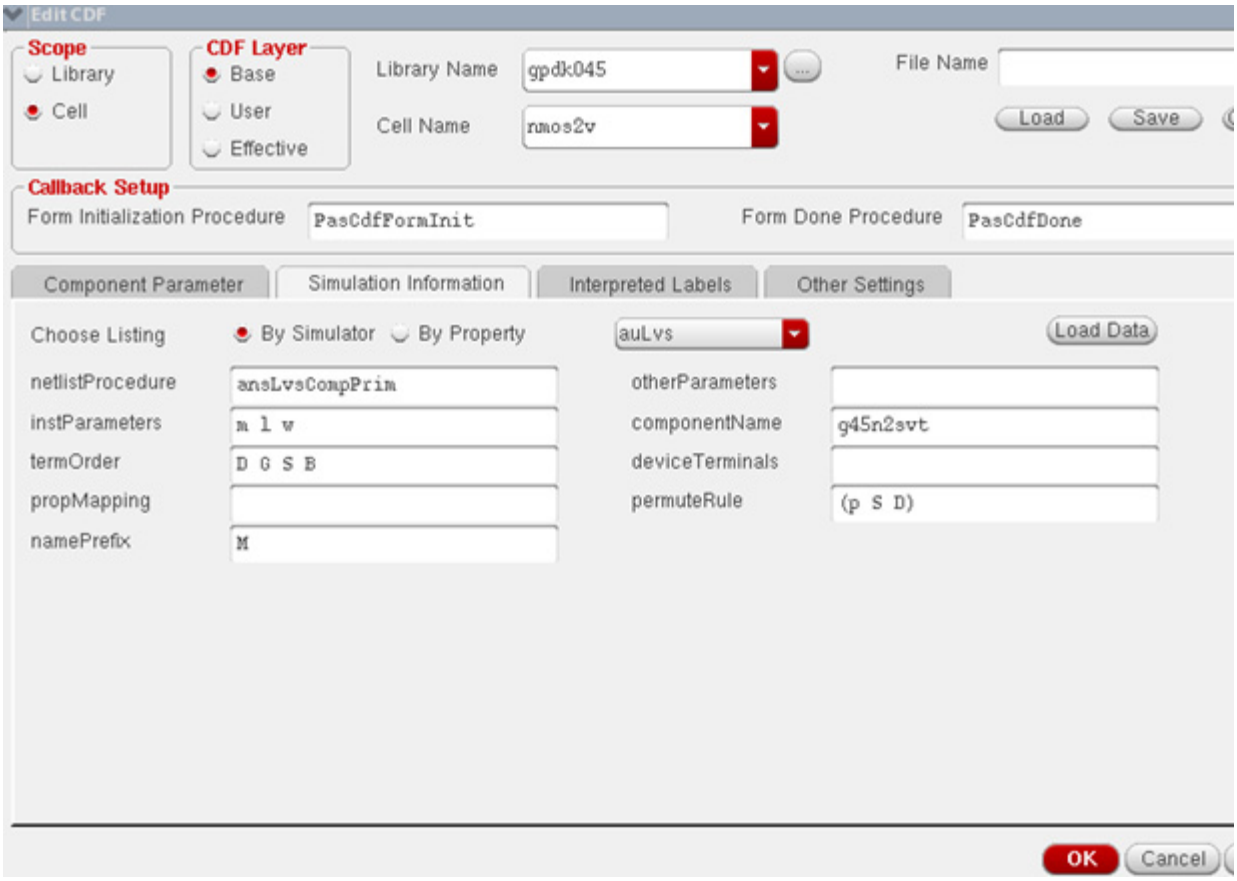
However, if it is acceptable to connect the terminals `S` and `D` interchangeably, we must specify the same using the appropriate CDF permute rule before launching the design in Layout XL.

To set the CDF permute rules:

1. Close the layout window, if open, and in the CIW, choose *Tools – CDF – Edit*.

© 2023 Cadence Design Systems, Inc.

The Edit CDF form displays, as shown below.



2. Set the value for *Scope* to *Cell*.
3. Set the value for *CDF Layer* to *Base*.
4. Specify the *Library Name* and *Cell Name*.

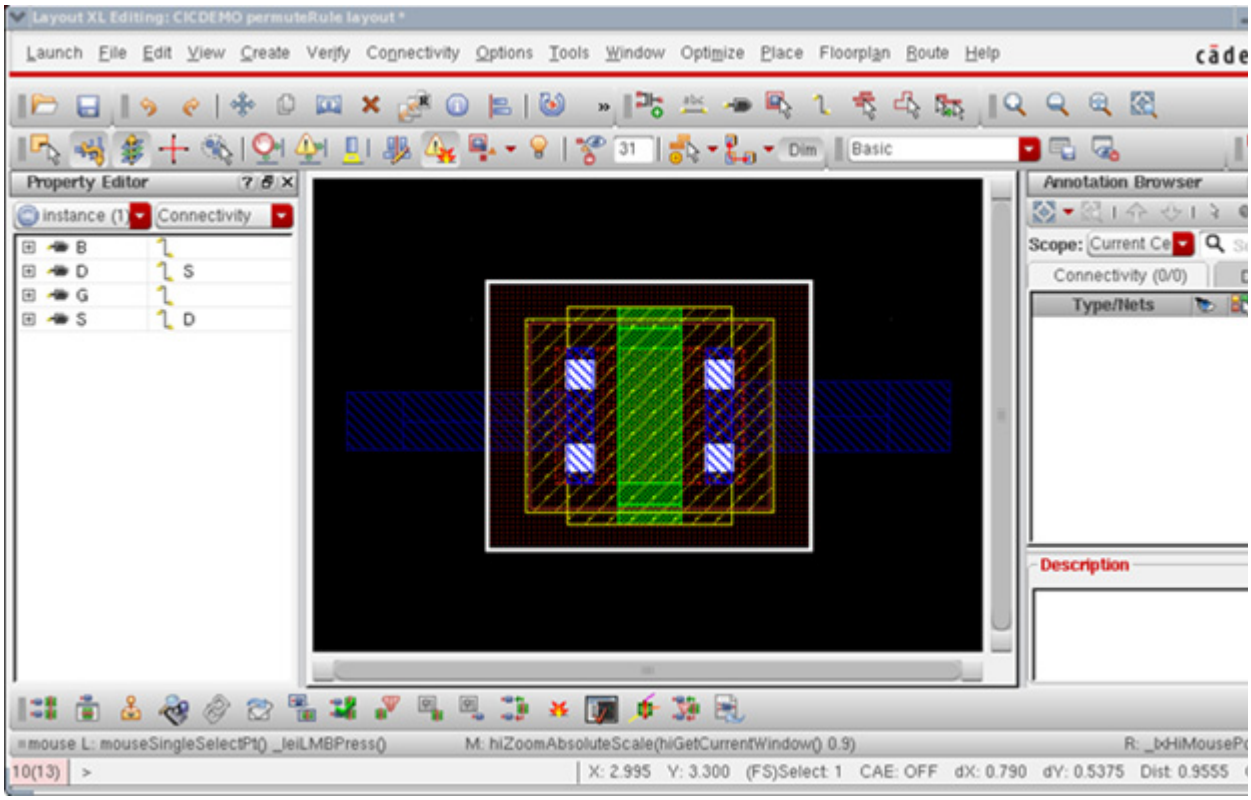
   For this example:

   - Use the *Library Name* `gpdk045`.
   - Use the *Cell Name* `nmos2v`.
5. Select the *Simulation Information* tab.
6. Select *auLvs* from the drop-down menu adjacent to *Choose Listing*.
7. In the *permuteRule* field, type the permute rule, such as (p S D).
8. Click *OK*.

For the CDF permute rule updates to take effect, open the design in Layout XL.

© 2023 Cadence Design Systems, Inc.

When the `nmos2v` is used in Layout XL, its `S` and `D` terminals can be connected interchangeably with no opens or short created, as shown below.



Additionally, when running *Update Binding,* it will use the permute rule, if required, to get the correct connectivity and increase the XL-compliance of the layout.

For more information on adding permute rules, see permuteRule Property.

Issues when Editing the Technology File to Add Stop Layers

In Layout XL, an extraction stop level `0` is sufficient to establish the connectivity, if connections are made to the pins of cells at the level below. However, if these connections are not made to the pins, the extraction level needs to be increased to consider the shapes at the level below.

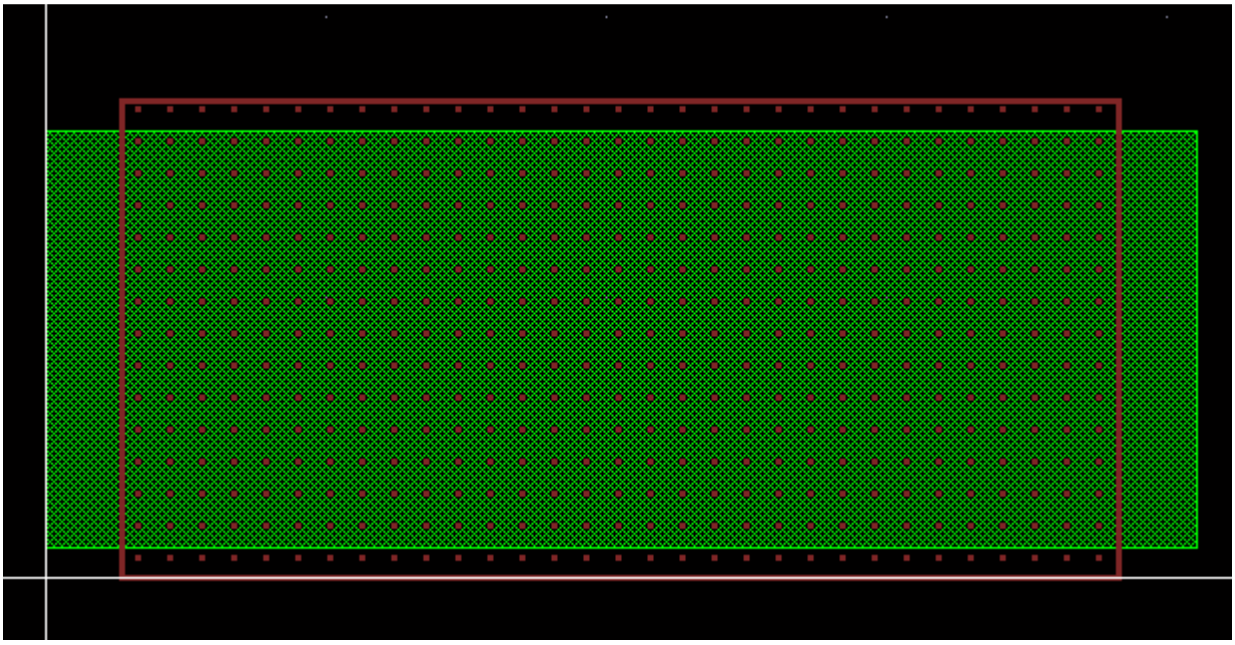
Layout XL now has the capability to extract through all the extractable shapes in the level below the specified extraction level. However, this can lead to shorts because the lower-levels cells were not originally created to support extraction. Therefore, a `stopLayer` may have to be defined to stop the extraction at the defined shape, and prevent a short being created.

© 2023 Cadence Design Systems, Inc.

Consider a layout that contains:

- Two pins, `A` and `B`, on `metal1`.
- Cell `I0` that contains two shapes–one on `Poly` and one on `Resdum`.

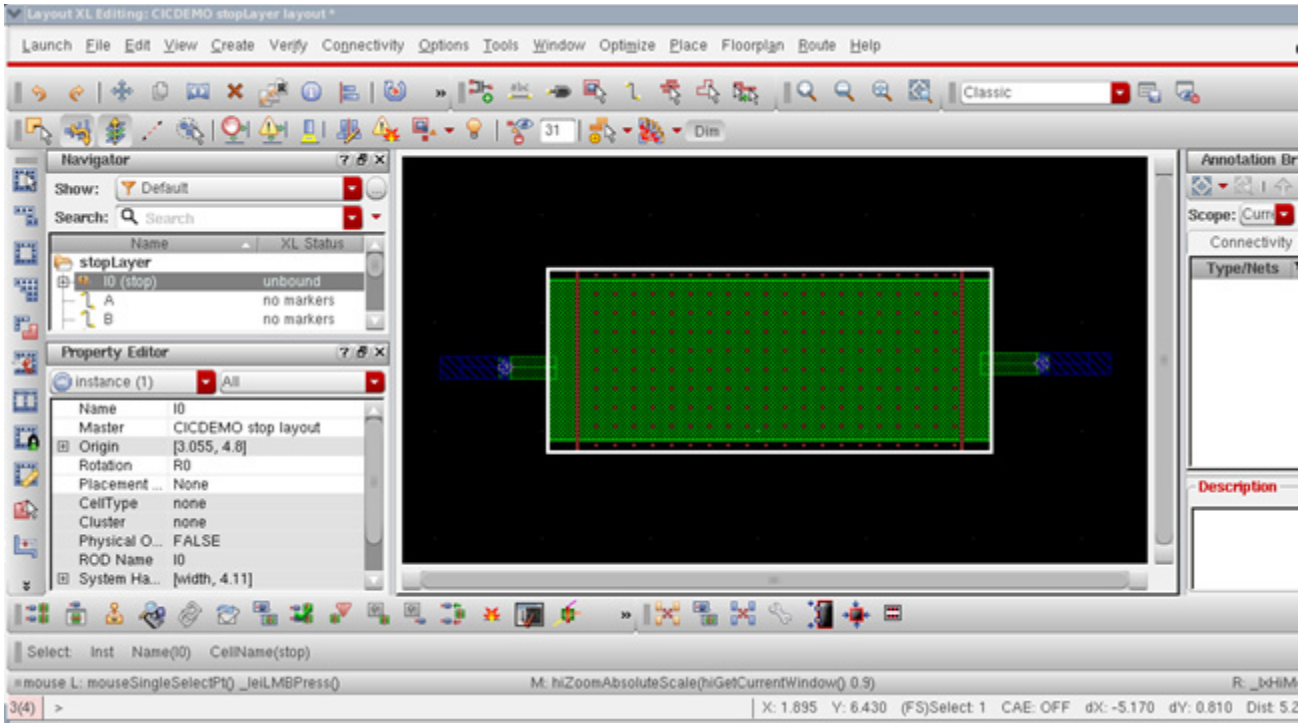
The figure below represents cell `I0` containing a poly shape (green) and a resdum shape (red dots).



At the top level (level `0`), routes are created from Pins `A` and `B` to connect to the poly shape in cell `I0`. If we extract at level `0` by setting the *Options – Connectivity – Connectivity –*

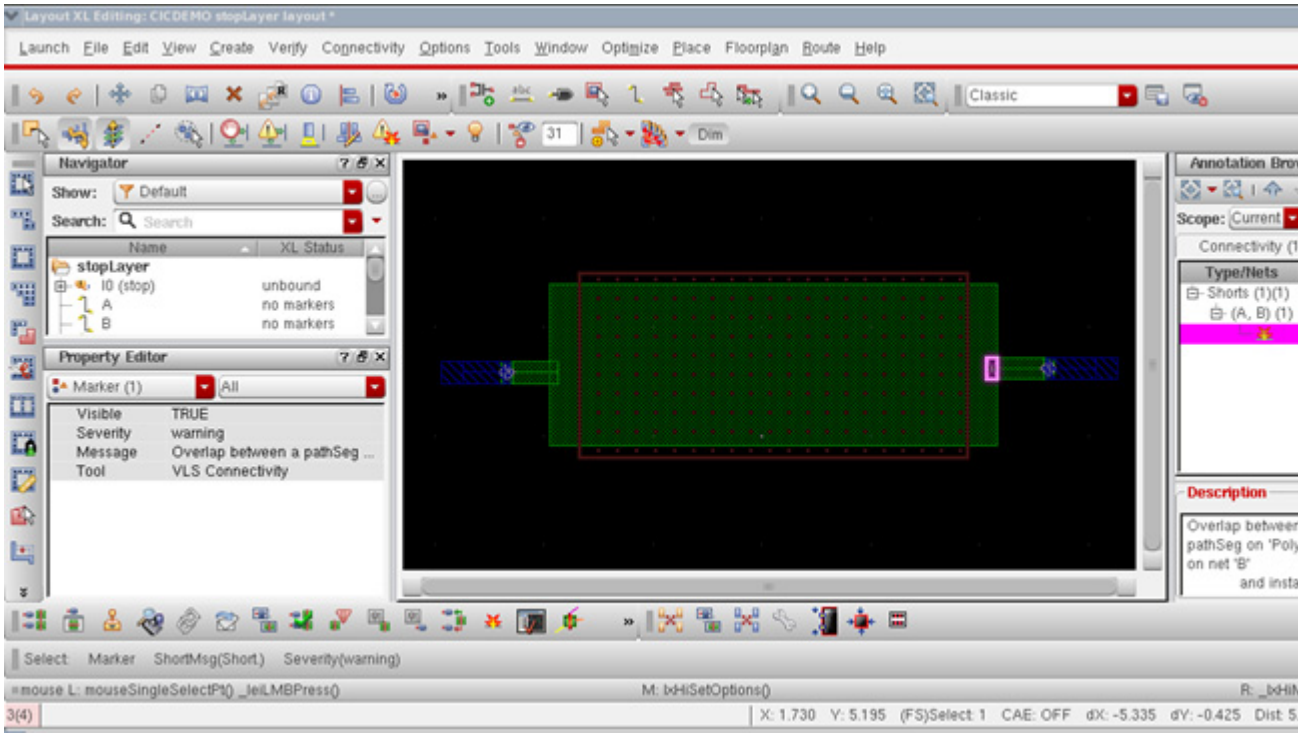
© 2023 Cadence Design Systems, Inc.

Hierarchy Control to `0`, no short is created between nets `A` and `B`, as shown below. This is so because the poly shape in `I0` is at level `1`, therefore, not considered for extraction.



© 2023 Cadence Design Systems, Inc.

However, if the extraction depth is increased to `1` and the extractor is run again, a short is created between net `A` and `B`, as shown below. This is expected because the extractor now considers the poly shape within `I0` as the extraction depth has been increased.



Although extracting at level `0` can prevent reporting shorts to lower levels, in some situations such as when finding off-pin connected at the level below, the extraction depth may need to be increased. This results in shorts being created. To avoid the shorts, a stop layer must be defined in the technology file. This can either be a real layer or a derived one.

For more information on adding stop layers, see validLayers – Stop Layers Specification.

Editing the LAM File to Increase Layout XL Compliance

What is a LAM file?

The Library and Attributes Mapping file—LAM file (`cph.lam`)—stores the schematic and layout library data, such as:

- symbol properties that drive layout generation
- component type information for a given library
- library and cell-level mapping information

© 2023 Cadence Design Systems, Inc.

The `cph.lam` file is stored in the design hierarchy at: `library/.cadence/dfII/cph/lam/cph.lam`

Currently, there is no graphical user interface available to help you create or edit a LAM file. Therefore, a new file or any edits should be created using your preferred text editor.

The values you set in the LAM file can also be set in Configure Physical Hierarchy (CPH), but if set in CPH, they are written to the physConfig view and will only apply to the current design. Setting the values in the LAM file ensures that every user benefits from the setting.

For more information on CPH, see Physical Hierarchy Configuration. For more information on the LAM file, see Library and Attributes Mapping File Syntax.

The values that can be set in a LAM file are:

- Component types – Layout XL uses component types to identify devices. For example, `nmos` and `pmos` component types are used to identify the device terminals for *Update Binding*. Additionally, the *width* parameter and the *folding threshold* are used by automatic folding.

  If the disableFolding environment variable is set and the folding threshold for the component type is set to `0`, folding is disabled.

- Terminal mapping between the schematic and the layout – Often there are terminal mismatches between schematic and layout and these are reported as connectivity differences in Layout XL. This terminal mismatch can also prevent *Update Binding* from giving a good result.

Setting Component Types in LAM

The setting of component types is recommended to help identify devices in Layout XL. It is good practice to define a component type for at least `nmos` and `pmos` devices and then map the appropriate cells to the component type, before the library is released to users.

The setting of component types allows the *Update Binding* command to recognize devices and their terminals.

Consider the following example. In the gpdk045 library there are two cells: `nmos2v` and `pmos2v`. To identify these devices as `nmos` and `pmos` devices respectively, we can create component types in the LAM file and map the cells to the correct component type. The following section of the sample LAM file illustrates this:

```
<PHYSICAL>
    <COMPONENT_TYPE_GROUPS>
        <LIB NAME="gpdk045">
```


© 2023 Cadence Design Systems, Inc.

```
            <COMPONENT_TYPE_GROUP NAME="nmos">
               <COMP_CLASS>NMOS</COMP_CLASS>
               <WIDTH>fw</WIDTH>
               <DRAIN>D</DRAIN>
               <GATE>G</GATE>
               <SOURCE>S</SOURCE>
               <BULK>B</BULK>
               <ACTIVE_LP>Oxide drawing</ACTIVE_LP>
               <FOLD_THRESH>1e-06</FOLD_THRESH>
            </COMPONENT_TYPE_GROUP>
            <COMPONENT_TYPE_GROUP NAME="pmos">
               <COMP_CLASS>PMOS</COMP_CLASS>
               <WIDTH>fw</WIDTH>
               <DRAIN>D</DRAIN>
               <GATE>G</GATE>
               <SOURCE>S</SOURCE>
               <BULK>B</BULK>
               <ACTIVE_LP>Oxide drawing</ACTIVE_LP>
               <FOLD_THRESH>1e-06</FOLD_THRESH>
            </COMPONENT_TYPE_GROUP>
        </LIB>
    </COMPONENT_TYPE_GROUPS>
    <MAPPINGS>
        <LIB NAME="gpdk045">
            <CELL NAME="nmos2v">
               <COMPONENT_TYPE>
                   <NAME>nmos</NAME>
                   <LIB>gpdk045</LIB>
               </COMPONENT_TYPE>
            </CELL>
            <CELL NAME="pmos2v">
               <COMPONENT_TYPE>
                   <NAME>pmos</NAME>
                   <LIB>gpdk045</LIB>
               </COMPONENT_TYPE>
            </CELL>
        </LIB>
    </MAPPINGS>
</PHYSICAL>
```


© 2023 Cadence Design Systems, Inc.

The sample above is a partial LAM file. To view a complete LAM file, see Example – Complete LAM File.

In our example, the `nmos` and `pmos` cells are in the `gpk045` library. The LAM file for this library will be automatically read when Layout XL is opened, and you can view it using the CPH window. In CPH, you can verify if the LAM file settings you made are correct.

To verify the LAM file update in CPH:

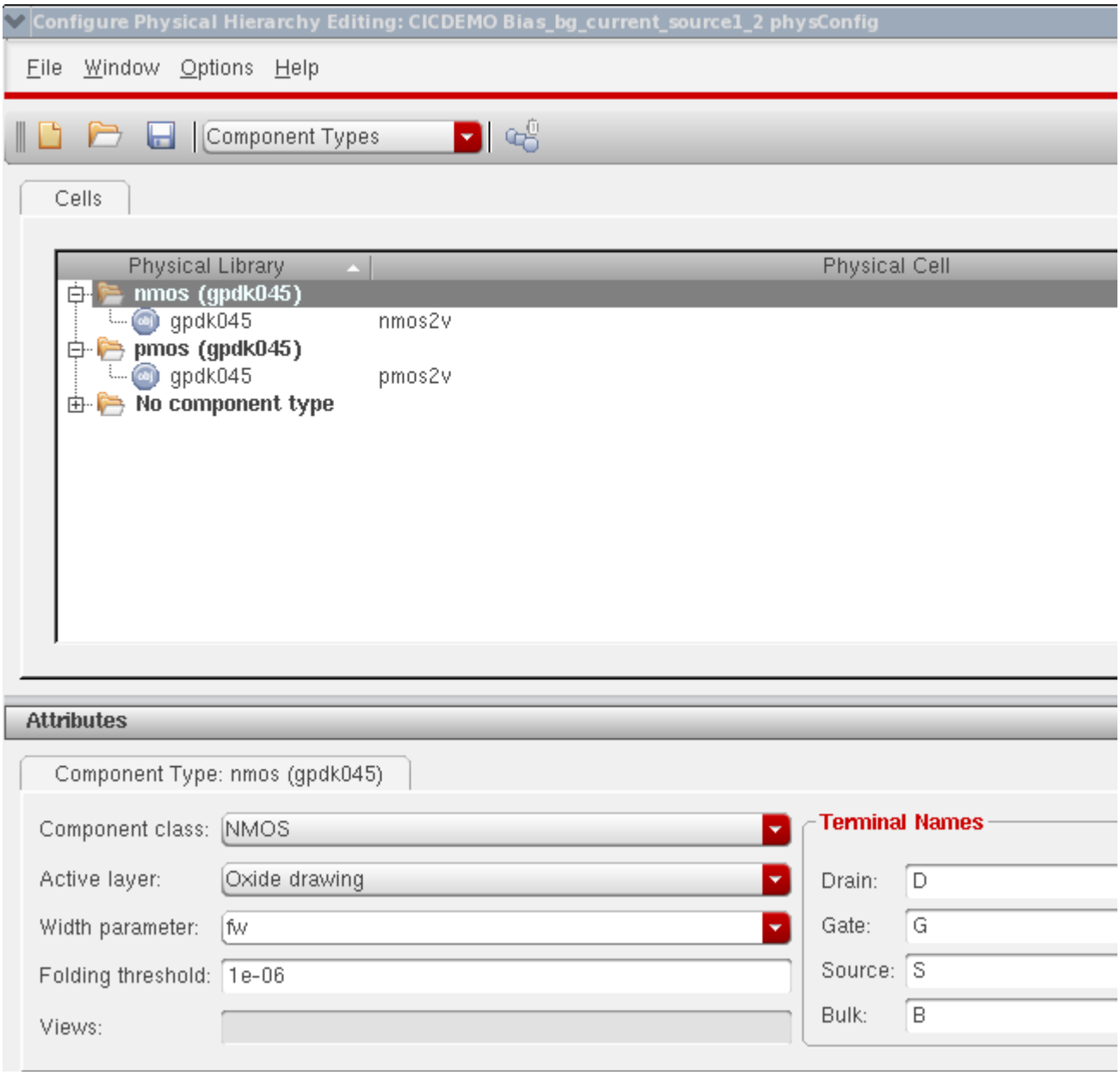
1. Choose *Launch – Configure Physical Hierarchy*.

   The *Configure Physical Hierarchy* window displays.

2. Select *Component Types* from the drop-down menu at the top of the window.

© 2023 Cadence Design Systems, Inc.

The CPH window displays the two component types – `nmos` and `pmos` – that you added using the LAM file, as shown in the figure below.

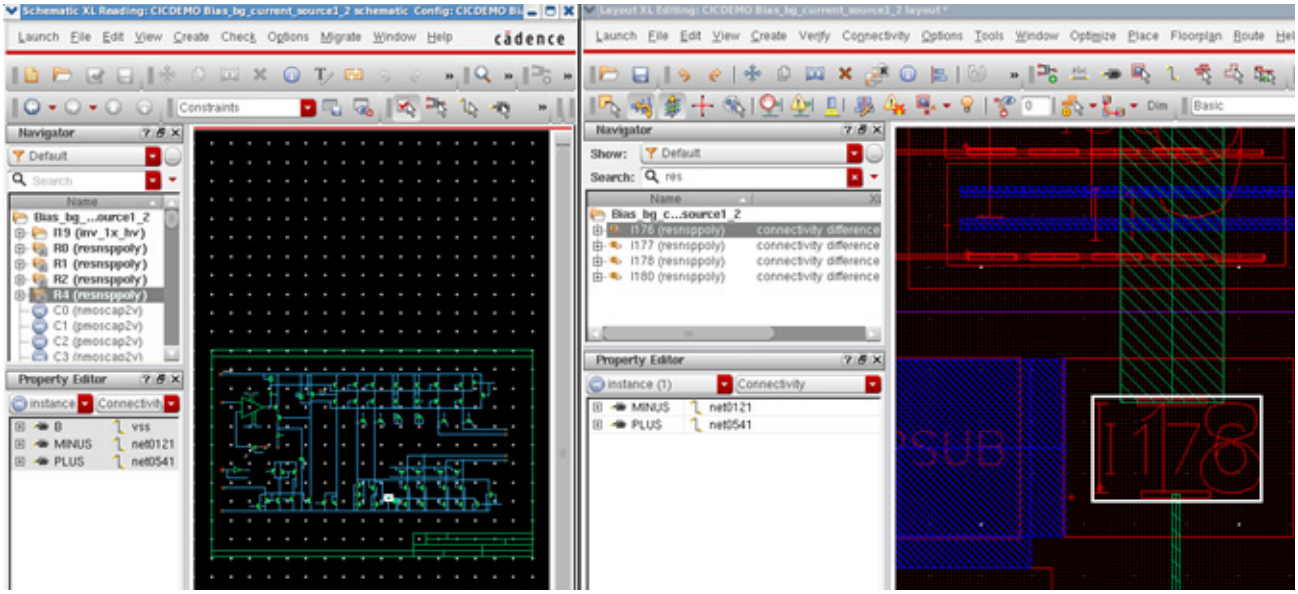


Resolving Terminal Mismatches between Schematic and Layout

Consider the example below. In the `gpdk045` library, there is a cell `resnspoply`. As can be seen in the screenshow below, the schematic has `3` terminals: `B`, `MINUS` and `PLUS` while the

© 2023 Cadence Design Systems, Inc.

layout has 2 terminals: `MINUS` and `PLUS`. This causes Layout XL to report a connectivity difference, as seen in the *XL Status* column of the Navigator.



The connectivity difference displayed in the figure above can also cause the *Update Binding* command to result in bad bindings. Therefore, it is good practice to resolve terminal mismatches between the schematic and the layout using the LAM file before running the *Update Binding* command.

To stop the connectivity differences from being reported, you can add an *ignore for generation* property to the `B` terminal in the library LAM file, as shown below.

In the previous example, we had to edit the physical section of the LAM file to set up the component types. In this example, we are defining how the layout is generated from the schematic. Therefore, we need to edit the logical section of the LAM file.

```
<LOGICAL>
    <LIB NAME="gpdk045">
        <CELL NAME="resnsppoly">
            <VIEW NAME="symbol">
                <PARAMETERS>
                    <IGNORE>
                        <CHECK></CHECK>
                        <GEN></GEN>
                    </IGNORE>
                </PARAMETERS>
                <TERMINALS>
                    <IGNORE>
```


© 2023 Cadence Design Systems, Inc.

```
                        <CHECK></CHECK>
                        <GEN>B</GEN>
                    </IGNORE>
                </TERMINALS>
            </VIEW>
        </CELL>
    </LIB>
</LOGICAL>
```

The sample above is a partial LAM file. To view a complete LAM file, see Example – Complete LAM File. To verify if the LAM file is set correctly, you can verify the settings in the Configure Physical Hierarchy window.

To verify the LAM settings in CPH:

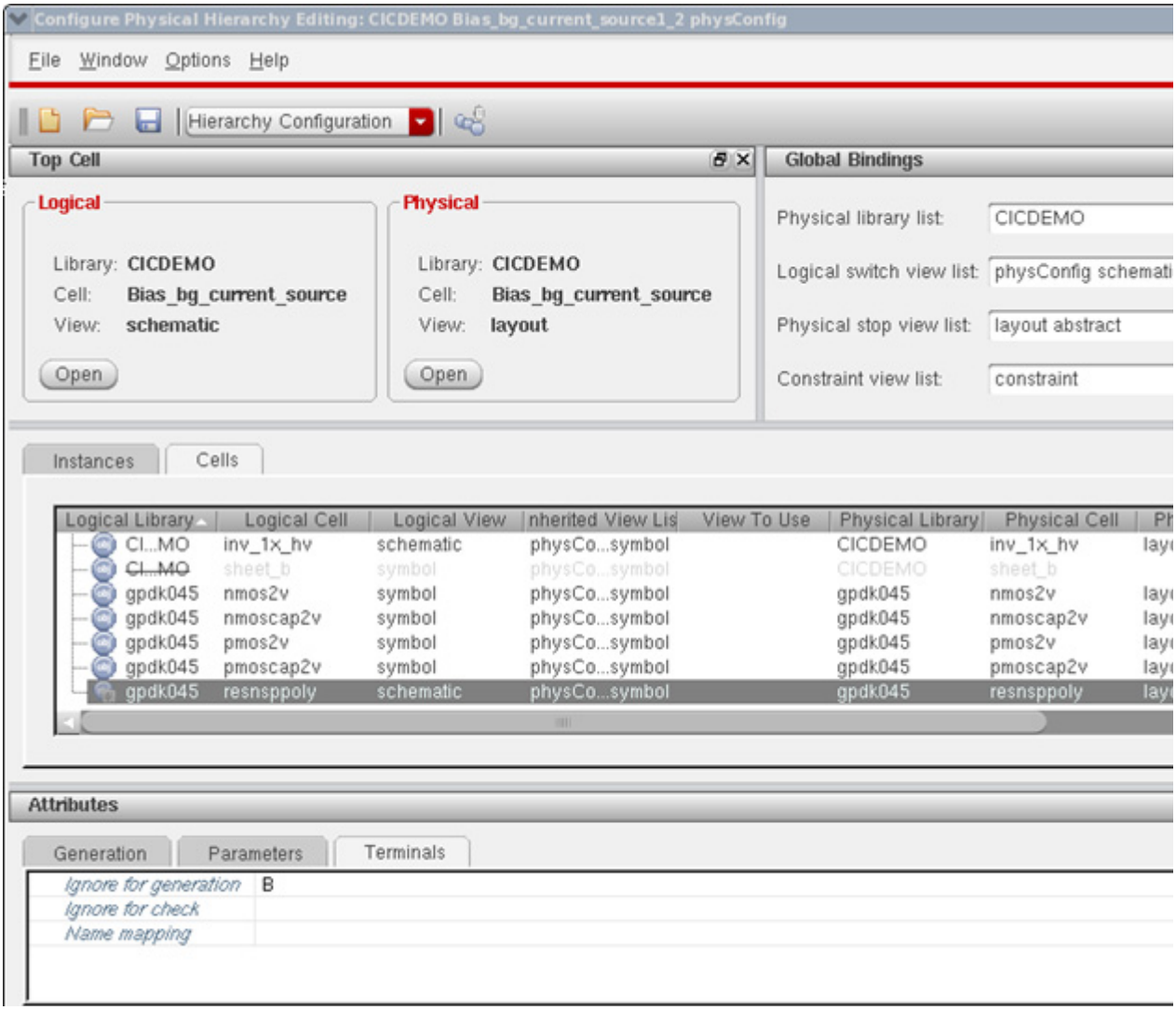
1. *Launch – Configure Physical Hierarchy*.

   The Configure Physical Hierarchy form opens in *Hierarchy Configuration* mode.

2. In the middle pane, select the *Cells* tab and then select the *resnsppoly* entry in the table.

© 2023 Cadence Design Systems, Inc.

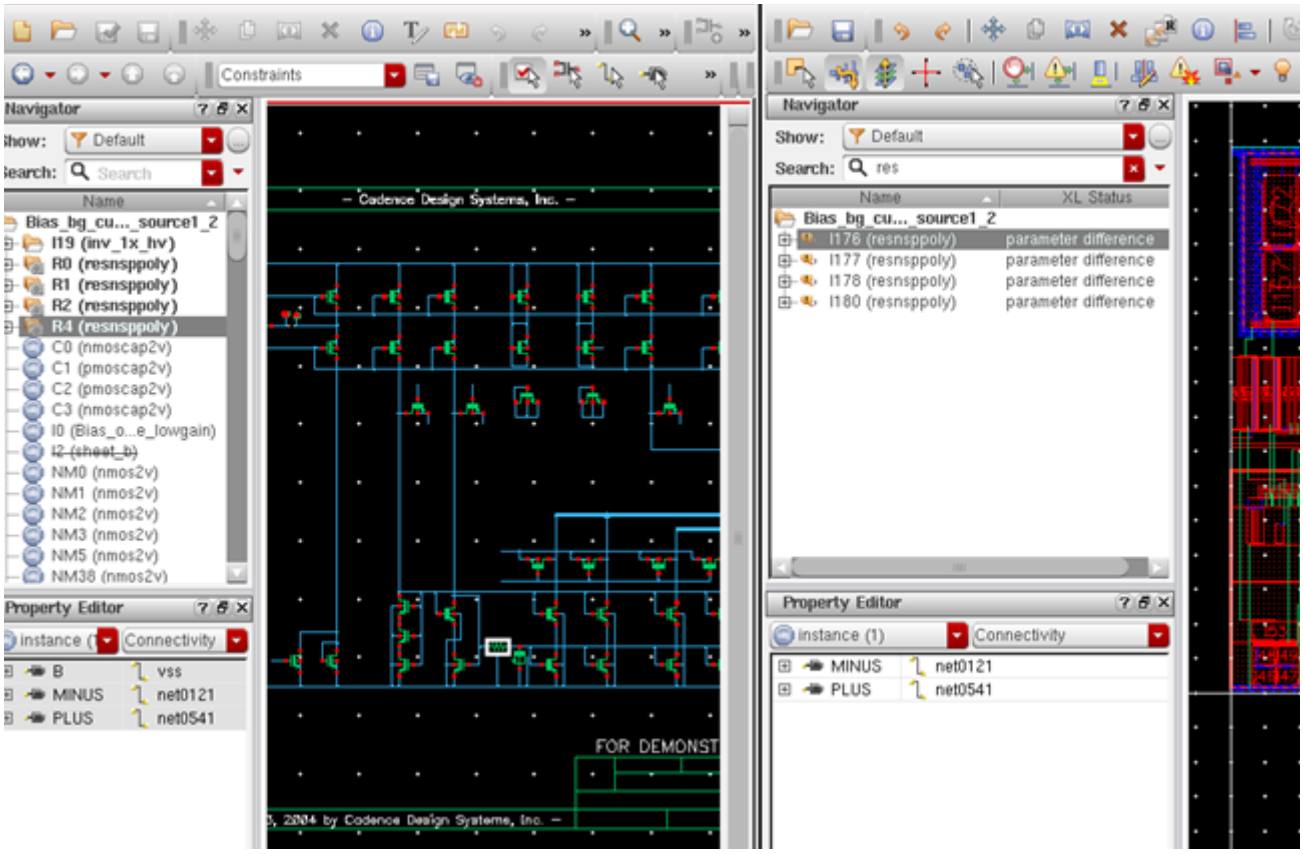
3. In the *Attributes* tab at the bottom of the window, select the *Terminals* tab, as displayed in the figure below.



4. Notice that the *Ignore for generation* property displays on terminal `B`, as set in the LAM file.

© 2023 Cadence Design Systems, Inc.

After editing the LAM file, if you open a cellview in Layout XL and generate a `resnsppoly` device from the schematic, it will no longer report a connectivity difference. This is because the `B` terminal is now ignored, as shown in the figure below.



In the two *Update Binding* examples illustrated above, the XL status still reported parameter differences, although the binding and connectivity was correct. These differences should be checked, and if needed, an ignore property can be set for these in the *Parameters* section in the LAM file.

The sample LAM file below illustrates the use of the `ignore` property in the *Parameters* section.

```
<PARAMETERS>
    <IGNORE>
        <CHECK>sa sb leftTap RightTap</CHECK>
        <GEN>simM</GEN>
    </IGNORE>
</PARAMETERS>
```


© 2023 Cadence Design Systems, Inc.

Example – Complete LAM File

The sample below represents a complete LAM File that is set to improve the Layout XL compliance for a design.

```
<!DOCTYPE LAM_FILE>
<LIBRARY_DATA>
    <VERSION NUMBER="0.2" />
    <LOGICAL>
        <LIB NAME="gpdk045">
            <CELL NAME="resnsppoly">
                <VIEW NAME="symbol">
                    <PARAMETERS>
                        <IGNORE>
                            <CHECK></CHECK>
                            <GEN></GEN>
                        </IGNORE>
                    </PARAMETERS>
                    <TERMINALS>
                        <IGNORE>
                            <GEN>B</GEN>
                        </IGNORE>
                    </TERMINALS>
                </VIEW>
            </CELL>
        </LIB>
    </LOGICAL>
    <PHYSICAL>
        <COMPONENT_TYPE_GROUPS>
            <LIB NAME="gpdk045">
                <COMPONENT_TYPE_GROUP NAME="nmos">
                    <COMP_CLASS>NMOS</COMP_CLASS>
                    <WIDTH>fw</WIDTH>
                    <DRAIN>D</DRAIN>
                    <GATE>G</GATE>
                    <SOURCE>S</SOURCE>
                    <BULK>B</BULK>
                    <ACTIVE_LP>Oxide drawing</ACTIVE_LP>
                    <FOLD_THRESH>1e-06</FOLD_THRESH>
                </COMPONENT_TYPE_GROUP>
                <COMPONENT_TYPE_GROUP NAME="pmos">
                    <COMP_CLASS>PMOS</COMP_CLASS>
```


© 2023 Cadence Design Systems, Inc.

```
                    <WIDTH>fw</WIDTH>
                    <DRAIN>D</DRAIN>
                    <GATE>G</GATE>
                    <SOURCE>S</SOURCE>
                    <BULK>B</BULK>
                    <ACTIVE_LP>Oxide drawing</ACTIVE_LP>
                    <FOLD_THRESH>1e-06</FOLD_THRESH>
                </COMPONENT_TYPE_GROUP>
            </LIB>
        </COMPONENT_TYPE_GROUPS>
        <MAPPINGS>
            <LIB NAME="gpdk045">
                <CELL NAME="nmos2v">
                    <COMPONENT_TYPE>
                        <NAME>nmos</NAME>
                        <LIB>gpdk045</LIB>
                    </COMPONENT_TYPE>
                </CELL>
                <CELL NAME="pmos2v">
                    <COMPONENT_TYPE>
                        <NAME>pmos</NAME>
                        <LIB>gpdk045</LIB>
                    </COMPONENT_TYPE>
                </CELL>
            </LIB>
        </MAPPINGS>
    </PHYSICAL>
</LIBRARY_DATA>
```


© 2023 Cadence Design Systems, Inc.

Troubleshoot Binding Update Issues

You can face following issues when updating bindings:

- Ignoring Route Cells
- Updating Instance Bindings after Running MakeCell

Ignoring Route Cells

If route cells are set to be ignored during *Update Binding* but some devices have stop layers defined while others do not, you can do either of the following for the cells to be processed appropriately during an *Update Binding* run.

- Add the stop layer for the devices that do not have them already defined and run *Update Binding* again.
- Run *Update Binding* without ignoring route cells and manually add ignores on route cells after they have been reported by *Update Binding*.

See validLayers – Stop Layers Specification.

Updating Instance Bindings after Running MakeCell

If you run the *MakeCell* command after flattening a device, the *Update Binding* command may not be able to update the instance bindings to reflect the change. To enable the *Update Binding* command to effectively update the instance bindings:

1. In the Navigator assistant or layout canvas, right-click the made cell instance and choose the *Hierarchy – Unset Transparent* command to prevent the binder from binding devices inside the made cell.
2. Right-click the made cellview and edit it to rename the terminals to match the schematic nmos device.
3. Relaunch Layout XL on the top cellview, where the made cell is instantiated.
4. Run the *Update Binding* command using the default options.

Related Topics

Update Binding Form

© 2023 Cadence Design Systems, Inc.

Troubleshoot Performance

You can face the following issues related to performance:

- Design Fails to Open or Takes a Long Time to Open
- Layout XL Is Slow

Design Fails to Open or Takes a Long Time to Open

If you design takes a long time to open in Layout XL, or fails to open at all, try the following.

- If your design has many devices which overlap but have not been abutted previously, and you start Layout XL with full connectivity extraction, by default the software attempts to automatically abut the devices in question. To prevent this, set the lxLocalAbutment environment variable to `t`.
- If your design is hierarchical, it might be that the logical elaboration performed by Configure Physical Hierarchy is causing the problem.

  You can use the physical stop view list to limit the logical elaboration by switching on the cphStopLogicalElabAtPhysLeaf environment variable. This stops the elaboration of the logical design hierarchy when a node is reached which maps to one of the view names specified in the *Physical stop view list* field.

  When `cphStopLogicalElabAtPhysLeaf` is set to `t`, global nets below the leaf instance in the schematic are not considered during the elaboration. If your design relies on such nets, Cadence recommends that you do not switch on this environment variable.

Layout XL Is Slow

The following considerations can help you improve Layout XL performance speed.

- Instead of displaying flight lines for all the incomplete nets in the design, use the highlighting features of the path command or probing to determine connections or display only nets you are working on or that are relevant to the current task.
- If you need extra pins in the layout for feedthrough nets or substrate connections, consider adding those after you have completed the main portion of the design. Adding extra pins increases the number of nets the extractor manages and degrades performance.
- Specify only layers to be used as interconnect in the `validLayers` constraint Using more layers than necessary causes the extractor to check all shapes on those layers,

© 2023 Cadence Design Systems, Inc.

which slows performance. If you do not have interconnect to wells, do not specify the well in the `validLayers` constraint.

Related Topics

Technology File Requirements for Layout XL

© 2023 Cadence Design Systems, Inc.

A

Layout XL Environment Variables

This appendix provides information on the environment variable names, descriptions, and graphical user interface equivalents for Virtuoso® Layout Suite XL layout editor (Layout XL).

Note: Only the environment variables documented in this chapter are supported for public use. All other Layout XL environment variables, regardless of their name or prefix, and undocumented aspects of the environment variables described below, are private and are subject to change at any time.

- List of Layout XL Environment Variables
- Batch Checker Environment Variables

Inherited Environment Variable Settings

Layout XL honors many graphic and layout environment variables that are useful for basic layout editing functions. For information on these environment variables, see Layout XL Basic Editing Environment Variables.

Setting Environment Variables

Environment variables control the values of the Layout XL options. For a list of all the Layout XL environment variables and their values, see List of Layout XL Environment Variables.

There are three ways in which you can set environment variables:

- To set an environment variable that is applied every time you start Layout XL, add the setting to your `.cdsenv` or `.cdsinit` file. For more information, see Setting Environment Variables in a .cdsenv or .cdsinit File.
- To set an environment variable that is applied for the duration of the current session, use the `envSetVal()` command in the CIW. For more information, see Setting Environment Variables in the CIW.

© 2023 Cadence Design Systems, Inc.

- To set an environment variable from the layout window, use the Layout XL commands or, if appropriate, the form associated with the command you are using. For more information, see Setting Environment Variables in the Connectivity Form.

  Note: The graphical user interface provides access only to specific environment variables.

Setting Environment Variables in a .cdsenv or .cdsinit File

To have your environment variables set automatically when you start Layout XL, do one of the following.

- Include the environment variables in the `.cdsenv` file in your home directory; for example,

  ```
  layoutXL chainFolds boolean t
  ```

- Include an `envSetVal( )` command in your `.cdsinit` file

  ```
  envSetVal("layoutXL" "chainFolds" 'boolean t)
  ```

Setting Environment Variables in the CIW

If you use any environment variable values consistently and do not want to set these values each time you use a command, you can set the variables to the value you normally use in the CIW and it will remain valid for the duration of the current session.

To set environment variables for a single session, do one of the following.

- Include `envSetVal( )` in any other Cadence SKILL file you load.
- Type `envSetVal( )` in the CIW.

For example, to set the Layout XL `chainFolds` variable, which causes Layout XL to abut the folds of newly folded devices into chains, type the following in the CIW or include it in a setup file.

```
envSetVal("layoutXL" "chainFolds" 'boolean t)
```

To determine the current value of any Layout XL environment variable, type the following in the CIW.

```
envGetVal("layoutXL" "chainFolds")
```

Setting Environment Variables in the Connectivity Form

To set environment variables in the Connectivity form:

© 2023 Cadence Design Systems, Inc.

1. From the layout window menu bar, choose *Options – Connectivity XL*.

   The Connectivity Form form is displayed.

2. Select one of the following.

   - *Global* to apply options to the current cellview and all other views of the current cell.
   - *Cellview* to apply options only to the current cellview. Selecting this option inactivates the *Extraction*, *Generation*, and *Parameters* tabs, along with certain options on the *General* and *Display* tabs.

3. Select the options you want to apply to your design.

4. In the *Load/Save* group box at the bottom of the *General* tab, choose whether you want to save the options settings to the current cellview, the design library, the technology library, or a file.

5. Click *OK* or *Apply*.

   The environment variable values you choose are enforced for all subsequent design sessions until you change the value of the variables again.

To load previously-saved options into this form, type the name of a file and click *Load from*. To delete settings from a cell or a library, click *Delete from*. You cannot delete environment values from a file.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

GUI Options and Corresponding Environment Variables

Hidden Environment Variables with No Corresponding GUI Option

© 2023 Cadence Design Systems, Inc.

List of Layout XL Environment Variables

abutAddUniqueDummyNet

abutChainEvent

abuttedPcellParamUpdate

abutSetEditedAsMoved

abutWithoutConnectivity

addIxStickyNetPropOnDummy

allowPinResizing

allowPinResizingInEdit

allowSelectedParamUpdate

alwaysGenerateProps

alwaysUseDescriptionForFiltering

autoAbutment

autoArrange

autoHideEmptyTabs

automaticHighlight

autoMirror

autoMirrorChains

autoMirrorFilter

autoPermuteNoAbutProp

autoSavePhysConfigUpdate

autoSpace

autoZoomEnabled

© 2023 Cadence Design Systems, Inc.

autoZoomFactor

autoZoomIsFixed

backAnnotateInstances

bindAllCellsInDesign

bindClearConn

bindCorrectMaster

bindCreateTransparent

bindCrossRefFile

bindCurrentLevel

bindExtract

bindExtractedNetlistFile

bindFactorWarnLimit

bindFile

bindFlattenLayoutCreate

bindIgnoreBulk

bindIgnoreDummies

bindIgnoreDummiesFullyConn

bindIgnoreRouteCells

bindIncr

bindIncrAddIgnore

bindIncrConn

bindIncrMessage

bindIncrPropagateConn

bindInitConn

© 2023 Cadence Design Systems, Inc.

bindInitInstTerms

bindLayoutStop

bindNameOnly

bindNewSplitParam

bindPermuteRules

bindPreserveExistingBindings

bindPreserveLayoutHierarchy

bindPreserveUserBindings

bindPVSRulesFile

bindPVSScale

bindRemoveUnboundTerminalsNoPins

bindReportCellsNoTerms

bindReportDummies

bindReportHierarchy

bindReportReduction

bindReportRouteCells

bindReportTerminalMismatch

bindReportUnconnInstTerms

bindSourceSkip

bindSourceStop

bindStrict

bindUpdateName

bindUseFingerWidth

bndRemoveDeviceConnectivity

© 2023 Cadence Design Systems, Inc.

capacitanceParamNames

casDisplayUnboundInsts

casGlyphs

casInfoBalloonOnHover

casMarkers

casMarkerTypeLimit

casSelected

casVisibleUnboundMarkers

chainAbutStrategy

chainAllowSingleBulk

chainCellOrient

chainDummyAbutment

chainEffort

chainExtendSelection

chainFolds

chainInPlace

chainLeftNet

chainMirror

chainMirrorEquivOrients

chainOptimize

chainPartitionByNumFins

chainPartitionByCompType

chainPartitionSizeMax

chainPermute

© 2023 Cadence Design Systems, Inc.

chainPermutePins

chainPreserveExistingChains

chainPreserveRows

chainSyncChains

chainUseAbutSpacing

chainUseDeviceOrder

checkRFUnimplementedTerms

checkBindings

checkCellsAsLeaf

checkCompType

checkDuplicateObjects

checkExcludeCellViews

checkExtractLayout

checkMasterCompType

checkMissingParamsOrProps

checkMissingTermPins

checkParamsOnly

checkPermute

checkRemoveDevice

checkShortsAndOpens

checkStateConfirmModeChange

checkStrictFingers

ciwWindow

cloneArrayFlipFirstColumn

© 2023 Cadence Design Systems, Inc.

cloneArrayFlipFirstRow

cloneArrayFlipNextFromPreviousColumn

cloneArrayFlipNextFromPreviousRow

cloneArrayRowOrColValue

cloneArrayXSpacing

cloneArrayYSpacing

cloneJustify

cloningAutomaticUpdateLayoutParameters

cloningDoExactMatch

cloneNumberPerRowOrCol

cloneSynchronizeUnlockedColor

cloneTransformFrom

cloningSearchIn

cloningSearchInConstraintOnly

cloningSearchInSelSetOnly

cloningUpdateConstraint

closeSchWindow

colorDraglines

colorOrNot

constraintAwareEditing

cphAutoUprev

cphCheckEmptyConfig

cphCheckTopCellName

cphHierForceDescendStopLibs

© 2023 Cadence Design Systems, Inc.

cphIncrementalElab

cphLayoutLibraryInLibList

cphLayoutSync

cphMasterLibraryOnly

cphNonDefaultAutomaticName

cphOpenWin

cphPromptConfigOrSchematic

cphScrollShowCellsMenu

cphShowInstanceColumns

cphStopLevel

cphStopLogicalElabAtPhysLeaf

cphStrictSchematicCheck

cphUseLayoutLibrary

cphUseLayoutViewName

cphVHSizeLimit

createBoundaryLabel

createImplicitBusTerminals

createPinsFromLabels

crossProbeHier

crossSelect

crossSelect

crossSelectInstTerm

crossSelectLayoutShape

deleteConfirmModeChange

© 2023 Cadence Design Systems, Inc.

deleteUnmatchedInsts

deleteUnmatchedPins

deviceExtractType

disableCASOptionsPopUp

disableFolding

disableVXLClonedModgenEditPopUp

displayConfigureTab

drdUseNetName

dummyBackAnnotateLocalNetActive

dummyBackAnnotateRelaxActiveRule

electricalPinSetup

enableCloneColoringSupport

equivalentAbutClasses

excludedViewList

extractShortLocatorAutoZoomToFit

extractShortLocatorCommonShapesColor

extractShortLocatorDisplayMode

extractShortLocatorHaloTransparency

extractShortLocatorHighlightLimit

extractShortLocatorNetColor1

extractShortLocatorNetColor2

extractShortLocatorNumThreads

extractShortLocatorOtherShapesColor

extractShortLocatorShowInstancePins

© 2023 Cadence Design Systems, Inc.

extractShortLocatorShowPinsAndLabels

extractShortLocatorShowVirtualLabelNetOnShapes

extractShortLocatorTrueColor

extractShortLocatorUseInstancePins

extractShortLocatorVirtualLabelNetOnShapesFontHeight

extractShowMustConnectMarkers

extractShowUnimplementedInstTerms

extractShowWeakMarkers

fingerSplit

fingerSplitSetWidth

flattenHierNameSep

flattenKeepLabels

flattenPreserveConn

flattenPreserveUnthreadedConn

flightLineEnable

flightLineForAllMustConnectInstTerms

foldEnforceWidthMatch

foldFingerSplitUseSchWidth

foldIgnoreMfactor

foldInPlace

foldInPlaceDirection

foldKeepFolds

foldsLimit

foldOptimizeSupplyAndGroundNets

© 2023 Cadence Design Systems, Inc.

foldUseThreshold

generateModgens

generatePinsDisplayLimit

generateXSpacing

generateYSpacing

hardBlockColor

hideDraglinesForGlobalNets

hideMarkersWhenBrowserHidden

hierarchyDepth

hierMfactorNames

hierSummaryColumns

hierSummaryLogFileName

highlightColors

highlightedIsSelectable

highlightObjRMB

highlightPotentialSwap

highlightPotentialSwapColor

highlightShapes

ignoreTermsWithNonRoutingLabels

incompleteNetFilter

incompleteNetFilter

incompleteNetFilterDefaultList

incompleteNetFilterMode

inductanceParamNames

© 2023 Cadence Design Systems, Inc.

infoWindow

infoWindowPos

initAreaFuncName

initAspectRatio

initAspectRatioOption

initCreateBoundary

initCreateInstances

initCreateMTM

initCreatePadPins

initCreatePins

initCreateSnapBoundary

initDoFolding

initDoStacking

initEstimateArea

initGlobalNetPins

initIOLabelType

initIOPinLayer

initIOPinLayerNames

initIOPinName

initIOPinPurposeNames

initPinHeight

initPinMultiplicity

initPinWidth

initPrBoundaryH

© 2023 Cadence Design Systems, Inc.

initPrBoundaryW

initUtilization

insertChain

insertChainAbutViolation

lamReadCdsLibs

lamReadFileMessage

lamReadUserLibs

lamReuseCache

layoutWindow

lswWindow

lxAbutPassGroupIdToEvent1

lxAllowPseudoParallelNets

lxChainAlignNMOS

lxChainAlignPMOS

lxCanSwapSDFunction

lxDeltaWidth

lxDummyBackAnnotateAll

lxDummyBackAnnotateMFactor

lxDummyBackAnnotateMFactorName

lxDummyBackAnnotateMissingTerm

lxEvalCDFCallbacks

lxFingeringNames

lxGenerateInBoundary

lxGenerationOrientation

© 2023 Cadence Design Systems, Inc.

lxGenerationTopLevelOnly

lxGetSignifDigits

lxGroundNetNames

lxInitResetSource

lxLocalAbutment

lxPositionCompact

lxPositionMinSep

lxPositionPinsOnBoundary

lxPositionUseInstPrB

lxRetainFoldOrient

lxSchematicDefaultApp

lxSchematicEditMode

lxSchExtractTopLevelOnly

lxStackMinimalFolding

lxStackPartitionParameters

lxSupplyNetNames

lxSwapSDFunction

lxUpdateFoldedWidth

lxUseLibList

lxValidateXLParameterEvaluation

lxWidthTolerance

makeCellCreatePins

makeCellPinsBelowBoundary

makeCellPRBoundary

© 2023 Cadence Design Systems, Inc.

makeCellPreserveConn

makeCellTransparent

matchedToIgnore

mfactorNames

mfactorSplit

moveAsGroup

netTracerAutoZoomOnSelected

netTracerColor

netTracerConnectedLayer

netTracerConstraintGroup

netTracerCrossHighlightNetInSchematic

netTracerDisplayDepth

netTracerDisplayMode

netTracerExcludePurposes

netTracerExcludePurposesName

netTracerFromNet

netTracerHaloFadeoutShapes

netTracerHaloForPins

netTracerHaloForShapes

netTracerHaloStyleThick

netTracerHaloTransparency

netTracerMaxCutShapes

netTracerNumShapesLimitForRefresh

netTracerNumThreads

© 2023 Cadence Design Systems, Inc.

netTracerOn

netTracerPopupSelectionWindow

netTracerPopupSelectionWindow

netTracerRange

netTracerRegion

netTracerSaveMethod

netTracerShowPreview

netTracerStartLevel

netTracerStepMode

netTracerStopLayer

netTracerStopLevel

netTracerUpdateIncrementally

netTracerViaLayer

openConnRef

openConnRefTab

openLocation

openMode

opticalPinSetup (Virtuoso Photonics Option)

paramsToCheck

paramsToIgnore

paramsToIgnoreForCheck

paramTolerance

passFigIdsToEvent3

phoAbutClass (Virtuoso Photonics Option)

© 2023 Cadence Design Systems, Inc.

phoAbutFunction (Virtuoso Photonics Option)

phoAbutNonPcells (Virtuoso Photonics Option)

phoComposeMaster (Virtuoso Photonics Option)

phoComposeMasterPromptOff (Virtuoso Photonics Option)

phoPinInputAngle (Virtuoso Photonics Option)

phoPinLayer (Virtuoso Photonics Option)

phoPinRadius (Virtuoso Photonics Option)

phoPinWidth (Virtuoso Photonics Option)

physOnlyTerminalsRemoveBang

pibCellName

pibCreateBlockageType

pibCreatePinLPPChoice

pibFloatPurposeNames

pibLayerNames

pibLibName

pibPinLPPName

pibPinPurposeName

pibPurposeNames

pibPushAllAsBlockage

pibPushBlockage

pibPushCloseWire

pibPushFloats

pibPushNonPowerGndNet

pibPushPowerGndNet

© 2023 Cadence Design Systems, Inc.

pibPushRows

pibPushMode

pibPushSelection

pibPushWhole

pibNetType

pibReMasterInst

pibViewName

pibWSPActivePattern

placeCloneAsArray

prBoundarySnap

prefixLayoutInstNamesWithPipe

preserveAreaBoundary

preserveBlockages

preserveClusters

preserveClusterBoundaries

preservePRBoundary

preserveRows

preserveSnapBoundary

preserveSnapPatterns

preserveTrackPattern

probeCycleHilite

probeDevice

probeDuringCreate

probeHiliteLayer

© 2023 Cadence Design Systems, Inc.

probeInfoInCIW

probeNet

probePin

processBatchViolations

propagateAssignNet

propsUsedToIgnoreObjs

propsUsedToIgnoreObjsForCheck

removeDuplicateObjects

renameClonesWhenFamilyIsRenamed

renameConflictingLayoutInstances

resistanceParamNames

saveViewSubType

schematicWindow

schLayLibraryPair

scopeLevel

segmentParamNames

setPPConn

sfactorNames

sfactorSplit

showDraglinesForDistantConns

showIncompleteNets

showPropagateConnectivityReport

showTabs

showUnplacedInstances

© 2023 Cadence Design Systems, Inc.

smartDeletePathSegBetweenVia

snapCpaToPlacementGrid

softBlockColor

stopList

stretchWithPRBoundary

substrateNetParamNames

suppressAbutFailMsgs

suppressExpansion

swapIncludeOrientation

swapOrigin

tabPosition

transistorWidthParamNames

turnInfixOffWhenSmartSnapping

unfoldUseLayoutWidths

updateBindMfactorSplit

updateBindPhysicalBinding

updateLayoutInstNames

updateLayoutNetNames

updateLayoutParameters

updateNetMinMaxVoltage

updateNetSigType

updateNetsOnly

updateOneToOneMappings

updateParamChangeNumFolds

© 2023 Cadence Design Systems, Inc.

updateParamCopyFolds

updateParamsForCheck

updatePlaceBelowBoundary

updatePlacementStatus

updatePlacementStatusInBoundary

updateReplacesMasters

updateSchParamWithEvalValue

updateSelectedComponents

updateWithMarkers

useConstantAreaStretch

useHierMfactorNames

viewList

vlGenAutoDetectCells

vlGenCreateCells

vlGenCreateCellsInPhysConfigLib

vplGenCreateCells

vplGenLibs

vplGenParams

vplGenSetDefaultLayout

xlComplianceCheckDevices

xlComplianceCreateCasMarkers

xlComplianceFilter

xlComplianceHtml

xlComplianceHtmlOpen

© 2023 Cadence Design Systems, Inc.

xlComplianceLogDir

xlComplianceLogs

xlStatus

© 2023 Cadence Design Systems, Inc.

abutAddUniqueDummyNet

```
layoutXL abutAddUniqueDummyNet boolean { t | nil }
```

Description

Enables chaining and interactive abutment of unconnected dummy devices—devices that have no source or drain instance terminals. Parameter updates performed on abutted Pcell instances are preserved.

The default is `t`.

When set to `nil`, unconnected dummy devices cannot be chained or interactively abutted because no automatic net assignment is supported.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "abutAddUniqueDummyNet")
envSetVal("layoutXL" "abutAddUniqueDummyNet" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

abutChainEvent

```
layoutXL abutChainEvent boolean { t | nil }
```

Description

Controls whether abutment Event 6 is triggered as the last event of the abutment (or unabutment) flow.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "abutChainEvent")
envSetVal("layoutXL" "abutChainEvent" 'boolean t)
```

Related Topics

Abutment Function Flow

Unabutment Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

abuttedPcellParamUpdate

```
layoutXL abuttedPcellParamUpdate boolean { t | nil }
```

Description

Retains the parameter updates performed on abutted Pcell instances.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "abuttedPcellParamUpdate")
envSetVal("layoutXL" "abuttedPcellParamUpdate" 'boolean nil)
```

Related Topics

Abutment of Dummy Shapes in Pcells

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

abutSetEditedAsMoved

```
layoutXL abutSetEditedAsMoved boolean { t | nil }
```

Description

Controls whether the edited instance among the two abutted instances is considered as the one that moved during internal reabutment.

The default is `nil`, which means that the moving instance from the last `abutGroup` continues to be considered as the one that moved after the edit during internal reabutment.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "abutSetEditedAsMoved")
envSetVal("layoutXL" "abutSetEditedAsMoved" 'boolean t)
```

Related Topics

Abutment of Dummy Shapes in Pcells

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

abutWithoutConnectivity

```
layoutXL abutWithoutConnectivity boolean { t | nil }
```

Description

Enables abutment of instances that have no connectivity information.

The default is `t`, which means that overlapping instances with no nets can get abutted.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "abutWithoutConnectivity")
envSetVal("layoutXL" "abutWithoutConnectivity" 'boolean nil)
```

Related Topics

Device Abutment

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

addlxStickyNetPropOnDummy

```
layoutXL addlxStickyNetPropOnDummy boolean { t | nil }
```

Description

Checks whether newly-created dummies have the `lxStickyNet` property already added and adds the property if it is not present.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "addlxStickyNetPropOnDummy")
envSetVal("layoutXL" "addlxStickyNetPropOnDummy" 'boolean t)
```

Related Topics

lxStickyNet Property

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

allowPinResizing

```
layoutXL allowPinResizing boolean { t | nil }
```

Description

Specifies whether soft block pins need to be resized during pin snapping.

The default is `t`.

GUI Equivalent

Command:	*Floorplan -> Floorplan Global Options*
Field:	*Allow Pin Resizing* (Floorplan Global Options)

Examples

```
envGetVal("layoutXL" "allowPinResizing")
envSetVal("layoutXL" "allowPinResizing" 'boolean t)
envSetVal("layoutXL" "allowPinResizing" 'boolean nil)
```

Related Topics

Snapping Soft Block Pins to Grid

Setting Floorplan Global Options

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

allowPinResizingInEdit

```
layoutXL allowPinResizingInEdit boolean { t | nil }
```

Description

Specifies whether soft block pins need to be resized during pin snapping for the current move operation.

The default is `t`, which means that the `allowPinResizingInEdit` setting that you make will be applicable only for the current move operation.

GUI Equivalent

Command:	*Edit – Move – F3*
Field:	*Allow Pin Resizing*

Examples

```
envGetVal("layoutXL" "allowPinResizingInEdit")
envSetVal("layoutXL" "allowPinResizingInEdit" 'boolean t)
envSetVal("layoutXL" "allowPinResizingInEdit" 'boolean nil)
```

Related Topics

Move Form

Snapping Soft Block Pins to Grid

Objects Moving

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

allowSelectedParamUpdate

```
layoutXL allowSelectedParamUpdate boolean { t | nil }
```

Description

Controls whether the selected parameter updates are enabled for an instance to resolve the parameter differences reported in the CAS tab after a *Check Against Source* run.

The default is `t`.

When set to `nil`, the *Update Selected Layout Parameters* and *Update Selected Schematic Parameters* commands are not listed in the shortcut menu for instances that are reported with parameter differences in the Annotation Browser CAS tab. Therefore, a parameter update can be made using the commands: *Update All Layout Parameters* or *Update All Schematic Parameters*. Both these commands update all the parameters, not only the selected ones.

GUI Equivalent

Command:	Right-click a parameter difference marker reported in the *Annotation Browser CAS* tab
Field:	*Update Selected Layout Parameters* (command) *Update Selected Schematic Parameters* (command)

Examples

```
envGetVal("layoutXL" "allowSelectedParamUpdate")
envSetVal("layoutXL" "allowSelectedParamUpdate" 'boolean nil)
```

Related Topics

CAS TAB Marker Categories

Design Check Against Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

alwaysGenerateProps

```
layoutXL alwaysGenerateProps boolean { t | nil }
```

Description

Transfers properties from schematic instances to layout instances during the *Generate All From Source* or *Generated Selected From Source* commands. Properties are transfered in addition to the CDF parameters, which are always transferred.

The default is `t`.

When set to `nil`, it uses the value of `checkParamsOnly` to either generate only CDF parameters or both CDF parameters and instance properties.

GUI Equivalent

Command:	*Options – Connectivity (Parameters tab)*
Field:	*Always consider properties for generation*

Examples

```
envGetVal("layoutXL" "alwaysGenerateProps")
envSetVal("layoutXL" "alwaysGenerateProps" 'boolean nil)
```

Related Topics

Parameters

Connectivity

Design Check Against Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

alwaysUseDescriptionForFiltering

```
ab alwaysUseDescriptionForFiltering boolean { t | nil }
```

Description

Controls whether the content of the *Description* column is always used by the Annotation Browser search filter widget.

The default is `t`, which means that the *Search* feature seeks matching patterns in all the displayed columns and in the *Description* column, even if the *Description* column is not displayed.

When set to `nil`, the *Search* feature seeks matching patterns only in the displayed columns.

GUI Equivalent

None

Examples

```
envGetVal("ab" "alwaysUseDescriptionForFiltering")
envSetVal("ab" "alwaysUseDescriptionForFiltering" 'boolean nil)
```

Related Topics

Annotation Browser Toolbar

Annotation Browser Assistant

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

autoAbutment

```
layoutXL autoAbutment boolean { t | nil }
```

Description

Turns on automatic abutment, which abuts prepared transistors so that they can share pins. Abutment is switched on for both automatic and interactive layout generation and during automatic placement using the Virtuoso custom digital placer.

The default is `t`, which means that devices that have not been abutted but are overlapping can be abutted. If the devices are already abutted, they are not reabutted.

Important

Auto-abutment works only when automatic extraction is on, when the `extractEnabled` environment variable is set to `t`.

GUI Equivalent

Command:	*Options – Connectivity* (*Generation* tab)
Field:	*Abut transistors*

Examples

```
envGetVal("layoutXL" "autoAbutment")
envSetVal("layoutXL" "autoAbutment" 'boolean t)
envSetVal("layoutXL" "autoAbutment" 'boolean nil)
```

Related Topics

extractEnabled

Connectivity Form

Prerequisites for Automatic Device Abutment in Layout XL

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

autoArrange

```
layoutXL autoArrange boolean { t | nil }
```

Description

Controls whether Layout XL automatically rearranges its three windows on your desktop when you launch the application.

The default is `t`, meaning that the windows are positioned based on the values specified for the following environment variables.

- `ciwWindow`
- `lamReadFileMessage`
- `schematicWindow`

If those environment variables are not set, or they are set to the default value – `"((0.0 0.0) (0.0 0.0))"` – Layout XL tiles the three windows based on the height and width of your screen.

GUI Equivalent

Command:	*Options – Connectivity* (*Display* tab)
Field:	*Auto arrange windows* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "autoArrange")
envSetVal("layoutXL" "autoArrange" 'boolean t)
envSetVal("layoutXL" "autoArrange" 'boolean nil)
```

Additional Information

If you are using dual monitors and you do not want Layout XL to occupy the entire space available, set the `USE_SINGLE_MONITOR` global environment variable before you start Virtuoso from the command line.

Related Topics

ciwWindow

© 2023 Cadence Design Systems, Inc.

lamReadFileMessage

schematicWindow

Connectivity Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

autoHideEmptyTabs

```
ab autoHideEmptyTabs boolean { t | nil }
```

Description

Automatically hides all the Annotation Browser tabs that are empty.

The default is `nil`.



Important

Even when no tab in the Annotation Browser has any marker to display and the `autoHideEmptyTabs` environment variable is set to `t`, the Annotation Browser still displays at least one tab. The displayed tab is either the currently active tab or the one that is assigned the first position.

GUI Equivalent

Command:	*Right-click any Annotation Browser tab*
Field:	*Auto Hide Empty Tabs (*shortcut menu option in Annotation Browser)

Examples

```
envGetVal("ab" "autoHideEmptyTabs")
envSetVal("ab" "autoHideEmptyTabs" 'boolean t)
```

Related Topics

Annotation Browser Shortcut Menus

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

automaticHighlight

```
ab automaticHighlight string { "New markers" | "New marker categories" | "Inherit
     from parent category" }
```

Description

Controls the highlight state set on new markers reported in the Annotation Browser.

- *New markers*: Highlights the new markers, irrespective of the highlight state of the category under which the marker is placed in the Annotation Browser.
- *New marker categories:* Highlights the new markers when they are placed under a new category in the Annotation Browser.
- *Inherit from parent category:* Highlights the new markers, if they fall under a category in the Annotation Browser for which the highlight state is not set to OFF.

The default is `"New marker categories"`.

GUI Equivalent

Command:	*Right-click the Browser Pane – Options*
Field:	*Auto*matic *highlighting*
	(Annotation Browser Options)

Examples

```
envGetVal("ab" "automaticHighlight")
envSetVal("ab" "automaticHighlight" 'string "New markers")
envSetVal("ab" "automaticHighlight" 'string "Inherit from parent category")
```

Related Topics

Navigator

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

autoMirror

```
layoutXL autoMirror boolean { t | nil }
```

Description

Specifies that if a short violation is created during abutment, Virtuoso Layout XL first mirrors the device in question to resolve the short. Only if this is unsuccessful, does Virtuoso Layout XL attempt to resolve the short by permuting the pins.

After mirroring an instance within its bounding box to resolve the short, abutment usually occurs, if the resulting overlap is between figures that can trigger abutment. However, the triggering of abutment is not guaranteed, especially in the case of asymmetric instances where no overlap is created post-mirroring.

The default is `t`.

If you switch the option off, Virtuoso Layout XL uses only pin permutation to attempt resolving the shorts.

GUI Equivalent

Command:	*Options – Connectivity* (*Generation* tab)
Field:	*Mirror transistors*

Examples

```
envGetVal("layoutXL" "autoMirror")
envSetVal("layoutXL" "autoMirror" 'boolean nil)
```

Additional Information

- If one of the devices is in any type of group (including a synchronous clone), the other device must be in the same group for automatic mirroring to occur.
- Automatic mirroring is disabled when constraint-aware editing is on and constraints exist between the instances and chains under consideration.

Related Topics

Connectivity Form

© 2023 Cadence Design Systems, Inc.

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

autoMirrorChains

```
layoutXL autoMirrorChains boolean { t | nil }
```

Description

Specifies that if a short violation is created during abutment of a chain of devices, then Layout XL first mirrors the chain in question in order to resolve the short. Only if this is unsuccessful does Layout XL attempts to resolve the short by permuting the pins.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "autoMirrorChains")
envSetVal("layoutXL" "autoMirrorChains" 'boolean nil)
```

Additional Information

Chains can be mirrored only if

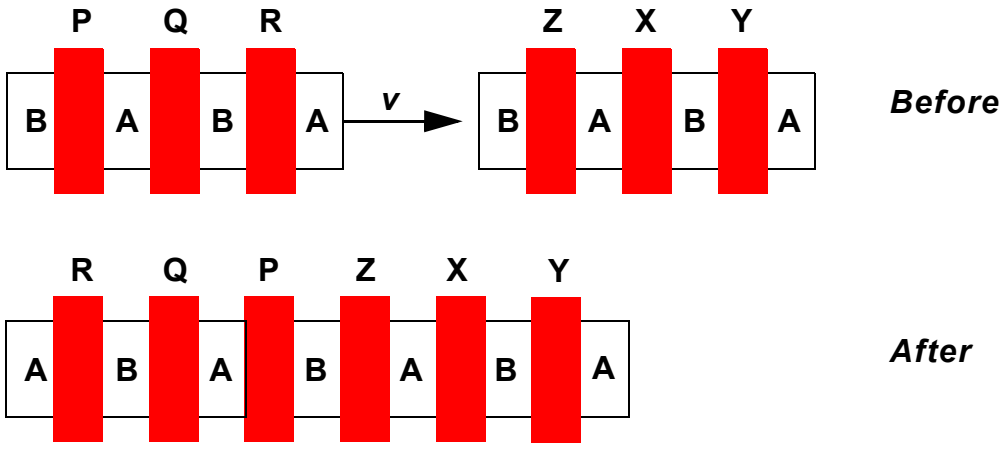
- The `autoMirror` environment variable is set to `t`
- They have no physical connectivity or routing defined
- The cells in the chain all belong to the same group (including synchronous clones) or no group
- The chain contains no multiply-abutted cells
- Any end-pin violations are valid left/right or top/bottom pairs

The following transformations are possible. Which of the three is employed is controlled internally by the abutment code. You are not required to choose a particular transformation. No transformation is performed which would break the chain.

- *Chain Mirror*

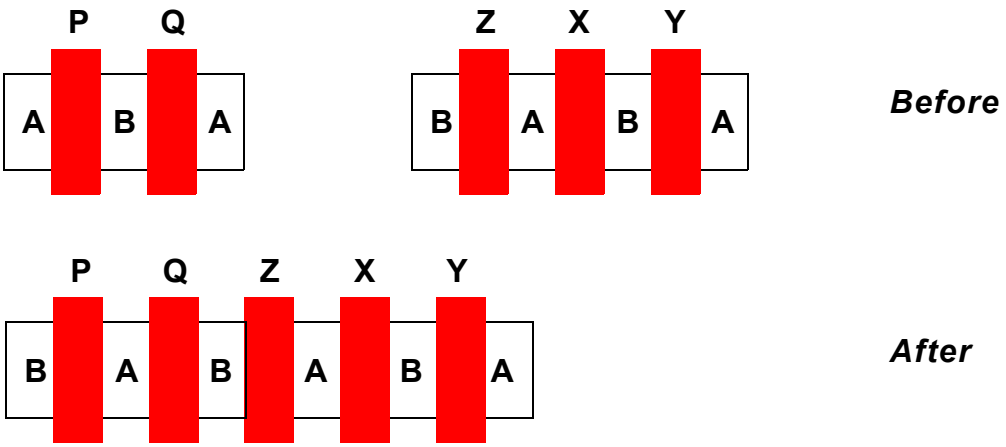
© 2023 Cadence Design Systems, Inc.

Mirrors the complete moving chain about its central Y axis to allow abutment.



- *Cascade Mirror*

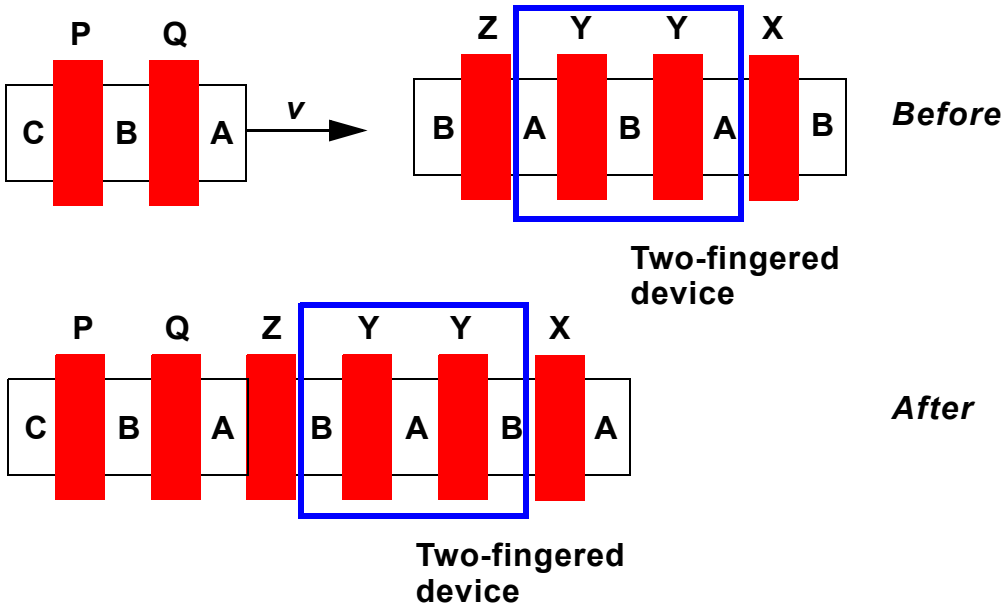
  Mirrors each individual device in the chain to allow abutment.



- *Cascade Mirror Permute* (2 nets)

© 2023 Cadence Design Systems, Inc.

Neither *Chain Mirror* nor *Cascade Mirror* alone can deliver a transformation that allows abutment. Instead, a mixture of *Cascade Mirror* and *Permute* is employed in order to achieve the desired result.



Related Topics

autoMirror

Connectivity Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

autoMirrorFilter

```
layoutXL autoMirrorFilter string "layer_name"
```

Description

Specifies that if a short violation is created by abutting a chain of devices, then Layout XL filters and ignores the violations on the supplied layer name when resolving the short.

The default is `""`, which means that no filtering is done.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "autoMirrorFilter")
envSetVal("layoutXL" "autoMirrorFilter" 'string "M0")
```

Related Topics

autoMirror

Generation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

autoPermuteNoAbutProp

```
layoutXL autoPermuteNoAbutProp boolean { t | nil }
```

Description

Prevents automatic permutation from running on pins that have a short violation but are missing abutment properties. The horizontal must-connect pin belongs to this category.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "autoPermuteNoAbutProp")
envSetVal("layoutXL" "autoPermuteNoAbutProp" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

autoSavePhysConfigUpdate

```
layoutXL autoSavePhysConfigUpdate boolean { t | nil }
```

Description

Automatically saves updates made to the physConfig view during Layout XL commands such as *Flatten* and *Create Physical Binding*.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "autoSavePhysConfigUpdate")
envSetVal("layoutXL" "autoSavePhysConfigUpdate" 'boolean nil)
```

Related Topics

Transparent Layout Hierarchy Generation

Shortcut menu for the Hierarchy Configuration Instances and Cells Tables

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

autoSpace

```
layoutXL autoSpace boolean { t | nil }
```

Description

Switches on automatic spacing, which allows components with the properties vxlInstSpacingDir and vxlInstSpacingRule to be spaced based on to the values specified in the properties.

The default is `t`.

GUI Equivalent

Command:	*Options – Connectivity* (*Generation* tab)
Field:	*Space components*

Examples

```
envGetVal("layoutXL" "autoSpace")
envSetVal("layoutXL" "autoSpace" 'boolean nil)
```

Additional Information

Automatic spacing is run only on pairs of instances that are not already abutted. This includes devices which overlap but which are not abutted (for example, instances where abutment has been attempted but has failed).

To establish whether an instance is abutted or not, type the following into the CIW.

```
inst~>groupMembers~>group~>name
```

This returns the names of the abutment groups to which the instance belongs. If the instance is not abutted, it returns nil.

If one of the devices is in any type of group (including a synchronous clone), the other device must be in the same group for automatic spacing to occur.

Related Topics

vxlInstSpacingDir Property

© 2023 Cadence Design Systems, Inc.

vxlInstSpacingRule Property

Generation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

autoZoomEnabled

```
ab autoZoomEnabled boolean { t | nil }
```

Description

Controls the *Auto Zoom* function of the Annotation Browser.

The default is `nil`.

Note: For more information on the Annotation browser *Auto Zoom*, see the section on the toolbar in Annotation Browser Graphical User Interface.

GUI Equivalent

Command:	*Windows – Assistants – Annotation Browser*
Field:	*Auto Zoom* (*Toolbar* button)

Examples

```
envGetVal("ab" "autoZoomEnabled")
envSetVal("ab" "autoZoomEnabled" 'boolean t)
```

Related Topics

Annotation Browser Toolbar

Annotation Browser Assistant

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

autoZoomFactor

`ab autoZoomFactor int` *`integer`*

Description

Controls the *Auto Zoom* scale of the Annotation Browser, which determines the level of zoom applied on the selected markers on a scale of 1 to 10.

- When *Auto Zoom* is set to *Fixed* mode, the `autoZoomFactor` environment variable fully controls the zoom level applied on the canvas.
- When *Auto Zoom* is set to *Minimal* mode, the Annotation Browser limits the change in zoom level on the canvas if the selected marker is visible on panning. Else, the zoom level is modified as in Fixed mode.

The default is `4`.

Note: For more information on the Annotation browser *Auto Zoom*, see the section on the toolbar in Annotation Browser Graphical User Interface.

GUI Equivalent

Command:	*Windows – Assistants – Annotation Browser*
Field:	*Auto Zoom* (*Toolbar* button)

Examples

```
envGetVal("ab" "autoZoomFactor")
envSetVal("ab" "autoZoomFactor" 'int 2)
envSetVal("ab" "autoZoomFactor" 'int 10)
```

Related Topics

Annotation Browser Toolbar

Annotation Browser Assistant

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

autoZoomIsFixed

```
ab autoZoomIsFixed boolean { t | nil }
```

Description

Controls whether the *Auto Zoom* function of the Annotation Browser defaults to *Fixed* mode or *Minimal* mode.

The default is `nil`, which means that the function is set in *Minimal* mode.

Note: For more information on the Annotation browser *Auto Zoom*, see the section on the toolbar in Annotation Browser Graphical User Interface.

GUI Equivalent

Command:	*Windows – Assistants – Annotation Browser* (*Auto Zoom* toolbar button)
Field:	*Fixed* and *Minimal* radio buttons (*Toolbar*)

Examples

```
envGetVal("ab" "autoZoomIsFixed")
envSetVal("ab" "autoZoomIsFixed" 'boolean t)
```

Related Topics

Annotation Browser Toolbar

Annotation Browser Assistant

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

backAnnotateInstances

```
layoutXL backAnnotateInstances cyclic { "individually" | "mfactored" | "iterated" }
```

Description

Specifies how instances are backannotated to the schematic.

- `individually`: Each instance is backannotated individually. This is the default.
- `mfactored`: Instances are backannotated as mfactored devices in the schematic.
- `iterated`: Instances are backannotated the same way as for mfactored, but some iterated notation is used. For example, `(<0:2>)`. Specify this value when the instance you want to backannotate is not a Pcell and when no mfactored parameter is defined.

Value set for `backAnnotateInstances` takes effect only when `lxDummyBackAnnotateMFactor` is set to `nil`.

This setting also apply to *Generate Selected from Layout* in the following solutions:

- Virtuoso RF (Virtuoso MultiTech Framework Only)
- Photonics (Virtuoso Photonics Option)

GUI Equivalent

Command:	*Connectivity – Backannotate – Dummy Instances – F3*
Field:	*Generate*

Command:	*Connectivity – Generate – Generate Selected From Layout – F3 (Virtuoso Photonics Option)*
Field:	*Generate*

Examples

```
envGetVal("layoutXL" "backAnnotateInstances")
envSetVal("layoutXL" "backAnnotateInstances" 'cyclic "mfactored")
envSetVal("layoutXL" "backAnnotateInstances" 'cyclic "iterated")
```


© 2023 Cadence Design Systems, Inc.

Related Topics

Back Annotate Dummies Form

lxDummyBackAnnotateMFactor

Generating Selected From Layout

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

backAnnotateIgnoreBoundMasters

```
layoutXL backAnnotateIgnoreBoundMasters boolean { t | nil }
```

Description

Specifies whether backannotation should use the library and cell of a layout device as the schematic master. Use this environment variable if you want a backannotated device to have a different symbol to that of the existing schematic devices bound to the instances of the specified layout library and cell.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "backAnnotateIgnoreBoundMasters")
envSetVal("layoutXL" "backAnnotateIgnoreBoundMasters" 'boolean t)
Schematic       Layout
I1 (lib/nmos_2/symbol) <-bound-> I1 (lib/nmos/layout)
                                 I2 (lib/nmos/layout)
```

If `backAnnotateIgnoreBoundMasters` is set to `nil`, `I2` will be backannotated as `/lib/nmos_2/symbol`.

If `backAnnotateIgnoreBoundMasters` is set to `t`, `I2` will be backannotated as `/lib/nmos/symbol`.

Related Topics

Back Annotate Dummies Form

lxDummyBackAnnotateMFactor

Generating Selected From Layout

© 2023 Cadence Design Systems, Inc.

backAnnotatePackingMethod

```
layoutXL backAnnotatePackingMethod cyclic { "fillRowsFirst" | "fillColsFirst" }
```

Description

Switches between column-priority packing and row-priority packing for the grid of instances being backannotated.

The default is `fillRowsFirst`.

GUI Equivalent

Command: *Connectivity – Backannotate – Dummy Instances – F3*

Connectivity – Generate – Selected From Layout – F3

Field:



Examples

```
envGetVal("layoutXL" "backAnnotatePackingMethod")
envSetVal("layoutXL" "backAnnotatePackingMethod" 'cyclic "fillColsFirst")
```

Related Topics

Back Annotate Dummies Form

Generate Selected From Layout Form

Generating Selected From Layout

© 2023 Cadence Design Systems, Inc.

backAnnotateXSpacing

```
layoutXL backAnnotateXSpacing float instanceSpacing
```

Description

Specifies the horizontal spacing between instances.

The default is `0.625`.

GUI Equivalent

Command:	*Connectivity – Backannotate – Dummy Instances – F3*
	Connectivity – Generate – Selected From Layout – F3
Field:	*X, Y Spacing*

Examples

```
envGetVal("layoutXL" "backAnnotateXSpacing")
envSetVal("layoutXL" "backAnnotateXSpacing" 'float 0.1)
```

Related Topics

Back Annotate Dummies Form

Generating Selected From Layout

Generate Selected From Layout Form

© 2023 Cadence Design Systems, Inc.

backAnnotateYSpacing

```
layoutXL backAnnotateYSpacing float instanceSpacing
```

Description

Specifies the vertical spacing between instances.

The default is `0.625`.

GUI Equivalent

Command:	*Connectivity – Backannotate – Dummy Instances – F3*
	Connectivity – Generate – Selected From Layout – F3
Field:	*X, Y Spacing*

Examples

```
envGetVal("layoutXL" "backAnnotateYSpacing")
envSetVal("layoutXL" "backAnnotateYSpacing" 'float 0.1)
```

Related Topics

Back Annotate Dummies Form

Generating Selected From Layout

Generate Selected From Layout Form

© 2023 Cadence Design Systems, Inc.

bindAllCellsInDesign

```
layoutXL bindAllCellsInDesign boolean { t | nil }
```

Description

Binds all nonleaf cellviews at current level by using their corresponding connectivity reference.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*All cells in design* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "bindAllCellsInDesign")
envSetVal("layoutXL" "bindAllCellsInDesign" 'boolean t)
```

Related Topics

Update Binding Form

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindClearConn

```
layoutXL bindClearConn boolean { t | nil }
```

Description

Deletes logical connectivity, when set to `t`.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*Clear* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "bindClearConn")
envSetVal("layoutXL" "bindClearConn" 'boolean t)
```

Related Topics

Update Binding Form

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindCorrectMaster

```
layoutXL bindCorrectMaster boolean { t | nil }
```

Description

Restricts initial name and connectivity-driven binding to bind schematic instances with those layout instances that have the master as defined using the Configure Physical Hierarchy Window.

The default is `nil`, which means if the correct master is not found, instances with most matching terminal names and matching connectivity are bound.



Cadence recommends that the environment variable should be changed from its default value by expert users or CAD team members only.

Note: The default value of the environment variable cannot be controlled using the *Correct Master* field on the *Update Binding* form. For any update required to the default value, the change must be saved to the `.cdsinit` file.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindCorrectMaster")
envSetVal("layoutXL" "bindCorrectMaster" 'boolean t)
```

Related Topics

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindCreateTransparent

```
layoutXL bindCreateTransparent boolean { t | nil }
```

Description

Controls whether transparent instances are created during hierarchical binding.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindCreateTransparent")
envSetVal("layoutXL" "bindCreateTransparent" 'boolean nil)
```

Related Topics

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindCrossRefFile

`layoutXL bindCrossRefFile string "`*`fileName`*`"`

Description

Specifies the PVS instance cross-reference file to be used for creating the binding file.

The default is `""`.

Arguments

A PVS LVS `.ixf` cross-reference file created in the `svdb` directory of the PVS run directory by using the PVS *Create Quantus Input Data* option.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*Instance Cross Reference* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "bindCrossRefFile")
envSetVal("layoutXL" "bindCrossRefFile" 'string "instanceCrossRefFileName")
```

Related Topics

Update Binding Form

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindCurrentLevel

```
layoutXL bindCurrentLevel boolean { t | nil }
```

Description

Binds the schematic and layout devices at the current level of hierarchy.

The default is `t`.

Note: Binding at current layout is most suitable when your design is largely LVS clean or you have same-connectivity mosaics in your design. If your design reports mismatches after an LVS run or the design has different-connectivity mosaics, you must select hierarchical binding by setting the environment variable to `nil` so that the hierarchical mismatches, if exist, are resolved and different-connectivity mosaics are first flattened before being bound.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*Current Level* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "bindCurrentLevel")
envSetVal("layoutXL" "bindCurrentLevel" 'boolean nil)
```

Related Topics

bindCurrentLevel

Update Binding Form

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindExtract

```
layoutXL bindExtract boolean { t | nil }
```

Description

Extracts the design to improve the bindings.

The default is `t`.

Note: The extractor can be disabled only when binding at the current level or when binding based on logical connectivity.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*Extract to Level* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "bindExtract")
envSetVal("layoutXL" "bindExtract" 'boolean nil)
```

Related Topics

Update Binding Form

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindExtractedNetlistFile

```
layoutXL bindExtractedNetlistFile string "fileName"
```

Description

Specifies the extracted netlist file to be used for creating the binding file.

The default is `""`.

Arguments

A PVS LVS `.net` extracted layout CDL netlist file created in the `svdb` directory of the PVS run directory by using the PVS *Create Quantus Input Data* option.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*Extracted Netlist* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "bindExtractedNetlistFile")
envSetVal("layoutXL" "bindExtractedNetlistFile" 'string
"extractedNetlistFileName")
```

Related Topics

Update Binding Form

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindFactorWarnLimit

`layoutXL bindFactorWarnLimit int` *`generationFactor`*

Description

Controls if a message should get issued to report a schematic instance that has a high generation factor and, therefore, can delay the launch of Virtuoso Layout Suite XL.

The default is `5000`. This means the message is issued if Virtuoso Layout Suite XL encounters schematic instances that have a generation factor greater than or equal to `5000`. The message is not issued for schematic instances that have a generation factor less than `5000`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindFactorWarnLimit")
envSetVal("layoutXL" "bindFactorWarnLimit" 'int 10000)
```

Related Topics

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindFile

```
layoutXL bindFile string "fileName"
```

Description

Specifies the layout instance binding file.

The default is `""`.

GUI Equivalent

Command:	*Connectivity – Update – Binding – Read File (group box)*
Field:	*Binding* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "bindFile")
envSetVal("layoutXL" "bindFile" 'string "bindingFileName")
```

Related Topics

Update Binding Form

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindFlattenLayoutCreate

```
layoutXL bindFlattenLayoutCreate cyclic { "Synchronized Family" | "Free Objects" |
     "Grouped Objects" }
```

Description

Controls which one of the following is created when instances are flattened:

- `Synchronized Family`: Synchronized family for the top-level instances that are flattened and figure groups for the lower-level flattened instances
- `Free Objects`: No Grouping
- `Grouped Objects`: Figure groups at all hierarchical levels

The default is `"Synchronized Family"`.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*Flatten Layout Instances and Create* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "bindFlattenLayoutCreate")
envSetVal("layoutXL" "bindFlattenLayoutCreate" 'cyclic "Free Objects")
```

Related Topics

Update Binding Form

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindIgnoreBulk

```
layoutXL bindIgnoreBulk boolean { t | nil }
```

Description

Ignores any bulk terminals and their connectivity when running the *Update Binding* command.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindIgnoreBulk")
envSetVal("layoutXL" "bindIgnoreBulk" 'boolean t)
```

Related Topics

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindIgnoreDummies

```
layoutXL bindIgnoreDummies boolean { t | nil }
```

Description

Ignores any unbound, potential dummy instances that are identified after an Update Binding run.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*Ignore Dummies* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "bindIgnoreDummies")
envSetVal("layoutXL" "bindIgnoreDummies" 'boolean t)
```

Related Topics

Binding Updates

Update Binding Form

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindIgnoreDummiesFullyConn

```
layoutXL bindIgnoreDummiesFullyConn boolean { t | nil }
```

Description

Ignores any unbound, potential dummy instances that are identified as fully connected by all power nets, ground nets, or by a single net of any type after an Update Binding Form run.

The default is `nil`, which means that the dummy instances are additionally identified if:

- The devices belong to one of the following component types: NMOS, PMOS, NFIN, PFIN, RES, CAP, or IND
- The devices are connected by one or more power nets or one or more ground nets
- The other terminals are unconnected

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindIgnoreDummiesFullyConn")
envSetVal("layoutXL" "bindIgnoreDummiesFullyConn" 'boolean t)
```

Related Topics

Update Binding Form

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindIgnoreRouteCells

```
layoutXL bindIgnoreRouteCells boolean { t | nil }
```

Description

Ignores any unbound, potential dummy instances that are identified after an Update Binding run.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*Ignore Route Cells* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "bindIgnoreRouteCells")
envSetVal("layoutXL" "bindIgnoreRouteCells" 'boolean t)
```

Related Topics

Binding Updates

Update Binding Form

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindIncr

```
layoutXL bindIncr boolean { t | nil }
```

Description

Switches on the incremental binder.

The default is `t`.

When set to `nil`, the incremental binder is switched off. This means if the layout is edited, any unbound layout instances or pins will remain unbound within the Layout XL session.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindIncr")
envSetVal("layoutXL" "bindIncr" 'boolean nil)
```

Related Topics

Incremental Binding

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindIncrAddIgnore

```
layoutXL bindIncrAddIgnore boolean { t | nil }
```

Description

Adds an `ignore` property on unbound instances during incremental binding to prevent the instances from getting bound. For the instances to be made available for binding, the `ignore` property must first be manually removed by using the *Remove Ignore* command, which is available in the shortcut menu of the instances that have the property set.

Note: Any existing bound instances will continue to get updated, if required, during incremental binding.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindIncrAddIgnore")
envSetVal("layoutXL" "bindIncrAddIgnore" 'boolean t)
```

Related Topics

Incremental Binding

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindIncrConn

```
layoutXL bindIncrConn boolean { t | nil }
```

Description

The incremental binder binds only when an exact connectivity match is found, which means the devices to be bound have the same master and their connectivity is unique.

The default is `t`.

When set to `nil`, the binder binds incrementally by name.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindIncrConn")
envSetVal("layoutXL" "bindIncrConn" 'boolean nil)
```

Related Topics

Incremental Binding

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindIncrMessage

```
layoutXL bindIncrMessage boolean { t | nil }
```

Description

Issues a message to report that instances, nets, or terminals were bound during incremental binding. If no new bindings are created during incremental binding, no message is issued,

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindIncrMessage")
envSetVal("layoutXL" "bindIncrMessage" 'boolean nil)
```

Related Topics

Incremental Binding

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindIncrPropagateConn

```
layoutXL bindIncrPropagateConn boolean { t | nil }
```

Description

Controls the propagation of logical connectivity from the schematic to the layout instance during incremental binding.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindIncrPropagateConn")
envSetVal("layoutXL" "bindIncrPropagateConn" 'boolean nil)
```

Related Topics

Incremental Binding

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindInitConn

```
layoutXL bindInitConn boolean { t | nil }
```

Description

Runs the connectivity-driven binder to bind the layout to the schematic when the design is opened in Layout XL.

The default is `nil`. This means the bindings from the last time the design was saved in Layout XL are restored. For any new bindings to be formed during initialization, the devices are bound only by name.

When set to `t`, in addition to binding by name, the binder binds instances based on logical and physical connectivity.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindInitConn")
envSetVal("layoutXL" "bindInitConn" 'boolean t)
```

Related Topics

Initial Binding

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindInitInstTerms

```
layoutXL bindInitInstTerms boolean { t | nil }
```

Description

Performs faster binder initialization and consumes much less memory compared to the regular binder initialization.

The default is `t`.

When set to `nil`:

- The binder initialization gives the same binding as regular binding, if all the bound local nets have matching names.
- If a layout is saved with bound local nets that have mismatching names, the nets are unbound at the next launch of Layout XL and the instances they connect reflect connectivity differences.

Note: To resolve the unbound instances, you can run Update Binding with *Preserve – Existing bindings* option selected or set the `bindInitInstTerms` environment variable to `t` and relaunch Layout XL.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindInitInstTerms")
envSetVal("layoutXL" "bindInitInstTerms" 'boolean nil)
```

Related Topics

Initial Binding

Update Binding Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindLayoutStop

`layoutXL bindLayoutStop string "(lib cell view)"`

Description

Specifies the list of layout library cellviews that the binder should consider as leaf-level instances to which to bind.

Once a specified layout library cellview is reached, the binder stops looking beyond for any devices to bind.

The default is `""`.

Note: When specifying the format for the library cellview, (`lib cell view`), the `lib`, `cell`, and `view` names can contain the `'\\*'` wildcard.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindLayoutStop")
envSetVal("layoutXL" "bindLayoutStop" 'string "(XLME INV layout\\*)"
envSetVal("layoutXL" "bindLayoutStop" 'string "(lib1 \\* \\*)")
envSetVal("layoutXL" "bindLayoutStop" 'string "(lib2 inv1 layout) (lib3 nand
layout)")
```

Related Topics

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindNameOnly

```
layoutXL bindNameOnly boolean { t | nil }
```

Description

Binds schematic and layout instances if they have matching names.

The default is `nil`, which means instances are bound based on connectivity.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*Name only* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "bindNameOnly")
envSetVal("layoutXL" "bindNameOnly" 'boolean t)
```

Related Topics

Update Binding Form

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindNewSplitParam

```
layoutXL bindNewSplitParam boolean { t | nil }
```

Description

Controls whether *Update – Components And Nets* generates missing split-fingered devices, when set to `t`.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindNewSplitParam")
envSetVal("layoutXL" "bindNewSplitParam" 'boolean t)
```

Related Topics

bindStrict

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindPermuteRules

```
layoutXL bindPermuteRules string "permuteRule"
```

Description

Provides an alternative method for defining a permute rule for cells that are symmetric and have terminals that can be permuted. The environment variable can be useful for defining permute rules in situations when the Edit CDF Simulation Information section of auLvs is not editable for some reason. The permute rules set using the `bindPermuteRules` environment variable can only be used for binding, *Update Binding*, and for checking Layout XL compliance.

The default is `""`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindPermuteRules")
envSetVal("layoutXL" "bindPermuteRules" 'string "(lib1 cell1 (p D S))(lib2 cell2
(p MINUS PLUS))")
```

Additional Information

The `bindPermuteRules` environment variable does not support setting permute rules for device terminals such as for abutment. For setting permute rules on device terminals, you must set the CDF `permuteRule`. Below is an example that illustrates how CDF `permuteRule` can be added to the `.cdsinit` file.

```
procedure( testSetCellCDFPermuteRule(lib cell permuteRule)
    let( (cellId cdf)
        if( lib  && cell && permuteRule then
         cellId = ddGetObj(lib cell)
         cdf    = cdfGetBaseCellCDF(cellId)
         cdf->simInfo->auLvs->permuteRule = permuteRule
         t
        )
    )
)
```


© 2023 Cadence Design Systems, Inc.

```
     testSetCellCDFPermuteRule("lib" "cell1" "(p X Y)")
     testSetCellCDFPermuteRule("lib" "cell2" "(p D S)")
…
```

Related Topics

Update Binding Flow

Design Check for Layout XL Compliance

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindPreserveExistingBindings

```
layoutXL bindPreserveExistingBindings boolean { t | nil }
```

Description

Preserves all existing bindings between the schematic and the layout devices.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*Existing bindings* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "bindPreserveExistingBindings")
envSetVal("layoutXL" "bindPreserveExistingBindings" 'boolean t)
```

Related Topics

Update Binding Form

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

List of Layout XL Environment Variables

© 2023 Cadence Design Systems, Inc.

bindPreserveLayoutHierarchy

```
layoutXL bindPreserveLayoutHierarchy boolean { t | nil }
```

Description

Creates complex bindings for hierarchy mismatches.

When set to `nil`, flattens the layout instances to match the source hierarchy.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*Flatten layout instances – Create – Complex Bindings* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "bindPreserveLayoutHierarchy")
envSetVal("layoutXL" "bindPreserveLayoutHierarchy" 'boolean nil)
```

Related Topics

Update Binding Form

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindPreserveUserBindings

```
layoutXL bindPreserveUserBindings boolean { t | nil }
```

Description

Preserves user bindings.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*User-defined bindings* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "bindPreserveUserBindings")
envSetVal("layoutXL" "bindPreserveUserBindings" 'boolean nil)
```

Related Topics

Update Binding Form

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindPVSRulesFile

```
layoutXL bindPVSRulesFile string "fileName"
```

The default is `""`.

Description

Specifies the PVS rule deck file to be used for determining the input-scale for running PVS.

Arguments

A PVS rule deck file.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*PVS Input Scale* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "bindPVSRulesFile")
envSetVal("layoutXL" "bindPVSRulesFile" 'string "PVSRuleFile")
```

Related Topics

Update Binding Form

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindPVSScale

`layoutXL bindPVSScale float` *`inputScale`*

Description

Specifies the floating point, *`input_scale`* for running PVS.

The default is `0`, which means `dbuPerUu` for the layout is used. But, if the PVS input scale is different from that of the layout, and the PVS rule file is specified, the PVS rule deck is parsed to determine the appropriate value. Alternatively, you can specify the PVS input scale to be used.

Arguments

`floating_point_number`

A floating point number that specifies the value to be used as the input scale for running PVS.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*PVS Input Scale* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "bindPVSScale")
envSetVal("layoutXL" "bindPVSScale" 'float 987.5)
```

Related Topics

Update Binding Form

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindRemoveIgnores

```
layoutXL bindRemoveIgnores boolean { t | nil }
```

Description

Removes the ignore property from the layout instances that can be bound to the schematic instances.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindRemoveIgnores")
envSetVal("layoutXL" "bindRemoveIgnores" 'boolean t)
```

Related Topics

Dummy Instance Generation

ignore Property

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindRemoveUnboundTerminalsNoPins

```
layoutXL bindRemoveUnboundTerminalsNoPins boolean { t | nil }
```

Description

Removes unnecessary unbound layout terminals with no pins and issues a message listing the deleted terminals.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindRemoveUnboundTerminalsNoPins")
envSetVal("layoutXL" "bindRemoveUnboundTerminalsNoPins" 'boolean nil)
```

Related Topics

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindReportCellsNoTerms

```
layoutXL bindReportCellsNoTerms boolean { t | nil }
```

Description

Controls whether the `lxUpdateBinding` SKILL function (or the *Connectivity – Update – Binding* command) should issue information messages to report cellviews with no terminals.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*Report – Cellviews with no terminals* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "bindReportCellsNoTerms")
envSetVal("layoutXL" "bindReportCellsNoTerms" 'boolean nil)
```

Related Topics

lxUpdateBinding

Update Binding Form

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindReportDummies

```
layoutXL bindReportDummies boolean { t | nil }
```

Description

Controls whether the `lxUpdateBinding` SKILL function (or the *Connectivity – Update – Binding* command) should issue information messages to report instances that have been identified as possible dummies.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*Report – Dummies* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "bindReportDummies")
envSetVal("layoutXL" "bindReportDummies" 'boolean nil)
```

Related Topics

lxUpdateBinding

Update Binding Form

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindReportHierarchy

```
layoutXL bindReportHierarchy boolean { t | nil }
```

Description

Controls whether the `lxUpdateBinding` SKILL function (or the *Connectivity – Update – Binding* command) should issue information messages to report instances that the hierarchical binder is descending inside.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*Report – Hierarchy binding* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "bindReportHierarchy")
envSetVal("layoutXL" "bindReportHierarchy" 'boolean nil)
```

Related Topics

lxUpdateBinding

Update Binding Form

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindReportReduction

```
layoutXL bindReportReduction boolean { t | nil }
```

Description

Reports the schematic and layout design statistics after parallel and serial reduction is performed.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindReportReduction")
envSetVal("layoutXL" "bindReportReduction" 'boolean t)
```

Related Topics

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindReportRouteCells

```
layoutXL bindReportRouteCells boolean { t | nil }
```

Description

Controls whether the `lxUpdateBinding` SKILL function (or the *Connectivity – Update – Binding* command) should issue information messages to report layout instances that have an `ignore` property set.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*Report – Route cells* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "bindReportRouteCells")
envSetVal("layoutXL" "bindReportRouteCells" 'boolean nil)
```

Related Topics

lxUpdateBinding

Update Binding Form

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindReportTerminalMismatch

```
layoutXL bindReportTerminalMismatch boolean { t | nil }
```

Description

Controls whether the `lxUpdateBinding` SKILL function (or the *Connectivity – Update – Binding* command) should issue information messages to report cells with terminals that do not match.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindReportTerminalMismatch")
envSetVal("layoutXL" "bindReportTerminalMismatch" 'boolean nil)
```

Related Topics

lxUpdateBinding

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindReportUnconnInstTerms

```
layoutXL bindReportUnconnInstTerms boolean { t | nil }
```

Description

Controls whether the `lxUpdateBinding` SKILL function (or the *Connectivity – Update – Binding* command) should issue information messages to report instance terminals that are unconnected.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindReportUnconnInstTerms")
envSetVal("layoutXL" "bindReportUnconnInstTerms" 'boolean nil)
```

Related Topics

lxUpdateBinding

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindSourceSkip

`layoutXL bindSourceSkip string "`*`lib cell view`*`"`

Description

Specifies the list of schematic library cellviews to ignore during a hierarchical Update Binding run.

The default is `""`.

Note: When specifying the format for the library cellview, (`lib cell view`), the `lib`, `cell`, and `view` names can contain the `'\\*'` wildcard.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindSourceSkip")
envSetVal("layoutXL" "bindSourceSkip" 'string "(lib1 inv1 sch\\*)"
envSetVal("layoutXL" "bindSourceSkip" 'string "(lib1 \\* \\*)")
envSetVal("layoutXL" "bindSourceSkip" 'string "(lib2 inv1 layout) (lib3 nand
layout)")
```

Related Topics

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindSourceStop

`layoutXL bindSourceStop string "`*`lib cell view`*`"`

Description

Specifies the list of schematic library cellviews that the binder should consider as leaf-level instances to which to bind.

Once a specified schematic library cellview is reached, the binder stops looking beyond for any devices to bind.

The default is `""`.

Note: When specifying the format for the library cellview, (`lib cell view`), the `lib`, `cell`, and `view` names can contain the `'\\*'` wildcard.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindSourceStop")
envSetVal("layoutXL" "bindSourceStop" 'string "(XLME INV sch\\*)"
envSetVal("layoutXL" "bindSourceStop" 'string "(lib1 \\* \\*)")
envSetVal("layoutXL" "bindSourceStop" 'string "(lib2 inv1 layout) (lib3 nand
layout)")
```

Related Topics

Update Binding Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindStrict

```
layoutXL bindStrict boolean { t | nil }
```

Description

Leaves the layout instances unbound if the mfactor, sfactor, or lxCombination value set on the corresponding schematic instances is reduced.

The default is `nil`, which means the layout instances will remain bound if the schematic mfactor, sfactor, or lxCombination is reduced.

When set to `t`, folding is allowed. But, any extra folded instances will be unbound if Update Binding is run, unless the folded instances are user bound.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bindStrict")
envSetVal("layoutXL" "bindStrict" 'boolean t)
```

Related Topics

Update Binding Flow

Incremental Binding

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindUpdateName

```
layoutXL bindUpdateName boolean { t | nil }
```

Description

Updates the name of the layout instance to match the name of the schematic instance it is bound to.

The default is `t`, which means, by default the name of the layout instance is updated after binding to match the name of the schematic instance.

When set to nil, the layout instance does not change its name after binding to match that of the corresponding schematic instance.

GUI Equivalent

Command:	*Connectivity – Define Device Correspondence*
Field:	*Update Layout Instance – Name* (Define Device Correspondence Form)

Examples

```
envGetVal("layoutXL" "bindUpdateName")
envSetVal("layoutXL" "bindUpdateName" 'boolean nil)
```

Related Topics

Define Device Correspondence Form

Device Correspondence Updates

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindUpdateTermName

```
layoutXL bindUpdateTermName boolean { t | nil }
```

Description

Updates the name of the terminal when binding through Define Device Correspondence form or by using the *Bind* option on the right mouse button shortcut menu.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Define Device Correspondence*
Field:	*Update Layout Terminal – Name* (Define Device Correspondence Form)

Examples

```
envGetVal("layoutXL" "bindUpdateTermName")
```

```
envSetVal("layoutXL" "bindUpdateTermName" 'boolean nil)
```

Related Topics

Define Device Correspondence Form

Device Correspondence Updates

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bindUseFingerWidth

```
layoutXL bindUseFingerWidth boolean { t | nil }
```

Description

Specifies if the parameter listed as *Width* for the selected component type needs to be considered as *Width* or the *Finger width*.

The default is `nil`, which means the specified parameter is considered as *Width*.

GUI Equivalent

Command:	*Configure Physical Hierarchy – Component Types Mode (Attributes Table)*
Field:	*Width parameter* (Component Types Attributes)

Examples

```
envGetVal("layoutXL" "bindUseFingerWidth")
envSetVal("layoutXL" "bindUseFingerWidth" 'boolean t)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

bndRemoveDeviceConnectivity

```
layoutXL bndRemoveDeviceConnectivity cyclic { "singleNetNoShortedTerminals" |
     "singleNetAllowShortedTerminals" | "equivalentNets" }
```

Description

Enables creating equivalent nets for remove devices when generating a layout or when updating existing bindings.

The default is `singleNetNoShortedTerminals`.

Arguments

`singleNetNoShortedTerminals`

Single net is created for remove device unless the device is connected to two terminals. In that case, two nets are created.

`singleNetAllowShortedTerminals`

Always creates a single net for remove device. Therefore, two terminals can be shorted.

`equivalentNets`

Two nets are created for remove device and they are set as equivalent nets.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "bndRemoveDeviceConnectivity")
envSetVal("layoutXL" "bndRemoveDeviceConnectivity" 'cyclic "equivalentNets")
```

Related Topics

Device Correspondence Updates

List of Layout XL Environment Variables

© 2023 Cadence Design Systems, Inc.

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

capacitanceParamNames

```
layoutXL capacitanceParamNames string "list_of_parameters"
```

Description

Lists the names of the schematic parameters used to specify capacitance.

The default is `"c C"`.

The capacitance value must be split among the generated series-connected devices during the following commands.

- *Generate All From Source*
- *Check Against Source*
- *Update Layout Parameters*.

Layout XL checks each schematic instance for one of the listed parameter names and updates the capacitance values of the matching layout parameters accordingly.

Arguments

list_of_parameters

A list of parameter names each separated by a space. The list must be enclosed in quotation marks; for example,

`"c C"`

GUI Equivalent

Command:	*Options – Connectivity* (Parameters)
Field:	*Capacitance*

Examples

```
envGetVal("layoutXL" "capacitanceParamNames")
envSetVal("layoutXL" "capacitanceParamNames" 'string "c C")
```


© 2023 Cadence Design Systems, Inc.

Related Topics

Generating All Components from Source

Design Check Against Source

Layout Parameters Updates

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

casAppendToLogFile

```
layoutXL casAppendToLogFile boolean { t | nil }
```

Description

Appends the report from a new CAS run to the specified log file.

The default is `nil`, which means the existing log file is cleared of all the information from the previous CAS run and populated with the information from the new run.

GUI Equivalent

Command:	*Connectivity – Check – Against Source* (Check Against Source Form)
Field:	*Overwrite log file*

Examples

```
envGetVal("layoutXL" "casAppendToLogFile")
envSetVal("layoutXL" "casAppendToLogFile" 'boolean t)
envSetVal("layoutXL" "casAppendToLogFile" 'boolean nil)
```

Related Topics

CAS TAB Marker Categories

Design Check Against Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

casDisplayBusTerminalsMismatch

```
layoutXL casDisplayBusTerminalsMismatch boolean { t | nil }
```

Description

Reports mismatches between the implicit bus terminals that exist in the layout cellview and the explicit bus terminals that exist in the schematic cellview.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Check – Against Source* (Check Against Source Form)
Field:	*Bus terminals*

Examples

```
envGetVal("layoutXL" "casDisplayBusTerminalsMismatch")
envSetVal("layoutXL" "casDisplayBusTerminalsMismatch" 'boolean t)
envSetVal("layoutXL" "casDisplayBusTerminalsMismatch" 'boolean nil)
```

Related Topics

CAS TAB Marker Categories

Design Check Against Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

casDisplayConnMismatch

```
layoutXL casDisplayConnMismatch boolean { t | nil }
```

Description

Reports connectivity issues on top-level pins and global nets, mismatched or missing terminals and instance terminals, and unbound nets in the layout.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Check – Against Source* (Check Against Source Form)
Field:	*Connectivity*

Examples

```
envGetVal("layoutXL" "casDisplayConnMismatch")
envSetVal("layoutXL" "casDisplayConnMismatch" 'boolean t)
envSetVal("layoutXL" "casDisplayConnMismatch" 'boolean nil)
```

Related Topics

CAS TAB Marker Categories

Design Check Against Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

casDisplayDummyMismatch

```
layoutXL casDisplayDummyMismatch boolean { t | nil }
```

Description

Reports dummy instances that are unbound, which means they have no correspondence to the schematic devices and have the `ignore` property set.

The default is `nil`. When set to `t`, reports dummies instances that are good candidates for backannotation.

GUI Equivalent

Command:	*Connectivity – Check – Against Source*
Field:	*Dummy devices*

Examples

```
envGetVal("layoutXL" "casDisplayDummyMismatch")
envSetVal("layoutXL" "casDisplayDummyMismatch" 'boolean t)
envSetVal("layoutXL" "casDisplayDummyMismatch" 'boolean nil)
```

Related Topics

Check Against Source Form

CAS TAB Marker Categories

Design Check Against Source

ignore Property

© 2023 Cadence Design Systems, Inc.

casDisplayMastersMismatch

```
layoutXL casDisplayMastersMismatch boolean { t | nil }
```

Description

Reports the instance masters that are missing or mismatched between the layout and the schematic.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Check – Against Source* (Check Against Source Form)
Field:	*Masters*

Examples

```
envGetVal("layoutXL" "casDisplayMastersMismatch")
envSetVal("layoutXL" "casDisplayMastersMismatch" 'boolean t)
envSetVal("layoutXL" "casDisplayMastersMismatch" 'boolean nil)
```

Related Topics

CAS TAB Marker Categories

Design Check Against Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

casDisplayNameMismatch

```
layoutXL casDisplayNameMismatch boolean { t | nil }
```

Description

Reports mismatches between the schematic and the layout instance and net names.

The default is `nil`, which means name mismatches are not reported, and no markers are generated to highlight the mismatches.

GUI Equivalent

Command:	*Connectivity – Check – Against Source* (Check Against Source Form)
Field:	*Names*

Examples

```
envGetVal("layoutXL" "casDisplayNameMismatch")
envSetVal("layoutXL" "casDisplayNameMismatch" 'boolean t)
envSetVal("layoutXL" "casDisplayNameMismatch" 'boolean nil)
```

Related Topics

CAS TAB Marker Categories

Design Check Against Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

casDisplayParamsMismatch

```
layoutXL casDisplayParamsMismatch boolean { t | nil }
```

Description

Reports mismatches between CDF parameters (and properties) set on the schematic and the layout views. The parameters (and properties) to be compared are set on the Parameters of the Connectivity form.

The default is `t`. The update binding command and SKILL functions `lxCheck`, `lxHierCheck`, and `lxCheckLib` calculate parameter differences only when `casDisplayParamsMismatch` is set to `t`.

GUI Equivalent

Command:	*Connectivity – Check – Against Source* (Check Against Source Form)
Field:	*Parameters*

Examples

```
envGetVal("layoutXL" "casDisplayParamsMismatch")
envSetVal("layoutXL" "casDisplayParamsMismatch" 'boolean t)
envSetVal("layoutXL" "casDisplayParamsMismatch" 'boolean nil)
```

Related Topics

CAS TAB Marker Categories

Design Check Against Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

casDisplayUnboundInsts

```
layoutXL casDisplayUnboundInsts boolean { t | nil }
```

Description

Generates markers in the Annotation Browser CAS TAB marker categories to report unbound instances found during a layout versus schematic check performed by running the *Check Against Source* command.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Check – Against Source* (Check Against Source Form)
Field:	*Unbound instances*

Examples

```
envGetVal("layoutXL" "casDisplayUnboundInsts")
envSetVal("layoutXL" "casDisplayUnboundInsts" 'boolean t)
envSetVal("layoutXL" "casDisplayUnboundInsts" 'boolean nil)
```

Related Topics

CAS TAB Marker Categories

Design Check Against Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

casDisplayedDiffsLimit

`layoutXL casDisplayedDiffsLimit int` *`integer`*

Description

Specifies the maximum number of schematic versus layout differences to be reported by a Check Against Source (CAS) run.

The default is `300`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "casDisplayedDiffsLimit")
envSetVal("layoutXL" "casDisplayedDiffsLimit" 'int 250)
```

Related Topics

CAS TAB Marker Categories

Design Check Against Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

casGlyphs

```
layoutXL casGlyphs boolean { t | nil }
```

Description

Controls the display of on-canvas glyphs for *Check Against Source* issues.
The default is `t`, which means that on-canvas glyphs are visible in the canvas.
When set to `nil`, on-canvas glyphs are not visible.

GUI Equivalent

Command:	*Connectivity – Check Against Source*
Field:	*Display*

Examples

```
envGetVal("layoutXL" "casGlyphs")
envSetVal("layoutXL" "casGlyphs" 'boolean nil)
```

Related Topics

Check Against Source Form

CAS TAB Marker Categories

Design Check Against Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

casInfoBalloonOnHover

```
layoutXL casInfoBalloonOnHover boolean { t | nil }
```

Description

Controls the display of info balloons for on-canvas glyphs.

The default is `nil`, which means that an info balloon is displayed when you left click a *Check Against Source* glyph.

When set to `t`, the info balloon is displayed when you hover your mouse pointer over the glyph.

GUI Equivalent

Command:	*Connectivity – Check Against Source*
Field:	*Info balloon*

Examples

```
envGetVal("layoutXL" "casInfoBalloonOnHover")
envSetVal("layoutXL" "casInfoBalloonOnHover" 'boolean t)
```

Related Topics

Check Against Source Form

CAS TAB Marker Categories

Design Check Against Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

casInfoWindow

```
layoutXL casInfoWindow boolean { t | nil }
```

Description

Prints the *Check Against Source* report in a separate Info window.

The default is `nil`, which means the schematic versus layout differences are reported in the form of markers generated in the CAS tab of the Annotation Browser assistant.

GUI Equivalent

Command:	*Connectivity – Check – Against Source* (Check Against Source Form)
Field:	*Display info window*

Examples

```
envGetVal("layoutXL" "casInfoWindow")
envSetVal("layoutXL" "casInfoWindow" 'boolean t)
envSetVal("layoutXL" "casInfoWindow" 'boolean nil)
```

Related Topics

CAS TAB Marker Categories

Design Check Against Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

casLogFileName

```
layoutXL casLogFileName string "logFileName"
```

Description

Specifies the name of the log file to which all the *Check Against Source* messages are written.

The default is `casLog`, which means the schematic versus layout differences are reported in the form of markers generated in the CAS tab of the Annotation Browser assistant.

GUI Equivalent

Command:	*Connectivity – Check – Against Source* (Check Against Source Form)
Field:	*Log File*

Examples

```
envGetVal("layoutXL" "casLogFileName")
envSetVal("layoutXL" "casLogFileName" 'string "casLog1")
```

Related Topics

CAS TAB Marker Categories

Design Check Against Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

casOpenWorkspace

```
layoutXL casOpenWorkspace boolean [ t| nil }
```

Description

Opens the Setting Environment Variables workspace after a *Check Against Source* run.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Check – Against Source*
Field:	*Open workspace*

Examples

```
envGetVal("layoutXL" "casOpenWorkspace")
envSetVal("layoutXL" "casOpenWorkspace" 'boolean t)
envSetVal("layoutXL" "casOpenWorkspace" 'boolean nil)
```

Related Topics

Check Against Source Form

CAS TAB Marker Categories

Design Check Against Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

casMarkers

```
layoutXL casMarkers boolean { t | nil }
```

Description

Opens the CAS workspace with the Annotation Browser CAS TAB marker categories displayed, which reports the markers from a *Check Against Source* run.

The default is `t`.

When set to `nil`, the mismatches identified between the schematic and the layout during a *Check Against Source* run are displayed in an information window.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "casMarkers")
envSetVal("layoutXL" "casMarkers" 'boolean t)
envSetVal("layoutXL" "casMarkers" 'boolean nil)
```

Related Topics

Workspaces in Layout XL

incrementalCas

CAS TAB Marker Categories

Design Check Against Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

casMarkerTypeLimit

`layoutXL casMarkerTypeLimit int` *`integer`*

Description

Limits the number of markers created for each type of difference reported after a *Check Against Source* (CAS) run.

For example, if the CAS marker type limit is set to `5000`, and the Check Against Source run identifies parameter differences between the schematic and the layout, not more than `5000` parameter differences are reported in the CAS tab.

The default is `5000`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "casMarkerTypeLimit")
envSetVal("layoutXL" "casMarkerTypeLimit" 'int 5000)
envSetVal("layoutXL" "casMarkerTypeLimit" 'int 2500)
```

Related Topics

Design Check Against Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

casSelected

```
layoutXL casSelected boolean { t | nil }
```

Description

Runs the *Check Against Source* command only on the selected instances.

The default is `nil`, which means the command runs on all the instances.

GUI Equivalent

Command:	*Connectivity – Check – Against Source* (Check Against Source Form)
Field:	*Selected only*

Examples

```
envGetVal("layoutXL" "casSelected")
envSetVal("layoutXL" "casSelected" 'boolean t)
envSetVal("layoutXL" "casSelected" 'boolean nil)
```

Related Topics

Design Check Against Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

casUngenRemoveSplitParamDiff

```
layoutXL casUngenRemoveSplitParamDiff boolean { t | nil }
```

Description

Creates split parameter diffs if source instance is expanded and has ungenerated instances.

The default is `nil`, which means that split parameter diffs are not created.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "casUngenRemoveSplitParamDiff")
envSetVal("layoutXL" "casUngenRemoveSplitParamDiff" 'boolean t)
```

Related Topics

Application Readiness Checker Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

casVisibleUnboundMarkers

```
layoutXL casVisibleUnboundMarkers boolean { t | nil }
```

Description

Displays unbound markers on the canvas.

The default is `t`.

When set to `nil`, no markers are displayed.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "casVisibleUnboundMarkers")
envSetVal("layoutXL" "casVisibleUnboundMarkers" 'boolean t)
envSetVal("layoutXL" "casVisibleUnboundMarkers" 'boolean nil)
```

Related Topics

Design Check Against Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainAbutStrategy

```
layoutXL chainAbutStrategy string { "sdFirst" | "dummyFirst" }
```

Description

Abuts on Source/Drain using the default abutment option, `sdFirst`.

Abuts on Source/Drain first if the default abutment value, `sdFirst`, is set. Else, abuts first on dummy poly if the abutment value is set to `dummyFirst`. If dummy poly abutment fails, Source/Drain abutment, using `sdFirst`, is carried out.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "chainAbutStrategy")
envSetVal("layoutXL" "chainAbutStrategy" 'string "sdFirst")
envSetVal("layoutXL" "chainAbutStrategy" 'string "dummyFirst")
```

Related Topics

Device Abutment

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainAllowSingleBulk

```
layoutXL chainAllowSingleBulk boolean { t | nil }
```

Description

Specifies whether instances with a single bulk or with different bulk connections can be chained. When both the environment variables, chainUseDeviceOrder and `chainAllowSingleBulk`, are set to `t`, chaining is updated to abut devices with only a single bulk.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "chainAllowSingleBulk")
envSetVal("layoutXL" "chainAllowSingleBulk" 'boolean t)
envSetVal("layoutXL" "chainAllowSingleBulk" 'boolean nil)
```

Related Topics

Device Abutment

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainCellOrient

```
layoutXL chainCellOrient string "cell_orientation"
```

Description

Sets the cell orientation to the specified value if the environment variable is set to "R0" or "R90". Otherwise, the cell orientation is calculated by computing how the S, G, and D are aligned relative to each other.

The default value is "".

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "chainCellOrient")
envSetVal("layoutXL" "chainCellOrient" 'string "R0")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainDummyFlexBothEndNets

```
layoutXL chainDummyFlexBothEndNets boolean { t | nil }
```

Description

Ensures that dummy instances can change their nets on both sides during interactive chaining to support abutment on both sides.

When set to `nil`, dummy instances can change their net on only one side.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "chainDummyFlexBothEndNets")
envSetVal("layoutXL" "chainDummyFlexBothEndNets" 'boolean t)
envSetVal("layoutXL" "chainDummyFlexBothEndNets" 'boolean nil)
```

Related Topics

Device Abutment

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainDummyAbutment

```
layoutXL chainDummyAbutment boolean { t | nil }
```

Description

Controls if the post process to combine distinct chains into one chain by abutting on the dummy pins is performed. This post process applies if all devices support event 5.

The default is `t`. To prevent this post process, set environment variable to `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "chainDummyAbutment")
envSetVal("layoutXL" "chainDummyAbutment" 'boolean nil)
```

Related Topics

Abutment Event 5

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainDummyUpdate

```
layoutXL chainDummyUpdate boolean { t | nil }
```

Description

Controls whether dummy devices can be changed during chaining.

The default is `t`, which means the dummy devices can change during chaining. When set to `nil`, the dummy devices are not changed in any way during chaining and there are no mirror, permute, or connectivity updates.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "chainDummyUpdate")
```

```
envSetVal("layoutXL" "chainDummyUpdate" 'boolean nil)
```

Related Topics

chainDummyFlexBothEndNets

chainDummyAbutment

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainEffort

```
layoutXL chainEffort cyclic { "Minimal" | "Nominal" | "Maximal" }
```

Description

Controls the effort applied during chaining. Increased effort can improve the quality of results, but it can be time-consuming. When the value is set to `Nominal` or `Maximal`, the environment variable also tries to improve the order of the device names.

The default is `Nominal`.

GUI Equivalent

Command:	*Options – Connectivity – Generation – Device Chaining*
Field:	*Chain effort*

Examples

```
envGetVal("layoutXL" "chainEffort")
envSetVal("layoutXL" "chainEffort" 'cyclic "Minimal")
envSetVal("layoutXL" "chainEffort" 'cyclic "Maximal")
```

Related Topics

Connectivity Form

Generate Chained Devices Form

Chaining Transistors Automatically during Layout Generation

lxChain

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainExtendSelection

```
layoutXL chainExtendSelection boolean { t | nil }
```

Description

When only part of a chain is selected, extends the selection to include the whole chain, including any mfactored instances and legs of folded devices.

The default is `nil`, which means you can select individual devices, mfactors, or folded legs within a chain without selecting the entire chain.

GUI Equivalent

Command:	*Connectivity – Generate – Chained Devices*
Field:	*Extend Selection* (Generate Chained Devices Form)

Examples

```
envGetVal("layoutXL" "chainExtendSelection")
envSetVal("layoutXL" "chainExtendSelection" 'boolean t)
envSetVal("layoutXL" "chainExtendSelection" 'boolean nil)
```

Related Topics

Device Abutment

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainFolds

```
layoutXL chainFolds boolean { t | nil }
```

Description

Abuts the folds of newly folded devices into chains. If any folds already exist before the new folds are created, all the existing folds except the selected ones are deleted. The first fold from the retained set is then used to create the new folds.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Generate – Folded Devices*
Field:	*Chain Folds* (Generate Folded Devices Form)

Examples

```
envGetVal("layoutXL" "chainFolds")
envSetVal("layoutXL" "chainFolds" 'boolean t)
envSetVal("layoutXL" "chainFolds" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainInPlace

```
layoutXL chainInPlace boolean { t | nil }
```

Description

Creates the device chains on layout canvas at the same location as the left-most instance in a row region that is selected for chaining, if at least one of the selected instances is on a row region. If none of the selected instances are on a row region, the device chain is created at the same location as the left-most instance selected for chaining.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Generate – Chained Devices*
Field:	*Create In Place* (Generate Chained Devices Form)

Examples

```
envGetVal("layoutXL" "chainInPlace")
envSetVal("layoutXL" "chainInPlace" 'boolean t)
envSetVal("layoutXL" "chainInPlace" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainInPlaceDirection

```
layoutXL chainInPlaceDirection string { "Increasing" | "Decreasing" }
```

Description

Controls the direction in which the chain is created relative to the left-most instance in a row region that is selected for chaining, if at least one of the selected instances is on a row region. Else the chain is created at the same location as the left-most selected instance.

The default is `"Increasing"`, which means the chain is created to the right of the left-most instance in a row region or right of the left-most selected instance.

When set to "`Decreasing`", the chain the created to the left of the right-most instance in a row region, or left of the right-most selected instance.

If the original instance is at an orientation, such as `R90`, that can place the chain vertically, the environment variable can be used to create the new chain above or below the original instance.

GUI Equivalent

Command:	*Connectivity – Generate – Chained Devices*
Field:	*Create In Place* (Generate Chained Devices Form)

Examples

```
envGetVal("layoutXL" "chainInPlaceDirection")
envSetVal("layoutXL" "chainInPlaceDirection" 'string "Increasing")
envSetVal("layoutXL" "chainInPlaceDirection" 'string "Decreasing")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainLeftNet

```
layoutXL chainLeftNet string { "Source" | "Drain" | "Either" }
```

Description

Controls whether source or drain nets are optimized to the left of generated chains. The default is `"Source"`, which means a chain is optimized so that one of its source nets is on the left-hand side (where possible).

Note: `"Source"` and `"drain"` here refer to schematic source and drain nets and not layout source and drain nets, which may have been permuted.

Set to `"Either"` if you have no preference (and to maintain the default behavior from previous releases). Values must be enclosed in quotation marks.

Arguments

`"Source"`	Optimizes the generated chain so that one of the schematic source nets is on the left-hand side of the chain (if possible).
`"Drain"`	Optimizes the generated chain so that one of the schematic drain nets is on the left-hand side of the chain (if possible).
`"Either"`	Preserves the default behavior from previous releases.

GUI Equivalent

Command:	*Options – Connectivity* (Generation)
Field:	*Chain Left Net*

Examples

```
envGetVal("layoutXL" "chainLeftNet")
envSetVal("layoutXL" "chainLeftNet" 'string "Drain")
envSetVal("layoutXL" "chainLeftNet" 'string "Either")
```

Related Topics

Connectivity Form

Generate Chained Devices Form

© 2023 Cadence Design Systems, Inc.

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainMirror

```
layoutXL chainMirror boolean { t | nil }
```

Description

Controls whether the instances in a chain are mirrored by the Generate Chained Devices Form command and the `lxChain` SKILL function.

The default is `t`.

Note: The `chainMirror` environment variable is used in conjunction with the chainPermutePins environment variable. Depending on the state of each environment variable, devices are mirrored, permuted, or left unchanged as indicated below.

- If both the `chainMirror` and `chainPermutePins` are set to `t`, odd devices are mirrored and even devices are permuted.
- If `chainMirror` is set to `t` and `chainPermutePins` is set to `nil`, odd devices are mirrored but even devices are not permuted.
- If `chainMirror` is set to `nil` and `chainPermutePins` is set to `t`, all devices are permuted.
- If both the environment variables are set to `nil`, no devices are mirrored or permuted.

Note: The optional arguments of `lxChain`, `mirror` and `permute` default to the value set for the `chainMirror` and `chainPermutePins` environment variables and can be used to override the existing environment variable values.

GUI Equivalent

Command:	*Connectivity – Generate – Chained Devices* (Generate Chained Devices Form)
Field:	*Mirror*

Examples

```
envGetVal("layoutXL" "chainMirror")
envSetVal("layoutXL" "chainMirror" 'boolean t)
envSetVal("layoutXL" "chainMirror" 'boolean nil)
```


© 2023 Cadence Design Systems, Inc.

Related Topics

`lxChain` SKILL function

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainMirrorEquivOrients

```
layoutXL chainMirrorEquivOrients boolean { t | nil }
```

Description

Controls the orientation in which devices are mirrored.

The default is `t`, which means devices are mirrored using equivalent orientations.

- For `R0` cells, `R0` and `MX` are mirrored to `MY` and `MY` and `R180` are mirrored to `R0`.
- For `R90` cells, `R270` and `MXR90` are mirrored to `MYR90` and `MYR90` and `R90` are mirrored to `R270`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "chainMirrorEquivOrients")
envSetVal("layoutXL" "chainMirrorEquivOrients" 'boolean t)
envSetVal("layoutXL" "chainMirrorEquivOrients" 'boolean nil)
```

Additional Information

The table below details the mirroring orientations for `R0` and `R90` cells, depending on the value of the `chainMirrorEquivOrients` environment variable.

Cell Orientation	Device Orientation	Mirroring Orientation with **chainMirrorEquivOrients = t**
R0	R0	MY
	MX	MY
	MY	R0
	R180	R0

© 2023 Cadence Design Systems, Inc.

R90	R270	MYR90
	MXR90	MYR90
	MYR90	R270
	R90	R270
Cell Orientation	**Device Orientation**	Mirroring Orientation with **chainMirrorEquivOrients = nil**
R0	R0	MY
	MX	R180
	MY	R0
	R180	MX
R90	R270	MYR90
	MXR90	R90
	MYR90	R270
	R90	MXR90

Related Topics

Generate Chained Devices Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainOptimize

```
layoutXL chainOptimize cyclic { "Tracks" | "Abutments" | "Both" }
```

Description

Optimizes the chains by minimizing the tracks or maximizing the abutments or both.

The default value is `"Both"`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "chainOptimize")
envSetVal("layoutXL" "chainOptimize" 'cyclic "Tracks")
envSetVal("layoutXL" "chainOptimize" 'cyclic "Abutments")
```

Related Topics

Connectivity Form

Generate Chained Devices Form

Chaining Transistors Automatically during Layout Generation

lxChain

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainPartitionByNumFins

```
layoutXL chainPartitionByNumFins { t | nil }
```

Description

Controls whether chaining partitions devices by number of fins and so only attempts to abut devices with the same number of fins. The default is nil.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "chainPartitionByNumFins")
envSetVal("layoutXL" "chainPartitionByNumFins" 'boolean t)
envSetVal("layoutXL" "chainPartitionByNumFins" 'boolean nil)
```

Related Topics

Device Abutment

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainPartitionByCompType

```
layoutXL chainPartitionByCompType { t | nil }
```

Description

Controls whether chaining partitions devices by component type and so only attempts to abut devices with the same component type. The default is nil.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "chainPartitionByCompType")
envSetVal("layoutXL" "chainPartitionByCompType" 'boolean t)
envSetVal("layoutXL" "chainPartitionByCompType" 'boolean nil)
```

Related Topics

Device Abutment

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainPartitionSizeMax

```
layoutXL chainPartitionSizeMax int integer
```

Description

Specifies the maximum number of devices allowed in a chain. The default is `8`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "chainPartitionSizeMax")
envSetVal("layoutXL" "chainPartitionSizeMax" 'int 5)
```

Related Topics

Device Abutment

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainPermute

```
layoutXL chainPermute boolean { t | nil }
```

Description

Controls whether the instances in a chain are permuted or not.

GUI Equivalent

Command:	*Connectivity – Generate – Chained Devices* (Generate Chained Devices Form)
Field:	*Permute*

Examples

```
envGetVal("layoutXL" "chainPermute")
envSetVal("layoutXL" "chainPermute" 'boolean t)
envSetVal("layoutXL" "chainPermute" 'boolean nil)
```

Related Topics

`lxChain` SKILL function

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainPermutePins

```
layoutXL chainPermutePins boolean { t | nil }
```

Description

Controls whether devices are permuted by the Generate Chained Devices Form command and the `lxChain` SKILL function.

The default is `t`.

Note: The `chainPermutePins` environment variable is used in conjunction with the chainMirror environment variable. Depending on the state of each environment variable, devices are mirrored, permuted, or left unchanged as indicated below.

- If both the `chainMirror` and `chainPermutePins` are set to `t`, odd devices are mirrored and even devices are permuted.
- If `chainMirror` is set to `t` and `chainPermutePins` is set to `nil`, odd devices are mirrored but even devices are not permuted.
- If `chainMirror` is set to `nil` and `chainPermutePins` is set to `t`, all devices are permuted.
- If both the environment variables are set to `nil`, no devices are mirrored or permuted.

Note: The optional arguments of `lxChain`, `mirror` and `permute` default to the value set for the `chainMirror` and `chainPermutePins` environment variables and can be used to override the existing environment variable values.

GUI Equivalent

Command:	*Connectivity – Generate – Chained Devices* (Generate Chained Devices Form)
Fields:	*Mirror, Permute*

Examples

```
envGetVal("layoutXL" "chainPermutePins")
envSetVal("layoutXL" "chainPermutePins" 'boolean t)
envSetVal("layoutXL" "chainPermutePins" 'boolean nil)
```


© 2023 Cadence Design Systems, Inc.

Related Topics

`lxChain` SKILL function

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainPreserveExistingChains

```
layoutXL chainPreserveExistingChains boolean { t | nil }
```

Description

Ensures that any existing chains in the design are not broken during interactive chaining.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Generate – Chained Devices*
Field:	*Preserve Existing Chains* (Generate Chained Devices Form)

Examples

```
envGetVal("layoutXL" "chainPreserveExistingChains")
envSetVal("layoutXL" "chainPreserveExistingChains" 'boolean t)
envSetVal("layoutXL" "chainPreserveExistingChains" 'boolean nil)
```

Related Topics

Device Abutment

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainPreserveRows

```
layoutXL chainPreserveRows boolean { t | nil }
```

Description

Accepts multiple rows as input and chains each row using the optional parameters supplied for the lxChain SKILL function and with the `useDeviceOrder` argument set. If a flat instance list is supplied, the instances are sorted into rows. If a list of rows is supplied, the rows are used as specified.

The default is `nil`.

Note: This functionality is specific to the `lxChain` SKILL function and is not supported by the interactive chaining command, *Generate Chained Devices.*

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "chainPreserveRows")
envSetVal("layoutXL" "chainPreserveRows" 'boolean t)
envSetVal("layoutXL" "chainPreserveRows" 'boolean nil)
```

Related Topics

lxChain

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainSyncChains

```
layoutXL chainSyncChains boolean { t | nil }
```

Description

Synchronizes chains across rows by first arranging the specified instances into rows and then by abutting the instances row-wise from right to left in each column pair.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "chainSyncChains")
envSetVal("layoutXL" "chainSyncChains" 'boolean t)
envSetVal("layoutXL" "chainSyncChains" 'boolean nil)
```

Related Topics

lxChain

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainSyncChainsSDFirst

```
layoutXL chainSyncChainsSDFirst boolean { t | nil }
```

Description

Synchronizes chains by attempting to abut using the `sdFirst` strategy which orients the devices so that the Source/Drain nets match for any subsequent abutment using the `dummyFirst` strategy.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "chainSyncChainsSDFirst")
envSetVal("layoutXL" "chainSyncChainsSDFirst" 'boolean t)
```

Related Topics

lxChain (Synchronized Chaining)

© 2023 Cadence Design Systems, Inc.

chainUseAbutSpacing

```
layoutXL chainUseAbutSpacing boolean { t | nil }
```

Description

Controls how chains are spaced.

If set to `t`, chains are spaced using the actual abutment spacing value. If set to `nil`, chains are spaced using the value of the lxPositionMinSep environment variable.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "chainUseAbutSpacing")
envSetVal("layoutXL" "chainUseAbutSpacing" 'boolean t)
envSetVal("layoutXL" "chainUseAbutSpacing" 'boolean nil)
```

Related Topics

lxChain

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

chainUseDeviceOrder

```
layoutXL chainUseDeviceOrder boolean { t | nil }
```

Description

Maintains the relative horizontal starting positions of the specified instances when forming the chain. Instances are sorted by the x-coordinates of their origins (from lowest to highest) and the resultant list used to abut the devices from right to left.

This argument does not always preserve device orientations; for example, it may be necessary to mirror an instance in order to abut it with its neighbor.

The default is `nil`.

Note: If `useDeviceOrder` is set to `t`, advanced node devices are chained even if their S/D pins are on different nets. In this case, the devices are abutted on dummy pins. In addition, when all the advanced node devices have the same bulk, all the individual chains created using these devices are merged into a single chain.

GUI Equivalent

Command:	*Connectivity – Generate – Chained Devices*
Field:	*Use Device Order* (Generate Chained Devices Form)

Examples

```
envGetVal("layoutXL" "chainUseDeviceOrder")
envSetVal("layoutXL" "chainUseDeviceOrder" 'boolean t)
envSetVal("layoutXL" "chainUseDeviceOrder" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

checkRFUnimplementedTerms

```
layoutXL checkRFUnimplementedTerms boolean { t | nil }
```

Description

Toggles the output of error messages and the creation of ungenerated CAS markers when *Check Against Source* is run within the context of a package, board, or module design.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "checkRFUnimplementedTerms")
envSetVal("layoutXL" "checkRFUnimplementedTerms" 'boolean nil)
```

Related Topics

Design Check Against Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

checkBindings

```
layoutXL checkBindings boolean { t | nil }
```

Description

Checks the binding for each cell in the library and displays a summary table showing the cells that need to be updated. The `checkBindings` environment variable is used by the `lxCheckLib` SKILL function.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "checkBindings")
envSetVal("layoutXL" "checkBindings" 'boolean t)
envSetVal("layoutXL" "checkBindings" 'boolean nil)
```

Related Topics

Design Check for Layout XL Compliance

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

checkCellsAsLeaf

`layoutXL checkCellsAsLeaf string "`*`name_of_the_cellviews`*`"`

Description

Considers the list of specified cellviews as leaf cells for an XL Compliance check of a hierarchical design or for an Update Binding run. The specified cellviews treated as leaf cells only have their terminals checked, not their instances and nets.

The default is `nil`.

Note: When specifying the format for the cellview, (`lib cell view`), the `lib`, `cell`, and `view` names can contain the `'\\*'` wildcard.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "checkCellsAsLeaf")
envSetVal("layoutXL" "checkCellsAsLeaf" 'string "(MOSLIB AND2 layout)")
envSetVal("layoutXL" "checkCellsAsLeaf" 'string "(\\* \\*AND2 layout\\*)")
```

Related Topics

Design Check for Layout XL Compliance

Binding Updates

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

checkCompType

```
layoutXL checkCompType boolean { t | nil }
```

Description

Controls whether the lxCheck SKILL function (or the *Connectivity – Check – XL Compliance* command) should issue information messages to report cells that may be missing component types.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "checkCompType")
envSetVal("layoutXL" "checkCompType" 'boolean t)
envSetVal("layoutXL" "checkCompType" 'boolean nil)
```

Related Topics

Design Check for Layout XL Compliance

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

checkDuplicateObjects

```
layoutXL checkDuplicateObjects boolean { t | nil }
```

Description

Checks and reports duplicate instances, shapes, and vias in the layout cellview.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*Report – Duplicate objects* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "checkDuplicateObjects")
envSetVal("layoutXL" "checkDuplicateObjects" 'boolean t)
envSetVal("layoutXL" "checkDuplicateObjects" 'boolean nil)
```

Related Topics

Update Binding Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

checkExcludeCellViews

```
layoutXL checkExcludeCellViews string "name_of_the_cellviews"
```

Description

Specifies the cellviews to be excluded from the following checks:

- lxHierCheck
- lxHierCheckAgainstSource
- lxCheckLib
- lxHierUpdateComponentsAndNets
- lxUpdateBinding (when run with `?allCellsInDesign` set to `t`)

The default is `""`.

Note: When specifying the format for the cellview, (`lib cell view`), the `lib`, `cell`, and `view` names can contain the `'\\*'` wildcard.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "checkExcludeCellViews")
envSetVal("layoutXL" "checkExcludeCellViews" 'string "(CHK INV sch\\*)"
envSetVal("layoutXL" "checkExcludeCellViews" 'string "(lib1 \\* \\*)")
envSetVal("layoutXL" "checkExcludeCellViews" 'string "(lib2 inv1 layout) (lib3
nand layout)")
```

Related Topics

Design Check for Layout XL Compliance

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

checkExtractLayout

```
layoutXL checkExtractLayout boolean { t | nil }
```

Description

Enables the extraction of nets during the following Layout XL-compliance checks:

- `lxCheck`
- `lxHierCheck`
- `lxCheckLib`

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "checkExtractLayout")
envSetVal("layoutXL" "checkExtractLayout" 'boolean t)
envSetVal("layoutXL" "checkExtractLayout" 'boolean nil)
```

Related Topics

Design Check for Layout XL Compliance

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

checkMasterCompType

```
layoutXL checkMasterCompType boolean { t | nil }
```

Description

Controls if cells found to have different masters are reported during a *Check Against Source* run.

When set to `t`, master differences in cells are ignored if the cells are found to have matching component types.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "checkMasterCompType")
envSetVal("layoutXL" "checkMasterCompType" 'boolean t)
envSetVal("layoutXL" "checkMasterCompType" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

checkMissingParamsOrProps

```
layoutXL checkMissingParamsOrProps boolean { t | nil }
```

Description

When set to `nil`, ignores parameters (and properties, if checkParamsOnly is also set to `nil`) that are present in one view but missing from the other.

The default is `nil`.

Set this to `t` if, for example, your layout has additional properties that you want to propagate to the schematic using the *Update Schematic Parameters* command.

This environment variable is honored by the following commands in the *Connectivity* menu.

- *Check Against Source*
- *Update Layout Parameters*
- *Update Schematic Parameters*

GUI Equivalent

Command:	*Options – Connectivity* (*Parameters* tab)
Field:	*Ignore missing parameters or properties* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "checkMissingParamsOrProps")
envSetVal("layoutXL" "checkMissingParamsOrProps" 'boolean t)
envSetVal("layoutXL" "checkMissingParamsOrProps" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

checkMissingTermPins

```
layoutXL checkMissingTermPins boolean { t | nil }
```

Description

Controls whether instance terminals with no pins are checked and reported during the Layout XL compliance check and Update Binding.

The default is `t`, which means that terminals with no pins are checked for and reported by the following Layout XL commands:

- *Connectivity – Check – XL Compliance* (lxCheck)
- *Connectivity – Update – Binding* (lxUpnoinding)

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "checkMissingTermPins")
envSetVal("layoutXL" "checkMissingTermPins" 'boolean t)
envSetVal("layoutXL" "checkMissingTermPins" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

checkParamsOnly

```
layoutXL checkParamsOnly boolean { t | nil }
```

Description

Specifies whether the *Check Against Source*, *Update Layout Parameters*, and *Update Schematic Parameters* commands consider CDF parameters only or CDF parameters *and* cellview properties when comparing values in schematic and layout.

- When set to `t`, the commands check only CDF parameters in the schematic against the parameters and properties in the layout.
- When set to `nil`, the commands check both CDF parameters and cellview properties in the schematic against parameters and properties in the layout. Use this setting to check or update user-defined properties in the layout and schematic or when you know that a particular instance has certain properties that you are interested in comparing or updating.

In both cases, the commands then report (or update) values that do not match, parameters that cannot be checked, and parameters that are missing from the layout view.

The default is `nil`.

GUI Equivalent

Command:	*Options – Connectivity* (*Parameters* tab)
Field:	*Consider parameters only* *Consider parameters and properties* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "checkParamsOnly")
envSetVal("layoutXL" "checkParamsOnly" 'boolean t)
envSetVal("layoutXL" "checkParamsOnly" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

checkPermute

```
layoutXL checkPermute boolean { t | nil }
```

Description

Controls whether the lxCheck SKILL function (or the *Connectivity – Check – XL Compliance* command) should issue information messages to report cells that may be missing permute rules.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "checkPermute")
envSetVal("layoutXL" "checkPermute" 'boolean t)
envSetVal("layoutXL" "checkPermute" 'boolean nil)
```

Related Topics

Design Check for Layout XL Compliance

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

checkRemoveDevice

```
layoutXL checkRemoveDevice boolean { t | nil }
```

Description

Controls whether the lxCheck SKILL function (or the *Connectivity – Check – XL Compliance* command) should issue information messages to report instances that may need to be ignored using the lxRemoveDevice Property property.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "checkRemoveDevice")
envSetVal("layoutXL" "checkRemoveDevice" 'boolean t)
envSetVal("layoutXL" "checkRemoveDevice" 'boolean nil)
```

Related Topics

Design Check for Layout XL Compliance

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

checkShortsAndOpens

```
layoutXL checkShortsAndOpens boolean { t | nil }
```

Description

Controls whether cellviews with shorts between instance terminals and opens between strongly- or weekly-connected pins are reported during a schematic versus layout check.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "checkShortsAndOpens")
envSetVal("layoutXL" "checkShortsAndOpens" 'boolean t)
envSetVal("layoutXL" "checkShortsAndOpens" 'boolean nil)
```

Related Topics

Update Binding Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

checkStateConfirmModeChange

```
ab checkStateConfirmModeChange boolean { t | nil }
```

Description

Controls whether or not you must confirm explicitly that a read-only cellview is to be re-opened in edit mode before setting the checked state of a marker in the Annotation Browser assistant.

The default is `t`, which means that when you try to use the *Check/Uncheck* command (or its browser pane equivalent) in a read-only cellview, the software prompts you to confirm that you want to re-open the cellview in edit mode before performing the requested operation. When set to `nil`, the software automatically re-opens the cellview in edit mode and performs the requested operation.



Tip

The deleteConfirmModeChange environment variable performs the same function for the Annotation Browser *Delete* command.

GUI Equivalent

None

Examples

```
envGetVal("ab" "checkStateConfirmModeChange")
envSetVal("ab" "checkStateConfirmModeChange" 'boolean t)
envSetVal("ab" "checkStateConfirmModeChange" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

checkStrictFingers

```
layoutXL checkStrictFingers boolean { t | nil }
```

Description

Checks the number of fingers of an `nmos` or `pmos` layout device against the source and triggers the update of the associated schematic or layout devices, as appropriate.

When set to `t`:

- *Check Against Source* reports the differences in the total number of fingers, if a varying finger count is found between the schematic and the layout devices.
- *Update Layout Parameters* splits the number of fingers found on the schematic instances to evenly distribute them among the layout instances.
- *Update Schematic Parameters* updates the fingers of the schematic instance to match the total number of fingers on the layout instance.

The default is `nil`.

Important

For the `checkStrictFingers` to be supported, ensure that the `fingers` parameter is not ignored for generation or check and that the parameter name is listed as supported by the lxFingeringNames environment variable.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "checkStrictFingers")
envSetVal("layoutXL" "checkStrictFingers" 'boolean t)
envSetVal("layoutXL" "checkStrictFingers" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

checkTermLabelAllowBracketDiff

```
layoutXL checkTermLabelAllowBracketDiff boolean { t | nil }
```

Description

Specifies whether the difference between the brackets `[]` and <> should be ignored during an XL Compliance terminal label check.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "checkTermLabelAllowBracketDiff")
envSetVal("layoutXL" "checkTermLabelAllowBracketDiff" 'boolean t)
```

Related Topics

Check XL Compliance Form

Design Check for Layout XL Compliance

© 2023 Cadence Design Systems, Inc.

ciwWindow

```
layoutXL ciwWindow string "pair_of_coordinates"
```

Description

Specifies the position and size of the command interpreter window (CIW) on the screen.

By default, the CIW is positioned in the bottom left corner of the screen.

Arguments

pair_of_coordinates

A pair of coordinates enclosed in quotation marks specifying the lower left and upper right corners of the CIW; for example,

```
"((0.0 0.0) (600.0 300.0))"
```

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "ciwWindow")
envSetVal("layoutXL" "ciwWindow" 'string "((3.0 3.0) (750.0 180.0))")
```

Related Topics

Using the Command Interpreter Window

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cloneArrayFlipFirstColumn

```
layoutXL cloneArrayFlipFirstColumn boolean { t | nil }
```

Description

Flips the first column of the generated clone array.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Generate – Clones – Array Settings*
Field:	*Flip First Column (*Clone Array Settings Form)

Example

```
envGetVal("layoutXL" "cloneArrayFlipFirstColumn")
envSetVal("layoutXL" "cloneArrayFlipFirstColumn" 'boolean t)
envSetVal("layoutXL" "cloneArrayFlipFirstColumn" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cloneArrayFlipFirstRow

```
layoutXL cloneArrayFlipFirstRow boolean { t | nil }
```

Description

Flips the first row of the generated clone array.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Generate – Clones – Array Settings*
Field:	*Flip First Row (*Clone Array Settings Form)

Example

```
envGetVal("layoutXL" "cloneArrayFlipFirstRow")
envSetVal("layoutXL" "cloneArrayFlipFirstRow" 'boolean t)
envSetVal("layoutXL" "cloneArrayFlipFirstRow" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cloneArrayFlipNextFromPreviousColumn

```
layoutXL cloneArrayFlipNextFromPreviousColumn boolean { t | nil }
```

Description

Flips each column of the clone array from the previous column.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Generate – Clones – Array Settings*
Field:	*Flip Next From Previous Column* (Clone Array Settings Form)

Example

```
envGetVal("layoutXL" "cloneArrayFlipNextFromPreviousColumn")
envSetVal("layoutXL" "cloneArrayFlipNextFromPreviousColumn" 'boolean t)
envSetVal("layoutXL" "cloneArrayFlipNextFromPreviousColumn" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cloneArrayFlipNextFromPreviousRow

```
layoutXL cloneArrayFlipNextFromPreviousRow boolean { t | nil }
```

Description

Flips each row of the clone array from the previous row.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Generate – Clones – Array Settings*
Field:	*Flip Next From Previous Row* (Clone Array Settings Form)

Example

```
envGetVal("layoutXL" "cloneArrayFlipNextFromPreviousRow")
envSetVal("layoutXL" "cloneArrayFlipNextFromPreviousRow" 'boolean t)
envSetVal("layoutXL" "cloneArrayFlipNextFromPreviousRow" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cloneArrayRowOrColValue

```
layoutXL cloneArrayRowOrColValue cyclic { "Row" | "Column" }
```

Description

Specifies whether the clones are placed in an array arrangement based on the maximum number of clones allowed per *Row* or *Column*.

GUI Equivalent

Command:	*Connectivity – Generate – Clones – Array Settings*
Field:	*Number of clones per Row/Column* (Clone Array Settings Form)

Examples

```
envGetVal("layoutXL" "cloneArrayRowOrColValue")
envSetVal("layoutXL" "cloneArrayRowOrColValue" 'cyclic "Row")
envSetVal("layoutXL" "cloneArrayRowOrColValue" 'cyclic "Column")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cloneArrayXSpacing

```
layoutXL cloneArrayXSpacing int integer
```

Description

Specifies the distance from the right-most edge of one clone column to the left-most edge of the other clone column for clone array generation.

The default is `0.1`.

GUI Equivalent

Command:	*Connectivity – Generate – Clones – Array Settings*
Field:	*Clone X Spacing* (Clone Array Settings Form)

Examples

```
envGetVal("layoutXL" "cloneArrayXSpacing")
envSetVal("layoutXL" "cloneArrayXSpacing" 'int 0.5)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cloneArrayYSpacing

```
layoutXL cloneArrayYSpacing int integer
```

Description

Specifies the distance from the top-most edge of one clone row to the bottom-most edge of the other clone row for clone array generation.

The default is `0.1`.

GUI Equivalent

Command:	*Connectivity – Generate – Clones – Array Settings*
Field:	*Clone Y Spacing* (Clone Array Settings Form)

Examples

```
envGetVal("layoutXL" "cloneArrayYSpacing")
envSetVal("layoutXL" "cloneArrayYSpacing" 'int 0.5)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cloneJustify

```
layoutXL cloneJustify cyclic { "Center" | "Bottom left" | "Bottom right"| "Top left"
     | "Top right" }
```

Description

Controls the alignment of the clone you want to generate before you place it in the layout view.

When the clone is large, this environment variable helps to position the clone closer to another object.

The default is `"Center"`.

GUI Equivalent

Command:	*Connectivity – Generate – Clones*
Field:	*Justification*

Examples

```
envGetVal("layoutXL" "cloneJustify")
envSetVal("layoutXL" "cloneJustify" 'cyclic "Bottom left")
envSetVal("layoutXL" "cloneJustify" 'cyclic "Top right")
```

Related Topics

Generate Clones Form

Clone Source Selection and Target Search and Generation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cloningAutomaticUpdateLayoutParameters

```
layoutXL cloningAutomaticUpdateLayoutParameters boolean { t | nil }
```

Description

Automatically updates the parameters and parameter values on the devices in a generated clone to match their counterparts in the schematic clone source. Parameters that are set on layout devices but not present on their schematic counterparts are not removed.

When switched off, the parameters and values are taken from the layout clone source.

GUI Equivalent

Command:	*Connectivity – Generate – Clones*
Field:	*Update Layout Parameters to Match Schematic* (Generate Clones Form)

Examples

```
envGetVal("layoutXL" "cloningAutomaticUpdateLayoutParameters")
envSetVal("layoutXL" "cloningAutomaticUpdateLayoutParameters" 'boolean t)
envSetVal("layoutXL" "cloningAutomaticUpdateLayoutParameters" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cloningDoExactMatch

```
layoutXL cloningDoExactMatch boolean { t | nil }
```

Description

Requires that the parameter values on target components match the parameter values on the source components exactly. When switched off, Layout XL will also suggest targets that have different parameters and values.

Note: If required, you can update the parameters and values when you generate a clone by setting the cloneJustify environment variable to `t`.

GUI Equivalent

Command:	*Connectivity – Generate – Clones*
Field:	*Exact Parameters* (Generate Clones Form)

Examples

```
envGetVal("layoutXL" "cloningDoExactMatch")
envSetVal("layoutXL" "cloningDoExactMatch" 'boolean t)
envSetVal("layoutXL" "cloningDoExactMatch" 'boolean nil)
```

Related Topics

Clone Source Selection and Target Search and Generation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cloneNumberPerRowOrCol

```
layoutXL cloneNumberPerRowOrCol int integer
```

Description

Specifies the maximum number of clones that can be placed in a column or row, depending on the value set for cloneArrayRowOrColValue.

- Specifies the maximum number of clones that can be placed in a row, if cloneArrayRowOrColValue is set to `Rows`.
- Specifies the maximum number of clones that can be placed in a column, if cloneArrayRowOrColValue is set to `Columns`.

The default is `10`, which means a maximum of 10 clones can be placed in a column or row for clone array generation.

Note: Alternatively, you can use the *Number of clone per Rows/Columns* field in the Clone Array Settings form to specify the maximum number of clones to be placed in a column or in a row.

GUI Equivalent

Command:	*Connectivity – Generate – Clones – Array Settings*
Field:	*Number of clones per Row/Column* (Generate Clones Form)

Examples

```
envGetVal("layoutXL" "cloneNumberPerRowOrCol")
envSetVal("layoutXL" "cloneNumberPerRowOrCol" 'int 2)
```

Related Topics

Clone Source Selection and Target Search and Generation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cloneSynchronizeUnlockedColor

```
layoutXL cloneSynchronizeUnlockedColor cyclic { "no" | "cut only" | "shape only" |
     "cut and shape"}
```

Description

Synchronizes unlocked colors for lockless flows.

- `no` synchronizes only `locked` colors. This is the default.
- `cut only` synchronizes only `unlocked` colors for cuts. Here, the via plates are considered as shapes.
- `shape only` synchronizes only `unlocked` colors for shapes. Shapes include rectangles, via plates, and path segments.
- `cut and shape` synchronizes `unlocked` colors for both cuts and shapes.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "cloneSynchronizeUnlockedColor")
envSetVal("layoutXL" "cloneSynchronizeUnlockedColor" 'cyclic "cut only")
envSetVal("layoutXL" "cloneSynchronizeUnlockedColor" 'cyclic "shape only")
envSetVal("layoutXL" "cloneSynchronizeUnlockedColor" 'cyclic "cut and shape")
```

Related Topics

Generate Clones Form

Clone Source Selection and Target Search and Generation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cloneTransformFrom

```
layoutXL cloneTransformFrom cyclic { "Previous" | "Source" }
```

Description

Applies the clone transformation value either from the previous clone or from the clone source.

The default is `"Source"`.

GUI Equivalent

Command:	*Connectivity – Generate – Clones*
Field:	*Transform From*

Examples

```
envGetVal("layoutXL" "cloneTransformFrom ")
envSetVal("layoutXL" "cloneTransformFrom " 'cyclic "Previous")
```

Related Topics

Generate Clones Form

Clone Source Selection and Target Search and Generation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cloningSearchIn

```
layoutXL cloningSearchIn cyclic { "Schematic" | "Layout" }
```

Description

Controls the scope and criteria used when searching for target structures that match the clone source structure.

The default is `"Schematic"`, which means the search for matching target structures is run in the selected schematic cellview. When set to `"Layout"`, the search is run in the selected layout cellview.

GUI Equivalent

Command:	*Connectivity – Generate – Clones*
Field:	*Search* (Generate Clones Form)

Examples

```
envGetVal("layoutXL" "cloningSearchIn")
envSetVal("layoutXL" "cloningSearchIn" 'cyclic "Layout")
```

Related Topics

Clone Source Selection and Target Search and Generation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cloningSearchInConstraintOnly

```
layoutXL cloningSearchInConstraintOnly boolean { t | nil }
```

Description

Searches for matching target structures only within the other Clone Defs that belong to the same Clone Family constraint as the source. The clone source must be a Clone Def from a Clone Family constraint.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Generate – Clones*
Field:	*Search – Constraint Only* (Generate Clones Form)

Examples

```
envGetVal("layoutXL" "cloningSearchInConstraintOnly")
envSetVal("layoutXL" "cloningSearchInConstraintOnly" 'boolean t)
envSetVal("layoutXL" "cloningSearchInConstraintOnly" 'boolean nil)
```

Related Topics

Using the Generate Clones Form to Generate the Clone Family Constraint

Clone Source Selection and Target Search and Generation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cloningSearchInSelSetOnly

```
layoutXL cloningSearchInSelSetOnly boolean { t | nil }
```

Description

Controls the scope of the clone search.

The default is `nil`.

When set to `t` and searching in:

- *Schematic* – The cloning engine searches for clone targets only within the selected set in the schematic.
- *Layout* – The cloning engine searches for clone targets only within the selected set in the layout. All the selected layout instances are added to the clone target search space, including the selected instances placed within the PR Boundary.

GUI Equivalent

Command:	*Connectivity – Generate – Clones*
Field:	*Search – Selected Set Only* (Generate Clones Form)

Examples

```
envGetVal("layoutXL" "cloningSearchInSelSetOnly")
envSetVal("layoutXL" "cloningSearchInSelSetOnly" 'boolean t)
envSetVal("layoutXL" "cloningSearchInSelSetOnly" 'boolean nil)
```

Related Topics

Clone Source Selection and Target Search and Generation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cloningUpdateConstraint

```
layoutXL cloningUpdateConstraint boolean { t | nil }
```

Description

Creates a new Clone Family constraint and the associated Clone Def in the Constraint Manager, based on the new synchronous clone figGroups that you created.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Generate – Clones*
Field:	*Create Options – Update Constraint* (Generate Clones Form)

Examples

```
envGetVal("layoutXL" "cloningUpdateConstraint")
envSetVal("layoutXL" "cloningUpdateConstraint" 'boolean t)
envSetVal("layoutXL" "cloningUpdateConstraint" 'boolean nil)
```

Related Topics

Clone Source Selection and Target Search and Generation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

closeSchWindow

```
layoutXL closeSchWindow boolean { t | nil }
```

Description

If the layout is opened first in Layout XL or a higher tier and the schematic window is automatically opened, the schematic window will be automatically closed when the layout window is closed. If the schematic is opened first and then a window is launched for Virtuoso Layout XL or higher tier from the schematic, the schematic window is not automatically closed when the layout window is closed.

Note: The default is `nil`.

GUI Equivalent

Command:	*Options – Connectivity – General (tab)*
Field:	*Close schematic when exiting layout* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "closeSchWindow")
envSetVal("layoutXL" "closeSchWindow" 'boolean t)
envSetVal("layoutXL" "closeSchWindow" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

colorDraglines

```
layoutXL colorDraglines boolean { t | nil }
```

Description

Displays each dragline in a different color. This environment variable is honored by the *Generate Selected From Source*, *Generate Clones*, *Move*, and *Stretch* commands.



Tip

For more information on displaying draglines, see also hideDraglinesForGlobalNets and showDraglinesForDistantConns.

GUI Equivalent

Command:	*Options – Connectivity* (*Display* tab)
Field:	*Draglines – Display Draglines – Use colors* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "colorDraglines")
envSetVal("layoutXL" "colorDraglines" 'boolean t)
envSetVal("layoutXL" "colorDraglines" 'boolean nil)
```

Additional Information

Displaying many draglines during interactive commands can impair the performance of Layout XL. To mitigate these effects, use the `maxDragFig` environment variable to limit the number of figures that can be in a dragset.

To set `maxDragFig` interactively so that it takes effect in the current session, type the following in the CIW, where *`win`* is the window to which the limit applies.

```
win~>maxDragFig = 1000
```

To set a default value for `maxDragFig`, which will be used in all future Layout XL sessions, put the following line in your `.cdsenv` file.

```
graphic maxDragFig int 1000 nil
```


© 2023 Cadence Design Systems, Inc.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

colorOrNot

```
layoutXL colorOrNot boolean { t | nil }
```

Description

When set to `t`, the environment variable enables the macro coloring scheme for hard blocks and soft blocks. The default highlight setting is `hilite drawing` for hard blocks and `hilite drawing1` for soft blocks.

However, you can use the `hardBlockColor` and `softBlockColor` environment variables to assign different highlight colors for hard blocks and soft blocks so that they are visually distinguishable in the layout canvas.

GUI Equivalent

Command:	*Floorplan – Floorplan Global Options*
Field:	*Enable macro coloring scheme*

Examples

```
envGetVal("layoutXL" "colorOrNot")
envSetVal("layoutXL" "colorOrNot" 'boolean t)
envSetVal("layoutXL" "colorOrNot" 'boolean nil)
```

Related Topics

Floorplan Global Options

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

constraintAwareEditing

```
layoutXL constraintAwareEditing boolean { t | nil }
```

Description

Controls Constraint-Aware Editing mode, which ensures that the *Copy*, *Move*, *Stretch*, and *Rotate* commands honor the following constraints: *Alignment*, *Fixed*, *Locked*, *Matched Parameters*, *Orientation*, *Matched Orientation*, and *Symmetry*.

Constraint-aware editing is on by default, meaning that Layout XL notifies you of any constraint violations in the design by placing a marker at the appropriate location in the design window. As you edit your design, Layout XL continuously monitors the changes you make and adds or removes violation markers appropriately as specific constraints are violated or satisfied.

Note: Constraint-aware editing is a session-wide setting. It applies to all existing and new windows opened in the current Virtuoso session.

GUI Equivalent

Command:	*Options – Connectivity* (*General* tab)
Field:	*Constraint-aware editing* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "constraintAwareEditing")
envSetVal("layoutXL" "constraintAwareEditing" 'boolean t)
envSetVal("layoutXL" "constraintAwareEditing" 'boolean nil)
```

Additional Information

- If you use design-rule-driven editing in Enforce mode in conjunction with constraint-aware editing, design rules are enforced not only for the object you are editing, but for all of its constraint partners as well.

  For information on design-rule-driven editing, see the *Virtuoso Design Rule Driven Editing User Guide*.

- Constraint-aware editing does not allow edits across pseudo-hierarchy levels. For example, if you have a symmetry constraint with one member inside a group and another member outside the group, the constraint cannot be satisfied.

© 2023 Cadence Design Systems, Inc.

- For detailed definitions of the supported constraints, see Default Constraint Types in the *Virtuoso Unified Custom Constraints User Guide*.
- For more information on toggling constraint-aware editing mode from the graphical user interface, see Constraint-Aware Editing Considerations.

Related Topics

Constraint-Aware Editing

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cphAutoUprev

```
layoutXL cphAutoUprev boolean { t | nil }
```

Description

Controls the automatic design uprev during layout generation, such as the automatic uprev of any predefined properties.

The auto uprev happens when the schematic is opened in Virtuoso Layout Suite XL for the first time or with dynamic evaluation turned off by using the shell environment variable `CPH_USE_SCHEM_PROP`.

The environment variable only controls design-level uprevs. This means that an uprev will be supported for properties that are defined on instances but not for properties that are defined on symbols.

Cadence recommends setting symbol properties in the LAM file. But if a property cannot be set in LAM and must be set on the symbol view, you must set the shell environment variable, `CPH_USE_SCHEM_PROP` to `1`, before starting the Virtuoso session.

The default value is `t`.



Tip

If you have already launched the Schematic XL and the layout properties have already been automatically uprevved, in addition to setting the `cphAutoUprev` environment variable to `nil`, you must delete the physConfig view from the disk before generating the layout to avoid an automatic design uprev.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "cphAutoUprev")
envSetVal("layoutXL" "cphAutoUprev" 'boolean t)
envSetVal("layoutXL" "cphAutoUprev" 'boolean nil)
```


© 2023 Cadence Design Systems, Inc.

Related Topics

How to Control CPH with Schematic Properties

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cphCheckEmptyConfig

```
layoutXL cphCheckEmptyConfig boolean { t | nil }
```

Description

Prevents hierarchy elaboration from switching into a *physConfig* that is zero sized. If the *physConfig* is zero sized, elaboration switches into the next view in the logical switch view list instead.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "cphCheckEmptyConfig")
envSetVal("layoutXL" "cphCheckEmptyConfig" 'boolean nil)
```

Related Topics

Physical Hierarchy Configuration

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cphCheckTopCellName

```
layoutXL cphCheckTopCellName boolean { t | nil }
```

Description

During hierarchy elaboration, checks whether the lower-level physConfig that CPH is switching to is referencing the correct top cellview library and cell name.

The default is `t`, which means that the lower-level physConfig is checked to verify the referenced top cellview library and cell name. During the check, if the library and cell name of the top cellview referenced by the lower-level physConfig do not match the expected library and cell name, CPH ignores the lower-level physConfig view and switches to the next available view in the switch list.

Note: If the automatic switch to a matching view is not possible, a warning is issued.

When set to `nil`, the lower-level physConfig is not checked to verify the referenced top cellview library and cell name.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "cphCheckTopCellName")
envSetVal("layoutXL" "cphCheckTopCellName" 'boolean t)
envSetVal("layoutXL" "cphCheckTopCellName" 'boolean nil)
```

Related Topics

Physical Hierarchy Configuration

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cphHierForceDescendStopLibs

```
layoutXL cphHierForceDescendStopLibs string "lib1 lib2"
```

Description

Determines where the hierarchical force descend will stop when the *Physical stop library* list is added.

Note: For maximum flexibility, hierarchal force descend can be added at the top level of any node.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "cphHierForceDescendStopLibs")
envSetVal("layoutXL" "cphHierForceDescendStopLibs" 'string "lib1 lib2")
```

Related Topics

Physical Hierarchy Configuration

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cphIncrementalElab

```
layoutXL cphIncrementalElab boolean { t | nil }
```

Description

Controls whether CPH traverses the entire schematic hierarchy or performs an incremental traversal, when launching VXL or during operations such as *Edit in Place* and *Descend Edit*. The capability to stop at the schematic hierarchy level that has existing layout IP blocks enables building the CPH tree incrementally, providing performance benefits for large designs.

The default is `t`, which means that the schematic traversal is performed incrementally.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "cphIncrementalElab")
envSetVal("layoutXL" "cphIncrementalElab" 'boolean t)
envSetVal("layoutXL" "cphIncrementalElab" 'boolean nil)
```

Related Topics

Physical Hierarchy Configuration

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cphLayoutLibraryInLibList

layoutXL cphLayoutLibraryInLibList boolean { t | nil }

Description

Adds layout library to the physical library list if it is different from the schematic layout.

GUI Equivalent

Examples

```
envGetVal("layoutXL" "cphLayoutLibraryInLibList")
envSetVal("layoutXL" "cphLayoutLibraryInLibList" 'boolean t)
```

Related Topics

Physical Hierarchy Configuration

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cphLayoutSync

```
layoutXL cphLayoutSync boolean { t | nil }
```

Description

Ensures that the *physConfig* view and the layout view are kept in sync when performing a *Discard Edit* operation.

The default is `t`. This means that layout editing commands like *Flatten* or *Set Transparent Instance* store the changes to the *Configure Physical Hierarchy*, such that the changes can be reverted if the layout view is modified.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "cphLayoutSync")
envSetVal("layoutXL" "cphLayoutSync" 'boolean nil)
```

Related Topics

Physical Hierarchy Configuration

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cphMasterLibraryOnly

```
layoutXL cphMasterLibraryOnly boolean { t | nil }
```

Description

Prevents CPH from searching in libraries other than the symbol master library for a corresponding physical view.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "cphMasterLibraryOnly")
envSetVal("layoutXL" "cphMasterLibraryOnly" 'boolean t)
```

Related Topics

Physical Hierarchy Configuration

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cphNonDefaultAutomaticName

```
layoutXL cphNonDefaultAutomaticName boolean { t | nil }
```

Description

Specifies whether automatically created `physConfigs` with read-only Layout XL or a higher sessions are given non-default names, such as `"physConfigAuto"` and`"physConfigAuto_2"`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "cphNonDefaultAutomaticName")
envSetVal("layoutXL" "cphNonDefaultAutomaticName" 'boolean t)
```

Related Topics

Create Physical Configuration View Form

Open Physical Configuration View Form

© 2023 Cadence Design Systems, Inc.

cphOpenWin

```
layoutXL cphOpenWin boolean { t | nil }
```

Description

Opens the graphical user interface and displays the physical configuration view associated with the specified ID.

The default is `t`.

GUI Equivalent

Command:	*New File* (form) – *OK* – Create Physical Configuration View Form
	OR
	Open File (form) – *OK* – Open Physical Configuration View Form
Field:	*Open CPH*

Examples

```
envGetVal("layoutXL" "cphOpenWin")
envSetVal("layoutXL" "cphOpenWin" 'boolean t)
envSetVal("layoutXL" "cphOpenWin" 'boolean nil)
```

Related Topics

Create Physical Configuration View Form

Open Physical Configuration View Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cphPromptConfigOrSchematic

```
layoutXL cphPromptConfigOrSchematic boolean { t | nil }
```

Description

Controls whether the user is prompted to open the physical configuration cellview or top cellview when descending into a physical configuration cellview in a schematic cellview.

When set to `nil`, the Open Configuration or Top CellView form will not be displayed. The schematic cellview will open in the context of the physical configuration cellview.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "cphPromptConfigOrSchematic")
envSetVal("layoutXL" "cphPromptConfigOrSchematic" 'boolean t)
envSetVal("layoutXL" "cphPromptConfigOrSchematic" 'boolean nil)
```

Related Topics

Physical Hierarchy Configuration

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cphScrollShowCellsMenu

```
layoutXL cphScrollShowCellsMenu boolean { t | nil }
```

Description

Enables a scroll bar for the *Show cells* menu in CPH Component Types window.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "cphScrollShowCellsMenu")
envSetVal("layoutXL" "cphScrollShowCellsMenu" 'boolean t)
envSetVal("layoutXL" "cphScrollShowCellsMenu" 'boolean nil)
```

Related Topics

Physical Hierarchy Configuration

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cphShowInstanceColumns

`layoutXL cphShowInstanceColumns string "`*`instance_column_name`*`"`

Description

Controls the display of columns in the CPH Instances table.

The default is `all`, which means all the instance columns are displayed.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "cphShowInstanceColumns")
envSetVal("layoutXL" "cphShowInstanceColumns" 'string "Name")
envSetVal("layoutXL" "cphShowInstanceColumns" 'string "viewList physLib stopList")
```

Related Topics

Physical Hierarchy Configuration

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cphStopLevel

```
layoutXL cphStopLevel int integer
```

Description

Stops the elaboration of the logical design hierarchy at the specified level. This lets you use the physical stop view list to limit the logical elaboration for large hierarchical designs that would otherwise take a long time to open.

The default is `0`, which means that a full elaboration is performed until a logical node is reached that maps to one of the view names specified in the *Physical stop view list* field.

Note: When the stop level is set using the CPH window, it overrides the value specified in the environment variable. But when the stop level is set using the Update Connectivity Reference form, the specified value overrides the value set in CPH and in the environment variable.

GUI Equivalent

Command:	*Connectivity – Update – Connectivity Reference*
Field:	*Stop Level*
	Update Connectivity Reference Form
Command:	*Launch – Configure Physical Hierarchy*
Field:	*Set Stop Level*
	CPH Toolbar

Examples

```
envGetVal("layoutXL" "cphStopLevel")
envSetVal("layoutXL" "cphStopLevel" 'int 2)
```

Related Topics

Physical Hierarchy Configuration

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cphStopLogicalElabAtPhysLeaf

```
layoutXL cphStopLogicalElabAtPhysLeaf boolean { t | nil }
```

Description

Stops the elaboration of the logical design hierarchy when a node is reached which maps to a physical leaf node. This lets you use the physical stop view list to limit the logical elaboration for large hierarchical designs that would otherwise take a long time to open or might not open at all in Layout XL.

The default is `t`, which means that the elaboration stops when it reaches a logical node that maps to one of the view names specified in the *Physical stop view list* field. In this case, global nets below the leaf instance in the schematic are not considered during the elaboration. If your design relies on such nets, Cadence recommends that you switch off this environment variable.

When set to `nil`, the logical design is fully elaborated when you launch the *Configure Physical Hierarchy* command.

Note: A change to this environment variable takes effect only after you re-elaborate the design. To force a full re-elaboration, do one of the following.

- Change the *Logical switch view list* in the *Global Bindings* pane in the Configure Physical Hierarchy window.
- Close and re-open Layout XL and then choose *Launch –Configure Physical Hierarchy* again.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "cphStopLogicalElabAtPhysLeaf")
envSetVal("layoutXL" "cphStopLogicalElabAtPhysLeaf" 'boolean t)
envSetVal("layoutXL" "cphStopLogicalElabAtPhysLeaf" 'boolean nil)
```

Related Topics

Physical Hierarchy Configuration

© 2023 Cadence Design Systems, Inc.

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cphStrictSchematicCheck

```
layoutXL cphStrictSchematicCheck boolean { t | nil }
```

Description

Stops the launch of the Layout XL window when the schematic does not match the top schematic specified in the physConfig view and displays a warning message.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "cphStrictSchematicCheck")
envSetVal("layoutXL" "cphStrictSchematicCheck" 'boolean t)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cphUseLayoutLibrary

```
layoutXL cphUseLayoutLibrary boolean { t | nil }
```

Description

When launching Layout XL using CPH in automatic mode, ensures that CPH directly checks the layout library to look for a corresponding physConfig.

The default is `nil`, which means CPH first checks the schematic library and then the layout library and if a corresponding physConfig is not found, it creates one in the layout library.

GUI Equivalent

Command:	*Update – Connectivity Reference*
Field:	*Use configuration view* (Update Connectivity Reference Form)

Examples

```
envGetVal("layoutXL" "cphUseLayoutLibrary")
envSetVal("layoutXL" "cphUseLayoutLibrary" 'boolean t)
envSetVal("layoutXL" "cphUseLayoutLibrary" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cphUseLayoutViewName

```
layoutXL cphUseLayoutViewName boolean { t | nil }
```

Description

When launching Layout XL or a higher tier using physConfig in automatic mode, checks if the layout view name uses a prefix or a postfix and uses the same naming convention to name the physConfig. For example, `abc_layout` leads to a physConfig named `abc_physConfig` and `layout_xyz` leads to a physConfig named `physConfig_xyz`. Any views named `physConfig` are ignored.

The default is `nil`, which means the view name `physConfig` is used.

GUI Equivalent

Command:	*Update – Connectivity Reference*
Field:	*Use configuration view* (Update Connectivity Reference Form)

Examples

```
envGetVal("layoutXL" "cphUseLayoutViewName")
envSetVal("layoutXL" "cphUseLayoutViewName" 'boolean t)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

cphVHSizeLimit

`layoutXL cphVHSizeLimit int` *`integer`*

Description

Specifies the block size at which CPH stops elaboration for virtual hierarchy generation and automatically creates a soft block.

The default is `0`, which means CPH elaboration is not controlled by the number of instances inside the virtual hierarchy.

GUI Equivalents

None

Examples

```
envGetVal("layoutXL" "cphVHSizeLimit")
envSetVal("layoutXL" "cphVHSizeLimit" 'int 50)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

createBoundaryLabel

```
layoutXL createBoundaryLabel boolean { t | nil }
```

Description

Specifies whether the *Generate All From Source* command automatically creates a label containing the library, cell, and view name of the design when creating a boundary from the Generate Layout form. The label is created on layer-purpose pair (`prBoundary boundary`).

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "createBoundaryLabel")
envSetVal("layoutXL" "createBoundaryLabel" 'boolean t)
envSetVal("layoutXL" "createBoundaryLabel" 'boolean nil)
```

Related Topics

Generate Layout Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

createImplicitBusTerminals

```
layoutXL createImplicitBusTerminals boolean { t | nil }
```

Description

Creates, checks, and updates implicit bus terminals in the layout cellview, based upon the explicit bus terminals in the schematic cellview. This avoids the need for running VerilogAnnotate to annotate the bus terminal and bus bit order (ascending or descending) information into the layout view. In addition, it enhances the interoperability of layout cellviews with the Innovus designs. The bus connectivity is created in the layout view as "implicit" objects so that they are not visible to other parts of Virtuoso. For example, the implicit bus terminals generated by Layout XL do not show up in the list of terminals associated with the layout cellview when accessed through SKILL.

Layout XL also sets the `isInterface` flag on the implicit bus terminal to `true` and that on the explicit bit terminals to `false`. This maintains the information as in the original schematic design.

The `createImplicitBusTerminals` environment variable also controls the annotation of bus terminals into layout cellviews generated by the Generate Physical Hierarchy command in Virtuoso GXL".

The default is `t`.

GUI Equivalent (Only supported by GFS)

Command:	*Options – Connectivity* (*Generation* tab)
Field:	*Create implicit bus terminals* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "createImplicitBusTerminals")
envSetVal("layoutXL" "createImplicitBusTerminals" 'boolean t)
envSetVal("layoutXL" "createImplicitBusTerminals" 'boolean nil)
```

Related Topics

Soft Block Mode

Design Check Against Source

© 2023 Cadence Design Systems, Inc.

Generation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

createPinsFromLabels

```
layoutXL createPinsFromLabels boolean { t | nil }
```

Description

Creates pins from labels during an *Update Binding* run, if the label text or text display matches an ungenerated schematic terminal name. Additionally, creates pins for terminals that already have pins but labels are swapped in the layout.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*Create – Ungenerated pins from labels* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "createPinsFromLabels")
envSetVal("layoutXL" "createPinsFromLabels" 'boolean t)
envSetVal("layoutXL" "createPinsFromLabels" 'boolean nil)
```

Related Topics

Update Binding Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

crossProbeHier

```
layoutXL crossProbeHier boolean { t | nil }
```

Description

Creates probes at higher levels in the layout so that the probes are not lost when returning from *Edit In Place* or *Descend Edit*.

Note: This environment variable can only be used for controlling the probes created using the *Probe* command in the shortcut menu.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "crossProbeHier")
envSetVal("layoutXL" "crossProbeHier" 'boolean t)
envSetVal("layoutXL" "crossProbeHier" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

crossSelect

```
ab crossSelect boolean { t | nil }
```

Description

Specifies whether cross selection is allowed between Annotation Browser and Navigator or Canvas.

The default is `t`, which means that the nets are cross selected.

GUI Equivalent

Command:	*Windows – Assistants – Annotation Browser – Options* (RMB)
Field:	*Cross selection*

Examples

```
envGetVal("ab" "crossSelect")
envSetVal("ab" "crossSelect" 'boolean nil)
```

Related Topics

Navigator

List of Layout XL Environment Variables

© 2023 Cadence Design Systems, Inc.

crossSelect

```
layoutXL crossSelect boolean { t | nil }
```

Description

Specifies whether cross-selection is operational. Cross-selection means that when you select a component in the layout, the corresponding component is selected in the schematic and vice versa.

The default is `t`.

GUI Equivalent

Command:	*Options – Connectivity* (*Display* tab)
Field:	*Cross-selection* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "crossSelect")
envSetVal("layoutXL" "crossSelect" 'boolean t)
envSetVal("layoutXL" "crossSelect" 'boolean nil)
```

Related Topics

Rapid Adoption Kit (*Introduction to Connectivity-Driven Design in VLS XL*)

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

crossSelectInstTerm

```
layoutXL crossSelectInstTerm boolean { t | nil }
```

Description

Enables cross-selection of instance terminals between the schematic and layout cellviews. This environment variable is available only for soft block pins and you should have Level-1 editing enabled to use it. This is because the corresponding instance terminal in the layout exists only in the lower-level and not the top design, which is available only in soft blocks.

Instance terminals selected in the schematic are automatically selected in the layout.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "crossSelectInstTerm")
envSetVal("layoutXL" "crossSelectInstTerm" 'boolean t)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

crossSelectLayoutShape

```
layoutXL crossSelectLayoutShape boolean { t | nil }
```

Description

Selects the entire schematic net when the corresponding layout shape or via is selected.

The default is `nil`, which means the corresponding schematic net is not selected when a layout shape or via is selected.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "crossSelectLayoutShape")
envSetVal("layoutXL" "crossSelectLayoutShape" 'boolean t)
envSetVal("layoutXL" "crossSelectLayoutShape" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

deleteConfirmModeChange

```
ab deleteConfirmModeChange boolean { t | nil }
```

Description

Controls whether or not you must confirm explicitly that a read-only cellview is to be re-opened in edit mode before deleting a marker in the Annotation Browser assistant.

The default is `t`, which means that when you try to use the *Delete* command (or its browser pane equivalent) in a read-only cellview, the software prompts you to confirm that you want to re-open the cellview in edit mode before performing the requested operation. When set to `nil`, the software automatically re-opens the cellview in edit mode and performs the requested operation.



Tip

The checkStateConfirmModeChange environment variable performs the same function for the Annotation Browser's *Check/Uncheck* command.

GUI Equivalent

None

Examples

```
envGetVal("ab" "deleteConfirmModeChange")
envSetVal("ab" "deleteConfirmModeChange" 'boolean t)
envSetVal("ab" "deleteConfirmModeChange" 'boolean nil)
```

Related Topics

Annotation Browser Assistant

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

deleteUnmatchedInsts

```
layoutXL deleteUnmatchedInsts boolean { t | nil }
```

Description

Automatically deletes layout instances that are no longer present in the schematic during the *Update Components And Nets* command.

The default is `nil`, which means unmatched instances are not deleted when updating components and nets, but are instead indicated with a marker in the layout view.

GUI Equivalent

Command:	*Connectivity – Update – Components And Nets* (*Update* tab)
Field:	*Delete Unmatched Instances* (Update Components and Nets Form)

Examples

```
envGetVal("layoutXL" "deleteUnmatchedInsts")
envSetVal("layoutXL" "deleteUnmatchedInsts" 'boolean t)
envSetVal("layoutXL" "deleteUnmatchedInsts" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

deleteUnmatchedPins

```
layoutXL deleteUnmatchedPins boolean { t | nil }
```

Description

Automatically deletes layout pins that are no longer present in the schematic during the *Update Components And Nets* command. Redundant nets and terminals are deleted from the layout view at the same time.

The default is `nil`, which means unmatched pins are not deleted when updating components and nets, but are instead indicated with a marker in the layout view.

GUI Equivalent

Command:	*Connectivity – Update – Components And Nets* (*Update* tab)
Field:	*Delete Unmatched Pins* (Update Components and Nets Form)

Examples

```
envGetVal("layoutXL" "deleteUnmatchedPins")
envSetVal("layoutXL" "deleteUnmatchedPins" 'boolean t)
envSetVal("layoutXL" "deleteUnmatchedPins" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

deviceExtractType

```
layoutXL deviceExtractType cyclic {"pcells" | "all" | "none"}
```

Description

Controls the connectivity extractor of the Virtuoso Space-based Router, which ensures that connectivity is set correctly on all the shapes around the connection points in a cell loaded in the router. By default, only Pcells are extracted. You can choose to extract all cells, or disable the spaced-based router extractor altogether.

This extraction is required mainly for Pcells in device-level routing, where the router will otherwise consider a gate shape to be a blockage, even though the underlying gate pin has connectivity assigned. The extraction happens in the internal data structures of the router; the original cell is not touched. You can avoid this step and its associated performance cost by updating the Pcells to ensure that all the shapes around connection points have appropriate connectivity defined.

Arguments

`pcells`	Extracts Pcells only.
`all`	Extracts all cells.
`none`	Disables the space-based router connectivity extractor.

Examples

```
envGetVal("layoutXL" "deviceExtractType")
envSetVal("layoutXL" "deviceExtractType" 'cyclic "pcells")
envSetVal("layoutXL" "deviceExtractType" 'cyclic "all")
envSetVal("layoutXL" "deviceExtractType" 'cyclic "none")
```

Related Topics

Space-based Router Extraction

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

disableCASOptionsPopUp

```
layoutXL disableCASOptionsPopUp boolean { t | nil }
```

Description

Prevents the Check Against Source Form form from being displayed every time you run the *Check Against Source* (CAS) command.

The default is `nil`, which means that the Check Against Source form is displayed after each CAS run.

To ensure that the form is not displayed after every CAS run, set the environment variable to `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "disableCASOptionsPopUp")
envSetVal("layoutXL" "disableCASOptionsPopUp" 'boolean t)
envSetVal("layoutXL" "disableCASOptionsPopUp" 'boolean nil)
```

Related Topics

Check Against Source Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

disableFolding

```
layoutXL disableFolding boolean { t | nil }
```

Description

Disables folding, if the folding threshold value for the selected component type is set to `0`.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "disableFolding")
envSetVal("layoutXL" "disableFolding" 'boolean t)
envSetVal("layoutXL" "disableFolding" 'boolean nil)
```

Related Topics

Generate Folded Devices Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

disableVXLClonedModgenEditPopUp

```
layoutXL disableVXLClonedModgenEditPopUp boolean { t | nil }
```

Description

Prevents the *Edit Cloned Modgen* pop-up form from displaying and edits the modgen instead of the pseudo-modgen, without displaying any warning.

The default is `nil`, which means the Edit Clones Modgen pop-up is displayed, if you attempt to edit a pseudo-modgen figGroup.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "disableVXLClonedModgenEditPopUp")
envSetVal("layoutXL" "disableVXLClonedModgenEditPopUp" 'boolean t)
envSetVal("layoutXL" "disableVXLClonedModgenEditPopUp" 'boolean nil)
```

Related Topics

Creating Modgens

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

displayConfigureTab

```
layoutXL displayConfigureTab { t | nil }
```

Description

Controls the display of the *Configure* tab in the Connectivity Form form. The default is that the tab is displayed (`t`).

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "displayConfigureTab")
envSetVal("layoutXL" "displayConfigureTab" 'boolean t)
envSetVal("layoutXL" "displayConfigureTab" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

drdUseNetName

```
layoutXL drdUseNetName boolean { t | nil }
```

Description

Controls whether an object that is being moved retains the net name of the pin or instance terminal to which it was connected prior to the move. This in turn lets the design-rule-driven (DRD) editing functionality use the name to flag short violations during the move operation.

For example, when using DRD editing, if you move an object on net A towards an object on net B, the system displays a message warning you about the short violation. If the move leads to the object becoming disconnected from an I/O pin or instance terminal, the `drdUseNetName` environment variable controls the behavior as follows.

- When set to `t`, the object retains the net name of the pin or instance terminal to which it was connected prior to the move and DRD editing uses this name to flag short violations.
- When set to `nil`, the object does not retain the net name, Layout XL is unaware of any connectivity associated with the object, and therefore no short message is displayed.

The original net assignment remains intact in both cases. The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "drdUseNetName")
envSetVal("layoutXL" "drdUseNetName" 'boolean t)
envSetVal("layoutXL" "drdUseNetName" 'boolean nil)
```

Additional Information

- The lxStickyNet Property property overrides any setting of `drdUseNetName`.
- For information on design-rule-driven editing, see the *Design Rule Driven Editing User Guide*.

© 2023 Cadence Design Systems, Inc.

Important

`drdUseNetName` has no effect when you stretch an object because assigned connectivity is always maintained during the stretch operation. However, if all the points of the object are selected prior to stretching, the stretch is in fact a move and the behavior is as described above.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

dummyBackAnnotateLocalNetActive

```
layoutXL dummyBackAnnotateLocalNetActive boolean { t | nil }
```

Description

Specifies whether dummy Pcell instances containing local nets should be considered as active during backannotation, even if they are tied to power or ground nets.

The default value is `nil`, which means that dummy Pcell instances are not considered as active.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "dummyBackAnnotateLocalNetActive")
envSetVal("layoutXL" "dummyBackAnnotateLocalNetActive" 'boolean t)
```

Related Topics

Dummy Instances Backannotation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

dummyBackAnnotateRelaxActiveRule

```
layoutXL dummyBackAnnotateRelaxActiveRule boolean { t | nil }
```

Description

Allows dummy instances with one or more power nets, or one or more ground nets, and no other nets, to be treated as inactive.This can be used to avoid them being backannotated.

The default value is `nil`, which means that dummy instances are considered active and backannotated.

When set to `t`, it allows the dummies tied to more than one power net, or tied to more than one ground net, to be treated as inactive.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "dummyBackAnnotateRelaxActiveRule")
envSetVal("layoutXL" "dummyBackAnnotateRelaxActiveRule" 'boolean t)
```

Related Topics

Dummy Instances Backannotation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

electricalPinSetup

```
layoutXL electricalPinSetup string "constraint_group_name"
```

Description

Specifies the name of the constraint group to use for determining the layers available for generating the electrical pins.

The default is `"virtuosoDefaultElectricalPinSetup"`.

Note: If the `virtuosoDefaultElectricalPinSetup` constraint group is not defined, the `validLayers` defined in the default Layout XL constraint group, `virtuosoDefaultSetup`, are used for pin generation.

GUI Equivalents

None

Examples

```
envGetVal("layoutXL" "electricalPinSetup")
envSetVal("layoutXL" "electricalPinSetup" 'string
"virtuosoDefaultElectricalPinSetup")
```

Related Topics

Prerequisites for Using Virtuoso Photonics Solution (Virtuoso Photonics Option)

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

enableCloneColoringSupport

```
layoutXL enableCloneColoringSupport boolean { t | nil }
```

Description

Enables the color state and the color of a “color locked” shape in a synchronous clone to be carried to the corresponding shapes in other synchronous clones.

The default is `t`.

- When set to `t`, and *Editing In Place* a clone, the color state and the color of the “color-locked” shapes inside the clone are automatically synchronized.
  - Color editing of shapes inside a synchronous clone by using the Multiple Patterning Toolbar is not supported.
- When set to `nil`, the color state and color of shapes inside the clone are not synchronized.
  - The Multiple Patterning Toolbar can be used to color edit the shapes in a synchronous clone.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "enableCloneColoringSupport")
envSetVal("layoutXL" "enableCloneColoringSupport" 'boolean t)
envSetVal("layoutXL" "enableCloneColoringSupport" 'boolean nil)
```

Related Topics

Synchronous Copy Generation

Clone Generation

Virtuoso Multi-Patterning Technology User Guide

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

equivalentAbutClasses

`layoutXL equivalentAbutClasses string "`*`list_of_equivalent_abut_class_sets`*`"`

Description

Allows abutment of instances that have different values set for the `abutClass` property but the `abutClass` values are defined as equivalent using this environment variable. The environment variable can be used to list multiple equivalent `abutClass` sets with each set separated by a semi colon (;).

In an equivalent `abutClass` set, the instances that have their `abutClass` property set to the first value in the set can be abutted to the instances that have the property set to the remaining values in the set. But, the instances set to one of the remaining `abutClass` values from a set cannot be abutted with each other.

For example, for equivalent abut classes, "`nhpdummy nhplvt nhphvt; phpdummy phplvt phphvt`", the instances set to `abutClass nhpdummy` are allowed to abut to the instances set to the remaining abutClasses in that set, `nhplvt` and `nhphvt`. But instances set to `nhplvt` and `nhphvt` cannot abut. Similarly, the instances set to `phpdummy` are allowed to abut to the instances set to the remaining abutClasses in that set, `phplvt` and `phphvt`. But instances set to `phplvt` and `phplvt` cannot abut.

The default is `""`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "equivalentAbutClasses")
envSetVal("layoutXL" "equivalentAbutClasses" 'string "nhpdummy nhplvt nhphvt; phpdummy phplvt phphvt")
envSetVal("layoutXL" "equivalentAbutClasses" 'string "pmos pmos1 pmos2; nmos nmos1 nmos2 nmos3")
```

Related Topics

abutClass Property

List of Layout XL Environment Variables

© 2023 Cadence Design Systems, Inc.

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

excludedViewList

`layoutXL excludedViewList string "`*`layout view names to be excluded`*`"`

Description

Specifies specific layout view names or layout view name patterns that should be excluded from opening in Layout XL.

- When a specific layout view name is provided, the environment variable performs an absolute match to exclude the view name.

  For example, if the specified view name to exclude is "`extracted avLayout`", the environment variable can exclude view names such as "`extracted`" and "`avLayout`". However, view names such as "`extractedLay`" or "`Lay_avLayout`" are allowed to open using Layout XL.

- When an exact view name is not specified but a view name pattern using one or more wildcard characters (*) is specified, the environment variable excludes all view names corresponding to the excluded pattern.

  Note: Each wildcard character (*) used in the view name can represent 0 to many characters.

  Some sample view name patterns using wildcard characters are: { `*` | `*abcd` | `ab*cd` | `abcd*` | `*abcd*` | `*ab*cd` | `*ab*cd*` | `ab*cd*` | `ab*c*d` }.

  For example;

  - If the specified view name pattern to exclude is `*`, all view names are excluded.
  - If the specified view name pattern to exclude is `ex*ed`, the view name `extracted` is excluded but `Lay_exABCed` is not.
  - If the specified view name pattern to exclude is `*ex*ed`, view names such as `extracted`, `LayexABCed`, and similar others are excluded.

By default, no cellviews are excluded from opening in Layout XL.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "excludedViewList")
```


© 2023 Cadence Design Systems, Inc.

```
envSetVal("layoutXL" "excludedViewList" 'string "extracted lay*out*")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

extractShortLocatorAutoZoomToFit

```
layoutXL extractShortLocatorAutoZoomToFit boolean { t | nil }
```

Description

Automatically zooms the layout to fit all the shapes in the current display mode.

The default is `nil`.

GUI Equivalents

Command:	Connectivity – Short Locator – Preferences (tab)
Field:	*Auto zoom to fit* (Short Locator Form)

Examples

```
envGetVal("layoutXL" "extractShortLocatorAutoZoomToFit")
envSetVal("layoutXL" "extractShortLocatorAutoZoomToFit" 'boolean t)
envSetVal("layoutXL" "extractShortLocatorAutoZoomToFit" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

extractShortLocatorCommonShapesColor

```
layoutXL extractShortLocatorCommonShapesColor string "color1"
```

Description

Specifies the color of the halo around the common shapes causing the short.

The default is `"cyan"`.

GUI Equivalents

Command:	Connectivity – Short Locator – Preferences (tab)
Field:	*Color – Common Shapes* (Short Locator Form)

Examples

```
envGetVal("layoutXL" "extractShortLocatorCommonShapesColor")
envSetVal("layoutXL" "extractShortLocatorCommonShapesColor" 'string "cyan")
envSetVal("layoutXL" "extractShortLocatorCommonShapesColor" 'string "red")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

extractShortLocatorDisplayMode

```
layoutXL extractShortLocatorDisplayMode cyclic { "Common shapes" | "All shortest
     paths" | "Entire net" }
```

Description

Specifies the mode in which shapes are displayed in the layout canvas.

The default is `"Common shapes"`.

GUI Equivalents

Command:	Connectivity – Short Locator – General (tab)
Field:	*Display* (Short Locator Form)

Examples

```
envGetVal("layoutXL" "extractShortLocatorDisplayMode")
envSetVal("layoutXL" "extractShortLocatorDisplayMode" 'cyclic "Common shapes")
envSetVal("layoutXL" "extractShortLocatorDisplayMode" 'cyclic "All shortest
paths")
envSetVal("layoutXL" "extractShortLocatorDisplayMode" 'cyclic "Entire net")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

extractShortLocatorHaloTransparency

`layoutXL extractShortLocatorHaloTransparency int` *`integer`*

Description

Specifies the extent to which the halo drawn around the common shapes is transparent.

The default is `50`.

GUI Equivalents

None

Examples

```
envGetVal("layoutXL" "extractShortLocatorHaloTransparency")
envSetVal("layoutXL" "extractShortLocatorHaloTransparency" 'int 50)
envSetVal("layoutXL" "extractShortLocatorHaloTransparency" 'int 75)
```

Related Topics

Short Locator Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

extractShortLocatorHighlightLimit

`layoutXL extractShortLocatorHighlightLimit int` *`integer`*

Description

Specifies the maximum number of shapes that will be highlighted. This can help improve the display performance for a large net.

The default is `100000`.

GUI Equivalents

Command:	Connectivity – Short Locator – Preferences (tab)
Field:	*Show net for virtual labels – Highlight limit* (Short Locator Form)

Examples

```
envGetVal("layoutXL" "extractShortLocatorHighlightLimit")
envSetVal("layoutXL" "extractShortLocatorHighlightLimit" 'int 50000)
envSetVal("layoutXL" "extractShortLocatorHighlightLimit" 'int 100000)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

extractShortLocatorNetColor1

```
layoutXL extractShortLocatorNetColor1 string "Color1"
```

Description

Specifies the highlight color for the first of the two shorted nets.

The default is `"lime"`.

GUI Equivalents

Command:	Connectivity – Short Locator – Preferences (tab)
Field:	*Colors* (Short Locator Form)

Examples

```
envGetVal("layoutXL" "extractShortLocatorNetColor1")
envSetVal("layoutXL" "extractShortLocatorNetColor1" 'string "lime")
envSetVal("layoutXL" "extractShortLocatorNetColor1" 'string "red")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

extractShortLocatorNetColor2

```
layoutXL extractShortLocatorNetColor2 string "Color1"
```

Description

Specifies the highlight color for the second of the two shorted nets.

The default is `"magenta"`.

GUI Equivalents

Command:	Connectivity – Short Locator – Preferences (tab)
Field:	*Colors* (Short Locator Form)

Examples

```
envGetVal("layoutXL" "extractShortLocatorNetColor2")
envSetVal("layoutXL" "extractShortLocatorNetColor2" 'string "magenta")
envSetVal("layoutXL" "extractShortLocatorNetColor2" 'string "red")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

extractShortLocatorNumThreads

`layoutXL extractShortLocatorNumThreads int` *`integer`*

Description

Controls the number of threads used to chase the net with the short.

- If set to `0`, the number of threads is computed automatically.
- If set to > `0`, the specified number of threads are used.

The default is `0`.

GUI Equivalents

None

Examples

```
envGetVal("layoutXL" "extractShortLocatorNumThreads")
envSetVal("layoutXL" "extractShortLocatorNumThreads" 'int 0)
envSetVal("layoutXL" "extractShortLocatorNumThreads" 'int 2)
```

Related Topics

Short Locator Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

extractShortLocatorOtherShapesColor

```
layoutXL extractShortLocatorOtherShapesColor string "color"
```

Description

Specifies the color of the halo around the shapes that are neither common shapes nor shapes of the two shorted nets.

The default is `"white"`.

GUI Equivalents

Command:	Connectivity – Short Locator – Preferences (tab)
Field:	*Colors – Other shapes on net (Short Locator)*

Examples

```
envGetVal("layoutXL" "extractShortLocatorOtherShapesColor")
envSetVal("layoutXL" "extractShortLocatorOtherShapesColor" 'string "red")
```

Related Topics

Short Locator Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

extractShortLocatorShowInstancePins

```
layoutXL extractShortLocatorShowInstancePins boolean { t | nil }
```

Description

Displays a diamond shape on the canvas to represent each instance pinFig.

The default is `nil`.

GUI Equivalents

Command:	Connectivity – Short Locator – General (tab)
Field:	*Show instance pins* (Short Locator Form)

Examples

```
envGetVal("layoutXL" "extractShortLocatorShowInstancePins")
envSetVal("layoutXL" "extractShortLocatorShowInstancePins" 'boolean t)
envSetVal("layoutXL" "extractShortLocatorShowInstancePins" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

extractShortLocatorShowPinsAndLabels

```
layoutXL extractShortLocatorShowPinsAndLabels boolean { t | nil }
```

Description

Displays a diamond shape on the canvas to represent each pin and label.

The default is `nil`.

GUI Equivalents

Command:	Connectivity – Short Locator – General (tab)
Field:	*Show pins and labels* (Short Locator Form)

Examples

```
envGetVal("layoutXL" "extractShortLocatorShowPinsAndLabels")
envSetVal("layoutXL" "extractShortLocatorShowPinsAndLabels" 'boolean t)
envSetVal("layoutXL" "extractShortLocatorShowPinsAndLabels" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

extractShortLocatorShowVirtualLabelNetOnShapes

```
layoutXL extractShortLocatorShowVirtualLabelNetOnShapes boolean { t | nil }
```

Description

Displays a label with the name of the net associated with the virtual label.

The default is `nil`.

GUI Equivalents

Command:	Connectivity – Short Locator – Preferences (tab)
Field:	*Show net for virtual labels* (Short Locator Form)

Examples

```
envGetVal("layoutXL" "extractShortLocatorShowVirtualLabelNetOnShapes")
envSetVal("layoutXL" "extractShortLocatorShowVirtualLabelNetOnShapes" 'boolean t)
envSetVal("layoutXL" "extractShortLocatorShowVirtualLabelNetOnShapes" 'boolean
nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

extractShortLocatorTrueColor

```
layoutXL extractShortLocatorTrueColor boolean { t | nil }
```

Description

Displays shapes corresponding to the selected display mode in their true color.

The default is `nil`.

GUI Equivalents

Command:	Connectivity – Short Locator – General (tab)
Field:	*True color* (Short Locator Form)

Examples

```
envGetVal("layoutXL" "extractShortLocatorTrueColor")
envSetVal("layoutXL" "extractShortLocatorTrueColor" 'boolean t)
envSetVal("layoutXL" "extractShortLocatorTrueColor" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

extractShortLocatorUseInstancePins

```
layoutXL extractShortLocatorUseInstancePins boolean { t | nil }
```

Description

Lets instance pinFigs be considered as sticky points for the shorted nets, in addition to pins, labels, and virtual labels.

The default is `nil`.

GUI Equivalents

Command:	Connectivity – Short Locator – General (tab)
Field:	*Use instance pins* (Short Locator Form)

Examples

```
envGetVal("layoutXL" "extractShortLocatorUseInstancePins")
envSetVal("layoutXL" "extractShortLocatorUseInstancePins" 'boolean t)
envSetVal("layoutXL" "extractShortLocatorUseInstancePins" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

extractShortLocatorVirtualLabelNetOnShapesFontHeight

```
layoutXL extractShortLocatorVirtualLabelNetOnShapesFontHeight float
     floating_point_number
```

Description

Specifies the font size for the net used to display the virtual labels.

The default is `1.0`.

GUI Equivalents

Command:	Connectivity – Short Locator – General (tab)
Field:	*Show net for virtual labels – Font size* (Short Locator Form)

Examples

```
envGetVal("layoutXL" "extractShortLocatorVirtualLabelNetOnShapesFontHeight")
envSetVal("layoutXL" "extractShortLocatorVirtualLabelNetOnShapesFontHeight"
'float 1.0)
envSetVal("layoutXL" "extractShortLocatorVirtualLabelNetOnShapesFontHeight"
'float 2.2)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

extractShowMustConnectMarkers

This environment variable has been replaced by extractVerifyMustConnectMarkers.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

extractShowUnimplementedInstTerms

This environment variable has been replaced by extractVerifyUnimplementedInstTerms.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

extractShowWeakMarkers

This environment variable has been replaced by extractVerifyWeakMarkers.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

fingerSplit

```
layoutXL fingerSplit boolean { t | nil }
```

Description

Controls whether each finger of a schematic device will be placed in the layout as a separate device.

The default is `nil`.

GUI Equivalents

Command:	Options – Connectivity– Generation
Field:	*Split fingered devices* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "fingerSplit")
envSetVal("layoutXL" "fingerSplit" 'boolean t)
envSetVal("layoutXL" "fingerSplit" 'boolean nil)
```

Related Topics

Splitting Fingered Devices

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

fingerSplitSetWidth

```
layoutXL fingerSplitSetWidth boolean { t | nil }
```

Description

During a finger split, sets the width to the specified value and triggers the appropriate CDF callbacks.

The default is `nil`.

GUI Equivalents

None

Examples

```
envGetVal("layoutXL" "fingerSplitSetWidth")
envSetVal("layoutXL" "fingerSplitSetWidth" 'boolean t)
envSetVal("layoutXL" "fingerSplitSetWidth" 'boolean nil)
```

Related Topics

Splitting Fingered Devices

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

flattenHierNameSep

```
layoutXL flattenHierNameSep boolean { t | nil }
```

Description

Controls whether a hierarchical name is displayed for the flattened instances when the *Flatten* command is run in Layout XL.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "flattenHierNameSep")
envSetVal("layoutXL" "flattenHierNameSep" 'boolean t)
envSetVal("layoutXL" "flattenHierNameSep" 'boolean nil)
```

Additional Information

By default, Layout XL displays a hierarchical name for the flattened instances. The hierarchical name includes both the original name of the instance and the name it gets after flattening. The two names, however, are separated using a pipe separator (|), in the format:

```
<original instance name> | <instance name after flattening>
```

A hierarchical name for the flattened instances makes it easier to identify instances that have been flattened.

Related Topics

Flattening a Layout Instance

Transparent Layout Hierarchy Generation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

flattenKeepLabels

```
layoutXL flattenKeepLabels boolean { t | nil }
```

Description

Controls whether labels on shapes are retained when the *Flatten* command is run in Layout XL.

The default is `t`.

GUI Equivalent

Command:	*Edit – Hierarchy – Flatten*
Field:	*Keep Labels*

Example

```
envSetVal("layoutXL" "flattenKeepLabels")
envSetVal("layoutXL" "flattenKeepLabels" 'boolean t)
envSetVal("layoutXL" "flattenKeepLabels" 'boolean nil)
```

Related Topics

Hierarchical Layout Instance Flattening

Flatten Form

© 2023 Cadence Design Systems, Inc.

flattenPreserveConn

```
layoutXL flattenPreserveConn boolean { t | nil }
```

Description

Controls whether the connectivity of the flattened instances is preserved when the *Flatten* command is run in Layout XL.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "flattenPreserveConn")
envSetVal("layoutXL" "flattenPreserveConn" 'boolean t)
envSetVal("layoutXL" "flattenPreserveConn" 'boolean nil)
```

Related Topics

Flattening a Layout Instance

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

flattenPreserveUnthreadedConn

```
layoutXL flattenPreserveUnthreadedConn boolean { t | nil }
```

Description

Controls whether the unthreaded connectivity of the flattened instances is preserved when the *Flatten* command is run in Layout XL.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "flattenPreserveUnthreadedConn")
envSetVal("layoutXL" "flattenPreserveUnthreadedConn" 'boolean t)
```

Related Topics

Flattening a Layout Instance

© 2023 Cadence Design Systems, Inc.

flightLineEnable

```
layoutXL flightLineEnable boolean { t | nil }
```

Description

Controls whether draglines are displayed during the *Copy, Move*, *Stretch*, *Generate Selected From Source*, and *Generate Clone* commands. The draglines connect the pins of the object you are moving to pins of the nearest objects.

The default is `t`.



Tip

For information on how to change the display of the draglines, see colorDraglines, hideDraglinesForGlobalNets, and showDraglinesForDistantConns.

GUI Equivalents

Command:	*Edit – Copy* (Copy Form) *Edit – Basic – Move* (Move Form) *Edit – Basic – Stretch* (Stretch Form)
Field:	*Display Draglines*

Command:	*Connectivity – Generate – Selected From Source* (Generate Selected Components Form) *Connectivity – Generate – Clones* (Generate Clones Form)
Field:	*Draglines*

Examples

```
envGetVal("layoutXL" "flightLineEnable")
envSetVal("layoutXL" "flightLineEnable" 'boolean t)
envSetVal("layoutXL" "flightLineEnable" 'boolean nil)
```


© 2023 Cadence Design Systems, Inc.

Additional Information

Displaying many draglines during interactive commands can impair the performance of Layout XL. To mitigate these effects, use the `maxDragFig` environment variable to limit the number of figures that can be in a dragset.

To set `maxDragFig` interactively so that it takes effect in the current session, type the following in the CIW, where *`win`* is the window to which the limit applies.

```
win~>maxDragFig = 1000
```

To set a default value for `maxDragFig`, which will be used in all future Layout XL sessions, put the following line in your `.cdsenv` file.

```
graphic maxDragFig int 1000 nil
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

flightLineForAllMustConnectInstTerms

```
layoutXL flightLineForAllMustConnectInstTerms boolean { t | nil }
```

Description

Controls whether draglines are displayed for all or only one must-connect instance terminal for a pin.

The default is `nil`, which means draglines are displayed only for one must-connect instance terminal.

GUI Equivalents

None

Examples

```
envGetVal("layoutXL" "flightLineForAllMustConnectInstTerms")
envSetVal("layoutXL" "flightLineForAllMustConnectInstTerms" 'boolean t)
envSetVal("layoutXL" "flightLineForAllMustConnectInstTerms" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

foldEnforceWidthMatch

```
layoutXL foldEnforceWidthMatch boolean { t | nil }
```

Description

Generates the device folds with the same total width as that of the corresponding schematic device.

The default is `nil`, which allows the folds to be generated but triggers a message warning about the total fold width being different from that of the corresponding schematic device.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "foldEnforceWidthMatch")
envSetVal("layoutXL" "foldEnforceWidthMatch" 'boolean t)
envSetVal("layoutXL" "foldEnforceWidthMatch" 'boolean nil)
```

Related Topics

lxFold

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

foldFingerSplitUseSchWidth

```
layoutXL foldFingerSplitUseSchWidth boolean { t | nil}
```

Description

Specifies how the width of each fold is calculated when folding multi-fingered devices or transistors in order to split fingers. Folding creates a new device for each fold by setting the *Fingers* parameter to `1`. If set to `t`, the width of the schematic instance is divided by the number of fingers and the calculated number is applied to the width of each new device created. If set to `nil`, the width of each new device relies on the fingers callback of the PDK to set the correct width. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "foldFingerSplitUseSchWidth")
envSetVal("layoutXL" "foldFingerSplitUseSchWidth" 'boolean t)
envSetVal("layoutXL" "foldFingerSplitUseSchWidth" 'boolean nil)
```

Related topics

Folding a Multi-fingered Device to Create Split Fingers

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

foldIgnoreMfactor

```
layoutXL foldIgnoreMfactor boolean { t | nil }
```

Description

Ignores the schematic multiplication factor when generating folded devices and sets the transistor width to the effective width in the schematic.

Note: The effective schematic width is derived as the product of the mfactor and the schematic gate width.

The default is `nil`.

GUI Equivalents

Command:	*Connectivity – Generate – Folded Devices*
Field:	*Ignore mfactor* (Generate Folded Devices Form)

Examples

```
envGetVal("layoutXL" "foldIgnoreMfactor")
envSetVal("layoutXL" "foldIgnoreMfactor" 'boolean t)
envSetVal("layoutXL" "foldIgnoreMfactor" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

foldInPlace

```
layoutXL foldInPlace boolean { t | nil }
```

Description

Creates device folds at the same location on the layout canvas as the original device.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Generate – Folded Devices*
Field:	*Create In Place*

Examples

```
envGetVal("layoutXL" "foldInPlace")
envSetVal("layoutXL" "foldInPlace" 'boolean nil)
```

Related Topics

Generate Folded Devices Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

foldInPlaceDirection

```
layoutXL foldInPlaceDirection string { "Increasing" | "Decreasing" }
```

Description

Controls the direction in which the fold is created relative to the location of the instance that is selected for folding.

Note: This environment variable is dependent on `foldInPlace` being set to `t`.

The default is `"Increasing"`, which means that the new folds start at the lower left of the selection when positioning the folds for horizontal and vertical orientation.

When set to `"Decreasing"`, the new folds start at the lower right of the selection when positioning the folds for horizontal orientation, and start at the upper left of the selection when positioning the folds for vertical orientation.

If the original instance is at an orientation, such as `R90`, that can place the folds vertically, the *Create In Place* option can be used to create the new folds above or below the original instance.

GUI Equivalent

Command:	*Connectivity – Generate – Folded Devices*
Field:	*Create In Place*

Examples

```
envGetVal("layoutXL" "foldInPlaceDirection")
envSetVal("layoutXL" "foldInPlaceDirection" 'string "Decreasing")
```

Related Topics

Generate Folded Devices Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

foldKeepFolds

```
layoutXL foldKeepFolds boolean { t | nil }
```

Description

Allows selectively folding the individual folds of an instance.

The default is `nil`.

Note: The environment variable sets the default value for the optional, keepFolds parameter of the lxFold SKILL function. The environment variable does not impact the Generate Folded Devices Form command.

GUI Equivalents

None

Examples

```
envGetVal("layoutXL" "foldKeepFolds")
envSetVal("layoutXL" "foldKeepFolds" 'boolean t)
envSetVal("layoutXL" "foldKeepFolds" 'boolean nil)
```

Additional Information

Let us consider an example that demonstrates how the `foldKeepFolds` environment variable can be used to selectively fold individual folds of an instance.

Let us assume that an instance has folds `I0.1` and `I0.2` with `w = 3u`. Let us assume that with the `foldKeepFolds` environment variable set to `true`, we select `I0.2` and fold it into `2` using:

```
lxFold(geGetSelSet() 5:5 2)
```

In this case:

- The original fold, `I0.1` remains untouched.
- Fold `I0.2` is further folded into `I0.2` and `I0.3` with the width of each new fold set to `w = 1.5u`, and the new folds are placed at `5:5`.

© 2023 Cadence Design Systems, Inc.

Related Topics

lxFold

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

foldsLimit

```
layoutXL foldsLimit int integer
```

Description

Limits the number of folds to be generated for the selected devices.

The default value is `500`.

GUI Equivalent

Command:	*Connectivity – Generate – Folded Devices*
Field:	*Number of Folds* (Generate Folded Devices Form)

Examples

```
envGetVal("layoutXL" "foldsLimit")
envSetVal("layoutXL" "foldsLimit" 'int 2)
envSetVal("layoutXL" "foldsLimit" 'int 10)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

foldOptimizeSupplyAndGroundNets

```
layoutXL foldOptimizeSupplyAndGroundNets boolean { t | nil }
```

Description

Updates folds to optimise any supply and ground nets to the outside of the folded chain when an even number of folds exists.

The default is `t`.



Important

Optimizing the supply and ground nets to the outside of the folded chain can impact the original orientation and the left net. If the priority is to maintain these, the environment variable must be set to `nil`.

GUI Equivalents

None

Examples

```
envGetVal("layoutXL" "foldOptimizeSupplyAndGroundNets")
envSetVal("layoutXL" "foldOptimizeSupplyAndGroundNets" 'boolean t)
envSetVal("layoutXL" "foldOptimizeSupplyAndGroundNets" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

foldUseThreshold

```
layoutXL foldUseThreshold boolean { t | nil }
```

Description

Specifies if the *Folding Threshold* value should be used to fold the selected devices in the layout.

The default is `nil`.

GUI Equivalents

Command:	*Connectivity – Generate – Folded Devices*
Field:	*Use Threshold* (Generate Folded Devices Form)

Examples

```
envGetVal("layoutXL" "foldUseThreshold")
envSetVal("layoutXL" "foldUseThreshold" 'boolean t)
envSetVal("layoutXL" "foldUseThreshold" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

generateModgens

```
layoutXL generateModgens boolean { t | nil }
```

Description

Generates a modgen array for the selected instances when running the *Generate Selected From Source* command in Array placement mode. The row and column specifications used for generating the modgen are as defined in the Generate Selected Components Form form.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Generate – Selected From Source*
	(Generate Selected Components Form)
Field:	*Placement Mode – Array*
	Array Type – Modgen

Examples

```
envGetVal("layoutXL" "generateModgens")
envSetVal("layoutXL" "generateModgens" 'boolean t)
envSetVal("layoutXL" "generateModgens" 'boolean nil)
```

Related Topics

Array Generation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

generatePinsDisplayLimit

```
layoutXL generatePinsDisplayLimit int integer
```

Description

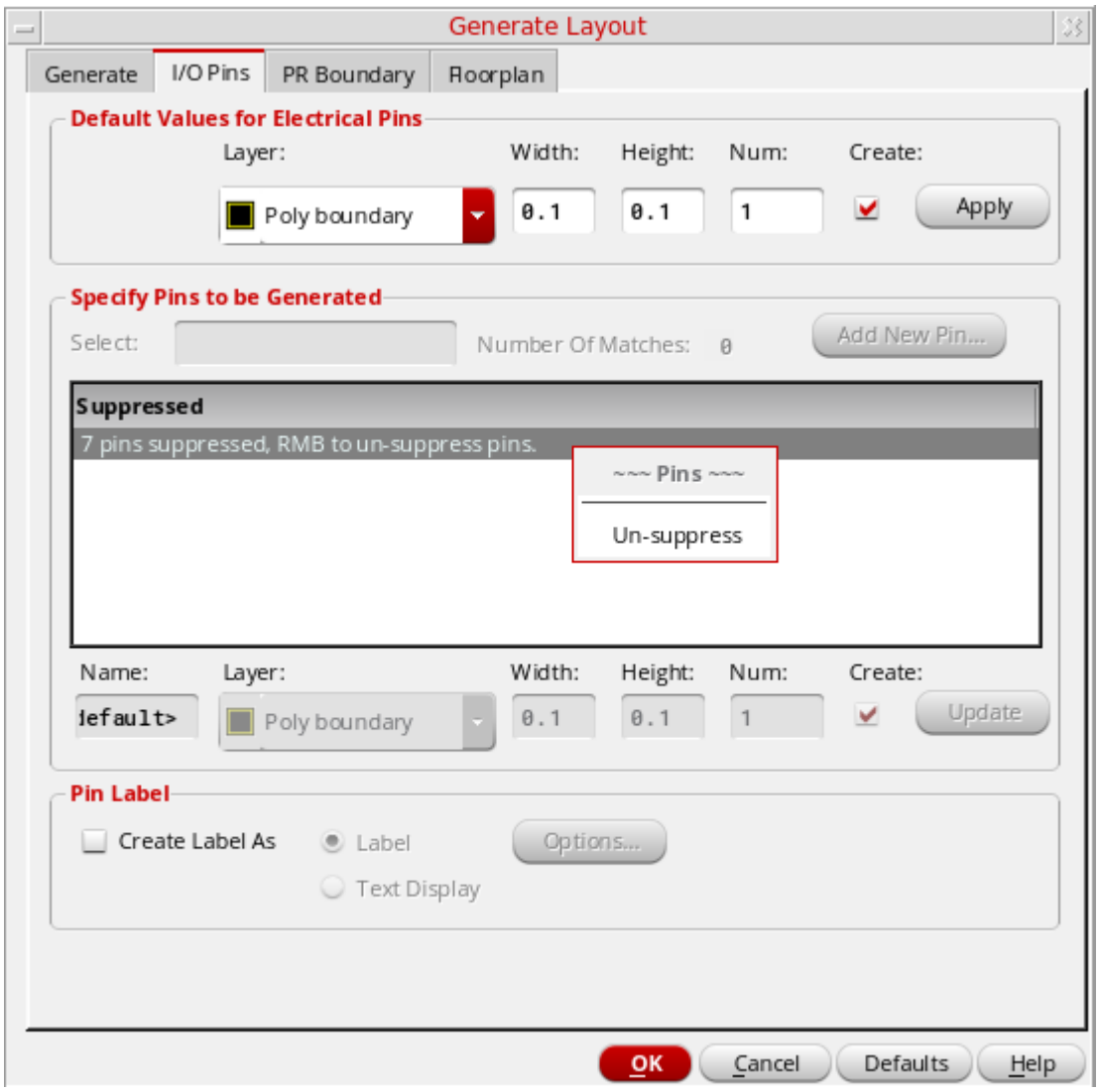
Specifies the maximum number of pins that are displayed in the *I/O Pins* tab of the Generate Layout Form and Update Components and Nets Form forms.

The default is `1000`.

If the number of pins to be displayed is above the specified limit, the display of additional pins in the forms is suppressed and the form is updated to display the number of suppressed pins.

© 2023 Cadence Design Systems, Inc.

A shortcut menu command, *Un-suppress*, is also enabled to allow the display of suppressed pins, if required, as shown in the figure below.



© 2023 Cadence Design Systems, Inc.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source – I/O Pins tab* (Generate Layout Form)
	Connectivity – Update – Components And Nets – I/O Pins tab (Update Components and Nets Form)
Field:	*Specify Pins to be Generated*

Examples

```
envGetVal("layoutXL" "generatePinsDisplayLimit")
envSetVal("layoutXL" "generatePinsDisplayLimit" 'int 550)
```

Related Topics

Generate Layout Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

generateXSpacing

`layoutXL generateXSpacing float` *`floating_point_number`*

Description

Controls the horizontal spacing between the instances generated as an array using the Generate Selected From Source command.

The default is `0.0`.

GUI Equivalent

Command:	*Connectivity – Generate – Selected From Source*
	(Generate Selected Components Form)
Field:	Placement Mode – Array
	Array Type – Instances

Examples

```
envGetVal("layoutXL" "generateXSpacing")
envSetVal("layoutXL" "generateXSpacing" 'float 0.3)
```

Related Topics

Array Generation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

generateYSpacing

`layoutXL generateYSpacing float` *`floating_point_number`*

Description

Controls the vertical spacing between the instances generated as an array using the *Generate Selected From Source* command.

The default is `0.0`.

GUI Equivalent

Command:	*Connectivity – Generate – Selected From Source*
	(Generate Selected Components Form)
Field:	*Placement Mode – Array*
	Array Type – Instances

Examples

```
envGetVal("layoutXL" "generateYSpacing")
envSetVal("layoutXL" "generateYSpacing" 'float 0.3)
```

Related Topics

Array Generation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

hardBlockColor

```
layoutXL hardBlockColor string "hilite_layer hilite_purpose"
```

Description

Specifies the highlight color for hard blocks in the layout canvas.

The default is `"hilite drawing"`.

Arguments

`hilite_layer hilite_purpose`

A list of parameter names separated by a space. The list must be enclosed in quotation marks; for example, `"hilite_layer hilite_drawing"`.

GUI Equivalent

Command:	*Floorplan* – Floorplan Global Options
Field:	*Hard block color*

Examples

```
envGetVal("layoutXL" "hardBlockColor")
envSetVal("layoutXL" "hardBlockColor" 'string "hilite drawing")
```

Related Topics

Floorplan Global Options Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

hideDraglinesForGlobalNets

```
layoutXL hideDraglinesForGlobalNets boolean { t | nil }
```

Description

Hides draglines for global nets and power and ground nets specified using the lxGroundNetNames and lxSupplyNetNames environment variables.

This environment variable is honored by the *Generate Selected From Source*, *Generate Clones*, *Move*, and *Stretch* commands.



Tip

For more information on displaying draglines, see also colorDraglines and showDraglinesForDistantConns.

GUI Equivalent

Command:	*Options – Connectivity* (*Display* tab)
Field:	*Draglines – Display Draglines – Hide global, power, and ground nets* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "hideDraglinesForGlobalNets")
envSetVal("layoutXL" "hideDraglinesForGlobalNets" 'boolean t)
envSetVal("layoutXL" "hideDraglinesForGlobalNets" 'boolean nil)
```

Additional Information

Displaying many draglines during interactive commands can impair the performance of Layout XL. To mitigate these effects, use the `maxDragFig` environment variable to limit the number of figures that can be in a dragset.

To set `maxDragFig` interactively so that it takes effect in the current session, type the following in the CIW, where *`win`* is the window to which the limit applies.

```
win~>maxDragFig = 1000
```


© 2023 Cadence Design Systems, Inc.

To set a default value for `maxDragFig`, which will be used in all future Layout XL sessions, put the following line in your `.cdsenv` file.

```
graphic maxDragFig int 1000 nil
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

hideMarkersWhenBrowserHidden

```
ab hideMarkersWhenBrowserHidden boolean { t | nil }
```

Description

Hides any highlighted markers in the canvas when you close the Annotation Browser assistant.

The default is `nil`, which means the markers remain highlighted in the canvas when you close the browser.

GUI Equivalent

Command:	*Windows – Assistants – Annotation Browser – Options* (RMB)
Field:	*Remove highlighting when browser is closed* (Annotation Browser Assistant)

Examples

```
envGetVal("ab" "hideMarkersWhenBrowserHidden")
envSetVal("ab" "hideMarkersWhenBrowserHidden" 'boolean t)
envSetVal("ab" "hideMarkersWhenBrowserHidden" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

hierarchyDepth

```
ab hierarchyDepth int { 0 | 1 | 2 ... 32 }
```

Description

Specifies the number of levels of hierarchy searched for markers. The value must lie in the range from 0 through 32.

The default is `3`.

GUI Equivalent

Command:	*Windows – Assistants – Annotation Browser – Options* (RMB)
Field:	*Hierarchy depth* (Annotation Browser Assistant)

Examples

```
envGetVal("ab" "hierarchyDepth")
envSetVal("ab" "hierarchyDepth" 'int 3)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

hierMfactorNames

`layoutXL hierMfactorNames string "`*`property_names_for_hier_mfactor_gen`*`"`

Description

Lists the names of schematic properties used to specify that hierarchical mfactors will be generated for transistors. The property names must each be separated by a space and enclosed in quotation marks, such as `"m M"`.

The default is `""`, which means no schematic properties will generate hierarchical mfactors.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "hierMfactorNames")
envSetVal("layoutXL" "hierMfactorNames" 'string "m M")
```

Related Topics

useConstantAreaStretch

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

hierSummaryColumns

```
layoutXL hierSummaryColumns string "column_names"
```

Description

Controls the columns displayed and their order in the summary report issued after running the *Update Binding* or XL Compliance commands.

To display all the columns in the report, the column names must be specified using the environment variable. Supported column names are: `Lib`, `Cell`, `View`, `Leaf`, `Ignored`, `ConnRef`, `Bound_Instances`, `Bound_Terminals`, `Bound_Nets`, `Unbound_Instances`, `Unbound_Terminals`, `Unbound_Nets`, `Ungenerated_Instances`, `Ungenerated_Terminals`, `Ungenerated_Nets`, `Connectivity_Differences`, `Master_Differences`, `Parameter_Differences`, `%Terminals`, `%Instances`, `%Nets`, `%Compliance`, `Path`, `Owner`

You can use the user-defined function `lxUserDefinedGetCellViewOwner` to find the owner of a cellview. If you have defined this function, the return value of the function will be displayed in the `Owner` column of the summary table. For example:

```
procedure(lxUserDefinedGetCellViewOwner(cv)
    cv~>ownerPropName
)
```

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "hierSummaryColumns")
```

```
envSetVal("layoutXL" "hierSummaryColumns" 'string "Lib Cell View Leaf Ignored
Connref %Terminals")
```

```
envSetVal("layoutXL" "hierSummaryColumns" 'string "Bound_Terminals Bound_Instances
Bound_Nets Unbound_Terminals Unbound_Instances Unbound_Nets Ungenerated_Terminals
Ungenerated_Instances Ungenerated_Nets Connectivity_Differences
Master_Differences Parameter_Differences")
```

Related Topics

Design Check Against Source

Design Check for Layout XL Compliance

© 2023 Cadence Design Systems, Inc.

Components and Nets Updates

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

hierSummaryLogFileName

`layoutXL hierSummaryLogFileName string "`*`logFileName`*`"`

Description

Controls whether the following SKILL functions print their summary information in a separate log file:

- `lxHierCheck`
- `lxHierCheckAgainstSource`
- `lxHierUpdateComponentsAndNets`
- `lxUpdateBinding` (when run with `?hierarchical` argument set to `t`)

The default is `""`, which means that the summary information is printed only to the default Virtuoso log file. When a log file name is specified, the summary information is also printed to the specified log file.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "hierSummaryLogFileName")
envSetVal("layoutXL" "hierSummaryLogFileName" 'string "./hierCheck.log")
```

Related Topics

Design Check Against Source

Design Check for Layout XL Compliance

Components and Nets Updates

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

highlightColors

```
ab highlightColors string "highlight_color_packet"
```

Description

Specifies the packets to be used for highlighting markers in the Annotation Browser. If a non-existing packet is specified, the Annotation Browser issues a warning message and resets the environment variable to the default highlight packet list.

Note: Any changes made to the value of the environment variable only impact the new windows.

The default is "`hilite hilite1 hilite2 hilite3 hilite4 hilite5 hilite6 hilite7 hilite8 hilite9`".

GUI Equivalent

None

Examples

```
envGetVal("ab" "highlightColors")
envSetVal("ab" "highlightColors" 'string "halo halo1 halo2 halo3")
```

Related Topics

Annotation Browser Assistant

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

highlightedIsSelectable

```
ab highlightedIsSelectable boolean { t | nil }
```

Description

Specifies that markers that are highlighted in the canvas can be selected even if marker selection has been switched off.

The default is `t`.

GUI Equivalent

Command:	*Windows – Assistants – Annotation Browser*
Field:	*Make Highlighted Markers Selectable* (toolbar button)
	(Annotation Browser Assistant)

Examples

```
envGetVal("ab" "highlightedIsSelectable")
envSetVal("ab" "highlightedIsSelectable" 'boolean t)
envSetVal("ab" "highlightedIsSelectable" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

highlightObjRMB

```
layoutXL highlightObjRMB boolean { t | nil }
```

Description

Displays the right-mouse button (RMB) menu for the highlighted object when there is no other object already selected in the canvas or the Navigator assistant, and when no layout command, such as *Copy* or *Move*, is active.

The default is `nil`.

Note:

- The RMB for the highlighted object displays just by hovering the mouse over the object, and does not need the object to be pre-selected.
  - The highlighted object can be any layout object, such as an instance, a pin, a shape, a via, a clone, a mosaic, a modgen, or a group.
- The environment variable is supported in all Virtuoso Layout Suite tiers.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "highlightObjRMB")
envSetVal("layoutXL" "highlightObjRMB" 'boolean t)
envSetVal("layoutXL" "highlightObjRMB" 'boolean nil)
```

Related Topics

Annotation Browser Assistant

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

highlightPotentialSwap

```
layoutXL highlightPotentialSwap boolean { t | nil }
```

Description

Highlights the instances that are selected for swapping. The default value is `t`.

By default, the selected instances are highlighted in yellow. Use the highlightPotentialSwapColor environment variable to change the highlight color.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "highlightPotentialSwap")
envSetVal("layoutXL" "highlightPotentialSwap" 'boolean t)
envSetVal("layoutXL" "highlightPotentialSwap" 'boolean nil)
```

Related Topics

Performing Assisted Placement

Swapping Components

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

highlightPotentialSwapColor

`layoutXL highlightPotentialSwapColor string "`*`highlight_color`*`"`

Description

Controls the color of the highlights on instances that are selected for swapping during assisted move. Valid values are `blue`, `purple`, `red`, `orange`, `cyan`, `lime`, `magenta`, `pink`, and `yellow`.

Default is `"yellow"`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "highlightPotentialSwapColor")
envSetVal("layoutXL" "highlightPotentialSwapColor" 'string "blue")
```

Related Topics

Performing Assisted Placement

Swapping Components

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

highlightShapes

```
layoutXL highlightShapes boolean { t | nil }
```

Description

Cross-highlights trimmed shapes on transparent nets in the layout.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "highlightShapes")
envSetVal("layoutXL" "highlightShapes" 'boolean t)
envSetVal("layoutXL" "highlightShapes" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

ignoreTermsWithNonRoutingLabels

```
layoutXL ignoreTermsWithNonRoutingLabels boolean { t | nil }
```

Description

Causes the following Layout XL commands to ignore instances of cells with missing terminals that have labels on non-routing layers:

- *Check Against Source*
- *Update Binding*
- *XL Compliance* (`lxCheck()` and `lxHierCheck()`)

Note: Running XL-compliance checks on the cells with labels on non-routing layers results in ungenerated terminals.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "ignoreTermsWithNonRoutingLabels")
envSetVal("layoutXL" "ignoreTermsWithNonRoutingLabels" 'boolean t)
envSetVal("layoutXL" "ignoreTermsWithNonRoutingLabels" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

incrementalCas

```
layoutXL incrementalCas boolean { t | nil }
```

Description

Controls whether incremental Check Against Source is enabled. If enabled, markers are interactively checked and created as changes occur.

The default is `nil`, which means incremental Check Against Source is not enabled.

GUI Equivalent

Toolbar: *Options*

Button: 

Examples

```
envGetVal("layoutXL" "incrementalCas")
envSetVal("layoutXL" "incrementalCas" 'boolean t)
```

Related Topics

Design Check Against Source

incrementalCasLimit

© 2023 Cadence Design Systems, Inc.

incrementalCasLimit

```
layoutXL incrementalCasLimit int integer
```

Description

Controls the number of objects that can be processed when incremental Check Against Source is enabled. The default is `100`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "incrementalCasLimit")
envSetVal("layoutXL" "incrementalCasLimit" 'int 150)
```

Related Topics

incrementalCas

Design Check Against Source

© 2023 Cadence Design Systems, Inc.

incompleteNetFilter

```
ab incompleteNetFilter boolean { t | nil }
```

Description

Specifies whether an incomplete net filter needs to be set by default in the Annotation Browser Connectivity tab.

The default is `t`, which means that the default incomplete net filter is applied and that it is driven by the incompleteNetFilterMode and incompleteNetFilterDefaultList environment variables.

Note: The `incompleteNetFilter` environment variable is supported only when the window edit cellview has no filter already applied. See the *Save To Cellview* option in Incomplete Net Filter form..

GUI Equivalent

Command:	*Right-click the Annotation Browser Connectivity tab browser pane*
Field:	*Incomplete net filter (*shortcut menu option*)*
	(Annotation Browser Assistant)

Examples

```
envGetVal("ab" "incompleteNetFilter")
envSetVal("ab" "incompleteNetFilter" 'boolean t)
envSetVal("ab" "incompleteNetFilter" 'boolean nil)
```

Related Topics

Incomplete Net Filter Form

Setting an Incomplete Net Filter

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

incompleteNetFilterDefaultList

```
ab incompleteNetFilterDefaultList string ""
```

Description

Specifies the default list of nets to be used for setting the incomplete net filters in the Annotation Browser Connectivity tab.

This environment variable is supported only when:

- The incompleteNetFilter environment variable is set to `t`.
- The window edit cellview has no filter already applied. See the *Save To Cellview* option in Incomplete Net Filter form.

Note: This environment variable accepts wildcards.

The default is `""`.

GUI Equivalent

Command:	*Right-click the Annotation Browser Connectivity tab browser pane – Incomplete net filter (form)*
Field:	*Defaults (button)*
	(Annotation Browser Assistant)

Examples

```
envGetVal("ab" "incompleteNetFilterDefaultList")
envSetVal("ab" "incompleteNetFilterDefaultList" 'string "net1 net2 net<*>")
```

Related Topics

Incomplete Net Filter Form

Setting an Incomplete Net Filter

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

incompleteNetFilterMode

```
ab incompleteNetFilterMode cyclic { "Hide" | "Show" }
```

Description

Controls whether the default behavior of the incomplete net filters in the Annotation Browser Connectivity tab shows or hides the nets specified using the incompleteNetFilterDefaultList environment variable.

This environment variable is supported only when:

- The incompleteNetFilter environment variable is set to `t`.
- The window edit cellview has no filter already applied. See the *Save To Cellview* option in Incomplete Net Filter form..

The default is `"Hide"`.

GUI Equivalent

Command:	*Right-click the Annotation Browser Connectivity tab browser pane – Incomplete net filter (form)*
Field:	*Filter Mode*
	(Annotation Browser Assistant)

Examples

```
envGetVal("ab" "incompleteNetFilterMode")
envSetVal("ab" "incompleteNetFilterMode" 'cyclic "Show")
envSetVal("ab" "incompleteNetFilterMode" 'cyclic "Hide")
```

Related Topics

Incomplete Net Filter Form

Setting an Incomplete Net Filter

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

inductanceParamNames

```
layoutXL inductanceParamNames string "list_of_parameters"
```

Description

Lists the names of the schematic parameters used to specify inductance.

The default is `"l L"`.

The inductance value must be split among the generated series-connected devices during the following *Connectivity* commands.

- *Generate All From Source*
- *Check Against Source*
- *Update Layout Parameters*.

Layout XL checks each schematic instance for one of the listed parameter names and updates the inductance values of the matching layout parameters accordingly.

Arguments

list_of_parameters

A list of parameter names each separated by a space. The list must be enclosed in quotation marks; for example,

`"l L"`

GUI Equivalent

Command:	*Options – Connectivity* (*Parameters* tab)
Field:	*Inductance* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "inductanceParamNames")
envSetVal("layoutXL" "inductanceParamNames" 'string "l L")
```


© 2023 Cadence Design Systems, Inc.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

infoWindow

```
layoutXL infoWindow boolean { t | nil }
```

Description

Controls where messages issued by the *Check Against Source*, *Update Layout Parameters*, and *Update Schematic Parameters* commands are displayed. When set to `t`, messages are displayed in a separate Info window. When set to `nil`, messages are sent to the CIW.

The default is `t`.

GUI Equivalent

Command:	*Options – Connectivity* (*Display* tab)
Field:	Redirect Layout XL messages to separate Info window (Connectivity Form)

Examples

```
envGetVal("layoutXL" "infoWindow")
envSetVal("layoutXL" "infoWindow" 'boolean t)
envSetVal("layoutXL" "infoWindow" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

infoWindowPos

```
layoutXL infoWindowPos string "pair_of_coordinates"
```

Description

Specifies the position and size of the information window on the screen. It is used by the autoArrange environment variable when rearranging the Layout XL windows on startup.

By default, the information window is positioned at the top left corner of the screen.

Arguments

`pair_of_coordinates`	A pair of coordinates enclosed in quotation marks specifying the lower left and upper right corners of the information window For example, “((0.0 0.0) (600.0 300.0))"

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "infoWindowPos")
envSetVal("layoutXL" "infoWindowPos" 'string "((1.0 4.0) (480.0 275.0))")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initAreaFuncName

`layoutXL initAreaFuncName string "`*`area_estimation_method`*`"`

Description

Sets the method of area estimation—`PR Boundary Based` or `BBox Based`—to be used during layout generation.

The default is `"PR Boundary Based"`.

Arguments

None

GUI Equivalent

Command:	*Connectivity – Generate — All From Source*
Field:	PR Boundary based (*PR Boundary* tab)
Command:	*Connectivity – Generate — All From Source*
Field:	BBox based (*PR Boundary* tab)

Examples

```
envGetVal("layoutXL" "initAreaFuncName")
envSetVal("layoutXL" "initAreaFuncName" 'string "BBox Based")
```

Related Topics

Area Estimation Function Registration to Adjust a Virtual Hierarchy Boundary

Area Estimation Function Registration to Adjust a Soft Block PR Boundary

© 2023 Cadence Design Systems, Inc.

initAspectRatio

`layoutXL initAspectRatio float` *`floating_point_number`*

Description

Specifies the width-to-height aspect ratio of the design to be generated or updated by the *Generate All From Source* and *Update Components And Nets* commands, or when defining soft blocks in the Configure Physical Hierarchy window.

The default is `1.0`, which specifies a square boundary. An aspect ratio of 0.5 specifies a boundary twice as high as it is wide. A value of 2 specifies a boundary twice as wide as it is high.

Arguments

`floating_point_number`

A float specifying the width-to-height ratio for the design boundary; for example,

`1.0`

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (PR Boundary tab) (Generate Layout Form)
	Connectivity – Update – Components And Nets (Update Components and Nets Form)
	Launch – Configure Physical Hierarchy (in Soft Block mode) (Configure Physical Hierarchy Window)
Field:	*Aspect Ratio (W/H)*

Examples

```
envGetVal("layoutXL" "initAspectRatio")
envSetVal("layoutXL" "initAspectRatio" 'float 1.0)
```


© 2023 Cadence Design Systems, Inc.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initAspectRatioOption

```
layoutXL initAspectRatioOption string { "Aspect Ratio W/H" | "prBoundary Width"
     | "prBoundary Height" }
```

Description

Specifies which boundary option is used by the Generate Layout Form and Update Components and Nets Form forms, or when defining soft blocks in the Configure Physical Hierarchy Window window.

The default is `"Aspect Ratio W/H"`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*PR Boundary* tab) (Generate Layout Form)
	Connectivity – Update – Components And Nets (*PR Boundary* tab) (Update Components and Nets Form)
	Launch – Configure Physical Hierarchy (in Soft Block mode) (Soft Block Mode)
Field:	*Aspect Ratio (W/H)*

Examples

```
envGetVal("layoutXL" "initAspectRatioOption")
envSetVal("layoutXL" "initAspectRatioOption" 'string "Aspect Ratio W/H")
envSetVal("layoutXL" "initAspectRatioOption" 'string "prBoundary Width")
envSetVal("layoutXL" "initAspectRatioOption" 'string "prBoundary Height")
```

Additional Information

To prepopulate the Generate Layout form with *Utilization%* and prBoundary height, set the following environment variables:

```
envSetVal("layoutXL" "initEstimateArea" 'boolean t)
envSetVal("layoutXL" "initAspectRatioOption" 'string "prBoundary Height")
envSetVal("layoutXL" "initPrBoundaryH" 'float 0.24)
```

Note: Set an appropriate value for the `initPrBoundaryH` environment variable. The value specified in the example above is used for representational purposes only.

© 2023 Cadence Design Systems, Inc.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initCreateBoundary

```
layoutXL initCreateBoundary boolean { t | nil }
```

Description

Controls whether Layout XL creates a place and route boundary in the layout canvas.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*Generate* tab)
Field:	*PR Boundary*

Examples

```
envGetVal("layoutXL" "initCreateBoundary")
envSetVal("layoutXL" "initCreateBoundary" 'boolean t)
envSetVal("layoutXL" "initCreateBoundary" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initCreateInstances

```
layoutXL initCreateInstances boolean { t | nil }
```

Description

Specifies that Layout XL layout editor is to create instances during the *Generate All From Source* command.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*Generate* tab)
Field:	*Instances*
	(Generate Layout Form)

Examples

```
envGetVal("layoutXL" "initCreateInstances")
envSetVal("layoutXL" "initCreateInstances" 'boolean t)
envSetVal("layoutXL" "initCreateInstances" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initCreateMTM

```
layoutXL initCreateMTM boolean { t | nil }
```

Description

Specifies that Layout XL is to preserve user-defined bindings during *Generate All From Source* and *Update Components And Nets*.

The default is `nil`.



Important

This option preserves only user-defined one-to-one, many-to-many, many-to-one, and one-to-many device correspondence defined in the Define Device Correspondence Form form. It does not report missing devices or shapes within a bound group. For more information, see Device Correspondence Updates.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*Generate* tab) (Generate Layout Form)
	Connectivity – Update – Components And Nets (*Update* tab) (Update Components and Nets Form)
Field:	*Preserve – User-Defined Bindings*

Examples

```
envGetVal("layoutXL" "initCreateMTM")
envSetVal("layoutXL" "initCreateMTM" 'boolean t)
envSetVal("layoutXL" "initCreateMTM" 'boolean nil)
```

Additional Information

The Generate Layout form uses this environment variable to set the value for its *Preserve User-Defined Bindings* option. If you change the setting for *Preserve user-Defined Bindings* in the form, the value of the environment variable changes as well. Conversely, the Update Components and Nets Form form always initializes with *Preserve User-Defined Bindings* switched on and never sets the value of `initCreateMTM`.

© 2023 Cadence Design Systems, Inc.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initCreatePadPins

```
layoutXL initCreatePadPins boolean { t | nil }
```

Description

Specifies that Layout XL generates both pins and pads for schematic pins that are connected to I/O pads (cells of type `pad`, `padSpacer`, or `padAreaIO`).

The default is `t`.



Tip

To generate only pads but no pins, set this environment variable to `nil`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*I/O Pins* tab) (Generate Layout Form)
	Connectivity – Update – Components And Nets (*I/O Pins* tab) (Update Components and Nets Form)
Field:	*Except Pad Pins*

Examples

```
envGetVal("layoutXL" "initCreatePadPins")
envSetVal("layoutXL" "initCreatePadPins" 'boolean t)
envSetVal("layoutXL" "initCreatePadPins" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initCreatePins

```
layoutXL initCreatePins boolean { t | nil }
```

Description

Specifies that Layout XL is to create pins.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*Generate* tab)
Field:	*I/O Pins* (Generate Layout Form)

Examples

```
envGetVal("layoutXL" "initCreatePins")
envSetVal("layoutXL" "initCreatePins" 'boolean t)
envSetVal("layoutXL" "initCreatePins" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initCreateSnapBoundary

```
layoutXL initCreateSnapBoundary boolean { t | nil }
```

Description

Controls whether Layout XL creates a rectangular snap boundary enclosing the generated PR boundary. (You can generate a snap boundary only if you also generate a PR boundary.)

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*Generate* tab)
Field:	*Snap Boundary* (Generate Layout Form)

Examples

```
envGetVal("layoutXL" "initCreateSnapBoundary")
envSetVal("layoutXL" "initCreateSnapBoundary" 'boolean t)
envSetVal("layoutXL" "initCreateSnapBoundary" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initDoFolding

```
layoutXL initDoFolding boolean { t | nil }
```

Description

Specifies that Layout XL is to divide prepared transistors into two or more folds.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*Generate* tab)
Field:	*Fold* (Generate Layout Form)
Command:	*Connectivity – Generate – Selected From Source*
Field:	*Fold* (Generate Selected Components Form)

Examples

```
envGetVal("layoutXL" "initDoFolding")
envSetVal("layoutXL" "initDoFolding" 'boolean t)
envSetVal("layoutXL" "initDoFolding" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initDoStacking

```
layoutXL initDoStacking boolean { t | nil }
```

Description

Specifies that Layout XL is to abut MOS transistors in a specific order.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*Generate* tab)
Field:	*Chain* (Generate Layout Form)
Command:	*Connectivity – Generate – Selected From Source*
Field:	*Chain* (Generate Selected Components Form)

Examples

```
envGetVal("layoutXL" "initDoStacking")
envSetVal("layoutXL" "initDoStacking" 'boolean t)
envSetVal("layoutXL" "initDoStacking" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initEstimateArea

```
layoutXL initEstimateArea boolean { t | nil }
```

Description

Determines whether Layout XL estimates the size of the design boundary based on the boundary area estimation options during the *Generate All From Source* and *Update Components And Nets* commands, and when defining soft blocks in the Configure Physical Hierarchy window. To support defining soft blocks in the Configure Physical Hierarchy window, the environment variable feeds the *Boundary* tab with the Area Estimator values.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*PR Boundary* tab) (Generate Layout Form)
	Connectivity – Update – Components And Nets (*PR Boundary* tab) (Update Components and Nets Form)
	Launch – Configure Physical Hierarchy (in Soft Block mode) (Soft Block Mode)
Field:	*Aspect Ratio (W/H)*

Examples

```
envGetVal("layoutXL" "initEstimateArea")
envSetVal("layoutXL" "initEstimateArea" 'boolean t)
envSetVal("layoutXL" "initEstimateArea" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initGlobalNetPins

```
layoutXL initGlobalNetPins boolean { t | nil }
```

Description

Determines whether the Layout XL layout editor *Generate All From Source* command creates layout pins for the global nets in the schematic.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*Generate* tab) (Generate Layout Form)
	Connectivity – Update – Components And Nets (*Update* tab) (Update Components and Nets Form)
Field:	*Create Label As*

Examples

```
envGetVal("layoutXL" "initGlobalNetPins")
envSetVal("layoutXL" "initGlobalNetPins" 'boolean t)
envSetVal("layoutXL" "initGlobalNetPins" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initIOLabelType

```
layoutXL initIOLabelType string { "Label" | "Text Display" | "None" }
```

Description

Specifies the type of label that Layout XL generates for I/O pins. When set to `None`, no pin label is created. This setting is honored by the *Generate All From Source, Generated Selected From Source*, and *Update Components And Nets* commands.

The default is `"Label"`.

Important

For `thi`s environment variable to take effect, you must also set the createPinLabel environment variable to `t`.

Arguments

`Label`	Creates pin labels on the layer specified by initIOPinLayer.
`Text Display`	Creates text displays on the layer specified by initIOPinLayer.
`None`	No pin labels or text displays are created.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*I/O Pins* tab) *Connectivity – Generate – Selected From Source* (Generate Layout Form)
	Connectivity – Update – Components And Nets (*I/O Pins* tab) (Update Components and Nets Form)
Field:	*Create Label As*

Examples

```
envGetVal("layoutXL" "initIOLabelType")
envSetVal("layoutXL" "initIOLabelType" 'string "Label")
envSetVal("layoutXL" "initIOLabelType" 'string "Text Display")
envSetVal("layoutXL" "initIOLabelType" 'string "None")
```


© 2023 Cadence Design Systems, Inc.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initIOPinLayer

```
layoutXL initIOPinLayer string "layer_purpose_pair"
```

Description

Specifies which of the layers and purposes defined in the technology file is used as the pin layer-purpose pair (LPP) when using the *Generate All From Source* and *Update Components And Nets* commands to generate and update pins, or when defining soft block pins using the Configure Physical Hierarchy window in the Soft Block mode.

If the environment variable is not set, the default pin LPP is selected based on the following precedence:

1. The current layer selected in the Layer Assistant is considered as the default pin LPP, assuming the current layer is defined as a valid layer in the technology file (see validLayers for more information).
2. Else, the first layer in the `validLayers` constraint section of the technology file that has a “pin” purpose is selected as the default pin LPP.
3. If a `validLayer` layer with a “pin” purpose does not exist, the first layer in the `validLayers` constraint with purpose “drawing” is selected as the default pin LPP.

Note: For CPH soft block mode, the default value of the pin layers on the I/O Pins tab is read from the environment variable, `initIOPinLayer`.

Important

To maintain a default LPP across the technology libraries that you use, Cadence recommends that you set the default pin LPP by using the `initIOPinLayer` environment variable.

© 2023 Cadence Design Systems, Inc.

Arguments

layer_purpose_pair

Specifies the layer and purpose on which pins are created. The string must be enclosed in quotation marks; for example,

```
"metal1 drawing"
```

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*I/O Pins* tab) (Generate Layout Form)
	Connectivity – Update – Components And Nets (*I/O Pins* tab) (Update Components and Nets Form)
	Launch – Configure Physical Hierarchy (in Soft Block mode) (Soft Block Mode)
Field:	*Layer*

Examples

```
envGetVal("layoutXL" "initIOPinLayer")
envSetVal("layoutXL" "initIOPinLayer" 'string "Metal1 drawing")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initIOPinLayerNames

```
layoutXL initIOPinLayerNames string "layer_names"
```

Description

- Specifies the layers that need to be listed in the *Layer* column in the *IO Pins* tab of the Configure Physical Hierarchy Window window (Soft Block mode).
- Filters the layers in the default and selected pin layer lists in the *IO Pins* tab of the Generate Layout Form form.
  - The Layer Purpose Pair (LPP) list on the Generate Layout form is filtered based on the values specified for the initIOPinLayerNames and initIOPinPurposeNames environment variables. This reduces the displayed LPP list, showing only the values that are supported.
  - For a layer purpose pair to be listed on the form, both the layer name and the purpose name must match. If the filter finds no matches, all the valid layer-purpose pair combinations are displayed.

The default value is `""`, which means the layers are displayed from `virtuosoDefaultExtractorSetup`.

Arguments

layer_names

Specifies a list of layers that need to be listed in the *Layer* column.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "initIOPinLayerNames")
envSetVal("layoutXL" "initIOPinLayerNames" 'string "Metal1 Metal2 Metal3 Poly")
```

Related Topics

Generating All Components from Source

© 2023 Cadence Design Systems, Inc.

Creating a Soft Pin

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initIOPinName

The `initIOPinName` environment variable is not required in OpenAccess designs and is no longer supported. You can remove references to it from your setup files.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initIOPinPurposeNames

```
layoutXL initIOPinPurposeNames string "purpose_names"
```

Description

Specifies the layer purposes to use on:

- The *IO Pins* tab of the Configure Physical Hierarchy Window (Soft Block mode).
- The *IO Pins* tab of the Generate Layout Form and Update Components and Nets Form form.
  - The layers are obtained from the `validLayers()` constraint and filtered based on the initIOPinLayerNames environment variable. The `initIOPinPurposeNames` environment variable then filters the purposes from the entire list and displays only the specified purposes.
  - For a layer-purpose pair to be listed on the form, it must match a valid layer-purpose pair. If the filter finds no match, the entire layer-purpose pair list is displayed.

The default value is `""`, which means all valid layer purpose pairs for the filtered layers are displayed.

Arguments

purpose_names

A list of layer purposes to use.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "initIOPinPurposeNames")
envSetVal("layoutXL" "initIOPinPurposeNames" 'string "pin drawing")
```

Related Topics

Generating All Components from Source

© 2023 Cadence Design Systems, Inc.

Creating a Soft Pin

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initPinHeight

```
layoutXL initPinHeight float floating_point_number
```

Description

Specifies the height of pins generated by the *Generate All From Source, Generated Selected From Source,* and *Update Components And Nets* commands, and when defining soft blocks in the Configure Physical Hierarchy (CPH) window. For CPH soft block, the environment variable feeds the I/O Pins tab with the default pin height values.

Note: This value is honored only if it is greater than the minimum width value obtained from the constraint lookup.

Arguments

floating_point_number

A float specifying the default height of pins generated by Layout XL; for example,

```
0.25
```

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*I/O Pins* tab) *Connectivity – Generate – Selected From Source* (Generate Layout Form) *Connectivity – Update – Components And Nets* (*I/O Pins* tab) (Update Components and Nets Form) *Launch – Configure Physical Hierarchy* (in Soft Block mode) (Soft Block Mode)
Field:	*Height*

Examples

```
envGetVal("layoutXL" "initPinHeight")
envSetVal("layoutXL" "initPinHeight" 'float 1.0)
```


© 2023 Cadence Design Systems, Inc.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initPinMultiplicity

`layoutXL initPinMultiplicity int` *`integer`*

Description

Specifies the number of pins of each type to be generated by the *Generate All From Source* and *Update Components And Nets* commands, and when defining soft blocks in the Configure Physical Hierarchy window. For CPH soft block, the environment variable feeds the I/O Pins tab with the default number of pins of each type to be generated.

The default is `1`.

Arguments

`integer`	Non-zero positive integer specifying a number of pins to be generated; for example, `1`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*I/O Pins* tab) (Generate Layout Form)
	Connectivity – Update – Components And Nets (*I/O Pins* tab) (Update Components and Nets Form)
	Launch – Configure Physical Hierarchy (in Soft Block mode) (Soft Block Mode)
Field:	*Num*, *Number*

Examples

```
envGetVal("layoutXL" "initPinMultiplicity")
envSetVal("layoutXL" "initPinMultiplicity" 'int 2)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initPinWidth

```
layoutXL initPinWidth float floating_point_number
```

Description

Specifies the width of pins generated by the *Generate All From Source, Generated Selected From Source,* and *Update Components And Nets* commands, and when defining soft blocks in the Configure Physical Hierarchy window. For CPH soft block, the environment variable feeds the I/O Pins tab with the default pin width values.

Note: This value is honored only if it is greater than the minimum width value obtained from the constraint lookup.

Arguments

floating_point_number

A float specifying the default width of pins generated by Layout XL; for example, 0.25.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*I/O Pins* tab) *Connectivity – Generate – Selected From Source* (Generate Layout Form)
	Connectivity – Update – Components And Nets (*I/O Pins* tab) (Update Components and Nets Form)
	Launch – Configure Physical Hierarchy (in Soft Block mode) (Soft Block Mode)
Field:	*Width*

Examples

```
envGetVal("layoutXL" "initPinWidth")
envSetVal("layoutXL" "initPinWidth" 'float 1.0)
```

Related Topics

List of Layout XL Environment Variables

© 2023 Cadence Design Systems, Inc.

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initPrBoundaryH

```
layoutXL initPrBoundaryH float floating_point_number
```

Description

Specifies the height of the design boundary created by the *Generate All From Source* and *Update Components And Nets* commands, and when defining soft blocks in the Configure Physical Hierarchy window. For CPH soft block mode, the environment variable feeds the Boundary tab with the default design boundary height values.

Arguments

floating_point_number

A float specifying the default height of the design boundary generated by Layout XL; for example, `8.0`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*PR Boundary* tab) (Generate Layout Form)
	Connectivity – Update – Components And Nets (*PR Boundary* tab) (Update Components and Nets Form)
	Launch – Configure Physical Hierarchy (in Soft Block mode) (Soft Block Mode)
Field:	*Height*

Examples

```
envGetVal("layoutXL" "initPrBoundaryH")
envSetVal("layoutXL" "initPrBoundaryH" 'float 471.31)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initPrBoundaryW

```
layoutXL initPrBoundaryW float floating_point_number
```

Description

Specifies the width of the design boundary created by the *Generate All From Source* and *Update Components And Nets* commands, and when defining soft blocks in the Configure Physical Hierarchy (CPH) window. For CPH soft block mode, the environment variable feeds the Boundary tab with the default design boundary width values.

Arguments

floating_point_number

A float specifying the default width of the design boundary generated by Layout XL; for example, `8.0`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*PR Boundary* tab) (Generate Layout Form)
	Connectivity – Update – Components And Nets (*PR Boundary* tab) (Update Components and Nets Form)
	Launch – Configure Physical Hierarchy (in Soft Block mode) (Soft Block Mode)
Field:	*Width*

Examples

```
envGetVal("layoutXL" "initPrBoundaryW")
envSetVal("layoutXL" "initPrBoundaryW" 'float 186.075)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

initUtilization

`layoutXL initUtilization float` *`floating_point_number`*

Description

Specifies the percentage area utilization to be used by the *Estimate Area* function in the *Generate All From Source* and *Update Components And Nets* commands, and when defining soft blocks in the Configure Physical Hierarchy window. For CPH soft block mode, the environment variable feeds the Boundary tab with the utilization percentage of the design area.

The default is `25.0`.

Arguments

`floating_point_number`

A float specifying the default area utilization for the design; for example, `25.0`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*PR Boundary* tab) (Generate Layout Form)
	Connectivity – Update – Components And Nets (*PR Boundary* tab) (Update Components and Nets Form)
	Launch – Configure Physical Hierarchy (in Soft Block mode) (Soft Block Mode)
Field:	*Utilization (%)*

Examples

```
envGetVal("layoutXL" "initUtilization")
envSetVal("layoutXL" "initUtilization" 'float 25.2)
```

Related Topics

List of Layout XL Environment Variables

© 2023 Cadence Design Systems, Inc.

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

insertChain

```
layoutXL insertChain boolean { t | nil }
```

Description

Allows the Move command to insert chains within the existing chains. Depending on the value of insertChainAbutViolation, the chains are inserted based on the `"First"` or `"Last"` valid abutment violation.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "insertChain")
envSetVal("layoutXL" "insertChain" 'boolean t)
envSetVal("layoutXL" "insertChain" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

insertChainAbutViolation

```
layoutXL insertChainAbutViolation string { "First" | "Last" }
```

Description

If insertChain is set to `t`, controls whether the Move command inserts chains within existing chains based on the `"First"` or `"Last"` valid abutment violation.

The default is `"First"`, which means chains are inserted based on the first valid abutment violation.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "insertChainAbutViolation")
envSetVal("layoutXL" "insertChainAbutViolation" 'string "First")
envSetVal("layoutXL" "insertChainAbutViolation" 'string "Last")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lamReadCdsLibs

```
layoutXL lamReadCdsLibs boolean { t | nil }
```

Description

Searches all the Cadence libraries registered in `cds.lib` to search for LAM files to read.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lamReadCdsLibs")
envSetVal("layoutXL" "lamReadCdsLibs" 'boolean t)
envSetVal("layoutXL" "lamReadCdsLibs" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lamReadFileMessage

```
layoutXL lamReadFileMessage boolean { t | nil }
```

Description

Issues a CIW message to indicate that the updated LAM file has been read by the current Layout XL session.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lamReadFileMessage")
envSetVal("layoutXL" "lamReadFileMessage" 'boolean t)
envSetVal("layoutXL" "lamReadFileMessage" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lamReadUserLibs

```
layoutXL lamReadUserLibs string "library_names"
```

Description

Searches only the specified Cadence libraries to search for LAM files to read.

The default is `""`, which means all Cadence libraries registered in `cds.lib` are searched.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lamReadUserLibs")
envSetVal("layoutXL" "lamReadUserLibs" 'string "customLibrary1")
envSetVal("layoutXL" "lamReadUserLibs" 'string "customLibrary1 myLib")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lamReuseCache

```
layoutXL lamReuseCache boolean { t | nil }
```

Description

Reads the LAM file from cache.

The default is `t`, which means the LAM file is not read all over again, even during a *Descend Edit* or an *Edit In Place* operation, resulting in better performance.

When set to `nil`, the LAM file is read again when descended into a different library or when editing in place.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lamReuseCache")
envSetVal("layoutXL" "lamReuseCache" 'boolean t)
envSetVal("layoutXL" "lamReuseCache" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

layoutWindow

```
layoutXL layoutWindow string "pair_of_coordinates"
```

Description

Specifies the position and size of the layout window on the screen.

By default, the layout window is positioned in the bottom left corner of the screen.

Arguments

pair_of_coordinates

A pair of coordinates enclosed in quotation marks specifying the lower left and upper right corners of the layout window; for example,

```
"((577.0 89.0) (1268.0 992.0))"
```

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "layoutWindow")
envSetVal("layoutXL" "layoutWindow" 'string "((435.0 52.0) (1588.0 1165.0))")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lswWindow

```
layoutXL lswWindow string "pair_of_coordinates"
```

Description

Resizes the Palette assistant when Layout XL is launched without a schematic reference.

By default, the Palette assistant is positioned at coordinates `"((0 0) (0 0))"`.

Arguments

None

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lswWindow")
envSetVal("layoutXL" "lswWindow" 'string "((0 3) (0 5))")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxAbutPassGroupIdToEvent1

```
layoutXL lxAbutPassGroupIdToEvent1 boolean { t | nil }
```

Description

Passes the abutment group ID to Abutment Event 1.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lxAbutPassGroupIdToEvent1")
envSetVal("layoutXL" "lxAbutPassGroupIdToEvent1" 'boolean t)
envSetVal("layoutXL" "lxAbutPassGroupIdToEvent1" 'boolean nil)
```

Related Topics

Abutment Function Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxAllowPseudoParallelNets

```
layoutXL lxAllowPseudoParallelNets boolean { t | nil }
```

Description

Automatically identifies nodes that qualify as pseudoparallel connections and defines them during chaining and abutment. A pseudoparallel net connects nodes that are always the same voltage, so the current does not pass through the net. Contacts are dropped automatically, if appropriate. However, if preserveTerminalContacts is switched on, the contacts are preserved.

The default is `nil`.

Important

This environment variable applies only when chaining is turned on. It does not control the automatic identification and creation of pseudoparallel nets during manual abutment. To do that, use the setPPConn environment variable.

GUI Equivalent

Command:	*Options – Connectivity* (*Generation* tab)
Field:	*Create interdigitated chains* (Connectivity Form)
Command:	*Connectivity – Generate Selected Components*
Field:	*Interdigitate Chains* (Generate Selected Components Form)

Examples

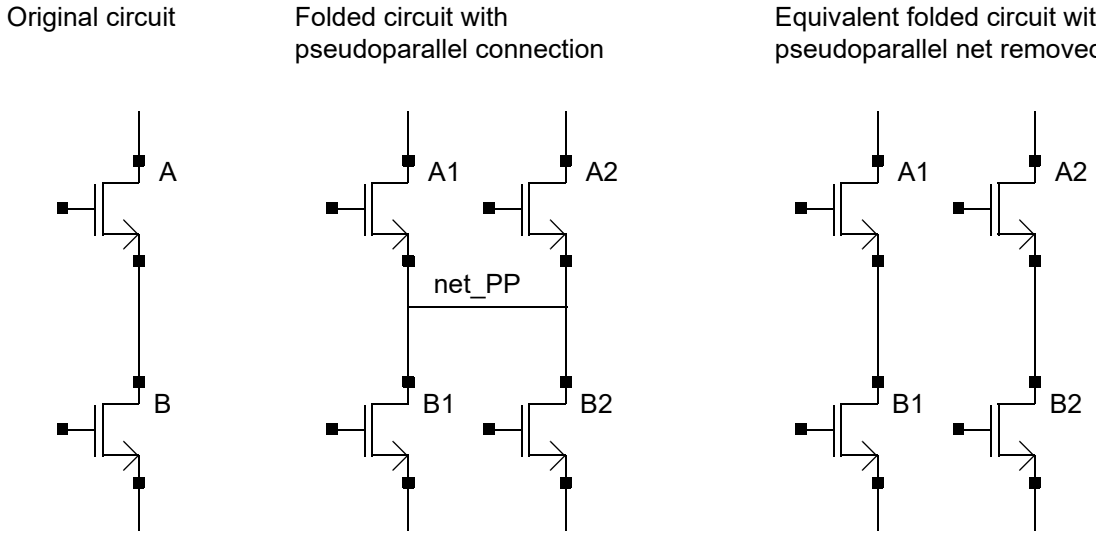
```
envGetVal("layoutXL" "lxAllowPseudoParallelNets")
envSetVal("layoutXL" "lxAllowPseudoParallelNets" 'boolean t)
envSetVal("layoutXL" "lxAllowPseudoParallelNets" 'boolean nil)
```

Additional Information

To see the significance of pseudoparallel nets, consider a symmetric series of N-transistors, A and B, tied in a series pull-down chain to ground, as shown in the following figure. Both A and B are folded into two legs, with A1, net_PP, and B1 connected in series to ground, and likewise A2 and B2 connected in series to ground, with both pairs in parallel. The nodes

© 2023 Cadence Design Systems, Inc.

between A1 and B1 and between A2 and B2 are electrically equivalent. In reality, no current passes through that connection and it need not physically be made.



For more information, see Pseudoparallel Connections.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxcuAllCellsInDesign

```
layoutXL lxcuAllCellsInDesign boolean { t | nil }
```

Description

Checks and updates all cells in the current cellview.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Check and Update*
Field:	*All cells in design*

Examples

```
envGetVal("layoutXL" "lxcuAllCellsInDesign")
envSetVal("layoutXL" "lxcuAllCellsInDesign" 'boolean t)
envSetVal("layoutXL" "lxcuAllCellsInDesign" 'boolean nil)
```

Related Topics

Application Readiness Checker Form

lxCheckAndUpdate

lxCheckAndUpdateRegister

lxCheckAndUpdateRemove

lxCheckAndUpdateSet

© 2023 Cadence Design Systems, Inc.

lxcuAllCellsInLibrary

```
layoutXL lxcuAllCellsInLibrary boolean { t | nil }
```

Description

Checks and updates all cells in the current library.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Check and Update*
Field:	*All cells in library*

Examples

```
envGetVal("layoutXL" "lxcuAllCellsInLibrary")
envSetVal("layoutXL" "lxcuAllCellsInLibrary" 'boolean t)
envSetVal("layoutXL" "lxcuAllCellsInLibrary" 'boolean nil)
```

Related Topics

Application Readiness Checker Form

lxCheckAndUpdate

lxCheckAndUpdateRegister

lxCheckAndUpdateRemove

lxCheckAndUpdateSet

© 2023 Cadence Design Systems, Inc.

lxcuApp

```
layoutXL lxcuApp string "application_name"
```

Description

Specifies the name of the application to check and update.

The default is "`Schematic v Layout`".

GUI Equivalent

Command:	*Connectivity – Check and Update*
Field:	*Application*

Examples

```
envGetVal("layoutXL" "lxcuApp")
envSetVal("layoutXL" "lxcuApp" 'string "Innovus")
```

Related Topics

Application Readiness Checker Form

lxCheckAndUpdate

lxCheckAndUpdateRegister

lxCheckAndUpdateRemove

lxCheckAndUpdateSet

© 2023 Cadence Design Systems, Inc.

lxcuBinder

```
layoutXL lxcuBinder cyclic  { "Current" | "Rebind" | "LVS" }
```

Description

Specifies the type of binding to perform during application update.

- `Current`: Uses the existing binding.
- `Rebind`: Updates using the *Update Binding* command.
- `LVS`: Rebinds using the binding information in the PVS svdb directory.

The default is "`Current`".

GUI Equivalent

Command:	*Connectivity – Check and Update*
Field:	*Binder*

Examples

```
envGetVal("layoutXL" "lxcuBinder")
envSetVal("layoutXL" "lxcuBinder" 'cyclic "Current")
```

Related Topics

Application Readiness Checker Form

lxCheckAndUpdate

lxCheckAndUpdateRegister

lxCheckAndUpdateRemove

lxCheckAndUpdateSet

© 2023 Cadence Design Systems, Inc.

lxcuCreateHTMLReport

```
layoutXL lxcuCreateHTMLReport  boolean { t | nil }
```

Description

Specifies whether to create an HTML report.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Check and Update*
Field:	*Create HTML Report*

Examples

```
envGetVal("layoutXL" "lxcuCreateHTMLReport")
envSetVal("layoutXL" "lxcuCreateHTMLReport" 'boolean t)
envSetVal("layoutXL" "lxcuCreateHTMLReport" 'boolean nil)
```

Related Topics

Application Readiness Checker Form

lxCheckAndUpdate

lxCheckAndUpdateRegister

lxCheckAndUpdateRemove

lxCheckAndUpdateSet

© 2023 Cadence Design Systems, Inc.

lxcuDetailedReturn

```
layoutXL lxcuDetailedReturn boolean { t | nil }
```

Description

Returns a list of checks as `Passed`, `Failed`, or `Needs attention` when `lxCheckAndUpdate` runs on a single cell and application.

For example,

```
list(list("Direction" "Passed")
list("Name - Nets" "Failed")
list("Parameter" "Needs attention")
list("Signal type" "Passed"))
```

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lxcuDetailedReturn")
envSetVal("layoutXL" "lxcuDetailedReturn" 'boolean t)
envSetVal("layoutXL" "lxcuDetailedReturn" 'boolean nil)
```

Related Topics

Application Readiness Checker Form

lxCheckAndUpdate

lxCheckAndUpdateRegister

lxCheckAndUpdateRemove

lxCheckAndUpdateSet

© 2023 Cadence Design Systems, Inc.

lxcuExcludeCells

```
layoutXL lxcuExcludeCells string "names_of_cells_to_exclude"
```

Description

Lists the cells to be excluded when the *All cells in library* or *All cells in library* option is selected.

The default is `""`.

GUI Equivalent

Command:	*Connectivity – Check and Update*
Field:	*Exclude Cells*

Examples

```
envGetVal("layoutXL" "lxcuExcludeCells")
envSetVal("layoutXL" "lxcuExcludeCells" 'string "triplexer")
```

Related Topics

Application Readiness Checker Form

lxCheckAndUpdate

lxCheckAndUpdateRegister

lxCheckAndUpdateRemove

lxCheckAndUpdateSet

© 2023 Cadence Design Systems, Inc.

lxcuExtract

```
layoutXL lxcuExtract boolean { t | nil }
```

Description

Specifies whether to extract the layout.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Check and Update*
Field:	*Extract*

Examples

```
envGetVal("layoutXL" "lxcuExtract")
envSetVal("layoutXL" "lxcuExtract" 'boolean t)
envSetVal("layoutXL" "lxcuExtract" 'boolean nil)
```

Related Topics

Application Readiness Checker Form

lxCheckAndUpdate

lxCheckAndUpdateRegister

lxCheckAndUpdateRemove

lxCheckAndUpdateSet

© 2023 Cadence Design Systems, Inc.

lxcuFilterMessage

```
layoutXL lxcuFilterMessage string "list_of_messages"
```

Description

Lists the messages to be filtered from the Check and Update report.

The default is `""`.

GUI Equivalent

Command:	*Connectivity – Check and Update*
Field:	*Filter Messages*

Examples

```
envGetVal("layoutXL" "lxcuFilterMessage")
envSetVal("layoutXL" "lxcuFilterMessage" 'string "LX-1001 LX-1005")
```

Related Topics

Application Readiness Checker Form

lxCheckAndUpdate

lxCheckAndUpdateRegister

lxCheckAndUpdateRemove

lxCheckAndUpdateSet

© 2023 Cadence Design Systems, Inc.

lxcuIncludeCells

```
layoutXL lxcuIncludeCells string "names_of_cells_to_include"
```

Description

List of cells to be included when the *All cells in library* or *All cells in library* option is selected.

The default is `""`.

GUI Equivalent

Command:	*Connectivity – Check and Update*
Field:	*Include Cells*

Examples

```
envGetVal("layoutXL" "lxcuIncludeCells")
envSetVal("layoutXL" "lxcuIncludeCells" 'string "triplexer")
```

Related Topics

Application Readiness Checker Form

lxCheckAndUpdate

lxCheckAndUpdateRegister

lxCheckAndUpdateRemove

lxCheckAndUpdateSet

© 2023 Cadence Design Systems, Inc.

lxcuLayoutComplete

```
layoutXL lxcuLayoutComplete boolean { t | nil }
```

Description

Runs hierarchical rebinding with clear logical connectivity when `lxCheckAndUpdate` is called and `lxcuUpdate` is set to `t`.

The `lxcuLayoutComplete` environment variable does not run the following updates:

- Removing unbound terminals, instances, or nets
- Generating terminals, instances, or nets
- Updating parameters

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Check and Update*
Field:	*Layout state*

Examples

```
envGetVal("layoutXL" "lxcuLayoutComplete")
envSetVal("layoutXL" "lxcuLayoutComplete" 'boolean t)
envSetVal("layoutXL" "lxcuLayoutComplete" 'boolean nil)
```

Related Topics

Application Readiness Checker Form

lxCheckAndUpdate

lxCheckAndUpdateRegister

lxCheckAndUpdateRemove

lxCheckAndUpdateSet

© 2023 Cadence Design Systems, Inc.

lxcuLeafCells

```
layoutXL lxcuLeafCells string "name_of_cells"
```

Description

Lists the cells to be treated as leaf cells. Only the terminals of leaf cells are checked when the *All cells in library* or *All cells in library* option is selected.

The default is `""`.

GUI Equivalent

Command:	*Connectivity – Check and Update*
Field:	*Leaf Cells*

Examples

```
envGetVal("layoutXL" "lxcuLeafCells")
envSetVal("layoutXL" "lxcuLeafCells" 'string "(STDLIB1 \\* \\*)" "(STDLIB2 NAND\\* \\*)")
```

Related Topics

Application Readiness Checker Form

lxCheckAndUpdate

lxCheckAndUpdateRegister

lxCheckAndUpdateRemove

lxCheckAndUpdateSet

© 2023 Cadence Design Systems, Inc.

lxcuLogFile

```
layoutXL lxcuLogFile boolean { t | nil }
```

Description

Specifies whether to save the report to a text log file.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Check and Update*
Field:	*Log File*

Examples

```
envGetVal("layoutXL" "lxcuLogFile")
envSetVal("layoutXL" "lxcuLogFile" 'boolean t)
envSetVal("layoutXL" "lxcuLogFile" 'boolean nil)
```

Related Topics

Application Readiness Checker Form

lxCheckAndUpdate

lxCheckAndUpdateRegister

lxCheckAndUpdateRemove

lxCheckAndUpdateSet

© 2023 Cadence Design Systems, Inc.

lxcuLogFileDirectory

```
layoutXL lxcuLogFileDirectory string "name_of_directory"
```

Description

Specify the directory to save the log file.

The default is `""`.

GUI Equivalent

Command:	*Connectivity – Check and Update*
Field:	*Log File Directory*

Examples

```
envGetVal("layoutXL" "lxcuLogFileDirectory")
envSetVal("layoutXL" "lxcuLogFileDirectory" 'string "./TopCellCheckAndUpdate")
```

Related Topics

Application Readiness Checker Form

lxCheckAndUpdate

lxCheckAndUpdateRegister

lxCheckAndUpdateRemove

lxCheckAndUpdateSet

© 2023 Cadence Design Systems, Inc.

lxcuOpenHTMLReport

```
layoutXL lxcuOpenHTMLReport boolean { t | nil }
```

Description

Specifies whether to open the HTML report in the default browser or to print the report in the CIW.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Check and Update*
Field:	*Open HTML Report*

Examples

```
envGetVal("layoutXL" "lxcuOpenHTMLReport")
envSetVal("layoutXL" "lxcuOpenHTMLReport" 'boolean t)
envSetVal("layoutXL" "lxcuOpenHTMLReport" 'boolean nil)
```

Related Topics

Application Readiness Checker Form

lxCheckAndUpdate

lxCheckAndUpdateRegister

lxCheckAndUpdateRemove

lxCheckAndUpdateSet

© 2023 Cadence Design Systems, Inc.

lxcuOpenWorkspace

```
layoutXL lxcuOpenWorkspace boolean { t | nil }
```

Description

Specifies whether to open the CAS workspace.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Check and Update*
Field:	*Open CAS Workspace*

Examples

```
envGetVal("layoutXL" "lxcuOpenWorkspace")
envSetVal("layoutXL" "lxcuOpenWorkspace" 'boolean t)
envSetVal("layoutXL" "lxcuOpenWorkspace" 'boolean nil)
```

Related Topics

Application Readiness Checker Form

lxCheckAndUpdate

lxCheckAndUpdateRegister

lxCheckAndUpdateRemove

lxCheckAndUpdateSet

© 2023 Cadence Design Systems, Inc.

lxcuSvdbPath

```
layoutXL lxcuSvdbPath string "path_to_the_directory"
```

Description

Specifies the path to the PVS svdb directory containing:

- *xyz*`.ixf` instance cross-reference file.
- *xyz*`.net` extracted layout CDL netlist file.

The default is "".

GUI Equivalent

Command:	*Connectivity – Check and Update*
Field:	*LVS svdb path*

Examples

```
envGetVal("layoutXL" "lxcuSvdbPath")
envSetVal("layoutXL" "lxcuSvdbPath" 'string "./svdb")
```

Related Topics

Application Readiness Checker Form

lxCheckAndUpdate

lxCheckAndUpdateRegister

lxCheckAndUpdateRemove

lxCheckAndUpdateSet

© 2023 Cadence Design Systems, Inc.

lxcuSvdbScale

```
layoutXL lxcuSvdbScale int integer
```

Description

The PVS rules file *input_scale* is required only if *input_scale* is different from the layout DBUPerUU.

The *input_scale* value can be found in the Gds_Map statement of the .gds.map file in the svdb directory.

Valid values are between 0 to 10000. The default is 0.

GUI Equivalent

Command:	*Connectivity – Check and Update*
Field:	*LCVS scale*

Examples

```
envGetVal("layoutXL" "lxcuSvdbScale")
envSetVal("layoutXL" "lxcuSvdbScale 'int 5)
```

Related Topics

Application Readiness Checker Form

lxCheckAndUpdate

lxCheckAndUpdateRegister

lxCheckAndUpdateRemove

lxCheckAndUpdateSet

© 2023 Cadence Design Systems, Inc.

lxcuUpdate

```
layoutXL lxcuUpdate boolean { t | nil }
```

Description

Specifies whether to run application updates.

The default is `nil`, which means to run the application checks only.

GUI Equivalent

Command:	*Connectivity – Check and Update*
Field:	*Update*

Examples

```
envGetVal("layoutXL" "lxcuUpdate")
envSetVal("layoutXL" "lxcuUpdate" 'boolean t)
envSetVal("layoutXL" "lxcuUpdate" 'boolean nil)
```

Related Topics

Application Readiness Checker Form

lxCheckAndUpdate

lxCheckAndUpdateRegister

lxCheckAndUpdateRemove

lxCheckAndUpdateSet

© 2023 Cadence Design Systems, Inc.

lxcuWithoutSchematic

```
layoutXL lxcuWithoutSchematic boolean { t | nil }
```

Description

Checks and updates layout cells without a schematic connectivity reference.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Check and Update*
Field:	*Cells without schematic*

Examples

```
envGetVal("layoutXL" "lxcuWithoutSchematic")
envSetVal("layoutXL" "lxcuWithoutSchematic" 'boolean t)
envSetVal("layoutXL" "lxcuWithoutSchematic" 'boolean nil)
```

Related Topics

Application Readiness Checker Form

lxCheckAndUpdate

lxCheckAndUpdateRegister

lxCheckAndUpdateRemove

lxCheckAndUpdateSet

© 2023 Cadence Design Systems, Inc.

lxChainAlignNMOS

```
layoutXL lxChainAlignNMOS string { "Top" | "Center" | "Bottom" }
```

Description

Controls the alignment of NMOS chains to default to bottom.

Arguments

Top

The NMOS chain alignment is set to “Top”.

Center

The NMOS chain alignment is set to “Center”.

Bottom

The NMOS chain alignment is set to “Bottom”.

GUI Equivalent

Command:	*Options – Connectivity* (*Generation* tab)
Field:	*Align N Devices* (Connectivity Form)
Command:	*Connectivity* – Generate Chained Devices Form
Field:	*Align NMOS*
Command:	*Connectivity* – Generate Folded Devices Form
Field:	*Align*

Examples

```
envGetVal("layoutXL" "lxChainAlignNMOS")
envSetVal("layoutXL" "lxChainAlignNMOS" 'string "Top")
envSetVal("layoutXL" "lxChainAlignNMOS" 'string "Center")
envSetVal("layoutXL" "lxChainAlignNMOS" 'string "Bottom")
```


© 2023 Cadence Design Systems, Inc.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxChainAlignPMOS

```
layoutXL lxChainAlignPMOS string { "Top" | "Center" | "Bottom" }
```

Description

Controls the alignment of PMOS chains to default to top.

Arguments

Top

The PMOS chain alignment is set to “Top”.

Center

The PMOS chain alignment is set to “Center”.

Bottom

The PMOS chain alignment is set to “Bottom”.

GUI Equivalent

Command:	*Options* – *Connectivity* (*Generation* tab)
Field:	*Align P Devices* (Connectivity Form)
Command:	*Connectivity* – Generate Chained Devices Form
Field:	*Align PMOS*
Command:	*Connectivity* – Generate Folded Devices Form
Field:	*Align*

Examples

```
envGetVal("layoutXL" "lxChainAlignPMOS")
envSetVal("layoutXL" "lxChainAlignPMOS" 'string "Top")
envSetVal("layoutXL" "lxChainAlignPMOS" 'string "Center")
envSetVal("layoutXL" "lxChainAlignPMOS" 'string "Bottom")
```


© 2023 Cadence Design Systems, Inc.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxCanSwapSDFunction

```
layoutXL lxCanSwapSDFunction string "user_Defined_SKILL_Function"
```

Description

Sets the name of a user-defined SKILL function that determines whether the source and drain pins can be swapped.

The user-defined SKILL function must return `t` if the Pcell master of the instance has a Pcell parameter that swaps the locations of the `S` and `D` pins, and if for this instance you want swapping to be enabled. Otherwise, it should return `nil`.

If the user-defined SKILL function set by this environment variable returns `t`, and Layout XL needs to swap the `S` and `D` pins, then the function defined in `lxSwapSDFunction` is called with an instance to update the Pcell parameter that swaps the locations of the `S` and `D` pins.

The default is "".



Important

The `S` and `D` pins can be swapped using the Pcell parameter only when both `lxCanSwapSDFunction` and `lxSwapSDFunction` are defined.

GUI Equivalent

None

Example

```
envGetVal("layoutXL" "lxCanSwapSDFunction")
envSetVal("layoutXL" "lxCanSwapSDFunction" 'string "canSwapSD")
```

Related Topics

lxSwapSDFunction

Permuting Pins Automatically

List of Layout XL Environment Variables

© 2023 Cadence Design Systems, Inc.

lxDeltaWidth

`layoutXL lxDeltaWidth float` *`floating_point_number`*

Description

Determines the effective width of folded transistors.

The effective total width is defined as

```
Weff = M (W + lxDeltaWidth)
```

Where `M` is the number of folds and `W` is the identical width of each transistor.

Arguments

`floating_point_number`

A float specifying the effective width of folded transistors; for example, `1.0`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lxDeltaWidth")
envSetVal("layoutXL" "lxDeltaWidth" 'float 1.0)
```

Related Topics

Transistor Folding

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxDummyBackAnnotateAll

```
layoutXL lxDummyBackAnnotateAll boolean { t | nil }
```

Description

Backannotates active and inactive dummies from the layout to the schematic view.

The default is `t`.

- If you select the *Connectivity – Back Annotate – All Dummy Instances* command after setting the environment variable to `t`, **all active and inactive** dummy instances are backannotated.
- If you select the *Connectivity – Back Annotate – Selected Dummy Instances* command after setting the environment variable to `t`, **all selected active and inactive** dummy instances are backannotated.

When set to `nil`, only active dummies are backannotated.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lxDummyBackAnnotateAll")
envSetVal("layoutXL" "lxDummyBackAnnotateAll" 'boolean t)
envSetVal("layoutXL" "lxDummyBackAnnotateAll" 'boolean nil)
```

Related Topics

Dummy Instances Backannotation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxDummyBackAnnotateMFactor

```
layoutXL lxDummyBackAnnotateMFactor boolean { t | nil }
```

Description

Allows dummies to be generated as mfactored devices in the schematic by merging those dummies during backannotation that have the same schematic master and connectivity.

Backannotation sets the multiplier parameter on the symbol to equal the number of merged devices. The parameter name used is the first name listed in the mfactorNames environment variable. If a parameter name is specified using the lxDummyBackAnnotateMFactorName environment variable, that name is used instead.

The default is `nil`.

For dummies to be effectively mfactored during backannotation, ensure that the parameters to ignore for check are appropriately set in the LAM file.

This environment variable is now deprecated. Instead, use backAnnotateInstances.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lxDummyBackAnnotateMFactor")
envSetVal("layoutXL" "lxDummyBackAnnotateMFactor" 'boolean t)
envSetVal("layoutXL" "lxDummyBackAnnotateMFactor" 'boolean nil)
```

Related Topics

mfactorNames

lxDummyBackAnnotateMFactorName

backAnnotateInstances

Dummy Instances Backannotation

List of Layout XL Environment Variables

© 2023 Cadence Design Systems, Inc.

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxDummyBackAnnotateMFactorName

```
layoutXL lxDummyBackAnnotateMFactorName string "mFactor_parameter_name"
```

Description

Allows specifying an override to the parameter name defined for dummy backannotation by the backAnnotateInstances environment variable. If more than one name is provided, the first name that matches a CDF parameter is used. If no name matches a CDF parameter, the first name is added as a property on the instance and used instead.

If no value is specified for `lxDummyBackAnnotateMFactorName`, dummy backannotation uses the first name listed in the mfactorNames environment variable. Else, the parameter name specified using the `lxDummyBackAnnotateMFactorName` environment variable takes precedence.

The default is `""`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lxDummyBackAnnotateMFactorName")
envSetVal("layoutXL" "lxDummyBackAnnotateMFactorName" 'string "simM")
envSetVal("layoutXL" "lxDummyBackAnnotateMFactorName" 'string "")
```

Related Topics

mfactorNames

lxDummyBackAnnotateMFactor

Dummy Instances BackannotationList of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxDummyBackAnnotateMissingTerm

```
layoutXL lxDummyBackAnnotateMissingTerm boolean { t | nil }
```

Description

Allows dummies to be backannotated when the layout instance has missing terminals compared to the schematic.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lxDummyBackAnnotateMissingTerm")
envSetVal("layoutXL" "lxDummyBackAnnotateMissingTerm" 'boolean t)
envSetVal("layoutXL" "lxDummyBackAnnotateMissingTerm" 'boolean nil)
```

Related Topics

Dummy Instances Backannotation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxEvalCDFCallbacks

```
layoutXL lxEvalCDFCallbacks boolean { t | nil }
```

Description

Causes all SKILL callbacks defined on Common Description Format (CDF) parameters to be evaluated by default when you run the following Layout XL commands. Callbacks defined on buttons are never evaluated.

- *Generate All From Source*
- *Generate Selected From Source*
- *Generate Folded Devices*
- *Update Components And Nets*
- *Update Layout Parameters*
- *Update Schematic Parameters*
- Back Annotate (Dummy backannotation)

The default is `nil`.

Note: Irrespective of the value of the `lxEvalCDFCallbacks` environment variable, the following callbacks are always evaluated:

- `Width` parameter callback for *Generate Folded Devices*
- `fingers` parameter callback for *Split Fingered Devices*
- `m` parameter callback, if changing the mfactor for backannotation
- `Width` or `fingers` parameter callback or both the callbacks for *Generate All From Source* with folding and finger split enabled.

The callbacks are evaluated in the order in which they are listed in the CDF, so it is important that you set the appropriate callback at the appropriate place in the list. For more information, see Additional Information and Defining Parameters in the *Component Description Format User Guide*.

© 2023 Cadence Design Systems, Inc.



Tip

To prevent any CDF parameters on generated instances from getting evaluated in the schematic, set the parameters as the ones to be ignored for generation. You can use the LAM file, the paramsToIgnore environment variable, or the Connectivity Form form to do this.



Important

Enabling this environment variable can adversely impact performance, especially on large designs with many callbacks.

GUI Equivalent

Command:	*Options – Connectivity* (*Parameters* tab)
Field:	Evaluate CDF callbacks during generation and update (Connectivity Form)

Examples

```
envGetVal("layoutXL" "lxEvalCDFCallbacks")
envSetVal("layoutXL" "lxEvalCDFCallbacks" 'boolean t)
envSetVal("layoutXL" "lxEvalCDFCallbacks" 'boolean nil)
```

Additional Information

CDF callbacks are SKILL expressions that are triggered when the value of a parameter is set in any of these forms—the Add Instance form, the Create Path Form form, the Property Editor assistant, or the Edit Properties form.

Note: If the PDK you are using has callbacks that edit the old Property Editor form or the Add Instance or Create Instance forms, Cadence recommends updating the PDK to remove such form-dependencies that can cause unexpected callback behavior.



Important

If used in batch mode, CDF callbacks cannot allow dialog boxes to display to invite user interaction and may cause Virtuoso to malfunction. CDF callbacks can be designed for interactive use with the Property Editor form but such callbacks are not compatible with batch evaluation.

© 2023 Cadence Design Systems, Inc.

The layoutXL environment variable `lxEvalCDFCallbacks` fires CDF callbacks in the following order:

1. For the changed parameters, filters out any parameters, if defined in `lxCDFCallbackParams`, or calls the callbacks on all the changed parameters in the order they appear in the Edit CDF form.
2. Fires all the other CDF callbacks in the order they appear in the Edit CDF form, apart from the button CDF parameters. If `lxCDFCallbackParams` is defined, only parameters in this list have their callbacks fired.

Note:

- `lxCDFCallbackParams` can be defined as both a global environment variable and as a CDF parameter itself, so you can customize it for a given cell.
- Even if `lxEvalCDFCallbacks` is set to `nil`, the `w` callback is triggered during folding, `fingers` callback is triggered for fingersplit, the `w` or `fingers` parameter callback or both the callbacks are triggered during layout generation, and the `m` callback is triggered for dummy backannotation, if using the m-factor model.
- The CDF `initProc` is always called before the CDF callbacks.
- When CDF callbacks are triggered, the CDF `doneProc` is called after all the parameters are set on the instance.



Tip

Because the order of the callbacks depends on the order they appear, it is recommended to have simple CDF parameters appear first and derived parameters appear after these.

For example, it is recommended to define `fw` (finger width) before total width (finger width * fingers). If the total width callback is called before the finger width callback, the finger width is not changed and the calculation is based on the previous value. If the finger width callback is called first, `fw` is set so that the callback uses the new value and is set correctly.



Important

Cadence advises against the use of CDF callbacks to compute the value of derived parameters. This is to avoid anomalies that are likely in automatic computation of the derived parameters. For more information, see Why Callbacks are not Advised for Deriving Parameter Values.

© 2023 Cadence Design Systems, Inc.

You can optionally set the `lxCDFCallbackParams` environment variable to list the callbacks you want to trigger. Some PDK set ups have callbacks that update all the CDF parameters. Therefore, triggering only a few callbacks may be required.

- If you use AEL expressions in CDF, such as `iPar()` or `pPar()`, you need to set the shell environment variable, `CDS_Netlisting_Mode` to `Analog` for Layout XL to evaluate the expressions in the schematic context.
  - The CDF `parseAsCEL` for the parameter should be set to `yes`. If you want a string, say "`2u`" to be a number, say, `2e-6` for Layout XL, you need to set `parseAsNumber` = `yes`.

Important

It is possible that even when `CDS_Netlisting_Mode` is set to `Analog` and `parseAsCEL` is set to `yes`, you may still not get an accurate layout. This is likely when the design uses callbacks with `pPar()`. Manually triggering the callbacks by setting lxEvalCDFCallbacks to `t` might help. See Issues when pPar and Callbacks are Used in a Design when Generating a Layout. But, Cadence advises against the combined use of CDF parameters with `pPar()` (AEL expressions) and CDF callbacks. This is meant to prevent getting an incorrect parameter value because the value of the expression is not known at the time the callback is fired.

- If the PDK is golden, you can override the CDF by using User CDF, though the User CDF will need to be loaded every time.

Why Callbacks are not Advised for Deriving Parameter Values

Triggering callbacks to derive the values of other parameters is not advised for the following reasons:

- ***Using callbacks to derive parameter values can prevent parameterization of the source parameters***

For situations where a source parameter is meant to derive the value of another parameter, the source parameter itself cannot be controlled by a variable or an expression, such as `pPar()` or `iPar()`.

For example, you might want to use the `pPar()` or `iPar()` expressions to control the value of a source parameter such that the parameter can be passed down the hierarchy. But, when the source parameter is being used to determine the value of a derived parameter, it can change depending on the value of `pPar()` at the time when the parameter value is being changed in Property Editor. Therefore, it is important to know the value of `pPar()` at the time of changing the parameter value to compute the derived parameter accurately. But, knowing the value of `pPar()` at the time of the parameter change is not possible. Therefore, to avoid

© 2023 Cadence Design Systems, Inc.

having an inaccurate derived parameter, you might need to avoid using a source parameter that depends on `pPar()` (or `iPar()`).

Cadence recommends that for parameter values that need to be derived, the value must be computed by the application by using the source parameter at the time of usage. This may require computing the derived parameter value at the time of simulation, physical verification, or at the Pcell level. When computing the derived parameter at the Pcell level, the computation can be done in the Pcell SKILL code instead of requiring a CDF callback to be triggered.

- ***Impossible to determine the correct order for firing multiple callbacks***

When multiple parameters are changed, multiple callbacks need to be triggered. This makes it impossible to predict the correct order in which the callbacks must be fired for a parameter value to be determined accurately.

For example, if you were to change the cell name on the Edit Properties form, all the derived parameters can lose their correspondence. So, you may need to trigger a callback for each derived parameter. But, to determine the correct order in which the callbacks must be fired is not simple, and can potentially cause inaccurate derived parameter values. Therefore, using callbacks to determine the value of derived parameters is not recommended.

Example

Let us assume that the Edit CDF form has the following parameters defined: `w`, `l`, `fingers`, `totalW`

- If we change `w` and `fingers`, the order of the callbacks will be: `w`, `fingers`, `l`, `totalW`
- If we just change `fingers`, the order of the callbacks will be: `fingers`, `w`, `l`, `totalW`
- If we now set `lxCDFCallbackParams` to "`w l`" and then:
  - Change `w` and `fingers`, the order of the callbacks will be: `w`, `l`
  - Change `fingers`, the order of the callbacks will be: `w`, `l`
  - Change `l`, the order of the callbacks will be: `l`, `w`

Related Topics

Dangers of CDF Callbacks

Issues when pPar and Callbacks are Used in a Design when Generating a Layout

© 2023 Cadence Design Systems, Inc.

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxFingeringNames

```
layoutXL lxFingeringNames string "list_of_param_names"
```

Description

Specifies parameter names that define Pcell gate fingering.

Folding and instance fingers are incompatible and this environment variable is used to filter out those instances when folding. If you do not set the names correctly, it can result in abutment failures for even fingered devices and folding.

The default is `"fingers finger numFingers numFinger"`.

Note:

- The `lxFingeringNames` environment variable is also used by the Module Generator when adding dummy devices to MOSFETs or resistors with the modgenMakeMinDummies environment variable set to `t`.

Arguments

list_of_param_names

List of parameter names each separated by a space. The list must be enclosed in quotation marks; for example,

```
"fingers finger numFingers numFinger"
```

GUI Equivalent

Command:	*Options – Connectivity* (*Generation* tab)
Field:	*Pcell fingering parameter names* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "lxFingeringNames")
envSetVal("layoutXL" "lxFingeringNames" 'string "fingers finger numFingers
numFinger")
```

Related Topics

List of Layout XL Environment Variables

© 2023 Cadence Design Systems, Inc.

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxGenerateInBoundary

```
layoutXL lxGenerateInBoundary boolean { t | nil }
```

Description

Generates layout representations within the design boundary when the *Generate All From Source command is run*.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*Generate* tab)
Field:	*In Boundary* (Generate Layout Form)

Examples

```
envGetVal("layoutXL" "lxGenerateInBoundary")
envSetVal("layoutXL" "lxGenerateInBoundary" 'boolean t)
envSetVal("layoutXL" "lxGenerateInBoundary" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxGenerationOrientation

```
layoutXL lxGenerationOrientation string { "preserve" | "R0" | "R90" | "R180" |
     "R270" | "MY" | "MYR90" | "MX" | "MXR90" }
```

Description

Controls the orientation of Layout XL layout devices generated from the schematic.

The default is `"R0"`, which means that devices are generated with `R0` orientation.

Arguments

`preserve`	Devices are generated with the same orientation as in the schematic.
`R0`	Devices are generated with R0 orientation.
`R90`	Generated devices are rotated 90 degrees clockwise.
`R180`	Generated devices are rotated 180 degrees clockwise.
`R270`	Generated devices are rotated 270 degrees clockwise.
`MY`	Generated devices are mirrored over the Y axis.
`MYR90`	Generated devices are mirrored over the Y axis and rotated 90 degrees clockwise.
`MX`	Generated devices are mirrored over the X axis.
`MXR90`	Generated devices are mirrored over the X axis and rotated by 90 degrees clockwise.

GUI Equivalent

Command:	*Options – Connectivity* (*Generation* tab)
Field:	*Device orientation* (Generate Layout Form)

Examples

```
envGetVal("layoutXL" "lxGenerationOrientation")
envSetVal("layoutXL" "lxGenerationOrientation" 'string "R0")
envSetVal("layoutXL" "lxGenerationOrientation" 'string "preserve")
```


© 2023 Cadence Design Systems, Inc.

```
envSetVal("layoutXL" "lxGenerationOrientation" 'string "MXR90")
```

Additional Information

If the device orientation is defined in the LAM file, the *Generate All From Source* command uses that value during layout generation. Else, the value defined using the `lxGenerationOrientation` is considered.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxGenerationTopLevelOnly

```
layoutXL lxGenerationTopLevelOnly boolean { t | nil }
```

Description

Generates layout instances only from top-level schematic instances during the *Generate All From Source* and the *Generate Selected From Source* commands. Hierarchical schematic instances are not generated in the layout.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lxGenerationTopLevelOnly")
envSetVal("layoutXL" "lxGenerationTopLevelOnly" 'boolean t)
envSetVal("layoutXL" "lxGenerationTopLevelOnly" 'boolean nil)
```

Related Topics

Generate Layout Form

Generate Selected Components Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxGetSignifDigits

```
layoutXL lxGetSignifDigits int { 0 | 3 ... 14 }
```

Description

Controls the number of significant digits used by Layout XL when calculating the value of layout CDF parameters.

When enabled `lxGetSignifDigits` overrides the setting of `aelGetSignifDigits` for calculations in the layout cellview. This lets you specify a different precision to be used in layout cellview calculations; for example, those involving parameterized cells.

Valid values are `0` to `14`. The default is `0`, which means that the environment variable is off. The higher the value you set, the more precise are the values used in the calculations.

Arguments

`0`	`lxGetSignifDigits` is disabled.
`1 ... 14`	An integer specifying the number of significant digits used by Layout XL. The minimum value is `1` and the maximum value `14`. The higher the value, the more precise the values used in calculations.

GUI Equivalent

Command:	*Options – Connectivity* (*Parameters* tab)
Field:	*Use significant digits when calculating parameter values* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "lxGetSignifDigits")
envSetVal("layoutXL" "lxGetSignifDigits" 'int 5)
```

Related Topics

List of Layout XL Environment Variables

© 2023 Cadence Design Systems, Inc.

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxGroundNetNames

```
layoutXL lxGroundNetNames string "list_of_ground_net_names"
```

Description

Specifies ground net names to help you achieve optimized chaining results in Layout XL device level schematics.

The list is also used by the Virtuoso custom digital placer to exclude ground nets during wire length optimization; and to determine the ground nets that are hidden when the hideDraglinesForGlobalNets environment variable is switched on.

The default values are `"gnd gnd! gnd: vss vss! vss:"`.

Arguments

list_of_ground_net_names

A list of ground net names each separated by a space. The list must be enclosed in quotation marks; for example,

```
"gnd gnd! gnd: vss vss! vss:"
```

GUI Equivalent

Command:	*Options – Connectivity* (*General* tab)
Field:	*Ground net names* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "lxGroundNetNames")
envSetVal("layoutXL" "lxGroundNetNames" 'string "gnd gnd! gnd: vss vss! vss:")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxInitResetSource

```
layoutXL lxInitResetSource boolean { t | nil }
```

Description

Prompts you to reset the source before starting the Layout XL layout editor from the layout.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lxInitResetSource")
envSetVal("layoutXL" "lxInitResetSource" 'boolean t)
envSetVal("layoutXL" "lxInitResetSource" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxLocalAbutment

```
layoutXL lxLocalAbutment boolean { t | nil }
```

Description

Controls device abutment on overlapping, unabutted pins when a full extraction is run.

The default is `nil`, which means that when you edit a parameter of an abutted instance, the instance is unabutted and then reabutted in order to take account of any differences in the diffusion sizes in wider or narrower transistors. To prevent this from happening, set `lxLocalAbutment` to `t`.

This environment variable also controls whether or not automatic abutment takes place when you launch Layout XL with a full connectivity extraction.

When set to `nil`, the tool automatically abuts any devices which overlap but have not been abutted previously. Any devices that are already abutted are not reabutted. To prevent devices from being abutted automatically when you open a cellview in Layout XL, set `lxLocalAbutment` to `t`. The software abuts only objects that are subsequently edited interactively.

GUI Equivalent

Command:	*Options – Connectivity* (*Generation* tab)
Field:	*Perform auto abutment only after a move or stretch*

Examples

```
envGetVal("layoutXL" "lxLocalAbutment")
envSetVal("layoutXL" "lxLocalAbutment" 'boolean t)
envSetVal("layoutXL" "lxLocalAbutment" 'boolean nil)
```

Additional Information

If you change this environment variable, the change takes effect only after you restart Layout XL. To toggle local abutment and have the change take effect without restarting the software, use the lxToggleLocalAbutment SKILL command.

This environment variable has no effect on the autoSpace functionality.

© 2023 Cadence Design Systems, Inc.

Related Topics

lxLocalAbutment Property

Connectivity Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxPositionCompact

```
layoutXL lxPositionCompact boolean { t | nil }
```

Description

Positions the layout instances as in the schematic and then compacts the instances to place them within the PR boundary.

The default is `t`.

When set to `nil`, the layout instances are placed using the same topology as the schematic but no compaction is performed.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lxPositionCompact")
envSetVal("layoutXL" "lxPositionCompact" 'boolean t)
envSetVal("layoutXL" "lxPositionCompact" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxPositionMinSep

```
layoutXL lxPositionMinSep float float
```

Description

Positions the instances in the layout at a minimum separation based on the user-defined value.

The environment variable can be used to spread out or compact the instances in a layout by increasing or decreasing the minimum separation value.

Note: The `lxPositionMinSep` environment variable takes precedence over lxGenerateInBoundary. Therefore, if the minimum separation value you specify using `lxPositionMinSep` is large, the instances may get positioned outside of the design boundary even if lxGenerateInBoundary is set.

Used by the following commands:

- *Generate All From Source*
- *Update Components And Nets*
- *Pick From Schematic*
- *Place As In Schematic*
- *Generate Folded Devices*

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*Generate* tab)
Field:	*Minimum Separation* (Generate Layout Form)

Examples

```
envGetVal("layoutXL" "lxPositionMinSep")
envSetVal("layoutXL" "lxPositionMinSep" 'float 0.5)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxPositionPinsOnBoundary

```
layoutXL lxPositionPinsOnBoundary boolean { t | nil }
```

Description

Positions the pins on the PR boundary.

The default is `t`.

Used by the following commands:

- *Generate All From Source*, when `lxGenerateInBoundary = t`
- *Place As In Schematic*

Arguments

`t`	Positions the pins on the boundary.
`nil`	Positions the pins inside the boundary.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lxPositionPinsOnBoundary")
envSetVal("layoutXL" "lxPositionPinsOnBoundary" 'boolean t)
envSetVal("layoutXL" "lxPositionPinsOnBoundary" 'boolean nil)
```

Related Topics

Generating All Components from Source

Placement Modes for Generating Selected Components from Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxPositionUseInstPrB

```
layoutXL lxPositionUseInstPrB boolean { t | nil }
```

Description

Controls whether the *Generate All From Source* and *Place As In Schematic* commands use instance master place and route boundary or instance bounding box to space instances during positioning.

The default is `nil`, which means the instance bounding box is used to space instances by lxPositionMinSep.

If set to `t`, instances are spaced using the instance master place and route boundary. If instance masters have shapes outside the place and route boundary, the *Generate All From Source* or *Place As In Schematic* positioning could result in shorts.

GUI Equivalent

None

Examples

```
envGetVal ("layoutXL" "lxPositionUseInstPrB")
envSetVal ("layoutXL" "lxPositionUseInstPrB" 'boolean t)
envSetVal ("layoutXL" "lxPositionUseInstPrB" 'boolean nil)
```

Related Topics

Generating All Components from Source

Placement Modes for Generating Selected Components from Source

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxRetainFoldOrient

```
layoutXL lxRetainFoldOrient boolean { t | nil }
```

Description

Preserves the original device orientation when running the *Generate Folded Devices* command or the *Generate All From Source* command with the *Fold* option switched on.

Note: If the *Chain* option is also switched on, instance orientation is not preserved.

GUI Equivalent

Command:	*Options – Connectivity* (*Generation* tab)
Field:	*Retain device orientation*

Examples

```
envGetVal("layoutXL" "lxRetainFoldOrient")
envSetVal("layoutXL" "lxRetainFoldOrient" 'boolean t)
envSetVal("layoutXL" "lxRetainFoldOrient" 'boolean nil)
```

Related Topics

Connectivity Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxSchematicDefaultApp

```
layoutXL lxSchematicDefaultApp cyclic { "L" | "XL" | "None" }
```

Description

Specifies which tier of the Virtuoso Schematic Editor is started when you launch Layout XL or GXL. If the environment variable is set to "`None`", the Layout XL is launched without the schematic editor being open.

The default is `"XL"`.



The schematic editor tier level does not change when you launch Layout XL from the Virtuoso Schematic Editor XL nor when you switch between layout editor tier levels *after* Layout XL or GXL have been launched.

Arguments

`L`	Launches Virtuoso Schematic Editor L
`XL`	Launches Virtuoso Schematic Editor XL
`None`	The Virtuoso Schematic Editor is not launched.

GUI Equivalent

Command:	*Options – Connectivity – General* tab
	(*Connectivity Reference* group box)
Field:	*Open with* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "lxSchematicDefaultApp")
envSetVal("layoutXL" "lxSchematicDefaultApp" 'cyclic "L")
envSetVal("layoutXL" "lxSchematicDefaultApp" 'cyclic "XL")
envSetVal("layoutXL" "lxSchematicDefaultApp" 'cyclic "None")
```


© 2023 Cadence Design Systems, Inc.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxSchematicEditMode

```
layoutXL lxSchematicEditMode boolean { t | nil }
```

Description

Controls whether the schematic cellview is opened in edit mode or not, depending on the mode in which the layout cellview is opened.

If the environment variable is set to `t` and the layout cellview is opened in edit mode, the schematic cellview is opened in edit mode as well, irrespective of the original mode of the schematic cellview. See Additional Information.

The default is `nil`.

Examples

```
envGetVal("layoutXL" "lxSchematicEditMode")
envSetVal("layoutXL" "lxSchematicEditMode" 'boolean t)
envSetVal("layoutXL" "lxSchematicEditMode" 'boolean nil)
```

Additional Information

The table below indicates the mode in which the schematic cellview is opened, depending on the state of the environment variable, `lxSchematicEditMode`, and the mode in which the layout cellview is opened.

`lxSchematicEditMode` set to ...	Layout cellview mode	Schematic cellview mode	Schematic cellview opens in mode
t	edit	edit	edit
t	edit	read	edit
t	read	edit	edit
t	read	read	read
nil	edit	edit	edit
nil	edit	read	read
nil	read	edit	edit
nil	read	read	read

© 2023 Cadence Design Systems, Inc.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxSchExtractTopLevelOnly

```
layoutXL lxSchExtractTopLevelOnly boolean { t | nil }
```

Description

Specifies that only the top level of the schematic is to be extracted during the following commands in the Connectivity menu.

- *Generate All From Source*
- *Generate Clones*
- *Check Against Source*
- *Update Components And Nets*
- *Define Device Correspondence*

The lower level schematics are not extracted, even if they need to be extracted.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lxSchExtractTopLevelOnly")
envSetVal("layoutXL" "lxSchExtractTopLevelOnly" 'boolean t)
envSetVal("layoutXL" "lxSchExtractTopLevelOnly" 'boolean nil)
```

Related Topics

Generating All Components from Source

Clone Generation

Design Check Against Source

Components and Nets Updates

Device Correspondence Updates

© 2023 Cadence Design Systems, Inc.

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxStackMinimalFolding

```
layoutXL lxStackMinimalFolding boolean { t | nil }
```

Description

Sets folding to yield the minimum number of legs for each folded device. It does not matter whether the number is even or odd.

The default is `t`.

GUI Equivalent

Command:	*Options – Connectivity* (*Generation* tab)
Field:	*Generate minimal folding*

Examples

```
envGetVal("layoutXL" "lxStackMinimalFolding")
envSetVal("layoutXL" "lxStackMinimalFolding" 'boolean t)
envSetVal("layoutXL" "lxStackMinimalFolding" 'boolean nil)
```

Related Topics

Connectivity Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxStackPartitionParameters

`layoutXL lxStackPartitionParameters string "(`*`integer integer`*`)"`

Description

Sets the device chaining partition parameters.

The first parameter controls how far up the hierarchy Layout XL looks for opportunities to abut devices. A value of 0 means that chaining can look for abutments only in the same hierarchical level. A value of N means that chaining can look up N levels of hierarchy in order to find abutments. The first parameter can take a value between `0` and `32`.

The second parameter sets the maximum number of devices permitted in a chain.

The default is `"(0 100)"`.



Important

The first parameter is always considered by the *Generate All From Source* and *Generate Selected From Source* commands. However, the *Generate Chained Devices* command considers this parameter only if there are 100 instances or more in the selected set.

© 2023 Cadence Design Systems, Inc.

Arguments

(integer integer)

A list of two integers enclosed in parentheses and quotation marks; for example,

`"(0 100)"`

Note: The first integer can take a value between `0` and `32`.

GUI Equivalent

Command:	*Options – Connectivity* (*Generation* tab)
Field:	*Search levels up for abutments* *Maximum number of devices in a chain*

Examples

```
envGetVal("layoutXL" "lxStackPartitionParameters")
envSetVal("layoutXL" "lxStackPartitionParameters" 'string "(0 100)")
```

Related Topics

Connectivity Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxSupplyNetNames

```
layoutXL lxSupplyNetNames string "list_of_supply_net_names"
```

Description

Specifies the list of power net names used to achieve optimized chaining results in Layout XL device level schematics.

The list is also used by the Virtuoso custom digital placer to exclude power nets during wire length optimization; and to determine the power nets that are hidden when the hideDraglinesForGlobalNets environment variable is switched on.

The default values are `"vcc vcc! vcc: vdd vdd! vdd:"`.

Arguments

list_of_supply_net_names

A list of power net names each separated by a space. The list must be enclosed in quotation marks; for example,

```
"vcc vcc! vcc: vdd vdd! vdd:"
```

GUI Equivalent

Command:	*Options – Connectivity* (*General* tab)
Field:	*Power net names*

Examples

```
envGetVal("layoutXL" "lxSupplyNetNames")
envSetVal("layoutXL" "lxSupplyNetNames" 'string "vcc vcc! vcc: vdd vdd! vdd:")
```

Related Topics

Connectivity Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxSwapSDFunction

`layoutXL lxSwapSDFunction string "`*`user_Defined_SKILL_Function`*`"`

Description

Sets the name of a user-defined SKILL function that swaps the `S` and `D` pins. This environment variable is read after setting `lxCanSwapSDFunction`.

The user-defined SKILL function is passed the database ID of the instance for which chaining needs to swap the `S` and `D` pins. The function must update a Pcell parameter on this instance, to perform the `S` and `D` pin swap, and then return `t`. If for some reason, the `S` and `D` pin swap cannot done for this instance, the function should return `nil`.

Important

The function defined in `lxSwapSDFunction` is called only if no permute rule exists. If a permute rule exists, chaining will permute using these rules. Additionally, the `S` and `D` pins can be swapped using the Pcell parameter only when both `lxCanSwapSDFunction` and `lxSwapSDFunction` are defined.

Additional Information

The Pcell parameter that swaps the `S` and `D` pins must be specified as `IGNORE` for `GENERATE` or `IGNORE` for `CHECK`. One way to do this is in the library and attributes mapping (LAM) file entry for this Pcell. If the parameter is not marked as ignored, then CAS will report any difference in that parameter value between the bound layout instance and its corresponding schematic instance. In this case, when you update the layout parameters, the layout parameter value will be updated to match the schematic value, which will break the chains and cause shorts. This is why, marking the parameter as ignored ensures that the *Update Components and Nets* or *Update Layout Parameters* commands do not overwrite the value of the parameter that has been swapped in the layout during chaining.

The default is `""`.

GUI Equivalent

None

Example

```
envGetVal("layoutXL" "lxSwapSDFunction")
```


© 2023 Cadence Design Systems, Inc.

```
envSetVal("layoutXL" "lxSwapSDFunction" 'string "swapSD")
```

Related Topics

lxCanSwapSDFunction

Permuting Pins Automatically

List of Layout XL Environment Variables

© 2023 Cadence Design Systems, Inc.

lxUpdateFoldedWidth

```
layoutXL lxUpdateFoldedWidth string {"No Change" | "Equalize" | "Distribute" }
```

Description

Specifies how the *Update Layout Parameters* command updates folded devices when the width value is changed in the schematic.

The default is `"No Change"`.

© 2023 Cadence Design Systems, Inc.

Arguments

`No Change` — Folded devices are not updated.

`Equalize` — Creates folds of equal width in the layout instance. The number of folds does not change, only (potentially) the width of each fold.

For example, the total width in the schematic is 12 and is distributed as follows in the layout.

```
layout widths I1.1 w = 2
              I1.2 w = 4
              I1.3 w = 6
```

If you change the width in the schematic to 15 and run *Update – Layout Parameters* with this option set, the new widths are as follows.

```
layout widths I1.1 w = 5
              I1.2 w = 5
              I1.3 w = 5
```

`Distribute` — Distributes the additional (or reduced) width equally amongst all the folds in the layout instance. Again, the number of folds does not change, only the width of each fold.

For example, the total width in the schematic is 12 and is distributed as follows in the layout.

```
layout widths I1.1 w = 2
              I1.2 w = 4
              I1.3 w = 6
```

If you change the width in the schematic to 18 and run *Update Layout Parameters* with this option set, the new widths are as follows.

```
layout widths I1.1 w = 4
              I1.2 w = 6
              I1.3 w = 8
```

An extra 2 units of width is added to each fold.

GUI Equivalent

Command: *Options – Connectivity* (*Generation* tab)

Field: *Update width parameter method*

© 2023 Cadence Design Systems, Inc.

Examples

```
envGetVal("layoutXL" "lxUpdateFoldedWidth")
envSetVal("layoutXL" "lxUpdateFoldedWidth" 'string "No Change")
envSetVal("layoutXL" "lxUpdateFoldedWidth" 'string "Equalize")
envSetVal("layoutXL" "lxUpdateFoldedWidth" 'string "Distribute")
```

Related Topics

Connectivity Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxUseLibList

`layoutXL lxUseLibList string "`*`list_of_libraries`*`"`

Description

Specifies a list of libraries for Layout XL to search for a layout view that corresponds to a particular schematic or symbol view.

When generating or updating layout components from a schematic or symbol view, Layout XL uses the layout cellview specified in the Configure Physical Hierarchy window for the schematic or symbol in question. If there is no cellview specified, Layout XL searches for a layout cellview of the same name stored in the same library as the schematic or symbol cellview. If there is no such cellview, the software searches any libraries specified by `lxUseLibList`.

The software searches the libraries in the sequence in which they are listed and uses the first layout cellview it finds with a name that matches the name of the schematic or symbol. If it is unable to find a corresponding layout cellview, the software issues a warning message.

The following commands in the *Connectivity* menu honor the `lxUseLibList` environment variable.

- *Generate All From Source*
- *Generate Clones*
- *Generate Selected From Source*
- *Generate Chained Devices*
- *Generate Folded Devices*
- *Check Against Source*
- *Update Components And Nets*
- *Update Layout Parameters*
- *Update Schematic Parameters*

© 2023 Cadence Design Systems, Inc.

Arguments

list_of_libraries

A list of library names each separated by a space. The list must be enclosed in quotation marks; for example,

```
"lib1 lib2 lib3"
```

GUI Equivalent

Command:	*Launch – Configure Physical Hierarchy*
Field:	*Physical library list* (Physical Hierarchy Configuration)

Examples

```
envGetVal("layoutXL" "lxUseLibList")
envSetVal("layoutXL" "lxUseLibList" 'string "lib1 lib2 lib3")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxValidateXLParameterEvaluation

```
layoutXL lxValidateXLParameterEvaluation string { "" | "Parallel(Block
     Generation)" }
```

Description

Controls the behavior of the tool when the specified mfactor or sfactor cannot be recognized as an integer.

The default is `""`.

Arguments

`""`	A single device is created and a warning is issued about the mfactor or sfactor value being invalid.
`"Parallel(Block Generation)"`	A warning is issued to indicate that the specified mfactor or sfactor value is invalid and that a single device will be generated. If the specified mfactor value is invalid, the following layout generation commands are blocked: ■ *Generate All From Source* ■ *Generate Selected From Source* ■ *Update Components and Nets* **Note:** The listed commands are not blocked if the sfactor value is found to be invalid.

GUI Equivalent

None

© 2023 Cadence Design Systems, Inc.

Examples

```
envGetVal("layoutXL" "lxValidateXLParameterEvaluation")
envSetVal("layoutXL" "lxValidateXLParameterEvaluation" 'string "")
envSetVal("layoutXL" "lxValidateXLParameterEvaluation" 'string "Parallel(Block
Generation)")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

lxWidthTolerance

```
layoutXL lxWidthTolerance string "(value value)"
```

Description

Sets the allowed device width variation for folding.

The default is `"(0.0 0.0)"`.

Arguments

(value value)

A list of two positive numbers separated by a space. The first is the absolute value of the negative tolerance; the second is the positive tolerance. The list must be enclosed in parentheses and quotation marks; for example,

`"(0.0 0.0)"`

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lxWidthTolerance")
envSetVal("layoutXL" "lxWidthTolerance" 'string "(0.25 0.25)"
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

makeCellCreatePins

```
layoutXL makeCellCreatePins boolean { t | nil }
```

Description

Controls whether the made cell created in Layout XL by using the *Make Cell* command has its pins created.

The default is `t`.

GUI Equivalent

Command:	*Edit – Hierarchy – Make Cell*
	Hierarchy – Make Cell (Right Mouse Button menu)
Field:	*Create Pins* (Make Cell Form)

Examples

```
envGetVal("layoutXL" "makeCellCreatePins")
envSetVal("layoutXL" "makeCellCreatePins" 'boolean t)
envSetVal("layoutXL" "makeCellCreatePins" 'boolean nil)
```

Related Topics

Creating a Made Cell

Make Cell Pin Creation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

makeCellPinsBelowBoundary

```
layoutXL makeCellPinsBelowBoundary boolean { t | nil }
```

Description

Specifies whether the pins of the made cell, created using the Make Cell command, are placed outside the design boundary (`t`). The default is `nil`.

This is only available if the `makeCellPreserveConn` and `makeCellCreatePins` variables are also `t`.

GUI Equivalent

Command:	*Edit – Hierarchy – Make Cell*
	OR
	Hierarchy – Make Cell (Right Mouse Button menu)
Field:	*Pins Below boundary* (*Make Cell* Form*)*

Examples

```
envGetVal("layoutXL" "makeCellPinsBelowBoundary")
envSetVal("layoutXL" "makeCellPinsBelowBoundary" 'boolean t)
```

Related Topics

Make Cell Form

Creating a Made Cell

Make Cell Pin Creation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

makeCellPRBoundary

```
layoutXL makeCellPRBoundary cyclic { "Rectangle" | "Polygon" | "None" }
```

Description

Controls whether a PR boundary is created for the made cell. When the PR boundary needs to be created, the environment variable also specifies the shape of the PR boundary.

The default is `Rectangle`. When set to `NONE`, no PR boundary is created for the made cell.

GUI Equivalent

Command:	*Edit – Hierarchy – Make Cell*
	OR
	Hierarchy – Make Cell (Right Mouse Button menu)
Field:	*P&R boundary* (*Make Cell Form*)

Examples

```
envGetVal("layoutXL" "makeCellPRBoundary")
envSetVal("layoutXL" "makeCellPRBoundary" 'cyclic "Rectangle")
envSetVal("layoutXL" "makeCellPRBoundary" 'cyclic "Polygon")
envSetVal("layoutXL" "makeCellPRBoundary" 'cyclic "None")
```

Related Topics

Make Cell Form

Creating a Made Cell

Make Cell Pin Creation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

makeCellPreserveConn

```
layoutXL makeCellPreserveConn boolean { t | nil }
```

Description

Controls whether the made cell created in Layout XL by using the *Make Cell* command has its connectivity preserved.

The default is `t`.

GUI Equivalent

Command:	*Edit – Hierarchy – Make Cell*
	(Instance shortcut menu) *Hierarchy – Make Cell*
Field:	*Preserve Connectivity* (Make Cell Form)

Examples

```
envGetVal("layoutXL" "makeCellPreserveConn")
envSetVal("layoutXL" "makeCellPreserveConn" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

makeCellTransparent

```
layoutXL makeCellTransparent boolean { t | nil }
```

Description

Controls whether the made cell occurrence created in Layout XL is transparent.

The default is `t`.

GUI Equivalent

Command:	*Edit – Hierarchy – Make Cell*
	(Instance shortcut menu) *Hierarchy – Make Cell*
Field:	*Create Pins* (Make Cell Form)

Examples

```
envGetVal("layoutXL" "makeCellTransparent")
envSetVal("layoutXL" "makeCellTransparent" 'boolean t)
envSetVal("layoutXL" "makeCellTransparent" 'boolean nil)
```

Related Topics

Transparent Layout Hierarchy Generation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

matchedToIgnore

This function has been included within this appendix for completeness. For details on its description, type, and value refer to the `matchedToIgnore` function in the *Virtuoso Unified Custom Constraints User Guide*.

When entering this function in *Virtuoso Layout Suite*, you should specify that the application is `layout` instead of `constraint`.

© 2023 Cadence Design Systems, Inc.

mfactorNames

```
layoutXL mfactorNames string "list_of_names"
```

Description

Lists the names of properties used in the schematic to specify the multiplication factor (`mfactor`) for transistors.

The default is `"m M"`.

Note:

- The `mfactor` property value can be expressed as an integer or an expression.
- For PDKs that use `simM` as the schematic multiplier, the override can be set using the lxDummyBackAnnotateMFactorName environment variable.

Arguments

list_of_names

A list of mfactor property names each separated by a space. The list must be enclosed in quotation marks; for example,

`"m M"`

GUI Equivalent

Command:	*Options – Connectivity* (*Parameters* tab)
Field:	*Parallel-connected factor* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "mfactorNames")
envSetVal("layoutXL" "mfactorNames" 'string "m M")
```

Related Topics

lxDummyBackAnnotateMFactor

lxDummyBackAnnotateMFactorName

© 2023 Cadence Design Systems, Inc.

Dummy Instances Backannotation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

mfactorSplit

```
layoutXL mfactorSplit boolean { t | nil }
```

Description

Controls whether Layout XL places schematic devices with the mfactor Property property as multiple devices in the layout.

The default is `t`.

Note: You can override this environment variable for a given instance by setting the lxMfactorSplit Property property for the instance in question.

GUI Equivalent

Command:	*Options – Connectivity* (*Generation* tab)
Field:	*Split mfactored devices*

Examples

```
envGetVal("layoutXL" "mfactorSplit")
envSetVal("layoutXL" "mfactorSplit" 'boolean t)
envSetVal("layoutXL" "mfactorSplit" 'boolean nil)
```

Additional Information

This environment variable controls *only* whether Layout XL generates one device or several devices in the layout view. It does not change the widths of the generated devices.

Related Topics

Connectivity Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

moveAsGroup

```
layoutXL moveAsGroup boolean { t | nil }
```

Description

Specifies that the currently selected instances are treated as a single group and maintain their relative positions during move and stretch operations.

If two cells in the selected group can be abutted to different cells, only one abutment will occur because to abut both would upset the relative position of the set. The cell chosen to abut is the one that causes the least movement within the selected set.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "moveAsGroup")
envSetVal("layoutXL" "moveAsGroup" 'boolean t)
envSetVal("layoutXL" "moveAsGroup" 'boolean nil)
```

Related Topics

Manual Device Abutment

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerAutoZoomOnSelected

```
layoutXL netTracerAutoZoomOnSelected boolean { t | nil }
```

Description

Automatically zooms into traces that are selected in the Trace Manager Form.

The default is `t`.

GUI Equivalent

Command:	*Right-click the menu bar – Toolbars – Net Tracer*
Field:	*Auto zoom to selected traces* (button)

Examples

```
envGetVal("layoutXL" "netTracerAutoZoomOnSelected")
envSetVal("layoutXL" "netTracerAutoZoomOnSelected" 'boolean t)
envSetVal("layoutXL" "netTracerAutoZoomOnSelected" 'boolean nil)
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerColor

```
layoutXL netTracerColor string { "Cycle" | "color1" }
```

Description

Lets you specify a color for the trace.

The default is `"Cycle"`, which selects a new color from the associated palette sequentially for creating the trace.

GUI Equivalent

Command:	*Connectivity – Net Tracer – Add/Remove*
Field:	*Color* (Net Tracer Options Form)

Examples

```
envGetVal("layoutXL" "netTracerColor")
envSetVal("layoutXL" "netTracerColor" 'string "black")
envSetVal("layoutXL" "netTracerColor" 'string "blue")
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerComputeNeighborNet

```
layoutXL netTracerComputeNeighborNet boolean { t | nil }
```

Description

Specifies whether the Net Tracer computes net names for the neighboring shapes of a traced shape.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Net Tracer – Neighbors*
Field:	*Compute net name*

Examples

```
envGetVal("layoutXL" "netTracerComputeNeighborNet")
envSetVal("layoutXL" "netTracerComputeNeighborNet" 'boolean t)
```

Related Topics

netTracerComputeNeighborNetMethod

netTracerNeighborsOnLayer

Net Tracer Neighbors Form

Finding Neighboring Shapes of a Traced Shape

© 2023 Cadence Design Systems, Inc.

netTracerComputeNeighborNetMethod

```
layoutXL netTracerComputeNeighborNetMethod cyclic { "logical" |
     "logicalandphysical" }
```

Description

Specifies the method that the Net Tracer uses to compute net names for the neighboring shapes of a traced shape.

- `logical`: Uses the same net name if a neighboring shape already has a net name. In a hierarchy, the top-most net is used as the net name for a neighboring shape. If a neighboring shape does not already have a net name, no net name is assigned.
- `logicalandphysical`: Uses the same net name if a neighboring shape already has a net name. If the neighboring shape does not already have a net name, the neighboring shape is physically traced until a shape with a net name is found, and that net name is used for the neighboring shape.

The default is `logical`.

GUI Equivalent

Command:	*Connectivity – Net Tracer – Neighbors*
Field:	*Compute From*

Examples

```
envGetVal("layoutXL" "netTracerComputeNeighborNetMethod")
envSetVal("layoutXL" "netTracerComputeNeighborNetMethod" 'cyclic
"logicalandphysical")
```

Related Topics

netTracerComputeNeighborNet

netTracerNeighborsOnLayer

Net Tracer Neighbors Form

Finding Neighboring Shapes of a Traced Shape

© 2023 Cadence Design Systems, Inc.

netTracerConnectedLayer

```
layoutXL netTracerConnectedLayer boolean { t | nil }
```

Description

Controls whether connected layers defined on the Net Tracer Custom Setup Form form must be traced through.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Net Tracer – Add/Remove*
Field:	*Use custom connected layers* (Net Tracer Custom Setup Form)

Examples

```
envGetVal("layoutXL" "netTracerConnectedLayer")
envSetVal("layoutXL" "netTracerConnectedLayer" 'boolean t)
envSetVal("layoutXL" "netTracerConnectedLayer" 'boolean nil)
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerConstraintGroup

```
layoutXL netTracerConstraintGroup string { "virtuosoDefaultExtractorSetUp" |
     "customCG" }
```

Description

Specifies the constraint group that defines the via layer, connected layer, and stop layers to be used for creating traces.

The default is `virtuosoDefaultExtractorSetUp`, which is also the default constraint group used by the Layout XL connectivity extractor.

GUI Equivalent

Command:	*Connectivity – Net Tracer – Add/Remove*
Field:	*Source* (Net Tracer Options Form)

Examples

```
envGetVal("layoutXL" "netTracerConstraintGroup")
envSetVal("layoutXL" "netTracerConstraintGroup" 'string
"virtuosoDefaultExtractorSetUp")
envSetVal("layoutXL" "netTracerConstraintGroup" 'string "customCG1")
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerCrossHighlightNetInSchematic

```
layoutXL netTracerCrossHighlightNetInSchematic boolean { t | nil }
```

Description

Cross-highlights a net in the schematic cellview when that net is traced in the layout cellview.

The default value is `nil`.

GUI Equivalent

Command:	*Connectivity – Net Tracer – Add/Remove*
Field:	*Cross-highlight net in schematic*

Examples

```
envGetVal("layoutXL" "netTracerCrossHighlightNetInSchematic")
envSetVal("layoutXL" "netTracerCrossHighlightNetInSchematic" 'boolean t)
```

Related Topics

Net Tracer Options Form

© 2023 Cadence Design Systems, Inc.

netTracerDisplayDepth

`layoutXL netTracerDisplayDepth int` *`integer`*

Description

Controls the hierarchy depth up to which the corresponding shapes for a traced net are displayed.

The default is `0`.

GUI Equivalent

Command:	*Options – Net Tracer*
Field:	*Display Depth* (Net Tracer Display Options Form)

Examples

```
envGetVal("layoutXL" "netTracerDisplayDepth")
envSetVal("layoutXL" "netTracerDisplayDepth" 'int 32)
envSetVal("layoutXL" "netTracerDisplayDepth" 'int 5)
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerDisplayMode

```
layoutXL netTracerDisplayMode cyclic { "True Color & Halo" | "True Color" | "Halo" }
```

Description

Controls whether the traces over shapes, pins, and other objects associated with a traced net are displayed in true color, and whether a trace displays a halo.

The default is `"True Color"`.

GUI Equivalent

Command:	*Options – Net Tracer*
Field:	*Display Mode* (Net Tracer Display Options Form)

Examples

```
envGetVal("layoutXL" "netTracerDisplayMode")
envSetVal("layoutXL" "netTracerDisplayMode" 'cyclic "True Color & Halo")
envSetVal("layoutXL" "netTracerDisplayMode" 'cyclic "True Color")
envSetVal("layoutXL" "netTracerDisplayMode" 'cyclic "Halo")
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerExcludePurposes

```
layoutXL netTracerExcludePurposes boolean { t | nil }
```

Description

Controls whether layer purposes are excluded from tracing.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Net Tracer – Add/Remove*
Field:	*Exclude purposes* (Net Tracer Options Form)

Examples

```
envGetVal("layoutXL" "netTracerExcludePurposes")
envSetVal("layoutXL" "netTracerExcludePurposes" 'boolean t)
envSetVal("layoutXL" "netTracerExcludePurposes" 'boolean nil)
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerExcludePurposesName

```
layoutXL netTracerExcludePurposesName string { "annotate boundary error flight
     label warning" }
```

Description

Specifies the names of the layer purposes to be excluded for tracing. Multiple purpose names can be space-separated.

The default is `""`.

GUI Equivalent

Command:	*Connectivity – Net Tracer – Add/Remove*
Field:	*Exclude purposes* (Net Tracer Options Form)

Examples

```
envGetVal("layoutXL" "netTracerExcludePurposesName")
envSetVal("layoutXL" "netTracerExcludePurposesName" 'string "annotate error")
envSetVal("layoutXL" "netTracerExcludePurposesName" 'string "boundary")
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerFromNet

```
layoutXL netTracerFromNet boolean { t | nil }
```

Description

Controls whether traces are added to highlight all the shapes that are physically connected on the same net.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Net Tracer – Add/Remove*
Field:	*From net* (Net Tracer Options Form)

Examples

```
envGetVal("layoutXL" "netTracerFromNet")
envSetVal("layoutXL" "netTracerFromNet" 'boolean t)
envSetVal("layoutXL" "netTracerFromNet" 'boolean nil)
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerHaloFadeoutShapes

```
layoutXL netTracerHaloFadeoutShapes boolean { t | nil }
```

Description

Controls whether a faded out halo is created around shapes when displaying net traces.

The default is `t`.

GUI Equivalent

Command:	*Options – Net Tracer*
Field:	*Fadeout shapes* (Net Tracer Options Form)

Examples

```
envGetVal("layoutXL" "netTracerHaloFadeoutShapes")
envSetVal("layoutXL" "netTracerHaloFadeoutShapes" 'boolean t)
envSetVal("layoutXL" "netTracerHaloFadeoutShapes" 'boolean nil)
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerHaloForPins

```
layoutXL netTracerHaloForPins boolean { t | nil }
```

Description

Controls whether the halos around pins are wider compared to those around shapes, or have the same width.

The default is `t`, which means the halos around pins are wider.

GUI Equivalent

Command:	*Options – Net Tracer*
Field:	*Pins* (Net Tracer Options Form)

Examples

```
envGetVal("layoutXL" "netTracerHaloForPins")
envSetVal("layoutXL" "netTracerHaloForPins" 'boolean t)
envSetVal("layoutXL" "netTracerHaloForPins" 'boolean nil)
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerHaloForShapes

```
layoutXL netTracerHaloForShapes boolean { t | nil }
```

Description

Controls whether a halo is displayed only around visible shapes.

The default is `nil`, which displays a halo around all the shapes on a trace, even if the shapes are not visible in the canvas.

GUI Equivalent

Command:	*Options – Net Tracer*
Field:	*Visible shapes only* (Net Tracer Options Form)

Examples

```
envGetVal("layoutXL" "netTracerHaloForShapes")
envSetVal("layoutXL" "netTracerHaloForShapes" 'boolean t)
envSetVal("layoutXL" "netTracerHaloForShapes" 'boolean nil)
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerHaloStyleThick

```
layoutXL netTracerHaloStyleThick boolean { t | nil }
```

Description

Controls whether a halo created around shapes and pins is displayed with a thick width to make the halo more prominent.

The default is `t`.

GUI Equivalent

Command:	*Options – Net Tracer*
Field:	*Thick width* (Net Tracer Options Form)

Examples

```
envGetVal("layoutXL" "netTracerHaloStyleThick")
envSetVal("layoutXL" "netTracerHaloStyleThick" 'boolean t)
envSetVal("layoutXL" "netTracerHaloStyleThick" 'boolean nil)
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerHaloTransparency

```
layoutXL netTracerHaloTransparency int integer
```

Description

Controls the extent of transparency for a halo. This environment variable takes effect only when the netTracerHaloFadeoutShapes environment variable is enabled.

The default is `50`.

GUI Equivalent

Command:	*Options – Net Tracer*
Field:	*Transparency* (Net Tracer Options Form)

Examples

```
envGetVal("layoutXL" "netTracerHaloTransparency")
envSetVal("layoutXL" "netTracerHaloTransparency" 'int 22)
envSetVal("layoutXL" "netTracerHaloTransparency" 'int 100)
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerMaxCutShapes

```
layoutXL netTracerMaxCutShapes string { "Low" | "Medium" | "High" | "Full" }
```

Description

Specifies the maximum number of cut shapes that are traced per overlap. You can choose to trace: 100 cut shapes per overlap (*Low*), 1000 (*Medium*), 10000 (*High*), or trace all the cut shapes (*Full*).

The default is `"Medium"`, which traces 1000 cut shapes per overlap.

GUI Equivalent

Command:	*Connectivity – Net Tracer – Add/Remove*
Field:	*Cut Shapes Extraction* (Net Tracer Options Form)

Examples

```
envGetVal("layoutXL" "netTracerMaxCutShapes")
envSetVal("layoutXL" "netTracerMaxCutShapes" 'string "Low")
envSetVal("layoutXL" "netTracerMaxCutShapes" 'string "Medium")
envSetVal("layoutXL" "netTracerMaxCutShapes" 'string "High")
envSetVal("layoutXL" "netTracerMaxCutShapes" 'string "Full")
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerNeighborsOnLayer

```
layoutXL netTracerNeighborsOnLayer cyclic { "same" | "any" }
```

Description

Specifies whether the Net Tracer finds the neighboring shapes on the same layer as the traced shape or on any extractable layer.

- `same`: Finds neighboring shapes on the same layer as the traced shape.
- `any`: Finds neighboring shapes on any extractable layer.

The default is `same`.

GUI Equivalent

Command:	*Connectivity – Net Tracer – Neighbors*
Field:	*Neighbors*

Examples

```
envGetVal("layoutXL" "netTracerNeighborsOnLayer")
envSetVal("layoutXL" "netTracerNeighborsOnLayer" 'cyclic "any")
```

Related Topics

netTracerComputeNeighborNet

netTracerComputeNeighborNetMethod

Net Tracer Neighbors Form

Finding Neighboring Shapes of a Traced Shape

© 2023 Cadence Design Systems, Inc.

netTracerNumShapesLimitForRefresh

`layoutXL netTracerNumShapesLimitForRefresh int` *`integer`*

Description

Controls the number of shapes that are retraced when an already traced region is selected.

The default is `10000`.

Note: If the number of retraced shapes is `>10000`, a pop up message prompts you to confirm whether to retrace or not.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "netTracerNumShapesLimitForRefresh")
envSetVal("layoutXL" "netTracerNumShapesLimitForRefresh" 'int 5678)
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerNumThreads

`layoutXL netTracerNumThreads int` *`integer`*

Description

Controls the number of threads used to chase the shapes with traces.

- If set to `0`, the number of threads to use for shape chasing is computed automatically.
- If set to `1`, a single thread is used for chasing the shapes with traces.
- If set to `n`, the specified number of threads are used to chase the shapes with traces.

The default is `0`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "netTracerNumThreads")
envSetVal("layoutXL" "netTracerNumThreads" 'int 4)
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerOn

```
layoutXL netTracerOn boolean { t | nil }
```

Description

Enables all the features related to the Net Tracer functionality—the *Net Tracer* command in the *Connectivity* menu, the *Net Tracer* toolbar, and all the related forms and features. See Tracing Nets for more information.

The default is `t`.

GUI Equivalent

Command: *Connectivity – Net Tracer – Add/Remove*

Examples

```
envGetVal("layoutXL" "netTracerOn")
envSetVal("layoutXL" "netTracerOn" 'boolean t)
envSetVal("layoutXL" "netTracerOn" 'boolean nil)
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerPopupSelectionWindow

```
layoutXL netTracerPopupSelectionWindow boolean { t | nil }
```

Description

Controls the display of the *Choose LPP to Trace* pop-up box when you click at multiple overlapping shapes while creating a trace. The pop-up box allows you to choose the layer purpose pair to use for creating the trace.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Net Tracer – Add/Remove*
Field:	*Pop up selection window* (Net Tracer Options Form)

Examples

```
envGetVal("layoutXL" "netTracerPopupSelectionWindow")
envSetVal("layoutXL" "netTracerPopupSelectionWindow" 'boolean t)
envSetVal("layoutXL" "netTracerPopupSelectionWindow" 'boolean nil)
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerRange

```
layoutXL netTracerRange cyclic { "Top to bottom" | "Top to current" | "Current to
     bottom" | "User defined" | "Top to stop level" | "Current to stop level" }
```

Description

Controls how deep in the hierarchy a trace is created.

The default is `"Top to bottom"`.

Choose:

- *Top to bottom* to create a trace on the selected net that passes through the entire hierarchy of the cellview.
- *Current to bottom* to create a trace on the selected net that passes from the current level of hierarchy to the level specified in the *Display Level To* field in the Display Options form (stop level).
- *Top to current* to create a trace on the selected net that passes from the level specified in the *Display Level From* field in the Display Options Form (start level) to the current level of hierarchy.
- *Top to stop level* to create a trace on the selected net that passes from the top level in the hierarchy to the level specified in the *Display Levels Stop* field in the Display Options form (stop level).
- *Current to stop level* to create a trace on the selected net that passes from the current level in the hierarchy to the level specified in the *Display Levels Stop* field in the Display Options Form (stop level).
- *User defined* to enable the *Start* and *Stop* fields. Use these fields to manually set the hierarchy start and stop levels for tracing nets.

GUI Equivalent

Command:	*Connectivity – Net Tracer – Add/Remove command*
Field:	*Hierarchy Range* (Net Tracer Options Form)

Examples

```
envGetVal("layoutXL" "netTracerRange")
envSetVal("layoutXL" "netTracerRange" 'cyclic "Top to bottom")
```


© 2023 Cadence Design Systems, Inc.

```
envSetVal("layoutXL" "netTracerRange" 'cyclic "Top to current")
envSetVal("layoutXL" "netTracerRange" 'cyclic "Current to stop level")
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerRegion

```
layoutXL netTracerRegion cyclic { "Entire cellview" | "Viewing area" | "Custom" }
```

Description

Controls the layout area in which nets are traced.

The default is `"Entire cellview"`.

Choose:

- *Entire cellview* to trace nets in the entire cellview.
- *Viewing area* to trace nets in the portion of the cellview that is currently displayed in the canvas.
- *Custom* to specify a rectangular or polygonal area for tracing nets.

GUI Equivalent

Command:	*Connectivity – Net Tracer – Add/Remove command*
Field:	*Region* (Net Tracer Options Form)

Examples

```
envGetVal("layoutXL" "netTracerRegion")
envSetVal("layoutXL" "netTracerRegion" 'cyclic "Entire cellview")
envSetVal("layoutXL" "netTracerRegion" 'cyclic "Viewing area")
envSetVal("layoutXL" "netTracerRegion" 'cyclic "Custom")
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerSaveMethod

```
layoutXL netTracerSaveMethod cyclic { "One view per trace" | "One view for all
     traces" | "Auto" }
```

Description

Specifies the method used for saving visible traces in a design.

- `One view per trace`: Saves each visible trace in a separate view. This helps to optimize the performance in case of a large trace with lots of figures.
- `One view for all traces`: Saves the visible traces into a single view.
- `Auto`: Lets the system to automatically determine the fastest method to be used for saving traces.
  - If only one trace is visible, the `One view per trace` method is used.
  - If at least one visible trace is large, the `One view per trace` method is used.
  - If all visible traces are small, the `One view for all traces` method is used.

The default value is `Auto`, implying that the system determines the method to be used to save traces.

GUI Equivalent

Command:	*Connectivity – Net Tracer – Save All Visible Traces*
Field:	*Method*

Examples

```
envGetVal("layoutXL" "netTracerSaveMethod")
envSetVal("layoutXL" "netTracerSaveMethod" 'cyclic "One view for all traces" )
```

Related Topics

Saving All Visible Traces

Load Traced Nets View Form

© 2023 Cadence Design Systems, Inc.

netTracerShowPreview

```
layoutXL netTracerShowPreview boolean { t | nil }
```

Description

Controls whether dynamic tracing is performed. Traces are displayed according to the selected *Display Mode: True Color*, *True Color & Halo*, or *Halo*.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Net Tracer – Add/Remove*
Field:	*Show preview* (Net Tracer Options Form)

Examples

```
envGetVal("layoutXL" "netTracerShowPreview")
envSetVal("layoutXL" "netTracerShowPreview" 'boolean t)
envSetVal("layoutXL" "netTracerShowPreview" 'boolean nil)
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerStartLevel

`layoutXL netTracerStartLevel int` *`integer`*

Description

Sets the hierarchy level at which tracing is started for a design.

The default is `0`.

Note: This environment variable takes effect only when netTracerRange is set to "`User defined`".

GUI Equivalent

Command:	*Connectivity – Net Tracer – Add/Remove*
Field:	*Hierarchy Range – User Defined – Start* (Net Tracer Options Form)

Examples

```
envGetVal("layoutXL" "netTracerStartLevel")
envSetVal("layoutXL" "netTracerStartLevel" 'int 32)
envSetVal("layoutXL" "netTracerStartLevel" 'int 3)
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerStepMode

```
layoutXL netTracerStepMode boolean { t | nil }
```

Description

Controls whether a trace is progressively highlighted on the layout canvas, with each step tracing an additional set of shapes.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Net Tracer – Add/Remove*
Field:	*Step mode*

Examples

```
envGetVal("layoutXL" "netTracerStepMode")
envSetVal("layoutXL" "netTracerStepMode" 'boolean t)
envSetVal("layoutXL" "netTracerStepMode" 'boolean nil)
```

Related Topics

Net Tracer Options Form

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerStopLayer

```
layoutXL netTracerStopLayer boolean { t | nil }
```

Description

Controls whether stop layers defined on the Net Tracer Custom Setup Form form are traced through.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Net Tracer – Add/Remove*
Field:	*Use custom stop layers* (Net Tracer Options Form)

Examples

```
envGetVal("layoutXL" "netTracerStopLayer")
envSetVal("layoutXL" "netTracerStopLayer" 'boolean t)
envSetVal("layoutXL" "netTracerStopLayer" 'boolean nil)
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerStopLevel

`layoutXL netTracerStopLevel int` *`integer`*

Description

Sets the hierarchy level at which tracing is stopped for a design.

The default is `0`.

Note: This environment variable takes effect only when netTracerRange is set to "`User defined`".

GUI Equivalent

Command:	*Connectivity – Net Tracer – Add/Remove*
Field:	*Hierarchy Range – User Defined – Stop* (Net Tracer Options Form)

Examples

```
envGetVal("layoutXL" "netTracerStopLevel")
envSetVal("layoutXL" "netTracerStopLevel" 'int 20)
envSetVal("layoutXL" "netTracerStopLevel" 'int 3)
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerUpdateIncrementally

```
layoutXL netTracerUpdateIncrementally boolean { t | nil }
```

Description

Controls whether traced nets for connected shapes are highlighted in the increments of 10K shapes.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Net Tracer – Add/Remove*
Field:	*Highlight incrementally* (Net Tracer Options Form)

Examples

```
envGetVal("layoutXL" "netTracerUpdateIncrementally")
envSetVal("layoutXL" "netTracerUpdateIncrementally" 'boolean t)
envSetVal("layoutXL" "netTracerUpdateIncrementally" 'boolean nil)
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

netTracerViaLayer

```
layoutXL netTracerViaLayer boolean { t | nil }
```

Description

Controls whether via layers defined on the Net Tracer Custom Setup Form form are traced through.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Net Tracer – Add/Remove*
Field:	*Use custom via layer* (Net Tracer Options Form)

Examples

```
envGetVal("layoutXL" "netTracerViaLayer")
envSetVal("layoutXL" "netTracerViaLayer" 'boolean t)
envSetVal("layoutXL" "netTracerViaLayer" 'boolean nil)
```

Related Topics

Tracing Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

openConnRef

```
layoutXL openConnRef boolean { t | nil }
```

Description

Specifies whether or not the source schematic cellview is opened when you use the *Edit In Place* or *Descend* commands.

When switched off, the layout cellview is still opened in Layout XL mode, but with no connectivity reference.

GUI Equivalent

Command:	*Options – Connectivity* (*General* tab)
Field:	*Open connectivity reference during Edit In Place/Descend*

Examples

```
envGetVal("layoutXL" "openConnRef")
envSetVal("layoutXL" "openConnRef" 'boolean t)
envSetVal("layoutXL" "openConnRef" 'boolean nil)
```

Related Topics

Connectivity Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

openConnRefTab

```
layoutXL openConnRefTab boolean { t | nil }
```

Description

Specifies whether or not the source cellview is opened in a new tab when you use the *Edit In Place* or *Descend* commands.

The default is `nil`.

Note: In the *Edit In Place* or *Descend* mode, if you run a command that needs the source schematic to be open, the environment variable automatically opens the schematic and launches the binder, if the connectivity reference is specified.

GUI Equivalent

Command:	*Options – Connectivity* (*General* tab)
Field:	*Open connectivity reference during Edit In Place/Descend:*
	■ *Open in new window*
	■ *Open in new tab*

Examples

```
envGetVal("layoutXL" "openConnRefTab")
envSetVal("layoutXL" "openConnRefTab" 'boolean t)
envSetVal("layoutXL" "openConnRefTab" 'boolean nil)
```

Related Topics

Net Tracer Options Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

openLocation

```
ab openLocation string { "current tab" | "new tab" | "new window" }
```

Description

Specifies whether a cellview descended into directly from the Annotation Browser assistant is opened in a new tab, the current tab, or a new window.

The default is `"current tab"`.

Arguments

`current tab`	The descended cellview is opened in the current tab.
`new tab`	`The descended cellview is opened in a new tab.`
`new window`	The descended cellview is opened in a new layout window.

GUI Equivalent

Command:	*Windows – Assistants – Annotation Browser – Options* (RMB)
Field:	*Default Open – Open in* (Navigator)

Examples

```
envGetVal("ab" "openLocation")
envSetVal("ab" "openLocation" 'string "current tab")
envSetVal("ab" "openLocation" 'string "new tab")
envSetVal("ab" "openLocation" 'string "new window")
```

Related Topics

Annotation Browser Assistant

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

openMode

```
ab openMode string { "edit" | "read" }
```

Description

Specifies whether a cellview descended into directly from the Annotation Browser assistant is opened in edit mode or read mode.

The default is `"edit"`.

Arguments

`edit`	The descended cellview is opened in edit mode.
`read`	The descended cellview is opened in read mode.

GUI Equivalent

Command:	*Windows – Assistants – Annotation Browser – Options* (RMB)
Field:	*Default Open – Mode* (Navigator)

Examples

```
envGetVal("ab" "openMode")
envSetVal("ab" "openMode" 'string "edit")
envSetVal("ab" "openMode" 'string "read")
```

Related Topics

Annotation Browser Assistant

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

opticalPinSetup (Virtuoso Photonics Option)

`layoutXL opticalPinSetup string "`*`constraint_group_name`*`"`

Description

Specifies the name of the constraint group to use for determining the layers available for generating the optical pins.

The default is `"virtuosoDefaultOpticalPinSetup"`.

Note: For designs that have both optical and electrical signals, if the `virtuosoDefaultOpticalPinSetup` constraint group is not defined, the `validLayers` defined in the default Layout XL constraint group, `virtuosoDefaultSetup`, are used for optical pin generation. However, if the `virtuosoDefaultElectricalPinSetup` constraint group is defined, the layers specified in the constraint group are used for electrical pin generation.

Note:



To use this environment variable in Layout XL, you must have the `Virtuoso_Photonics_Option` license checked out. For information on obtaining the required license, contact your local Cadence representative.

GUI Equivalents

None

Examples

```
envGetVal("layoutXL" "opticalPinSetup")
envSetVal("layoutXL" "opticalPinSetup" 'string "virtuosoDefaultOpticalPinSetup")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

paramsToCheck

```
layoutXL paramsToCheck string "list_of_parameters"
```

Description

Specifies the parameters to be checked by the *Check Against Source* command.

The default is `""`, which means the parameters to check are not specified. In this case, the paramsToIgnore list is used.



Important

Cadence recommends that the parameters to check be defined in the LAM file. The environment variable should be used only if an exhaustive list of parameters to be checked is provided.

Note:

- If the paramsToCheck environment variable is set, it takes precedence over the paramsToIgnoreForCheck environment variable. This means, when a list of *parameters to check* is specified, it is considered, even when a list of *parameters to ignore for check* is also specified.
- If the list of parameters to check is specified using the LAM file, the CPH window, or using the *Connectivity – Parameters to check* option, setting the paramsToCheck environment variables is not required.
- When the *parameters to check* list is set up, the parameters in the list are the only ones that are updated by the Update Layout Parameters or Update Schematic Parameters commands.

© 2023 Cadence Design Systems, Inc.

Arguments

list_of_parameters

A list of parameter names each separated by a space. The list must be enclosed in quotation marks.

GUI Equivalent

Command:	*Options – Connectivity* (*Parameters* tab)
Field:	*Parameters to check* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "paramsToCheck")
envSetVal("layoutXL" "paramsToCheck" 'string "fw w l")
```

Related Topics

Specification To Check Parameters

Parameters

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

paramsToIgnore

```
layoutXL paramsToIgnore string "list_of_parameters"
```

Description

Specifies the parameters that are ignored by the following commands.

- *Generate All From Source*
- *Generate Selected From Source*
- *Generate Clones*
- *Check Against Source*
- *Update Components And Nets*
- *Update Layout Parameters*
- *Update Schematic Parameters*
- *Backannotation*

The default is `""`.

Note: Alternatively, you can update the LAM file to specify the parameters to be ignored. See LAM File Logical Section.

The following parameters are always ignored.

```
instancesLastChanged          lxPlacementStatus
instNamePrefix                lxRounding
lxIgnoreParamForCAS           lxStopList
lxIgnoredParams               lxTimeStamp
lxMFactorNum                  lxUseCell
lxParamsToIgnore              pin#
lxParamsToIgnoreForCheck      posi
```


© 2023 Cadence Design Systems, Inc.

Arguments

list_of_parameters

A list of parameter names each separated by a space. The list must be enclosed in quotation marks.

GUI Equivalent

Command: *Options – Connectivity* (*Parameters* tab)

Field:
- *Parameters to ignore during check*
- *Parameters to ignore during generation and update*

Examples

```
envGetVal("layoutXL" "paramsToIgnore")
envSetVal("layoutXL" "paramsToIgnore" 'string "effW w myParam")
```

Related Topics

Connectivity Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

paramsToIgnoreForCheck

```
layoutXL paramsToIgnoreForCheck string "list_of_parameters"
```

Description

Lists the parameters to be ignored during the *Check Against Source* command. Mismatches for any of the listed parameters are not reported by the check.

The default is `""`.

The list automatically includes all the parameters specified by the paramsToIgnore environment variable.

Arguments

list_of_parameters

A list of parameter names each separated by a space. The list must be enclosed in quotation marks.

GUI Equivalent

Command:	*Options – Connectivity* (*Parameters* tab)
Field:	*Parameters to ignore during check*

Examples

```
envGetVal("layoutXL" "paramsToIgnoreForCheck")
envSetVal("layoutXL" "paramsToIgnoreForCheck" 'string "w l myParam")
```

Related Topics

Connectivity Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

paramTolerance

`layoutXL paramTolerance float` *`floating_point_number`*

Description

Specifies the relative tolerance allowed when comparing values between the layout and the schematic.

This tolerance is applied by any function that deals with parameters, including the following commands in the *Connectivity* menu.

- *Generate All From Source*
- *Generate Folded Devices*
- *Check Against Source*
- *Update Layout Parameters*
- *Update Schematic Parameters*

The default is `1e-6`.

Arguments

`floating_point_number`

A float specifying the relative tolerance applied when comparing parameter values; for example,

`1e-6`

Note: The environment variable does not accept `0` as a valid value.

GUI Equivalent

Command:	*Options – Connectivity* (*Parameters* tab)
Field:	*Tolerance*

Examples

```
envGetVal("layoutXL" "paramTolerance")
```


© 2023 Cadence Design Systems, Inc.

```
envSetVal("layoutXL" "paramTolerance" 'float 1e-06)
```

Related Topics

Connectivity Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

passFigIdsToEvent3

```
layoutXL passFigIdsToEvent3 boolean { t | nil }
```

Description

Passes the IDs of Pcell pinFigs involved in abutment to abutment Event 3.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "passFigIdsToEvent3")
envSetVal("layoutXL" "passFigIdsToEvent3" 'boolean t)
envSetVal("layoutXL" "passFigIdsToEvent3" 'boolean nil)
```

Related Topics

Abutment Function Flow

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

phoAbutClass (Virtuoso Photonics Option)

```
layoutXL phoAbutClass string "valid_abutment_class"
```

Description

Specifies that two optical pins can abut only when they belong to the same abutment class. Examples of abutment classes that can be used for abutting pins include the pin layer, pin edge, and so on.

The default is "", which means that the pin layer is used to get the `abutClass` property because the property is neither set on the Pcell, nor defined by this environment variable.

Note: If the Pcell or non-Pcell instances do not have the abutClass Property abutment property set on the abutting pinFigs, the value set for the `phoAbutClass` environment variable is used for abutment.

Important

To use this environment variable in Layout XL, you must have the `Virtuoso_Photonics_Option` license checked out. For information on obtaining the required license, contact your local Cadence representative.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "phoAbutClass")
envSetVal("layoutXL" "phoAbutClass" 'string "pin_layer")
```

Related Topics

Abutment in Virtuoso Photonics Solution (Virtuoso Photonics Option)

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

phoAbutFunction (Virtuoso Photonics Option)

`layoutXL phoAbutFunction string "`*`valid_function_name`*`"`

Description

Specifies a user-defined callback to process abutment and unabutment.

The default is "`phoWGAbut`", which means that this function is used to abut Pcells, non-Pcells, or both in photonic designs.

Note: If the Pcell or non-Pcell instances do not have the abutFunction Property abutment property set on the abutting pinFigs, the value set for the `phoAbutFunction` environment variable is used for abutment.

Important

To use this environment variable in Layout XL, you must have the `Virtuoso_Photonics_Option` license checked out. For information on obtaining the required license, contact your local Cadence representative.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "phoAbutFunction")
envSetVal("layoutXL" "phoAbutFunction" 'string "phoWGAbut")
```

Related Topics

Abutment in Virtuoso Photonics Solution (Virtuoso Photonics Option)

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

phoComposeMaster (Virtuoso Photonics Option)

```
layoutXL phoComposeMaster string "master_cellview_name"
```

Description

Specifies the library cellview of the cell master to be used for creating the composite waveguide.

The default is " "

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "phoComposeMaster")
envSetVal("layoutXL" "phoComposeMaster" 'string "pholib waveMaster layout")
```

Related Topics

phoComposeMasterPromptOff (Virtuoso Photonics Option)

Abutment in Virtuoso Photonics Solution (Virtuoso Photonics Option)

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

phoComposeMasterPromptOff (Virtuoso Photonics Option)

```
layoutXL phoComposeMasterPromptOff boolean { t | nil }
```

Description

Uses the bound schematic instances, if available, to determine the master library cellview to use for creating the composite waveguide. If the bound schematic instances are not available, uses a previous selection to determine the master cellview to use.

The default is "`nil`", which means a form is raised that enables you to specify the master library cellview to be used.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "phoComposeMasterPromptOff")
envSetVal("layoutXL" "phoComposeMasterPromptOff" 'boolean t)
envSetVal("layoutXL" "phoComposeMasterPromptOff" 'boolean nil)
```

Related Topics

phoComposeMaster (Virtuoso Photonics Option)

Abutment in Virtuoso Photonics Solution (Virtuoso Photonics Option)

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

phoAbutNonPcells (Virtuoso Photonics Option)

```
layoutXL phoAbutNonPcells boolean { t | nil }
```

Description

Controls whether the abutment of photonic non-Pcell instances is supported.

The default is "`t`", which means:

- Top-level pins can abut with instance pins.
- Photonic non-Pcell instances can abut with both non-Pcell and Pcell instances.

Important

To use the `phoAbutNonPcells` environment variables and the other environment variables listed here, you must have the `Virtuoso_Photonics_Option` license checked out. For information on obtaining the required license, contact your local Cadence representative.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "phoAbutNonPcells")
envSetVal("layoutXL" "phoAbutNonPcells" 'boolean t)
envSetVal("layoutXL" "phoAbutNonPcells" 'boolean nil)
```

Related Topics

Abutment in Virtuoso Photonics Solution (Virtuoso Photonics Option)

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

phoPinInputAngle (Virtuoso Photonics Option)

`layoutXL phoPinInputAngle float` *`floating_point_number`*

Description

Specifies the default angle to use for generating the optical input pins in the layout.

The default is `90.0`.

Arguments

`floating_point_number`

A floating point number that specifies the default angle to use for creating the optical input pins in the layout.



Important

To use this environment variable in Layout XL, you must have the `Virtuoso_Photonics_Option` license checked out. For information on obtaining the required license, contact your local Cadence representative.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source – I/O Pins – Type – Optical*
Field:	*Input Angle*

Examples

```
envGetVal("layoutXL" "phoPinInputAngle")
envSetVal("layoutXL" "phoPinInputAngle" 'float 45.0)
```

Related Topics

Generation of RF TLine Chains (Virtuoso MultiTech Framework)

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

phoPinLayer (Virtuoso Photonics Option)

`layoutXL phoPinLayer string "`*`valid_photonic_layer_name`*`"`

Description

Specifies the default photonic layer to use for generating optical pins in the layout.

The default is "".

If no default layer is specified in the environment variable, the first photonic layer in the list of `validLayers` is used for optical pin generation.

Important

To use this environment variable in Layout XL, you must have the Virtuoso_Photonics_Option license checked out. For information on obtaining the required license, contact your local Cadence representative.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source – I/O Pins – Type – Optical*
Field:	*Layer*

Examples

```
envGetVal("layoutXL" "phoPinLayer")
envSetVal("layoutXL" "phoPinLayer" 'string "waveguide drawing")
```

Related Topics

Generation of RF TLine Chains (Virtuoso MultiTech Framework)

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

phoPinRadius (Virtuoso Photonics Option)

`layoutXL phoPinRadius float` *`floating_point_number`*

Description

Specifies the default radius to use for generating optical pins in the layout.

The default is `0.0`.

Arguments

`floating_point_number`

A floating point number that specifies the default radius for creating the circular optical pins in the layout.



Important

To use this environment variable in Layout XL, you must have the Virtuoso_Photonics_Option license checked out. For information on obtaining the required license, contact your local Cadence representative.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source – I/O Pins – Type – Optical*
Field:	*Radius*

Examples

```
envGetVal("layoutXL" "phoPinRadius")
envSetVal("layoutXL" "phoPinRadius" 'float 0.2)
```

Related Topics

Generation of RF TLine Chains (Virtuoso MultiTech Framework)

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

phoPinWidth (Virtuoso Photonics Option)

`layoutXL phoPinWidth float` *`floating_point_number`*

Description

Specifies the default width to use for generating optical pins in the layout.

The default is `0.0`.

Arguments

`floating_point_number`

A floating point number that specifies the default width for creating the optical pins in the layout.

Important

To use this environment variable in Layout XL, you must have the Virtuoso_Photonics_Option license checked out. For information on obtaining the required license, contact your local Cadence representative.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source – I/O Pins – Type – Optical*
Field:	*Width*

Examples

```
envGetVal("layoutXL" "phoPinWidth")
envSetVal("layoutXL" "phoPinWidth" 'float 0.2)
```

Related Topics

Generation of RF TLine Chains (Virtuoso MultiTech Framework)

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

physOnlyTerminalsRemoveBang

```
layoutXL physOnlyTerminalsRemoveBang boolean { t | nil }
```

Description

Removes the trailing “!” from the physical-only terminal names.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "physOnlyTerminalsRemoveBang")
envSetVal("layoutXL" "physOnlyTerminalsRemoveBang" 'boolean t)
envSetVal("layoutXL" "physOnlyTerminalsRemoveBang" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibCellName

```
layoutXL pibCellName string "Cell"
```

Description

Specifies the name of the target cell that contains the soft block in which shapes are to be pushed.

The default value is "", which indicates the current cell.

GUI Equivalent

Command	*Floorplan – Soft Block Update*
Field	*Push Into Blocks – Target View – Cell (*Push Into Blocks*)*

Examples

```
envGetVal("layoutXL" "pibCellName")
envSetVal("layoutXL" "pibCellName" 'string "myCell")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibCreateBlockageType

```
layoutXL pibCreateBlockageType cyclic { "routing" | "fill" }
```

Description

Sets the blockage type for routing objects, such as pins and wires, when they are pushed into blocks by the *Push Into Blocks* command. This environment variable applies to both *Push as Blockage* and *Push as Route* modes. You can set the blockage type to one of the following values:

- `routing`: Pushes routing objects as routing blockages.
- `fill`: Pushes routing objects as fill blockages.

In *Push as Blockage* mode, all the routing objects are pushed as blockage, irrespective of their connectivity statuses.

In *Push as Route* mode, if connectivity is absent for a top-level routing object inside a block, it is pushed in as a blockage. If connectivity is present, it is pushed in as a corresponding object.

The default value is `routing`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "pibCreateBlockageType")
envSetVal("layoutXL" "pibCreateBlockageType" 'cyclic "fill")
```

Related Topics

Push Into Blocks Form

Pushing Top-level Power Structures into Soft Blocks

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibCreatePinLPPChoice

```
layoutXL pibCreatePinLPPChoice cyclic {"Same LPP" | "Custom Purpose" | "Custom LPP"
     }
```

Description

Specifies the choice of LPP selection for pin shapes when the top-level power structures are pushed to the block level. The available options are:

- `Same LPP`: Specifies that pins are to be created on the same layers and purposes as the source pin shapes.
- `Custom Purpose`: Creates pins on the same layers as the source pin shapes. You can specify a purpose using pibPinPurposeName.
- `Custom LPP`: Creates pins on the specified layer and purpose. Use pibPinLPPName to specify the LPP.

The default value is `"Custom Purpose"`.

GUI Equivalent

Command	*Floorplan – Soft Block Update*
Field	*Push Into Blocks – Create Pin Layer Selection (*Push Into Blocks*)*

Examples

```
envGetVal("layoutXL" "pibCreatePinLPPChoice")
envSetVal("layoutXL" "pibCreatePinLPPChoice" 'cyclic "Custom LPP")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibFloatPurposeNames

```
layoutXL pibFloatPurposeNames string "floatingPurpose"
```

Description

Specifies the purposes of float shapes that can be pushed into soft blocks.

The default value is "", which indicates all purposes.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "pibFloatPurposeNames")
envSetVal("layoutXL" "pibFloatPurposeNames" 'string "fill actFill")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibLayerNames

```
layoutXL pibLayerNames string "layerNames"
```

Description

Specifies the layer in which pins are to be created when you run the Push Into Blocks command.

The default value is "", which indicates that by default, pins are created in the same LPP.

GUI Equivalent

Command	*Floorplan – Soft Block Update*
Field	*Push Into Blocks – Target View – Create Pin Layer Selection – Custom LPP (*Push Into Blocks*)*

Examples

```
envGetVal("layoutXL" "pibLayerNames")
envSetVal("layoutXL" "pibLayerNames" 'string "Metal1 Metal2")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibLibName

```
layoutXL pibLibName string "Library"
```

Description

Specifies the name of the target library that contains the soft block in which shapes are to be pushed.

The default value is "", which indicates the current library.

GUI Equivalent

Command	*Floorplan – Soft Block Update*
Field	*Push Into Blocks – Target View – Library (*Push Into Blocks*)*

Examples

```
envGetVal("layoutXL" "pibLibName")
envSetVal("layoutXL" "pibLibName" 'string "myLib")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibPinLPPName

```
layoutXL pibPinLPPName string "LPP"
```

Description

Specifies the LPP on which pins are to be created when the top-level power structures are pushed to the block level. This environment variable takes effect when pibCreatePinLPPChoice is set to `Custom LPP`.

The default value is `metal1 drawing`.

GUI Equivalent

Command	*Floorplan – Soft Block Update*
Field	*Push Into Blocks – Create Pin Layer Selection – Custom LPP (*Push Into Blocks*)*

Examples

```
envGetVal("layoutXL" "pibPinLPPName")
envSetVal("layoutXL" "pibPinLPPName" 'string "metal4 pin")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibPinPurposeName

`layoutXL pibPinPurposeName string "`*`Purpose`*`"`

Description

Specifies the purpose of the layer on which pins are to be created when the top-level power structures are pushed to the block level. This environment variable takes effect when pibCreatePinLPPChoice is set to `Custom Purpose`.

The default value is `drawing`.

GUI Equivalent

Command	*Floorplan – Soft Block Update*
Field	*Push Into Blocks – Create Pin Layer Selection – Custom Purpose (*Push Into Blocks*)*

Examples

```
envGetVal("layoutXL" "pibPinPurposeName")
envSetVal("layoutXL" "pibPinPurposeName" 'string "drawing")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibPurposeNames

```
layoutXL pibPurposeNames string ""
```

Description

In Layout EXL or higher tier, specifies the purposes of layers to be pushed into soft blocks.

The default value is "", which indicates that all purposes are selected.

GUI Equivalent

Command	*Floorplan – Soft Block Update*
Field	*Push Into Blocks – LPP Filter – LPP (*Push Into Blocks*)*

Examples

```
envGetVal("layoutXL" "pibPurposeNames")
envSetVal("layoutXL" "pibPurposeNames" 'string "pin boundary")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibPushAllAsBlockage

```
layoutXL pibPushAllAsBlockage boolean { t | nil }
```

Description

Pushes the routes that have connectivity in the block as blockages and not as routes with pins created on them.

The default value is `nil`, in which case routes are pushed into blocks as they are.

GUI Equivalent

Command	*Floorplan – Soft Block Update*
Field	*Push Into Blocks – Push All As Blockages (*Push Into Blocks*)*

Examples

```
envGetVal("layoutXL" "pibPushAllAsBlockage")
envSetVal("layoutXL" "pibPushAllAsBlockage" 'boolean t)
envSetVal("layoutXL" "pibPushAllAsBlockage" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibPushBlockage

```
layoutXL pibPushBlockage boolean { t | nil }
```

Description

Pushes blockages inside their instances.

The default value is `nil`, in which case blocks are not pushed.

GUI Equivalent

Command	*Floorplan – Soft Block Update*
Field	*Push Into Blocks – Push Blockage(s) (*Push Into Blocks*)*

Examples

```
envGetVal("layoutXL" "pibPushBlockage")
envSetVal("layoutXL" "pibPushBlockage" 'boolean t)
envSetVal("layoutXL" "pibPushBlockage" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibPushCloseWire

```
layoutXL pibPushCloseWire boolean { t | nil }
```

Description

Pushes the pre-route objects that are not overlapping the soft block, but are within close proximity, while respecting the minSpacing value.

The default value is `nil`.

GUI Equivalent

Command	*Floorplan – Soft Block Update*
Field	*Push Into Blocks – Push Close Outside Wire(s) As Blockage(s) (*Push Into Blocks*)*

Examples

```
envGetVal("layoutXL" "pibPushCloseWire")
envSetVal("layoutXL" "pibPushCloseWire" 'boolean t)
envSetVal("layoutXL" "pibPushCloseWire" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibPushFloats

```
layoutXL pibPushFloats boolean { t | nil }
```

Description

Specifies whether floating shapes are to be pushed into blocks.

The default value is `nil`, in which case they are not pushed.

GUI Equivalent

Command	*Floorplan – Soft Block Update*
Field	*Push Into Blocks – Net Filter – Float (*Push Into Blocks*)*

Examples

```
envGetVal("layoutXL" "pibPushFloats")
envSetVal("layoutXL" "pibPushFloats" 'boolean t)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibPushNonPowerGndNet

```
layoutXL pibPushNonPowerGndNet boolean { t | nil }
```

Description

Selects nets other than power and ground nets to be pushed into blocks.

The default value is `nil`, in which case all nets are selected.

GUI Equivalent

Command	*Floorplan – Soft Block Update*
Field	*Push Into Blocks – Net Selection – Regular (non-Power/Ground) Net(s) (*Push Into Blocks*)*

Examples

```
envGetVal("layoutXL" "pibPushNonPowerGndNet")
envSetVal("layoutXL" "pibPushNonPowerGndNet" 'boolean t)
envSetVal("layoutXL" "pibPushNonPowerGndNet" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibPushPowerGndNet

```
layoutXL pibPushPowerGndNet boolean { t | nil }
```

Description

Selects power and ground nets to be pushed into blocks.

The default value is `t`, in which case all nets are selected.

GUI Equivalent

Command	*Floorplan – Soft Block Update*
Field	*Push Into Blocks – Net Selection – Power - Ground Net(s) (*Push Into Blocks*)*

Examples

```
envGetVal("layoutXL" "pibPushPowerGndNet")
envSetVal("layoutXL" "pibPushPowerGndNet" 'boolean t)
envSetVal("layoutXL" "pibPushPowerGndNet" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibPushRows

```
layoutXL pibPushRows boolean { t | nil }
```

Description

Pushes standard and custom rows inside blocks. Overlapping rows and the rows that are completely inside the boundary are not pushed inside.

The default value is `nil`, which case all above rows are pushed inside blocks.

GUI Equivalent

Command	*Floorplan – Soft Block Update*
Field	*Push Into Blocks – Push Row(s) inside Soft Block(s) (*Push Into Blocks*)*

Examples

```
envGetVal("layoutXL" "pibPushRows")
envSetVal("layoutXL" "pibPushRows" 'boolean t)
envSetVal("layoutXL" "pibPushRows" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibPushMode

```
layoutXL pibPushMode cyclic { "Auto Push" | "Create Net & Push with Feed Thru Pins"
     | "Push as Blockage" }
```

Description

Specifies the mode for pushing shapes into soft blocks. The valid options are:

- `Auto Push`: Automatically pushes all shapes into the selected soft blocks.
- `Create Net & Push with Feed Thru Pins`: Creates missing nets before pushing all shapes into the selected soft blocks.
- `Push as Blockage`: Pushes all shapes as blockages. The default value is `Auto Push`.

GUI Equivalent

Command	*Floorplan – Soft Block Update*
Field	*Push Into Blocks – Net Filter (*Push Into Blocks*)*

Examples

```
envGetVal("layoutXL" "pibPushMode")
envSetVal("layoutXL" "pibPushMode" 'cyclic "Push as Blockage")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibPushSelection

```
layoutXL pibPushSelection boolean { t | nil }
```

Description

Determines whether the Push Into Blocks command can be run on selected nets.

The default is `nil`, which means the Push Into Blocks command runs on all the nets crossing the PR boundary.

When set to `t`, the Push Into Blocks command runs on the nets and top-level shapes selected in the Navigator assistant.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "pibPushSelection")
envSetVal("layoutXL" "pibPushSelection" 'boolean t)
envSetVal("layoutXL" "pibPushSelection" 'boolean nil)
```

Related

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibPushWhole

```
layoutXL pibPushWhole boolean { t | nil }
```

Description

Specifies that whole shapes that overlap the selected soft block must not be clipped when they are pushed into soft blocks.

The default value is `nil`.

GUI Equivalent

Command	*Floorplan – Soft Block Update*
Field	*Push Into Blocks – Options – Push Whole Shapes (*Push Into Blocks*)*

Examples

```
envGetVal("layoutXL" "pibPushWhole")
envSetVal("layoutXL" "pibPushWhole" 'boolean t)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibNetType

`layoutXL pibNetType string "`*`Net`*`"`

Description

In Layout EXL or higher tier, specifies the net shapes to be pushed into the selected soft block. The valid values are: `Power/Ground`, `Net`, and `Float`.

The default value is `All`, which indicates that power/ground, signal, and float nets are to be pushed into the block.

GUI Equivalent

Command	*Floorplan – Soft Block Update*
Field	*Push Into Blocks – Net Filter (*Push Into Blocks*)*

Examples

```
envGetVal("layoutXL" "pibNetType")
envSetVal("layoutXL" "pibNetType" "Power/Ground")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibReMasterInst

```
layoutXL pibReMasterInst boolean { t | nil }
```

Description

Re-masters the current soft block as per the target design.

The default value is `t`.

GUI Equivalent

Command	*Floorplan – Soft Block Update*
Field	*Push Into Blocks – Re-master Instance(s) With Target View (*Push Into Blocks*)*

Examples

```
envGetVal("layoutXL" "pibReMasterInst")
envSetVal("layoutXL" "pibReMasterInst" 'boolean t)
envSetVal("layoutXL" "pibReMasterInst" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibViewName

```
layoutXL pibViewName string "Library"
```

Description

Specifies the name of the target view that contains the soft block in which shapes are to be pushed.

The default value is "", which indicates the current view.

GUI Equivalent

Command	*Floorplan – Soft Block Update*
Field	*Push Into Blocks – Target View – View (*Push Into Blocks*)*

Examples

```
envGetVal("layoutXL" "pibViewName")
envSetVal("layoutXL" "pibViewName" 'string "myView")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

pibWSPActivePattern

```
layoutXL pibWSPActivePattern boolean { t | nil }
```

Description

Pushes top-level WSP patterns into sub-blocks such that the top-level WSP patterns and the soft block WSP patterns are aligned.

The default value is `nil`.

GUI Equivalent

Command	*Floorplan – Soft Block Update*
Field	*Push Into Blocks – Push Width Spacing Pattern (*Push Into Blocks*)*

Examples

```
envGetVal("layoutXL" "pibWSPActivePattern")
envSetVal("layoutXL" "pibWSPActivePattern" 'boolean t)
envSetVal("layoutXL" "pibWSPActivePattern" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

placeCloneAsArray

```
layoutXL placeCloneAsArray boolean { t | nil }
```

Description

Places the clones selected from the *Clones Found* list in the Generate Clones form as a row, column, or as an array.

The default is `nil`, which means the selected clones are placed one-at-a time.

See Clone Placement as an Array.

GUI Equivalent

Command:	*Connectivity – Generate Clones*
Field:	*Place Clone as Array* (Generate Clones Form)

Examples

```
envGetVal("layoutXL" "placeCloneAsArray")
envSetVal("layoutXL" "placeCloneAsArray" 'boolean t)
envSetVal("layoutXL" "placeCloneAsArray" 'boolean nil)
```

Related Topics

Clone Source Selection and Target Search and Generation

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

prBoundarySnap

```
layoutXL prBoundarySnap cyclic { "None" | "Lower Left" | "All Points" }
```

Description

Controls the snapping of PR boundary and the area boundaries when using the *Generate All From Source* command or the *Adjust Boundary* command.

The default is `"None"`, which means that no boundary snapping takes place.

When set to `"Lower Left"`, creates the PR boundary of the specified size and the lower-left point of the boundary bbox is snapped to the grid.

When set to `"All Points"`, snaps all points of the PR boundary bbox to the grid. This may result in the boundary bbox being larger than expected.

GUI Equivalent

Command:	*Options – Connectivity – Generation tab*
Field:	*Boundary snap*

Examples

```
envGetVal("layoutXL" "prBoundarySnap")
envSetVal("layoutXL" "prBoundarySnap" 'cyclic "Lower Left")
envSetVal("layoutXL" "prBoundarySnap" 'cyclic "All Points")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

prefixLayoutInstNamesWithPipe

```
layoutXL prefixLayoutInstNamesWithPipe boolean { t | nil }
```

Description

Controls the names of instances generated by Layout XL.

- When set to `t`, Layout XL prefixes all instance names with an OR bar: `|P0`
- When set to `nil`, Layout XL generates instance names without an OR bar: `P0`

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "prefixLayoutInstNamesWithPipe")
envSetVal("layoutXL" "prefixLayoutInstNamesWithPipe" 'boolean t)
envSetVal("layoutXL" "prefixLayoutInstNamesWithPipe" 'boolean nil)
```

Additional Information

Retaining the default value of `nil` for this environment variable facilitates better interoperability with the OpenAccess database and the Cadence® Innovus Digital Implementation System. You might need to update any SKILL code that relies on the OR bar in the names.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

preserveAreaBoundary

```
layoutXL preserveAreaBoundary boolean { t | nil }
```

Description

Specifies that any existing area boundaries in the layout are preserved when you run the *Generate All From Source* command.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*Floorplan* tab)
Field:	*Preserve Floorplanning Objects – Area Boundaries* (Generate Layout Form)

Examples

```
envGetVal("layoutXL" "preserveAreaBoundary")
envSetVal("layoutXL" "preserveAreaBoundary" 'boolean t)
envSetVal("layoutXL" "preserveAreaBoundary" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

preserveBlockages

```
layoutXL preserveBlockages boolean { t | nil }
```

Description

Specifies that any existing blockage objects in the layout are preserved when you run the *Generate All From Source* command.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*Floorplan* tab)
Field:	*Preserve Floorplanning Objects – Blockages* (Generate Layout Form)

Examples

```
envGetVal("layoutXL" "preserveBlockages")
envSetVal("layoutXL" "preserveBlockages" 'boolean t)
envSetVal("layoutXL" "preserveBlockages" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

preserveClusters

```
layoutXL preserveClusters boolean { t | nil }
```

Description

Specifies that any existing clusters in the layout are preserved when you run the *Generate All From Source* command.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*Floorplan* tab)
Field:	*Preserve Floorplanning Objects – Clusters* (Generate Layout Form)

Examples

```
envGetVal("layoutXL" "preserveClusters")
envSetVal("layoutXL" "preserveClusters" 'boolean t)
envSetVal("layoutXL" "preserveClusters" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

preserveClusterBoundaries

```
layoutXL preserveClusterBoundaries boolean { t | nil }
```

Description

Specifies that any existing cluster boundaries in the layout are preserved when you run the *Generate All From Source* command.

The default is `nil`.

Note: Cluster boundaries are preserved only if clusters are preserved.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*Floorplan* tab)
Field:	*Preserve Floorplanning Objects – Cluster Boundaries* (Generate Layout Form)

Examples

```
envGetVal("layoutXL" "preserveClusterBoundaries")
envSetVal("layoutXL" "preserveClusterBoundaries" 'boolean t)
envSetVal("layoutXL" "preserveClusterBoundaries" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

preservePRBoundary

```
layoutXL preservePRBoundary boolean { t | nil }
```

Description

Preserves any existing PR boundaries during a *Generate All From Source* run.

The default value is `nil`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source (Floorplan tab)*
Field:	*PR Boundary*

Examples

```
envGetVal("layoutXL" "preservePRBoundary")
envSetVal("layoutXL" "preservePRBoundary" 'boolean t)
```

Related Topics

Generate Layout Form

preserveSnapBoundary

© 2023 Cadence Design Systems, Inc.

preserveRows

```
layoutXL preserveRows boolean { t | nil }
```

Description

Specifies that any existing rows and custom placement areas in the layout are preserved when you run the *Generate All From Source* command.

`preserveRows` also preserves `rowRegionSpec` and row regions in the layout when *Generate All From Source* is run. For more information on row regions, see Layout Support for Row Regions in the *Virtuoso Placer User Guide*.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*Floorplan* tab)
Field:	*Preserve Floorplanning Objects – Rows and Custom Placement Areas* (Generate Layout Form)

Examples

```
envGetVal("layoutXL" "preserveRows")
envSetVal("layoutXL" "preserveRows" 'boolean t)
envSetVal("layoutXL" "preserveRows" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

preserveSnapBoundary

```
layoutXL preserveSnapBoundary boolean { t | nil }
```

Description

Preserves any existing snap boundary only if PR boundary is also being preserved during a *Generate All From Source* run.

The default value is `nil`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source (Floorplan tab)*
Field:	*Snap Boundary*

Examples

```
envGetVal("layoutXL" "preserveSnapBoundary")
envSetVal("layoutXL" "preserveSnapBoundary" 'boolean t)
```

Related Topics

Generate Layout Form

preservePRBoundary

© 2023 Cadence Design Systems, Inc.

preserveSnapPatterns

```
layoutXL preserveSnapPatterns boolean { t | nil }
```

Description

Specifies that any existing snap patterns and width spacing patterns are preserved when you run the *Generate All From Source* command.

Width spacing pattern constraints are always preserved.

The default is `nil`.

For more information on snap pattern shapes, see Snapping Objects to Local Snap Pattern Shapes in the *Virtuoso Layout Suite XL: Basic Editing User Guide*. For more information on width spacing patterns, see Using Width Spacing Patterns in *Virtuoso Width Spacing Patterns User Guide*.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*Floorplan* tab)
Field:	*Preserve Floorplanning Objects – Snap Patterns* (Generate Layout Form)

Examples

```
envGetVal("layoutXL" "preserveSnapPatterns")
envSetVal("layoutXL" "preserveSnapPatterns" 'boolean t)
envSetVal("layoutXL" "preserveSnapPatterns" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

preserveTrackPattern

```
layoutXL preserveTrackPattern boolean { t | nil }
```

Description

Specifies that any existing track patterns in the layout are preserved when you run the *Generate All From Source* command.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Generate – All From Source* (*Floorplan* tab)
Field:	*Preserve Floorplanning Objects – Track Patterns* (Generate Layout Form)

Examples

```
envGetVal("layoutXL" "preserveTrackPattern")
envSetVal("layoutXL" "preserveTrackPattern" 'boolean t)
envSetVal("layoutXL" "preserveTrackPattern" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

probeCycleHilite

```
layoutXL probeCycleHilite boolean { t | nil }
```

Description

Specifies whether the *XL Probe* command cycles through the `hilite drawing` entry layers used for showing probes.

The default is `t`.

GUI Equivalent

Command:	*Options – Highlight*
Field:	*Display Layer – Cycle* (Highlight Options)

Examples

```
envGetVal("layoutXL" "probeCycleHilite")
envSetVal("layoutXL" "probeCycleHilite" 'boolean t)
envSetVal("layoutXL" "probeCycleHilite" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

probeDevice

```
layoutXL probeDevice boolean { t | nil }
```

Description

Controls whether the *Probe* command highlights devices on the schematic and layout.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – XL Probe*
Field:	*Devices* (XL Probe Form)

Examples

```
envGetVal("layoutXL" "probeDevice")
envSetVal("layoutXL" "probeDevice" 'boolean t)
envSetVal("layoutXL" "probeDevice" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

probeDuringCreate

```
layoutXL probeDuringCreate boolean { t | nil }
```

Description

Controls whether or not nets that are tapped during interactive editing are automatically highlighted in the layout canvas.

GUI Equivalent

Command:	*Create – Shape – Rectangle*
	Create – Shape – Polygon
	Create – Shape – Path
	Create – Wiring – Wire
	Options – Connectivity (*Display* tab)
Field:	*Probe nets during object creation*

Examples

```
envGetVal("layoutXL" "probeDuringCreate")
envSetVal("layoutXL" "probeDuringCreate" 'boolean t)
envSetVal("layoutXL" "probeDuringCreate" 'boolean nil)
```

Related Topics

Create Rectangle Form

Create Polygon Form

Create Path Form

Create Wire Form

Connectivity Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

probeHiliteLayer

`layoutXL probeHiliteLayer string "`*`layer_purpose_pair`*`"`

Description

Specifies the layer used to highlight probed components in the canvas.

The default is "`hilite drawing`".

GUI Equivalent

Command:	*Connectivity – XL Probe – Probing Options*
Field:	*Display Layer* (Highlight Options)

Examples

```
envGetVal("layoutXL" "probeHiliteLayer")
envSetVal("layoutXL" "probeHiliteLayer" 'string "hilite drawing")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

probeInfoInCIW

```
layoutXL probeInfoInCIW boolean { t | nil }
```

Description

Sends the output of the *Probe* command to the CIW as well as the XL Probe form.

The default is `nil`, meaning that the output is visible only in the XL Probe form.

GUI Equivalent

Command:	*Connectivity – XL Probe*
Field:	*Send Messages to CIW* (XL Probe Form)

Examples

```
envGetVal("layoutXL" "probeInfoInCIW")
envSetVal("layoutXL" "probeInfoInCIW" 'boolean t)
envSetVal("layoutXL" "probeInfoInCIW" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

probeNet

```
layoutXL probeNet boolean { t | nil }
```

Description

Enables the *Probe* command to highlight nets in the schematic and layout.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – XL Probe*
Field:	*Nets* (XL Probe Form)

Examples

```
envGetVal("layoutXL" "probeNet")
envSetVal("layoutXL" "probeNet" 'boolean t)
envSetVal("layoutXL" "probeNet" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

probePin

```
layoutXL probePin boolean { t | nil }
```

Description

Enables the *Probe* command to highlight pins in the schematic and layout.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – XL Probe*
Field:	*Pins* (XL Probe Form)

Examples

```
envGetVal("layoutXL" "probePin")
envSetVal("layoutXL" "probePin" 'boolean t)
envSetVal("layoutXL" "probePin" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

processBatchViolations

```
layoutXL processBatchViolations boolean { t | nil }
```

Description

Controls whether or not the XL fixer engines, such as *auto-permute*, *auto-mirror, auto-spacing*, and *auto-abutment*, should process the violations during batch extraction.

Note:

The default is `nil`.

GUI Equivalent

Command:	*Options – Connectivity (Generation tab)*
Field:	*Process all violations during batch extraction* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "processBatchViolations")
envSetVal("layoutXL" "processBatchViolations" 'boolean t)
envSetVal("layoutXL" "processBatchViolations" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

propagateAssignNet

```
layoutXL propagateAssignNet boolean { t | nil }
```

Description

Propagates connectivity to physically connected shapes, vias, instance pins, and pathSegs during assignment of nets.

The default value is `nil`.

GUI Equivalent

Command	`leHiAssignNet` SKILL function – F3 key
Field	*Propagate connectivity*

Examples

```
envGetVal("layoutXL" "propagateAssignNet")
envSetVal("layoutXL" "propagateAssignNet" 'boolean t)
```

Related Topics

Assign Net Form

leHiAssignNet

© 2023 Cadence Design Systems, Inc.

propsUsedToIgnoreObjs

```
layoutXL propsUsedToIgnoreObjs string "list_of_properties"
```

Description

Lists the properties that cause pins and instances to be ignored by the following *Connectivity* commands. Any object with one of the listed properties set to `t` is ignored during these operations.

- *Generate All From Source*
- *Generate Selected From Source*
- *Generate Clones*
- *Check Against Source*
- *Update Components And Nets*
- *Update Layout Parameters*
- *Update Schematic Parameters*
- *Update Binding*

By default, the list contains the following properties: `"ignore lxRemoveDevice nlAction"`

Note: For information on the `nlAction` property, see Netlisting in the *Virtuoso NC Verilog Environment User Guide*.

© 2023 Cadence Design Systems, Inc.

Arguments

list_of_properties

A list of property names each separated by a space. The list must be enclosed in quotation marks; for example,

```
"ignore lxRemoveDevice  nlAction"
```

GUI Equivalent

Command:	*Options – Connectivity* (*Parameters* tab)
Field:	*Properties used to ignore objects during generation and update*

Examples

```
envGetVal("layoutXL" "propsUsedToIgnoreObjs")
envSetVal("layoutXL" "propsUsedToIgnoreObjs" 'string "ignore lxRemoveDevice
nlAction")
```

Related Topics

Connectivity Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

propsUsedToIgnoreObjsForCheck

```
layoutXL propsUsedToIgnoreObjsForCheck string "list_of_properties"
```

Description

Lists the properties used to ignore objects during the *Check Against Source* command. Any object with one of the listed properties set to `t` is ignored during this operation.

The default is lvsIgnore Property, but the list also inherits all the properties specified in the propsUsedToIgnoreObjs environment variable.

Arguments

list_of_properties

A list of property names each separated by a space. The list must be enclosed in quotation marks.

GUI Equivalent

Command:	*Options – Connectivity* (*Parameters* tab)
Field:	*Properties used to ignore objects during check*

Examples

```
envGetVal("layoutXL" "propsUsedToIgnoreObjsForCheck")
envSetVal("layoutXL" "propsUsedToIgnoreObjsForCheck" 'string "lvsIgnore")
```

Related Topics

Connectivity Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

removeDuplicateObjects

```
layoutXL removeDuplicateObjects boolean { t | nil }
```

Description

Checks and removes duplicate instances, shapes, and vias from the layout cellview.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*Remove – Duplicate objects* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "removeDuplicateObjects")
envSetVal("layoutXL" "removeDuplicateObjects" 'boolean t)
envSetVal("layoutXL" "removeDuplicateObjects" 'boolean nil)
```

Related Topics

Update Binding Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

renameClonesWhenFamilyIsRenamed

```
layoutXL renameClonesWhenFamilyIsRenamed boolean { t | nil }
```

Description

Renames the member clones with the new clone family name using the `<Clone Family Name>_index` nomenclature.

If the clone family is renamed to `myDiffPair` and `renameClonesWhenFamilyIsRenamed` is set to `t`, the member clones are renamed as `myDiffPair_0`, `myDiffPair_1`, `myDiffPair_2`, and so on.

When `renameClonesWhenFamilyIsRenamed` is set to `nil`, the clone members retain their original group name even when the clone family is renamed.

The default is `t`, which means the member clones are renamed when the family name is changed.

GUI Equivalent

Command:	*Connectivity – Update – Clone Families – Rename Family*
Field:	*Rename Clones* (Update Clone Families Form)

Examples

```
envGetVal("layoutXL" "renameClonesWhenFamilyIsRenamed")
envSetVal("layoutXL" "renameClonesWhenFamilyIsRenamed" 'boolean t)
envSetVal("layoutXL" "renameClonesWhenFamilyIsRenamed" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

renameConflictingLayoutInstances

```
layoutXL renameConflictingLayoutInstances boolean { t | nil }
```

Description

Controls whether the *Generate Selected From Source* command automatically renames existing instances in the layout with instance names that conflict with the schematic instances being transferred. The default value is `nil`.

When set to `t`, where a conflicting instance name exists, the existing instance in layout is renamed with the suffix `_old`, for example `I1` to `I1_old`. The newly generated layout instance matches the schematic instance, for example `I1`.

When set to `nil`, the existing instance in layout is unchanged and the transferred instance is generated in layout with an incremental suffix, for example `I1` to `I1_1`.

Note: For folded, m-factored, series-connected, or lxCombination instances the next unique number is used. For example, for the folded instance `I1.1`, the new instances in layout would be renamed to `I1.2`; for the series-connected instance `I1.s1`, new instances would be renamed `I1.s2`, and so on.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "renameConflictingLayoutInstances")
envSetVal("layoutXL" "renameConflictingLayoutInstances" 'boolean t)
envSetVal("layoutXL" "renameConflictingLayoutInstances" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

resistanceParamNames

```
layoutXL resistanceParamNames string "list_of_parameters"
```

Description

Lists the names of the schematic parameters used to specify resistance.

The default is `"r R"`.

The resistance value must be split among the generated series-connected devices during the following *Connectivity* commands.

- *Generate All From Source*
- *Check Against Source*
- *Update Layout Parameters*.

Layout XL checks each schematic instance for one of the listed parameter names and updates the resistance values of the matching layout parameters accordingly.

Arguments

list_of_parameters

A list of parameter names each separated by a space. The list must be enclosed in quotation marks; for example,

`"r R"`

GUI Equivalent

Command:	*Options – Connectivity* (*Parameters* tab)
Field:	*Resistance*

Examples

```
envGetVal("layoutXL" "resistanceParamNames")
envSetVal("layoutXL" "resistanceParamNames" 'string "r R")
```


© 2023 Cadence Design Systems, Inc.

Related Topics

Connectivity Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

saveViewSubType

```
layoutXL saveViewSubType boolean { t | nil }
```

Description

Automatically opens a Layout cellview in the same application in which it was last saved (`t`). The default is `t`.

When set to t, this overrides the default application set for the current cellview or all layout cellviews on the Set Default Application form or using the `maskLayoutDefaultApp` environment variable.

GUI Equivalent

None

Example

```
envGetVal ("layoutXL" "saveViewSubType" )
envSetVal ("layoutXL" "saveViewSubType" 'boolean t)
envSetVal ("layoutXL" "saveViewSubType" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

schematicWindow

```
layoutXL schematicWindow string "pairs_of_coordinates"
```

Description

Specifies the position and size of the schematic window on the screen.

By default, the schematic window is positioned in the bottom left corner of the screen.

Arguments

pair_of_coordinates

A pair of coordinates enclosed in quotation marks specifying the lower left and upper right corners of the schematic window; for example,

```
"((132.0 292.0) (565.0 992.0))"
```

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "schematicWindow")
envSetVal("layoutXL" "schematicWindow" 'string "((545.0 319.0) (1172.0 1160.0))")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

schLayLibraryPair

`layoutXL schLayLibraryPair string "`*`sch_lay_library_pair`*`"`

Description

Specifies the schematic and layout library pairs to be used when launching Layout XL or Layout EXL to create a new library cellview or to open an existing one.

The default value is `""`, which means the schematic and layout from the same library are used.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "schLayLibraryPair")
envSetVal("layoutXL" "schLayLibraryPair" 'string "((\"*_sch\" \"*_lay\"))")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

scopeLevel

```
ab scopeLevel string "Current Cellview Only"
```

Description

Controls the default scope level when the Annotation Browser is first opened in a window.

The default is `"Current Cellview Only"`.

Arguments

Current Cellview Only

Controls the default scope level within the current cellview only.

Current Cellview to Depth

Controls the default scope level up to the specified depth in the current cellview.

Controls the default scope level within the top cellview only.

Top Cellview to Depth

Controls the default scope level up to the specified depth in the top cellview.

GUI Equivalent

None

Examples

When setting the environment variable in .cdsenv:

```
ab scopeLevel string "Top Cellview To Depth"
```

When setting the environment variable in CIW:

```
envSetVal("ab" "scopeLevel" 'string "Top Cellview To Depth")
```


© 2023 Cadence Design Systems, Inc.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

segmentParamNames

```
layoutXL segmentParamNames string "segments"
```

Description

Enables a resistor to be implemented as a single instance that has one or many segments. The name of each segment is unique and can be set by using the environment variable.

The default segment name is “segments”.

GUI Equivalent

Command:	*Options – Connectivity* (*Parameters* tab)
Field:	*Schematic Parameter Names – Segments*

Examples

```
envGetVal("layoutXL" "segmentParamNames")
envSetVal("layoutXL" "segmentParamNames" 'string "segments Segments")
```

Related Topics

Connectivity Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

setPPConn

```
layoutXL setPPConn boolean { t | nil }
```

Description

Automatically identifies nodes that qualify as pseudoparallel connections and defines them during chaining and abutment. Contacts are dropped automatically, if appropriate. However, if the preserveTerminalContacts environment variable is switched on, the contacts are preserved.

Note: A pseudoparallel net connects nodes that are always the same voltage, so the current does not pass through the net.

The default is `t`.



Tip

If set to `nil`, you must define pseudoparallel nets explicitly using the *Define Pseudo Parallel Connect Pins* command.

GUI Equivalent

Command:	*Options – Connectivity* (*Generation* tab)
Field:	*Create pseudoparallel connections*

Examples

```
envGetVal("layoutXL" "setPPConn")
envSetVal("layoutXL" "setPPConn" 'boolean t)
envSetVal("layoutXL" "setPPConn" 'boolean nil)
```

Related Topics

Connectivity Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

sfactorNames

```
layoutXL sfactorNames string "list_of_names"
```

Description

Note: Lists the names of properties used in the schematic to specify the number of series-connected (`sfactor`) devices to be generated in the layout.

Layout XL checks each instance for a property matching one of the names on the list and uses the value of that property to generate the appropriate number of series-connected devices. The `sfactor` property value can be expressed as an integer or an expression.

The default is `"s S"`.

Note: This environment variable is also used by the Module Generator when adding dummy devices to MOSFETs or resistors with the modgenMakeMinDummies environment variable set to `t`.

Arguments

list_of_names

A list of sfactor property names each separated by a space. The list must be enclosed in quotation marks; for example,

`"s S"`

GUI Equivalent

Command:	*Options – Connectivity* (*Parameters* tab)
Field:	*Series-connected factor*

Examples

```
envGetVal("layoutXL" "sfactorNames")
envSetVal("layoutXL" "sfactorNames" 'string "s S")
```

Related Topics

Connectivity Form

© 2023 Cadence Design Systems, Inc.

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

sfactorSplit

```
layoutXL sfactorSplit boolean { t | nil }
```

Description

Controls whether Layout XL places schematic devices with the sfactor Property property as series-connected devices in the layout.

The default is `t`.

Note: You can override this environment variable for a given instance by setting the lxSfactorSplit Property property for the instance in question.

GUI Equivalent

Command:	*Options – Connectivity* (*Generation* tab)
Field:	*Split sfactored devices* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "sfactorSplit")
envSetVal("layoutXL" "sfactorSplit" 'boolean t)
envSetVal("layoutXL" "sfactorSplit" 'boolean nil)
```

Additional Information

This environment variable controls *only* whether Layout XL generates one device or several devices in the layout view. It does not change the widths of the generated devices.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

showDraglinesForDistantConns

```
layoutXL showDraglinesForDistantConns boolean { t | nil }
```

Description

Displays draglines for all the connections to the instance that is being manipulated. When switched off, draglines are shown only for the connections closest to the current instance.

This environment variable is honored by the *Generate Selected From Source*, *Generate Clones*, *Move*, and *Stretch* commands.



Tip

For more information on displaying draglines, see also colorDraglines and hideDraglinesForGlobalNets.

GUI Equivalent

Command:	*Options – Connectivity* (*Display* tab)
Field:	*Initial Display – Draglines – Show distant connections* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "showDraglinesForDistantConns")
envSetVal("layoutXL" "showDraglinesForDistantConns" 'boolean t)
envSetVal("layoutXL" "showDraglinesForDistantConns" 'boolean nil)
```

Additional Information

Displaying many draglines during interactive commands can impair the performance of Layout XL. To mitigate these effects, use the `maxDragFig` environment variable to limit the number of flightlines that can be added to a dragset.

To set `maxDragFig` interactively so that it takes effect in the current session, type the following in the CIW, where *`win`* is the window to which the limit applies.

```
win~>maxDragFig = 1000
```

To set a default value for `maxDragFig`, which will be used in all future Layout XL sessions, put the following line in your `.cdsenv` file.

© 2023 Cadence Design Systems, Inc.

```
envSetVal("layout" "maxDragFig" 'int 500)
```

Where, `500` is the default number of flightlines that can be added to a dragset.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

showIncompleteNets

```
layoutXL showIncompleteNets boolean { t | nil }
```

Description

On launch of Layout XL, displays all the incomplete nets in the design canvas.

The default is `nil`, which means the incomplete nets are not displayed when Layout XL is launched.

Examples

```
envGetVal("layoutXL" "showIncompleteNets")
envSetVal("layoutXL" "showIncompleteNets" 'boolean t)
envSetVal("layoutXL" "showIncompleteNets" 'boolean nil)
```

Related Topics

lxToggleShowAllIncompleteNets

Incomplete Nets

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

showPropagateConnectivityReport

```
layoutXL showPropagateConnectivityReport boolean { t | nil }
```

Description

Reports in CIW the number of shapes, vias, instance pins, and pathSegs to which connectivity is propagated when you assign a net by using the `leHiAssignNet` function. The report is printed when the `propagateAssignNet` environment variable is enabled or the *Propagate connectivity* check box is selected in the Assign Net form.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "showPropagateConnectivityReport")
envSetVal("layoutXL" "showPropagateConnectivityReport" 'boolean t)
```

Related Topics

leHiAssignNet

Assign Net Form

propagateAssignNet

© 2023 Cadence Design Systems, Inc.

showTabs

```
ab showTabs string { "tab1 tab2 tab3 tabNameX" | "all" }
```

Description

Controls the display of tabs in the Annotation Browser when the assistant is launched. You can choose to display the tabs selectively, or display all.

The default is `all`.

Note: You can also right-click any tab name and use the displayed shortcut menu to choose the tabs to be displayed. The shortcut menu also gives you an option to hide all the tabs. This will close all the tabs, except either the currently active tab or the one that is assigned the first position.

GUI Equivalent

Command:	*Right-click any Annotation Browser tab*
Field:	*Name of the tab to be displayed (Click the tab name)* Each tab name clicked in the shortcut menu displays the corresponding tab in the Annotation Browser.

Examples

```
envGetVal("ab" "showTabs")
envSetVal("ab" "showTabs" 'string { "Connectivity" "DRC/DFM" } )
```

Related Topics

Annotation Browser Shortcut Menus

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

showUnplacedInstances

```
layoutXL showUnplacedInstances cyclic { "Yes" | "No" | "Select" }
```

Description

Specify whether unplaced instances are to be displayed when you descend into a hierarchy. When set to `"Select"`, a popup message is displayed when you descend into a hierarchy. You can specify whether you want to show or hide all unplaced instances.

The default value is `"Select"`.

GUI Equivalent

Command	*Floorplan – Auto Generate Hierarchy*
Field	*Show unplaced instances on descend*

Examples

```
envGetVal("layoutXL" "showUnplacedInstances")
envSetVal("layoutXL" "showUnplacedInstances" 'cyclic "Yes")
envSetVal("layoutXL" "showUnplacedInstances" 'cyclic "No")
```

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

smartDeletePathSegBetweenVia

```
layoutXL smartDeletePathSegBetweenVia boolean { t | nil }
```

Description

Deletes a pathSeg up to the closest orthogonal grid of the nearest via. When there are vias on both sides of a pathSeg, the pathSeg is deleted up to the closest orthogonal grid of both the vias. When there are vias only on one side, the pathSeg is deleted up to the closest orthogonal grid of the nearest via and up to the end of the pathSeg on the other side.

The default is `nil`, implying that the entire pathSeg is deleted when you delete a pathSeg.

GUI Equivalent

Command:	*Edit – Delete*
Field:	*NA*

Examples

```
envGetVal("layoutXL" "smartDeletePathSegBetweenVia")
envSetVal("layoutXL" "smartDeletePathSegBetweenVia" 'boolean t)
```

Related Topics

Deleting Objects

© 2023 Cadence Design Systems, Inc.

snapCpaToPlacementGrid

```
layoutXL snapCpaToPlacementGrid boolean { t | nil }
```

Description

Snaps the Custom Placement Area to the placement grid if the placement grid is defined.

The default is `nil`.

When set to `nil`, the custom placement area is snapped to the X Snap and Y Snap spacing, provided the following conditions are fulfilled:

1. X Snap and Y Snap spacing are multiples of the manufacturing grid.
2. Both X Snap and Y Snap spacing are equal to or greater than the manufacturing grid.

If either of the snapping conditions is not met, the custom placement area is snapped to the manufacturing grid.

When set to `t`, the custom placement area is snapped to the placement grid. If the placement grid is unavailable, the conditions for snapping to the X/Y spacing are verified. If both the conditions are fulfilled, the placement area is snapped to the X-snap and Y-snap spacing. Else, the placement area is snapped to the manufacturing grid.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "snapCpaToPlacementGrid")
envSetVal("layoutXL" "snapCpaToPlacementGrid" 'boolean t)
envSetVal("layoutXL" "snapCpaToPlacementGrid" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

softBlockColor

```
layoutXL softBlockColor string "hilite_layer hilite_purpose"
```

Description

Specifies the highlight color for soft blocks in the layout canvas.

The default is `"hilite drawing1"`

Arguments

`hilite_layer hilite_purpose`

A list of parameter names separated by a space. The list must be enclosed in quotation marks; for example, "`hilite_layer hilite_drawing`".

GUI Equivalent

Command:	*Floorplan – Floorplan Global Options*
Field:	*Soft block color* (Floorplan Global Options)

Examples

```
envGetVal("layoutXL" "softBlockColor")
envSetVal("layoutXL" "softBlockColor" 'string "hilite drawing1")
```

Related Topics

Floorplan Global Options

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

stopList

```
layoutXL stopList string "list_of_viewNames"
```

Description

Specifies the default value for the *Physical stop view list* field when a new physical configuration is created.

The physical stop view list is used to determine the view at which the hierarchical traversal of a design stops; i.e., when traversing a hierarchy, the system stops when it encounters a view with one of the specified names.

The default is `"layout compacted symbolic"`.

Important

This environment variable is used only when creating a new physical configuration view. Changing the value has no effect on an existing physical configuration view.

Arguments

list_of_viewNames

A list of view names each separated by a space. The list must be enclosed in quotation marks; for example,

```
"layout compacted symbolic"
```

GUI Equivalent

Command:	*Launch – Configure Physical Hierarchy*
Field:	*Physical stop view list* (in the *Global Bindings* pane)
	(Configure Physical Hierarchy Window)

Examples

```
envGetVal("layoutXL" "stopList")
envSetVal("layoutXL" "stopList" 'string "layout compacted symbolic")
```


© 2023 Cadence Design Systems, Inc.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

stretchWithPRBoundary

```
layoutXL stretchWithPRBoundary boolean { t | nil }
```

Description

Stretches along with the PR boundary, the shapes intersecting an edge of a PR boundary or abutted to an edge of a PR boundary. The supported shapes are orthogonal pathSegs, rectilinear pins, rectangles and row regions.

The default is `nil`.

GUI Equivalent

Command	*Edit – Stretch*
Field	*Boundary: Stretch with P&R boundary (*Stretch Form*)*

Examples

```
envGetVal("layoutXL" "stretchWithPRBoundary")
envSetVal("layoutXL" "stretchWithPRBoundary" 'boolean t)
envSetVal("layoutXL" "stretchWithPRBoundary" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

substrateNetParamNames

```
layoutXL substrateNetParamNames string "parameterName"
```

Description

Lists the names of parameters that are used on layout devices to specify the net to which any extra terminals of the devices can be connected.

The default is "`sub sub_inh bn`", which means that the parameters are looked up in that order to determine the net to which an extra terminal on the layout device, such as the bulk terminal, can be connected. The value of the parameter must be a net name, such as `vdd!`, to which the extra terminal of a layout device can be connected.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "substrateNetParamNames")
envSetVal("layoutXL" "substrateNetParamNames" 'string "bn")
```

Related Topics

Abutment of Dummy Shapes in Pcells

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

suppressAbutFailMsgs

```
layoutXL suppressAbutFailMsgs boolean { t | nil }
```

Description

Suppresses all chaining and folding abutment fail messages.

The default is `nil`, which means that all messages are issued.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "suppressAbutFailMsgs")
envSetVal("layoutXL" "suppressAbutFailMsgs" 'boolean t)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

suppressExpansion

`ab suppressExpansion int` *`integer`*

Description

Specifies in 1000s the number of violations which prevents a node from being expanded in the Annotation Browser assistant. If the number of entries under a particular node exceeds the limit, the node is not expanded by default.

The default is `1`, which means that any node with more than 1000 violations under it is not expanded by default. The value you enter must lie in the range 0 through 1000.

Arguments

`integer`	Non-negative integer in the range 0 through 1000. For example, to suppress the expansion of nodes containing more than 5000 violations, set this environment variable to `5`.

GUI Equivalent

Command:	*Windows – Assistants – Annotation Browser – Options* (RMB)
Field:	*Suppress expansion at* (Navigator)

Examples

```
envGetVal("ab" "suppressExpansion")
envSetVal("ab" "suppressExpansion" 'int 2)
```

Additional Information

If you select markers in the canvas and their parent category is not expandable in the browser pane, you can still use the *Go To The Next Selected Marker* and *Go To The Previous Selected Marker* buttons in the Annotation Browser toolbar to cycle through the selected markers.

Related Topics

Annotation Browser Assistant

© 2023 Cadence Design Systems, Inc.

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

swapIncludeOrientation

```
layoutXL swapIncludeOrientation boolean { t | nil }
```

Description

Retains the original orientation of the instances when swapping the instances with the swap originating at the origin of the instance.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "swapIncludeOrientation")
envSetVal("layoutXL" "swapIncludeOrientation" 'boolean t)
envSetVal("layoutXL" "swapIncludeOrientation" 'boolean nil)
```

Related Topics

Swap Components Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

swapOrigin

```
layoutXL swapOrigin cyclic { "upperLeft" | "upperCenter" | "upperRight" |
     "centerLeft" | "centerCenter" | "centerRight" | "lowerLeft" | "lowerCenter" |
     "lowerRight" | "instOrigin" }
```

Description

Specifies whether the point of origin of the swap is at the instance origin, or at one of the specified orientations of the instance bounding box.

The default is `"instOrigin"`.

Arguments

None

GUI Equivalent

Command:	*Edit – Advanced – Swap Components*
Field:	*Swap Origin* (Swap Components Form)

Examples

```
envGetVal("layoutXL" "swapOrigin")
envSetVal("layoutXL" "swapOrigin" 'cyclic "centerRight")
```

Related Topics

Swap Components Form

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

tabPosition

```
ab tabPosition cyclic { "horizontal" | "vertical" }
```

Description

Controls the position of tabs in the Annotation Browser assistant.

The default is `horizontal`, which means that the Annotation Browser tabs are lined up horizontally at the top.

When set to `vertical`, the Annotation Browser tabs are lined up vertically on the left.

GUI Equivalent

None

Examples

```
envGetVal("ab" "tabPosition")
envSetVal("ab" "tabPosition" 'cyclic "horizontal")
envSetVal("ab" "tabPosition" 'cyclic "vertical")
```

Related Topics

Annotation Browser Assistant

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

transistorWidthParamNames

```
layoutXL transistorWidthParamNames string "list_of_parameters"
```

Description

Lists the names of the schematic parameters used to specify transistor width.

The default is `"w"`.

The transistor width value must be split among the generated parallel-connected devices during the following *Connectivity* commands.

- *Generate All From Source*
- *Check Against Source*
- *Update Layout Parameters*.

Layout XL checks each schematic instance for one of the listed parameter names and updates the width values of the matching layout parameters such that

```
number of mfactored instances * mfactorSplit value = source parameter value
```

Note: For cyclic split parameters, if the exact value is not matched, the closest value will be used instead.

This environment variable is also used by the Module Generator when adding dummy devices to MOSFETs or resistors with the modgenMakeMinDummies environment variable set to `t`.

© 2023 Cadence Design Systems, Inc.

Arguments

list_of_parameters

A list of parameter names each separated by a space. The list must be enclosed in quotation marks; for example,

```
"w"
```

GUI Equivalent

Command:	*Options – Connectivity* (*Parameters* tab)
Field:	*Transistor width* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "transistorWidthParamNames")
envSetVal("layoutXL" "transistorWidthParamNames" 'string "w")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

turnInfixOffWhenSmartSnapping

```
layoutXL turnInfixOffWhenSmartSnapping boolean { t | nil }
```

Description

Switches the Infix mode OFF when creating a ruler using the smart snapping options or when creating a wire.

The default is `t`.

Examples

```
envGetVal("layoutXL" "turnInfixOffWhenSmartSnapping")
envSetVal("layoutXL" "turnInfixOffWhenSmartSnapping" 'boolean t)
envSetVal("layoutXL" "turnInfixOffWhenSmartSnapping" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

unfoldUseLayoutWidths

```
layoutXL unfoldUseLayoutWidths boolean { t | nil }
```

Description

Unfolds a folded device to the total layout width.

The default value is `nil`, which means the device is folded back to the width of the schematic instance.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "unfoldUseLayoutWidths")
envSetVal("layoutXL" "unfoldUseLayoutWidths" 'boolean t)
envSetVal("layoutXL" "unfoldUseLayoutWidths" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

updateBindMfactorSplit

```
layoutXL updateBindMfactorSplit boolean { t | nil }
```

Description

Controls the update of *Split mfactored devices* option in the CPH window during a hierarchical Update Binding run.

The default value is `t`, which means the *Split mfactored devices* option is automatically updated if hierarchical binding cannot take place using the specified schematic mfactor value.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "updateBindMfactorSplit")
envSetVal("layoutXL" "updateBindMfactorSplit" 'boolean t)
envSetVal("layoutXL" "updateBindMfactorSplit" 'boolean nil)
```

Related Topics

Splitting Mfactored Devices

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

updateBindPhysicalBinding

```
layoutXL updateBindPhysicalBinding boolean { t | nil }
```

Description

Controls the update of physical bindings in the CPH window during a hierarchical Update Binding run.

The default value is `t`, which means the physical bindings in CPH are automatically updated if the current physical binding does not match the bound layout instance.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "updateBindPhysicalBinding")
envSetVal("layoutXL" "updateBindPhysicalBinding" 'boolean t)
envSetVal("layoutXL" "updateBindPhysicalBinding" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

updateBindUsePhysicalBinding

```
layoutXL updateBindUsePhysicalBinding boolean { t | nil }
```

Description

Binds schematic instances with those layout instances that have the master defined using the Configure Physical Hierarchy Window.

The default value is `t`.

GUI Equivalent

Command:	*Connectivity – Update – Binding*
Field:	*Use Physical bindings* (Update Binding Form)

Examples

```
envGetVal("layoutXL" "updateBindUsePhysicalBinding")
envSetVal("layoutXL" "updateBindUsePhysicalBinding" 'boolean t)
envSetVal("layoutXL" "updateBindUsePhysicalBinding" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

updateLayoutInstNames

```
layoutXL updateLayoutInstNames boolean { t | nil }
```

Description

Renames the layout instances during an *Update Components And Nets* run to match the instance names in the schematic view.

The default value is `t`.

When set to `nil`, the layout instances are not renamed to match their schematic counterparts.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "updateLayoutInstNames")
envSetVal("layoutXL" "updateLayoutInstNames" 'boolean t)
envSetVal("layoutXL" "updateLayoutInstNames" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

updateLayoutNetNames

```
layoutXL updateLayoutNetNames boolean { t | nil }
```

Description

Renames the layout nets during an *Update Components And Nets* run to match the net names in the schematic view.

The default value is `t`.

When set to `nil`, the layout nets are not renamed to match their schematic counterparts.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "updateLayoutNetNames")
envSetVal("layoutXL" "updateLayoutNetNames" 'boolean t)
envSetVal("layoutXL" "updateLayoutNetNames" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

updateLayoutParameters

```
layoutXL updateLayoutParameters boolean { t | nil }
```

Description

Specifies that the *Update Components And Nets* command updates the parameters and parameter values on layout instances to match those on their schematic counterparts.

Note: Parameters that are set in layout instances but are not present on their schematic counterparts are not removed.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Update – Components And Nets* (*Update* tab)
Field:	*Update Layout Parameters* (Update Components and Nets Form)

Examples

```
envGetVal("layoutXL" "updateLayoutParameters")
envSetVal("layoutXL" "updateLayoutParameters" 'boolean t)
envSetVal("layoutXL" "updateLayoutParameters" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

updateNetMinMaxVoltage

```
layoutXL updateNetMinMaxVoltage boolean { t | nil }
```

Description

Specifies that the minimum and maximum voltages assigned to nets in the schematic are transferred to the layout view by the *Generate All From Source* and *Update Components And Nets* commands.

Note: The *Update Components And Nets* command transfers only user-defined net voltages to the layout. The mismatches that may arise between the layout and the schematic due to this selective transfer of voltages are not reported by the Check Against Source command.

The default is `t`.

For information about how net voltage updates are controlled using this environment variable in the VDR flow, see Schematic Driven VDR Flow.

GUI Equivalent

Command:	*Connectivity – Update – Components And Nets* (*Update* tab)
Field:	*Update Net MinV/MaxV* (Update Components and Nets Form)

Examples

```
envGetVal("layoutXL" "updateNetMinMaxVoltage")
envSetVal("layoutXL" "updateNetMinMaxVoltage" 'boolean t)
envSetVal("layoutXL" "updateNetMinMaxVoltage" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

updateNetSigType

```
layoutXL updateNetSigType boolean { t | nil }
```

Description

Specifies that the signal types assigned to nets in the schematic are transferred to the layout view by the *Update Components And Nets* command.

Note: If enabled, terminals with nets that have a *Signal Type* of `power` or `ground` in the schematic are automatically given a *Route Method* of `geometric` when the *Update Components and Nets* command is run.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Update – Components And Nets* (*Update* tab)
Field:	*Update Net Signal Type* (Update Components and Nets Form)

Examples

```
envGetVal("layoutXL" "updateNetSigType")
envSetVal("layoutXL" "updateNetSigType" 'boolean t)
envSetVal("layoutXL" "updateNetSigType" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

updateNetsOnly

```
layoutXL updateNetsOnly boolean { t | nil }
```

Description

Specifies that only net assignments and instance, terminal, and net names are updated by the *Update Components And Nets* command.

Note: When set to `t`, user-defined bindings are preserved. In addition, any pins that have got swapped due to a schematic ECO have their original positions retained in the layout.

The default is `nil`.

GUI Equivalent

Command:	*Connectivity – Update – Components And Nets* (*Update* tab)
Field:	*Update Nets and Instance Name Mismatches Only* (Update Components and Nets Form)

Examples

```
envGetVal("layoutXL" "updateNetsOnly")
envSetVal("layoutXL" "updateNetsOnly" 'boolean t)
envSetVal("layoutXL" "updateNetsOnly" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

updateOneToOneMappings

```
layoutXL updateOneToOneMappings boolean { t | nil }
```

Description

Updates and reports any connectivity and parameter differences for user-defined one-to-one mappings of devices between the schematic and the layout.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "updateOneToOneMappings")
envSetVal("layoutXL" "updateOneToOneMappings" 'boolean t)
envSetVal("layoutXL" "updateOneToOneMappings" 'boolean nil)
```

Related Topics

Parameters

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

updateParamChangeNumFolds

```
layoutXL updateParamChangeNumFolds boolean { t | nil }
```

Description

Creates or deletes folds while updating layout parameters to fix differences in width.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "updateParamChangeNumFolds")
envSetVal("layoutXL" "updateParamChangeNumFolds" 'boolean t)
```

Related Topics

Updating Parameter Values for Folded Devices

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

updateParamCopyFolds

```
layoutXL updateParamCopyFolds boolean { t | nil }
```

Description

Specifies whether the *Update Layout Parameters* and *Update Components and Nets* commands generate a new fold with same parameter as the existing device.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "updateParamCopyFolds")
envSetVal("layoutXL" "updateParamCopyFolds" 'boolean t)
```

Related Topics

Updating Parameter Values for Folded Devices

Layout Parameters Updates

Components and Nets Updates

© 2023 Cadence Design Systems, Inc.

updateParamsForCheck

```
layoutXL updateParamsForCheck boolean { t | nil }
```

Description

Specifies that parameters that are ignored for check do not get updated during a schematic or layout update even if these parameters are not consistent across the two views.

The default is `t`, which means the *Connectivity – Update* command uses the paramsToIgnoreForCheck environment variable to determine the list of parameters to ignore during check.

When set to `nil`, the *Connectivity – Update* command uses the paramsToIgnore environment variable to determine the list of parameters to ignore for generation and update.

GUI Equivalent

Command:	*Options – Connectivity* (*Parameters* tab)
	(Connectivity Form)
Field:	*(Cyclic)*
	Update based on parameters to ignore for – generate
	Update based on parameters to ignore for – check

Examples

```
envGetVal("layoutXL" "updateParamsForCheck")
envSetVal("layoutXL" "updateParamsForCheck" 'boolean t)
envSetVal("layoutXL" "updateParamsForCheck" 'boolean nil)
```

Related Topics

Parameters

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

updatePlaceBelowBoundary

```
layoutXL updatePlaceBelowBoundary boolean { t | nil }
```

Description

Places instances below the PR boundary after an *Update Components And Nets* run, if the instances overlap other instances during an update. The overlapping instances are the ones that have changed in size, and if placed at their original location, may overlap other unchanged instances.

The default value is `nil`.

GUI Equivalent

Command:	*Connectivity – Update – Components And Nets* (*Update* tab)
Field:	*Move Changed Overlapping Instances Below PR Boundary* (Update Components and Nets Form)

Examples

```
envGetVal("layoutXL" "updatePlaceBelowBoundary")
envSetVal("layoutXL" "updatePlaceBelowBoundary" 'boolean t)
envSetVal("layoutXL" "updatePlaceBelowBoundary" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

updatePlacementStatus

```
layoutXL updatePlacementStatus boolean { t | nil }
```

Description

Automatically updates the placement status of instances and pins that are moved into or out of the PR boundary, including situations where the PR boundary is moved or stretched to enclose or exclude an instance or pin.

Instances and pins with status `unknown` that are wholly inside the PR boundary after the edit are updated to status `placed`. Instances and pins with status `placed` that are wholly outside the PR boundary after the edit are updated to `unknown`.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "updatePlacementStatus")
envSetVal("layoutXL" "updatePlacementStatus" 'boolean t)
envSetVal("layoutXL" "updatePlacementStatus" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

updatePlacementStatusInBoundary

```
layoutXL updatePlacementStatusInBoundary cyclic { "placed" | "firm" | "locked" }
```

Description

Sets the placement status of instances and pins that are moved into the PR boundary to the value set for `updatePlacementStatusInBoundary` unless the current status of the instances and pins is a higher constrained status.

Note:

- In terms of the constraint status, locked > firm > placed.
- The environment variable `updatePlacementStatusInBoundary` is only used if `updatePlacementStatus = t`.

The default is `"placed"`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "updatePlacementStatusInBoundary")
envSetVal("layoutXL" "updatePlacementStatusInBoundary" 'cyclic "placed")
envSetVal("layoutXL" "updatePlacementStatusInBoundary" 'cyclic "firm")
envSetVal("layoutXL" "updatePlacementStatusInBoundary" 'cyclic "locked")
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

updateReplacesMasters

```
layoutXL updateReplacesMasters boolean { t | nil }
```

Description

Updates any existing instances that use an incorrect master with instances that use the correct master during the *Update Components And Nets* command.

When set to `t`,

- With updateWithMarkers also set to `t`, the system puts a marker on the incorrect instance in the layout canvas and renames it *`name`*`_old`. It then creates a new instance with the correct master and places it below the design boundary.
- With `updateWithMarkers` set to `nil`, the system updates the instance in place to use the correct master.

When set to `nil`, the binding to the incorrect master is accepted with a warning message in the log file.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Update – Components And Nets* (*Update* tab)
Field:	*Update Instance Masters by (Replacing the Old* or *Creating a New) Instance* (Update Components and Nets Form)

Examples

```
envGetVal("layoutXL" "updateReplacesMasters")
envSetVal("layoutXL" "updateReplacesMasters" 'boolean t)
envSetVal("layoutXL" "updateReplacesMasters" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

updateSchParamWithEvalValue

```
layoutXL updateSchParamWithEvalValue boolean { t | nil }
```

Description

Updates the schematic parameter value with the layout parameter value, when the two values are found to be different during evaluation. The mismatch is also reported by the *Check – Against Source* command.

The default is `t`. For information about the behavior of the environment variable when set to `nil`, see Additional Information.

GUI Equivalent

Command:	*Options – Connectivity* (*Update* tab)
Field:	*Update schematic parameter expressions with evaluated value* (Connectivity Form)

Examples

```
envGetVal("layoutXL" "updateSchParamWithEvalValue")
envSetVal("layoutXL" "updateSchParamWithEvalValue" 'boolean t)
envSetVal("layoutXL" "updateSchParamWithEvalValue" 'boolean nil)
```

Additional Information

When the updateSchParamWithEvalValue environment variable is set to `nil`, the value of the schematic parameter is not updated to that of the layout parameter, even if the two evaluated values are found to be different and the schematic parameter is defined as an expression. The *Check – Against Source* command continues to report the parameter mismatch.

Let us consider an example. Let us assume that after an evaluation of the parameters, where needed, the schematic and the layout parameter values are found to be as displayed in the table below.

Schematic Parameters	Layout Parameters

© 2023 Cadence Design Systems, Inc.

`w = "3u"`	`w = "4u"*`
	** Value changed by the designer in the layout*
`width = iPar("w")`	`width = "3u"`

- With the updateSchParamWithEvalValue environment variable set to `t` (*default*), the schematic parameter values will be updated to match that of the layout:

  ```
  w = "4u"
  width = "3u"
  ```

  Tip

  To ensure that a schematic parameter value is not updated to match the corresponding layout value even when the updateSchParamWithEvalValue environment variable set to `t`, add the parameter to the list of parameters to ignore for check (paramsToIgnoreForCheck) or manually edit the parameters to match the layout counterparts.

- With the updateSchParamWithEvalValue environment variable set to `nil`, the schematic parameter values will continue to be:

  ```
  w = "3u"
  width = iPar("w")
  ```

  Note: When updateSchParamWithEvalValue is set to `nil` and lxEvalCDFCallbacks is set to `t`, all the callbacks are fired as usual after the schematic parameters that need an update (other than the ones that are defined as expressions) are updated.

  - It is assumed that the CDF callbacks are written such that they are compatible with any use of AEL expressions.

Conditions that Trigger Evaluation of Schematic Parameters

Schematic parameters (or properties) are considered for evaluation if:

- The shell environment variable, `CDS_Netlisting_Mode`, is set to `Analog`. Else, the schematic parameter expression, such as `iPar("w")`, is treated as a regular string and updated based on the value of the layout string, if the values across the two views are found to be different.
- If the parameter is a CDF parameter **and** `parseAsCEL` is set to `yes`

© 2023 Cadence Design Systems, Inc.

- a schematic parameter that only has `parseAsNumber = yes` and `parseAsCEL = no` is not considered as an evaluated parameter.

- If the parameter (or property) is an NLP expression, the parameter (or property) is always considered as evaluated, regardless of the value of the `CDS_Netlisting_Mode` environment variable.
- If the property is a string (not a CDF parameter) and the evaluated value is different from the unevaluated value.
- If the property is an `ilExpr`.

Related Topics

Connectivity form (*Parameters* tab)

Updating Schematic Parameter Values in Preselection Mode

Updating Schematic Parameter Values in Postselection Mode

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

updateSelectedComponents

```
layoutXL updateSelectedComponents boolean { t | nil }
```

Description

Specifies that the *Update Components And Nets* command updates only the instances and pins currently selected in the layout window.

The default is `nil`.

Note: This environment variable automatically disables the generation of new components. However, if updateWithMarkers is set to `t`, Layout XL may generate new instances to replace existing instances with incorrect masters; you have the option to switch on automatic chaining and folding for these new instances.

GUI Equivalent

Command:	*Connectivity – Update – Components And Nets* (*Update* tab)
Field:	*Update Selected Layout Components Only* (Update Components and Nets Form)

Examples

```
envGetVal("layoutXL" "updateSelectedComponents")
envSetVal("layoutXL" "updateSelectedComponents" 'boolean t)
envSetVal("layoutXL" "updateSelectedComponents" 'boolean nil)
```

Related Topics

Updating Selected Layout Components

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

updateWithMarkers

```
layoutXL updateWithMarkers boolean { t | nil }
```

Description

Specifies how the *Update Components And Nets* command updates an instance with an incorrect master.

- The default is `nil`, which means the command automatically removes the incorrect instance and replaces it with an instance of the correct master in the same location.
- When set to `t`, the command puts a marker on the instance with an incorrect master and renames it *`name`*`_old` during the *Update Components And Nets* command. It then creates a new instance with the correct master and places it below the PR boundary.

Important

This environment variable is honored only when updateReplacesMasters is set to `t`.

GUI Equivalent

Command:	*Connectivity – Update – Components And Nets* (*Update* tab)
Field:	*Update Instance Masters by (Replacing the Old* or *Creating a New) Instance*
	(Update Components and Nets Form)

Examples

```
envGetVal("layoutXL" "updateWithMarkers")
envSetVal("layoutXL" "updateWithMarkers" 'boolean t)
envSetVal("layoutXL" "updateWithMarkers" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

useConstantAreaStretch

```
layoutXL useConstantAreaStretch boolean { t | nil }
```

Description

Maintains a constant area for soft blocks, top-level PR boundaries, and virtual hierarchy when their boundaries are stretched.

The default is `nil`.

GUI Equivalent

Command:	*Edit – Stretch – press* `F3` (*Menu* bar)
Field:	Constant area stretch

Examples

```
envGetVal("layoutXL" "useConstantAreaStretch")
envSetVal("layoutXL" "useConstantAreaStretch" 'boolean t)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

useHierMfactorNames

```
layoutXL useHierMfactorNames boolean { t | nil }
```

Description

Enables the hierMfactorNames environment variable.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "useHierMfactorNames")
envSetVal("layoutXL" "useHierMfactorNames" 'boolean t)
envSetVal("layoutXL" "useHierMfactorNames" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

viewList

```
layoutXL viewList string "list_of_viewNames"
```

Description

Specifies the default value for the *Logical switch view list* field when a new physical configuration view is created.

The logical switch view list specifies the design views that are used to descend into a hierarchical design to find stop views (see stopList).

The default is `"schematic netlist symbol layout compacted symbolic"`.



Important

This environment variable is used only when creating a new physical configuration view. Changing the value has no effect on an existing physical configuration view.

Arguments

list_of_viewNames

A list of view names each separated by a space. The list must be enclosed in quotation marks; for example,

`"schematic netlist symbol layout compacted symbolic"`

GUI Equivalent

Command:	*Launch – Configure Physical Hierarchy*
Field:	*Logical switch view list* (in the *Global Bindings* pane)
	(Configure Physical Hierarchy Window)

Examples

```
envGetVal("layoutXL" "viewList")
envSetVal("layoutXL" "viewList" 'schematic netlist symbol layout compacted
symbolic")
```


© 2023 Cadence Design Systems, Inc.

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

vlGenAutoDetectCells

envSetVal("layoutXL" "vlGenAutoDetectCells" 'boolean nil)

Description

Automatically sets for VLGen all schematic symbols across the hierarchy that do not have their CDF parameters defined and have no existing layouts.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "vlGenAutoDetectCells")
envSetVal("layoutXL" "vlGenAutoDetectCells" 'boolean t)
envSetVal("layoutXL" "vlGenAutoDetectCells" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

vlGenCreateCells

```
layoutXL vlGenCreateCells boolean { t | nil }
```

Description

Enables the Virtuoso layout generator functionality for generating schematic symbols that have no existing layouts and no CDF parameters set.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "vlGenCreateCells")
envSetVal("layoutXL" "vlGenCreateCells" 'boolean t)
envSetVal("layoutXL" "vlGenCreateCells" 'boolean nil)
```

Related Topics

(Non)parameterized Layout Cells Generation (VLGen)

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

vlGenCreateCellsInPhysConfigLib

```
layoutXL vlGenCreateCellsInPhysConfigLib boolean { t | nil }
```

Description

Creates VLGen in the same library as the top-level design.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "vlGenCreateCellsInPhysConfigLib")
envSetVal("layoutXL" "vlGenCreateCellsInPhysConfigLib" 'boolean t)
envSetVal("layoutXL" "vlGenCreateCellsInPhysConfigLib" 'boolean nil)
```

Related Topics

(Non)parameterized Layout Cells Generation (VLGen)

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

vplGenCreateCells

```
layoutXL vplGenCreateCells boolean { t | nil }
```

Description

Enables the Virtuoso parameterized layout generator functionality.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "vplGenCreateCells")
envSetVal("layoutXL" "vplGenCreateCells" 'boolean nil)
```

Related Topics

Generating VPLGen Pcell Instances

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

vplGenLibs

```
layoutXL vplGenLibs string "list_of_libraries"
```

Description

Specifies a list of libraries and the order of lookup to find VPLGen core layouts than can be re-used when generating a VPLGen instance in the current design.

If a VPLGen core layout is found in one of the specified libraries, the system saves the mapping in your new VPLGen instance. If no VPLGen core layout is found, the system creates a new core layout in your current library.

Arguments

list_of_libraries

List of library names each separated by a space.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "vplGenLibs")
envSetVal("layoutXL" "vplGenLibs" 'string "lib1 lib2 lib3")
```

Related Topics

Generating VPLGen Pcell Instances

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

vplGenParams

```
layoutXL vplGenParams string "list_of_parameters"
```

Description

Specifies a list of default parameter names and values for Virtuoso parameterized layout generators (VPLGens). The list is used to populate the *VPLGen* field in the *Parameters* tab in the Configure Physical Hierarchy window. If your VPLGens all have the same parameters, you can set this environment variable to save you having to enter these each time in the field.

The default is "" (empty), which means you have to type the parameters and values into the *VPLGen* field for each cell you want to instantiate as a VPLGen.

Arguments

list_of_parameters

List of parameter names (and default values for non-CDF parameters), each separated by a semicolon. For example, `"pw;nw;a 1"`.

If the parameter in question is a CDF parameter (like `pw` or `nw` in the example), you do not need to specify a default value; the default set in the schematic is used.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "vplGenParams")
envSetVal("layoutXL" "vplGenParams" 'string "pw;nw;a 1")
```

Related Topics

Generating VPLGen Pcell Instances

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

vplGenSetDefaultLayout

```
layoutXL vplGenSetDefaultLayout boolean { t | nil }
```

Description

Automatically sets the default layout to be the VPLGen core layout with default parameter values; i.e., the one created the first time you generate a particular VPLGen in the layout view.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "vplGenSetDefaultLayout")
envSetVal("layoutXL" "vplGenSetDefaultLayout" 'boolean t)
```

Related Topics

Generating VPLGen Pcell Instances

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

xlComplianceCheckDevices

```
layoutXL xlComplianceCheckDevices boolean { t | nil }
```

Description

Checks devices when an *XL Compliance* check is run.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "xlComplianceCheckDevices")
envSetVal("layoutXL" "xlComplianceCheckDevices" 'boolean t)
```

Related Topics

Design Check for Layout XL Compliance

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

xlComplianceCreateCasMarkers

```
layoutXL xlComplianceCreateCasMarkers boolean { t | nil }
```

Description

Creates CAS markers when an *XL Compliance* check is run.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "xlComplianceCreateCasMarkers")
envSetVal("layoutXL" "xlComplianceCreateCasMarkers" 'boolean t)
```

Related Topics

Design Check for Layout XL Compliance

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

xlComplianceFilter

```
layoutXL xlComplianceFilter string "elements_to_remove"
```

Description

Applies filters on the XL Compliance report to remove messages related to any design components or elements that are unwanted or to remove specific message IDs. For example, to avoid reporting missing pin information for instance terminals, the environment variable can be set to filter out the corresponding message ID. This will ensure the related message is not issued in the report.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "xlComplianceFilter")
envSetVal("layoutXL" "xlComplianceFilter" 'string "LX-2149")
```

Related Topics

Design Check for Layout XL Compliance

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

xlComplianceHtml

```
layoutXL xlComplianceHtml boolean { t | nil }
```

Description

Controls whether the `report.html` is created in the `XLComplianceLogs` directory after performing an XL Compliance check.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "xlComplianceHtml")
envSetVal("layoutXL" "xlComplianceHtml" 'boolean t)
envSetVal("layoutXL" "xlComplianceHtml" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

xlComplianceHtmlOpen

```
layoutXL xlComplianceHtmlOpen boolean { t | nil }
```

Description

Controls whether the XL compliance check and update binding reports are displayed in Virtuoso using the default HTML browser.

The default is `nil`, which means the reports are only issued in CIW.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "xlComplianceHtmlOpen")
envSetVal("layoutXL" "xlComplianceHtmlOpen" 'boolean t)
envSetVal("layoutXL" "xlComplianceHtmlOpen" 'boolean nil)
```

Related Topics

Design Check for Layout XL Compliance

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

xlComplianceLogDir

```
layoutXL xlComplianceLogDir string "name_of_the_directory"
```

Description

Specifies the directory in which the log files generated during an XL Compliance check are saved.

The default is `""`, which means the generated logs are saved to the `XLComplianceLogs` directory in the current working directory.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "xlComplianceLogDir")
envSetVal("layoutXL" "xlComplianceLogDir" 'string "Lay1_XlCompl_Logs")
```

Related Topics

Design Check for Layout XL Compliance

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

xlComplianceLogs

```
layoutXL xlComplianceLogs boolean { t | nil }
```

Description

Controls whether separate log files are generated and saved during an XL Compliance check.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "xlComplianceLogs")
envSetVal("layoutXL" "xlComplianceLogs" 'boolean t")
envSetVal("layoutXL" "xlComplianceLogs" 'boolean nil")
```

Related Topics

`lxHierCheck`

Design Check for Layout XL Compliance

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

xlStatus

```
layoutXL xlStatus boolean { t | nil }
```

Description

Sets the *XL Status* column in the Navigator assistant to display by default.

The default is `t`, which means the *XL Status* column is displayed in the Navigator assistant by default.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "xlStatus")
envSetVal("layoutXL" "xlStatus" 'boolean t)
envSetVal("layoutXL" "xlStatus" 'boolean nil)
```

Related Topics

Navigator

List of Layout XL Environment Variables

Setting Environment Variables

© 2023 Cadence Design Systems, Inc.

xlStatusHierShorts

```
layoutXL xlStatusHierShorts boolean { t | nil }
```

Description

Specifies whether illegal hierarchical connections are reported as XL status shorts in Navigator during interactive editing.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "xlStatusHierShorts")
envSetVal("layoutXL" "xlStatusHierShorts" 'boolean t)
```

Related Topics

Navigator

© 2023 Cadence Design Systems, Inc.

Batch Checker Environment Variables

The following section describes the environment variables used to set options for the Width Spacing Pattern (WSP) Active check and the placement checks in the Batch Checker GUI.

- WSP | SP Active Checker Environment Variables
- Placement Checker Environment Variables

WSP | SP Active Checker Environment Variables

These `layoutXL.wspCheck` environment variables are used to set options for the Width Spacing Pattern (WSP) | SP Active check in the Batch Checker GUI.

- activeOn
- color
- depth
- ignoreValidJogs
- mergeShapes
- selectLayer
- selectPurposes
- vioLimit
- width
- wireType

© 2023 Cadence Design Systems, Inc.

activeOn

```
layoutXL.wspCheck activeOn boolean { t | nil }
```

Description

Specifies whether the *WSP Active* check is enabled when the Batch Checker is first opened in a session. The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.wspCheck" "activeOn")
envSetVal("layoutXL.wspCheck" "activeOn" 'boolean t)
envSetVal("layoutXL.wspCheck" "activeOn" 'boolean nil)
```

Related Topics

WSP | SP Active Checker Environment Variables

Checking WSP | SP Active Using Batch Checker

wspCheckActive SKILL function

© 2023 Cadence Design Systems, Inc.

color

```
layoutXL.wspCheck color boolean { t | nil }
```

Description

Specifies whether the shapes are to be checked for conformance to the track colors of the active WSP. The default value is `t`.

GUI Equivalent

Command	*Verify – Design*
Field	*Color* (WSP | SP Active)

Examples

```
envGetVal("layoutXL.wspCheck" "color")
envSetVal("layoutXL.wspCheck" "color" 'boolean t)
envSetVal("layoutXL.wspCheck" "color" 'boolean nil)
```

Related Topics

WSP | SP Active Checker Environment Variables

Checking WSP | SP Active Using Batch Checker

wspCheckActive SKILL function

© 2023 Cadence Design Systems, Inc.

depth

```
layoutXL.wspCheck depth int integer_number
```

Description

Sets the hierarchy depth up to which the cellview will be checked. Valid values are 0 through 32. The default is 0.

GUI Equivalent

Command	*Verify – Design*
Field	*Hierarchy Depth* (WSP | SP Active)

Examples

```
envGetVal("layoutXL.wspCheck" "depth")
envSetVal("layoutXL.wspCheck" "depth" 'int 32)
```

Related Topics

WSP | SP Active Checker Environment Variables

Checking WSP | SP Active Using Batch Checker

wspCheckActive SKILL function

© 2023 Cadence Design Systems, Inc.

ignoreValidJogs

```
layoutXL.wspCheck ignoreValidJogs boolean { t | nil }
```

Description

Ignores jogs that are defined in the `allowedWidthRanges` constraint for the layer in the jog direction before checking. The default value is `nil`.

GUI Equivalent

Command	*Verify – Design*
Field	*Ignore Valid Jogs* (WSP | SP Active)

Examples

```
envGetVal("layoutXL.wspCheck" "ignoreValidJogs")
envSetVal("layoutXL.wspCheck" "ignoreValidJogs" 'boolean t)
envSetVal("layoutXL.wspCheck" "ignoreValidJogs" 'boolean nil)
```

Related Topics

WSP | SP Active Checker Environment Variables

Checking WSP | SP Active Using Batch Checker

wspCheckActive SKILL function

© 2023 Cadence Design Systems, Inc.

mergeShapes

```
layoutXL.wspCheck mergeShapes boolean { t | nil }
```

Description

Specifies whether same-color shapes and uncolored shapes that are connected and on the same layer are to be merged before checking. If multiple purposes are selected, merging applies to shapes on only those purposes. The default value is `t`.

GUI Equivalent

Command	*Verify – Design*
Field	*Merge Same Layer Shapes* (WSP | SP Active)

Examples

```
envGetVal("layoutXL.wspCheck" "mergeShapes")
envSetVal("layoutXL.wspCheck" "mergeShapes" 'boolean t)
envSetVal("layoutXL.wspCheck" "mergeShapes" 'boolean nil)
```

Related Topics

WSP | SP Active Checker Environment Variables

Checking WSP | SP Active Using Batch Checker

wspCheckActive SKILL function

© 2023 Cadence Design Systems, Inc.

selectLayer

```
layoutXL.wspCheck selectLayer string "layer_name"
```

Description

Specifies the layer for the shapes to be checked.

- *layer_name* specifies the layer name.
- *All* specifies all the layers in the active WSP.

The default is *All*.

GUI Equivalent

Command	*Verify – Design*
Field	*Layer* (WSP | SP Active)

Examples

```
envGetVal("layoutXL.wspCheck" "selectLayer")
envSetVal("layoutXL.wspCheck" "selectLayer" 'string "All")
envSetVal("layoutXL.wspCheck" "selectLayer" 'string "Metal1 Metal2")
```

Related Topics

WSP | SP Active Checker Environment Variables

Checking WSP | SP Active Using Batch Checker

wspCheckActive SKILL function

© 2023 Cadence Design Systems, Inc.

selectPurposes

`layoutXL.wspCheck selectPurposes string "`*`purpose_names`*`"`

Description

Specifies the layer for the shapes to be checked.

`purpose_names` specifies one or more purpose names.

The default is `drawing`.

GUI Equivalent

Command	*Verify – Design*
Field	*Purposes* (WSP | SP Active)

Examples

```
envGetVal("layoutXL.wspCheck" "selectPurposes")
envSetVal("layoutXL.wspCheck" "selectPurposes" 'string "drawing")
```

Related Topics

WSP | SP Active Checker Environment Variables

Checking WSP | SP Active Using Batch Checker

wspCheckActive SKILL function

© 2023 Cadence Design Systems, Inc.

vioLimit

```
layoutXL.wspCheck vioLimit int integer_number
```

Description

Sets the maximum number of violations. The default is 1000.

GUI Equivalent

Command	*Verify – Design*
Field	*Marker Limit* (WSP | SP Active)

Examples

```
envGetVal("layoutXL.wspCheck" "vioLimit")
envSetVal("layoutXL.wspCheck" "vioLimit" 'int 32)
```

Related Topics

WSP | SP Active Checker Environment Variables

Checking WSP | SP Active Using Batch Checker

wspCheckActive SKILL function

© 2023 Cadence Design Systems, Inc.

width

```
layoutXL.wspCheck width boolean { t | nil }
```

Description

Specifies the type of width checking to be run on the shapes.

- `t` checks whether the width of the shapes matches the width of the active WSP.
- `nil` checks whether the centerline of each shape is aligned with the centerline of the active WSP.

The default value is `t`.

GUI Equivalent

Command	*Verify – Design*
Field	*Width* (WSP | SP Active)

Examples

```
envGetVal("layoutXL.wspCheck" "width")
envSetVal("layoutXL.wspCheck" "width" 'boolean t)
envSetVal("layoutXL.wspCheck" "width" 'boolean nil)
```

Related Topics

WSP | SP Active Checker Environment Variables

Checking WSP | SP Active Using Batch Checker

wspCheckActive SKILL function

© 2023 Cadence Design Systems, Inc.

wireType

```
layoutXL.wspCheck wireType boolean { t | nil }
```

Description

Specifies whether the shapes are to be checked for conformance with the wireTypes of the active WSP.

The default value is `t`.

GUI Equivalent

Command	*Verify – Design*
Field	*Wire Type* (WSP | SP Active)

Examples

```
envGetVal("layoutXL.wspCheck" "wireType")
envSetVal("layoutXL.wspCheck" "wireType" 'boolean t)
envSetVal("layoutXL.wspCheck" "wireType" 'boolean nil)
```

Related Topics

WSP | SP Active Checker Environment Variables

Checking WSP | SP Active Using Batch Checker

wspCheckActive SKILL function

© 2023 Cadence Design Systems, Inc.

Placement Checker Environment Variables

These `layoutXL.APverify` environment variables are used to set options for the placement checks in the Batch Checker GUI.

- constraints
- rowOrGrid
- spacing
- tapCell
- padding
- minPinAccess
- colorAware
- reportViolations
- createMarkers
- selOnly

© 2023 Cadence Design Systems, Inc.

constraints

```
layoutXL.APVerify constraints 'boolean { t | nil }
```

Description

In Layout EXL or higher tier, if set, checks all elements in the design for conformance with the Alignment, Orientation, Symmetry, and Distance constraints.

Default is `t`.

GUI Equivalent

Command:	*Verify – Design*
Field:	*Placement – Check for – Constraints*

Examples

```
envGetVal("layoutXL.APVerify" "constraints")
envSetVal("layoutXL.APVerify" "constraints" 'boolean t)
envSetVal("layoutXL.APVerify" "constraints" 'boolean nil)
```

Related Topics

Batch Checker Form

Checking Placement Using Batch Checker

© 2023 Cadence Design Systems, Inc.

rowOrGrid

```
layoutXL.APVerify rowOrGrid 'boolean { t | nil }
```

Description

In Layout EXL or higher tier, checks whether instances are snapped to the correct rows or grids.

Default is `t`.

GUI Equivalent

Command:	*Verify – Design*
Field:	*Placement – Check for – Row/Grid*

Examples

```
envGetVal("layoutXL.APVerify" "rowOrGrid")
envSetVal("layoutXL.APVerify" "rowOrGrid" 'boolean t)
envSetVal("layoutXL.APVerify" "rowOrGrid" 'boolean nil)
```

Related Topics

Batch Checker Form

Checking Placement Using Batch Checker

© 2023 Cadence Design Systems, Inc.

spacing

```
layoutXL.APVerify spacing 'boolean { t | nil }
```

Description

In Layout EXL or higher tier, checks instances for conformance with applicable spacing rules.

Default is `nil`.

GUI Equivalent

Command:	*Verify – Design*
Field:	*Placement – Check for – Spacing*

Examples

```
envGetVal("layoutXL.APVerify" "spacing")
envSetVal("layoutXL.APVerify" "spacing" 'boolean t)
envSetVal("layoutXL.APVerify" "spacing" 'boolean nil)
```

Related Topics

Batch Checker Form

Checking Placement Using Batch Checker

© 2023 Cadence Design Systems, Inc.

tapCell

```
layoutXL.APVerify tapCell 'boolean { t | nil }
```

Description

In Layout EXL or higher tier, checks whether tap cells are inserted correctly, as per their specifications on the *Automatic* tab of the Placement Options form.

Default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.APVerify" "tapCell")
envSetVal("layoutXL.APVerify" "tapCell" t)
envSetVal("layoutXL.APVerify" "tapCell" nil)
```

Related Topics

Batch Checker Form

Checking Placement Using Batch Checker

© 2023 Cadence Design Systems, Inc.

padding

```
layoutXL.APVerify padding 'boolean { t | nil }
```

Description

In Layout EXL or higher tier, checks whether the spacing on either side of the master are in conformance with the *Cell Padding* value specified on the *Common* tab of the Placement Options form.

Default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.APVerify" "padding")
envSetVal("layoutXL.APVerify" "padding" 'tapCell t)
envSetVal("layoutXL.APVerify" "padding" 'tapCell nil)
```

Related Topics

Batch Checker Form

Checking Placement Using Batch Checker

© 2023 Cadence Design Systems, Inc.

minPinAccess

```
layoutXL.APVerify minPinAccess 'boolean { t | nil }
```

Description

In Layout EXL or higher tier, checks WSP tracks for conformance with the *Minimum Pin Access Count* specified on the *Common* tab of the Placement Options form.

Default is `nil`.

GUI Equivalent

Command:	*Verify – Design*
Field:	*Placement – Check for – Minimum Pin Access*

Examples

```
envGetVal("layoutXL.APVerify" "minPinAccess")
envSetVal("layoutXL.APVerify" "minPinAccess" 'boolean t)
envSetVal("layoutXL.APVerify" "minPinAccess" 'boolean nil)
```

Related Topics

Batch Checker Form

Checking Placement Using Batch Checker

© 2023 Cadence Design Systems, Inc.

colorAware

```
layoutXL.APVerify colorAware 'boolean { t | nil }
```

Description

In Layout EXL or higher tier, checks shapes for conformance with color locks assigned to them, if any.

Default is `nil`.

GUI Equivalent

Command:	*Verify – Design*
Field:	*Placement – Check for – Color Aware*

Examples

```
envGetVal("layoutXL.APVerify" "colorAware")
envSetVal("layoutXL.APVerify" "colorAware" 'boolean t)
envSetVal("layoutXL.APVerify" "colorAware" 'boolean nil)
```

Related Topics

Batch Checker Form

Checking Placement Using Batch Checker

© 2023 Cadence Design Systems, Inc.

reportViolations

```
layoutXL.APVerify reportViolations 'boolean { t | nil }
```

Description

In Layout EXL or higher tier, displays detailed error and warning messages in the CIW reporting all violations that were detected while running batch checker.

Default is `nil`.

GUI Equivalent

Command:	*Verify – Design*
Field:	*Placement – Reports – Report Violations*

Examples

```
envGetVal("layoutXL.APVerify" "reportViolations")
envSetVal("layoutXL.APVerify" "reportViolations" 'boolean t)
envSetVal("layoutXL.APVerify" "reportViolations" 'boolean nil)
```

Related Topics

Batch Checker Form

Checking Placement Using Batch Checker

© 2023 Cadence Design Systems, Inc.

createMarkers

```
layoutXL.APVerify createMarkers 'boolean { t | nil }
```

Description

In Layout EXL or higher tier, displays markers in the design to indicate the locations where violations are reported.

Default is `nil`.

GUI Equivalent

Command:	*Verify – Design*
Field:	*Placement – Reports – Create Markers*

Examples

```
envGetVal("layoutXL.APVerify" "createMarkers")
envSetVal("layoutXL.APVerify" "createMarkers" 'boolean t)
envSetVal("layoutXL.APVerify" "createMarkers" 'boolean nil)
```

Related Topics

Batch Checker Form

Checking Placement Using Batch Checker

© 2023 Cadence Design Systems, Inc.

selOnly

```
layoutXL.APVerify selOnly 'boolean { t | nil }
```

Description

In Layout EXL or higher tier, runs batch checker on the selected instances only.

Default is `nil`.

GUI Equivalent

Command:	*Verify – Design*
Field:	*Placement – Scope – Selected Only*

Examples

```
envGetVal("layoutXL.APVerify" "selOnly")
envSetVal("layoutXL.APVerify" "selOnly" 'boolean t)
envSetVal("layoutXL.APVerify" "selOnly" 'boolean nil)
```

Related Topics

Batch Checker Form

Checking Placement Using Batch Checker

© 2023 Cadence Design Systems, Inc.

GUI Options and Corresponding Environment Variables

Form	Option	Environment Variable
A		
Navigator	*Default open in*	openLocation
	Default open mode	openMode
	Remove highlighting when browser is closed	hideMarkersWhenBrowserHidden
	Hierarchy depth	hierarchyDepth
	Highlighted markers can be selected in canvas	highlightedIsSelectable
	Suppress expansion at	suppressAbutFailMsgs
C		
Check Against Source Form	*Do not show me this dialog again*	disableCASOptionsPopUp
Configure Physical Hierarchy Window	*Logical switch view list*	viewList
	Physical library list	lxUseLibList
	Physical stop view list	stopList
Create Path Form	*Probe nets*	probeDuringCreate
Define Device Correspondence Form	*Probe nets*	probeDuringCreate
Define Device Correspondence Form	*Probe nets*	probeDuringCreate
Create Wire Form	*Probe nets*	probeDuringCreate
G		
Generate Clones Form	*Draglines*	flightLineEnable
	Exact Parameters	cloningDoExactMatch

© 2023 Cadence Design Systems, Inc.

Form	Option	Environment Variable
	Update Layout Parameters to Match Schematic	cloneJustify

© 2023 Cadence Design Systems, Inc.

Form	Option	Environment Variable
Generate Layout Form		initIOPinName
	Aspect Ratio W/H	initAspectRatio
	Boundary Area Estimation	initEstimateArea
	Boundary Height	initPrBoundaryH
	Boundary Width	initPrBoundaryW
	Create	initCreatePins
	Height	initPinHeight
	Except Global Pins	initGlobalNetPins
	Instances	initCreateInstances
	Layer	initIOPinLayer
	Num	initPinMultiplicity
	Create Label As	initIOLabelType
	Pin Width	initPinWidth
	PR Boundary	initCreateBoundary
	Preserve Floorplanning Objects – Area Boundaries	preserveAreaBoundary
	Preserve Floorplanning Objects – Blockages	preserveBlockages
	Preserve Floorplanning Objects – Clusters	preserveClusters
	Preserve Floorplanning Objects – Cluster Boundaries	preserveClusterBoundaries
	Preserve Floorplanning Objects – Rows and custom Placement Areas	preserveRows
	Preserve Floorplanning Objects – Track Patterns	preserveTrackPattern
	Preserve User-Defined Bindings	initCreateMTM

© 2023 Cadence Design Systems, Inc.

Form	Option	Environment Variable
Generate Selected Components Form	*Draglines*	flightLineEnable
Generate Chained Devices Form	*Extend Selection*	chainExtendSelection
	Preserve Existing Chains	chainPreserveExistingChains
	Use Device Order	chainUseDeviceOrder
Generate Folded Devices Form	*Chain Folds*	chainFolds
L		
Connectivity –General	*Constraint-aware editing*	constraintAwareEditing
	Open connectivity reference	openConnRef
	Open connectivity reference in	lxSchematicDefaultApp
Connectivity –Display	*Auto arrange windows*	autoArrange
	Cross-selection	crossSelect
	Hide global, power, and ground nets	hideDraglinesForGlobalNets
	Probe nets during object creation	probeDuringCreate
	Show distant connections	showDraglinesForDistantConns
	Use colors	colorDraglines
	Redirect Layout XL messages to separate Info window	infoWindow
Connectivity – Connectivity	*Spaced-based Router – Extract*	deviceExtractType

© 2023 Cadence Design Systems, Inc.

Form	Option	Environment Variable
Connectivity – Generation	*Abut transistors*	abutWithoutConnectivity
	Generate minimal folding	lxStackMinimalFolding
	Mirror transistors	autoMirror
	Pcell fingering parameter names	lxFingeringNames
	Retain device orientation	lxRetainFoldOrient
	Space components	autoSpace
	Split mfactored devices	mfactorSplit
	Update layout instances	updateReplacesMasters
	Update width parameter method	lxUpdateFoldedWidth
	Create implicit bus terminals	createImplicitBusTerminals

© 2023 Cadence Design Systems, Inc.

Form	Option	Environment Variable
Connectivity – Parameters	*Capacitance*	capacitanceParamNames
	Consider parameters only and *Consider parameters and properties*	checkParamsOnly
	Ignore missing parameters or properties	checkMissingParamsOrProps
	Inductance	inductanceParamNames
	Parallel-connected factor	mfactorNames
	Parameters to ignore during check	paramsToIgnoreForCheck
	Parameters to ignore during generation	paramsToIgnore
	Properties used to ignore objects during check	propsUsedToIgnoreObjsForCheck
	Properties used to ignore objects during generation	propsUsedToIgnoreObjs
	Resistance	resistanceParamNames
	Series-connected factor	sfactorNames
	Tolerance	paramTolerance
	Transistor width	transistorWidthParamNames
M		
Move Cells Form	*Display Draglines*	flightLineEnable
P		
Highlight Options	*Display Layer*	probeHiliteLayer
	Display Layer – Cycle	probeCycleHilite

© 2023 Cadence Design Systems, Inc.

Form	Option	Environment Variable
S		
Set Pin Label Text Style Form	*Layer Name – Same As Pin*	pinTextSameLayer
	Layer Purpose – Same As Pin	pinTextSamePurpose
Swap Components Form	*Display Draglines*	flightLineEnable
U		
Update Components and Nets Form – *Update* tab	*Delete Unmatched Instances*	deleteUnmatchedInsts
	Delete Unmatched Pins	deleteUnmatchedPins
X		
XL Probe Form	*Instances*	probeDevice
	Nets	probeNet
	Pins	probePin
	Send Messages to CIW	probeInfoInCIW

© 2023 Cadence Design Systems, Inc.

Hidden Environment Variables with No Corresponding GUI Option

© 2023 Cadence Design Systems, Inc.

© 2023 Cadence Design Systems, Inc.

B

Command Quick Reference

Listed below are all the commands you can use in the Virtuoso® Layout Suite XL layout editor (Layout XL) with a brief summary of what each command does and a link to more complete documentation of the command.

Command	Function	Documented
Launch Menu		
Configure Physical Hierarchy	Lets you define how the physical implementation is generated from the schematic, define and edit component types, and define and edit soft blocks.	Physical Hierarchy Configuration
File Menu		
Load Physical View	Loads an OpenAccess template cellview containing pin, boundary, and placement information into the current cellview.	Load Physical View Form
Edit Menu		
– *Swap Components*	Exchanges the position of two selected components.	Swapping Components
Soft Blocks	Lets you modify the attributes of individual soft blocks in your design.	Soft Block Editing
Component Types	Controls how components are placed in rows and defines their chaining and folding parameters.	Component Types Mode in CPH

© 2023 Cadence Design Systems, Inc.

Command	Function	Documented
Connectivity Menu		
Pins – Permute	Switches the position of pins set up for permutation within an instance.	Pin Permutation
Nets – Assign	Assigns instance pins with or without a connectivity source or schematic to nets in the layout.	Assigning Instance Pins to a Net
Nets – Edit Attributes	Lets you modify the criticality and signal type of existing nets.	Editing Net Attributes and Properties
Nets – Set Order	Lets you create an ordered net constraint on a group of nets.	
Nets – Show/Hide All Incomplete Nets	Shows and hides all the incomplete nets in the design.	Showing Common Incomplete Nets for Selected Instances
Nets – Show/Hide Selected Incomplete Nets	Shows and hides any incomplete nets associated with selected objects in the specified layout window.	Showing Common Incomplete Nets for Selected Instances
Nets – Show/Hide Current Incomplete Nets	Toggles the visibility of the current set of incomplete nets displayed in the design canvas.	Showing and Hiding Current Incomplete Nets
XL Probe	Highlights the design element in the layout or schematic when you click the corresponding one in the other window.	Design Probing
Generate – All From Source	Creates and places layout cells below the boundary in the layout canvas.	Generating All Components from Source

© 2023 Cadence Design Systems, Inc.

Command	Function	Documented
Generate – Selected From Source	Creates and places interactively selected schematic elements in the layout.	Placement Modes for Generating Selected Components from Source
Generate – Place As In Schematic	Moves components generated in the layout into the design boundary, maintaining the relative positions of the components from the schematic.	Moving Generated Components inside the PR Boundary (Placing As In Schematic)
Generate – Clones	Copies parts of schematic or layout including connectivity info to use as other parts.	Clone Generation
Generate – Chained Devices	Abuts prepared MOS transistors into chains.	Manual Device Abutment Debugging Abutment
Generate – Folded Devices	Divides prepared MOS transistors or transistor chains into individual folds (smaller devices connected in parallel).	Folding a Transistor or FinFET Device
Check – Against Source	Reports the devices and parameters in the connectivity source that are missing in the schematic.	Design Check Against Source
Update – Components And Nets	Updates the layout to add devices, pins, or connectivity changes added to the schematic.	Components and Nets Updates
Update – Layout Constraints	Transfers constraints from the schematic view to the layout view.	Updating Layout Constraints
Update – Layout Parameters	Updates parameters of instances in the layout to match those in the schematic.	Layout Parameters Updates

© 2023 Cadence Design Systems, Inc.

Command	Function	Documented
Update – Schematic Parameters	Updates parameters of devices in the schematic to match those in the layout.	Schematic Parameter Updates
Update – Connectivity Reference	Updates the connectivity source and physical configuration for a given layout view.	Updating the Connectivity Reference
Update – Clone Families	Lets you manage the synchronized clone families in your design.	Updating Clone Families
Update – Extract Layout	Verifies the connectivity of the layout design and generates markers to highlight open and short circuits and illegal overlaps.	Connectivity Extraction Use Model
Define Device Correspondence	Creates user-defined device correspondence between selected devices in the schematic and layout views; allows implementation of one or many to many or one bindings.	Device Correspondence Updates
Options Menu		
Layout XL	Sets values of Layout XL environment variables.	Layout Editor Basics
Place Menu		
Pin Placement	Sets constraints on and places the pins in your design.	Pin Placement
Update Placement Status	Updates the placement status of instances and pins in the current cellview.	Updating the Placement Status Automatically
Custom Digital – Placement Planning	Lets you create, change, and delete rows	Placement Planning

© 2023 Cadence Design Systems, Inc.

Command	Function	Documented
Custom Digital – Placer	Runs the automatic placer	Auto Placer

© 2023 Cadence Design Systems, Inc.

© 2023 Cadence Design Systems, Inc.

C

Layout XL Properties

This section provides property names, descriptions, and graphical user interface (GUI) equivalents for the Virtuoso® Layout Suite XL layout editor (Layout XL).

Only the properties documented in this chapter are supported for public use. All other Layout XL properties, regardless of their name or prefix, and undocumented aspects of the properties described below, are private and are subject to change at any time.

abNetsColumnValue Property

abutAccessDir Property

abutClass Property

abutCondInclusion Property

abutFunction Property

abutGateNet Property

abutMosStretchMat Property

abutStretchMat Property

abutOffset Property

caeIgnoreInCluster Property

extractStopLevel Property

extractToLevel Property

ignore Property

lvsIgnore Property

lxAutoAbut Property

© 2023 Cadence Design Systems, Inc.

lxAutoSpace Property

lxCombination Property

lxLocalAbutment Property

lxMfactorSplit Property

lxNetNamePrefix Property

lxParamsToCheck Property

lxParamsToIgnore Property

lxParamsToIgnoreForCheck Property

lxRemoveDevice Property

lxRounding

lxSeriesTerms Property

lxSfactorSplit Property

lxStickyNet Property

lxStopList Property

lxUseCell Property

lxViewList Property

mfactor Property

permuteRule Property

physOnly Property

sfactor Property

vxlInstSpacingDir Property

vxlInstSpacingRule Property

© 2023 Cadence Design Systems, Inc.

Setting Layout XL Properties

You can set some of the properties described in this chapter in the graphical user interface. This section tells you how to access the appropriate form.

Accessing the Symbol Master Add Property Form

To add a user property to a symbol master:

1. From the Virtuoso Symbol Editing window, choose *Edit – Properties – Cellview*.

   The Edit Cellview Properties form appears.
2. In the *User Property* section, click *Add*.

   The Add Property form appears.

Accessing the Add CDF Parameter Form

To add a CDF parameter to a layout or schematic master:

1. From the CIW, choose *Tools – CDF – Edit*.

   The Edit Component CDF form is displayed.
2. Specify the library and cell names and click *Apply*.

   The form is populated with the options you can set.
3. In the *Component Parameters* section, click *Add*.

   The Add CDF Parameter form is displayed.

Accessing the Layout Instance Add Property Form

To add a user property to an instance in the layout:

1. In the layout canvas, select the instance to which you want to add a property.
2. From the menu bar, choose *Edit – Basic – Properties*.

   The Edit Instance Properties form is displayed.
3. Choose the *Property* radio button and click *Add*.

   The Add Property form is displayed.

© 2023 Cadence Design Systems, Inc.

Related Topics

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

abNetsColumnValue Property

`abNetsColumnValue string` *`t_net_names`*

Description

Specifies net names to be displayed in the *Nets* column of Annotation Browser tree views, instead of getting the names from the objects associated to the marker.

The net names should be separated by space.

You can specify the nets in any order, Annotation Browser reorders them alphabetically automatically.

The *Nets* column can be used to group the markers in the *View By* options of Annotation Browser

Example

```
dbCreateProp(marker1 "abNetsColumnValue" "string" "/a/b /a/c")
dbCreateProp(marker2 "abNetsColumnValue" "string" "/a/c /a/b")
```

Nets are ordered automatically and `marker1` and `marker2` will both have (`/a/b`, `/a/c`) displayed in the *Nets* column of Annotation Browser.

© 2023 Cadence Design Systems, Inc.

abutAccessDir Property

```
abutAccessDir list (left right top bottom)
```

Description

Defines the direction in which pins are able to abut, ensuring that the correct edges are abutting; for example, a left side to a right side, or a bottom to a top.

Cells with rail pins need to add access directions for both sides because there is one pin for both right and left sides of the cell. MOS cells have just one pin per side, so they need only one direction.

Examples

```
dbReplaceProp(objId "abutAccessDir" "list" list("left"))
dbReplaceProp(objId "abutAccessDir" "list" list("left" "right"))
```

Related Topics

abutClass Property

abutCondInclusion Property

abutFunction Property

abutGateNet Property

abutMosStretchMat Property

abutStretchMat Property

abutOffset Property

Device Abutment

Creating CMOS Pcells to Use with Abutment

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

abutClass Property

`abutClass string` *`t_user_defined_name`*

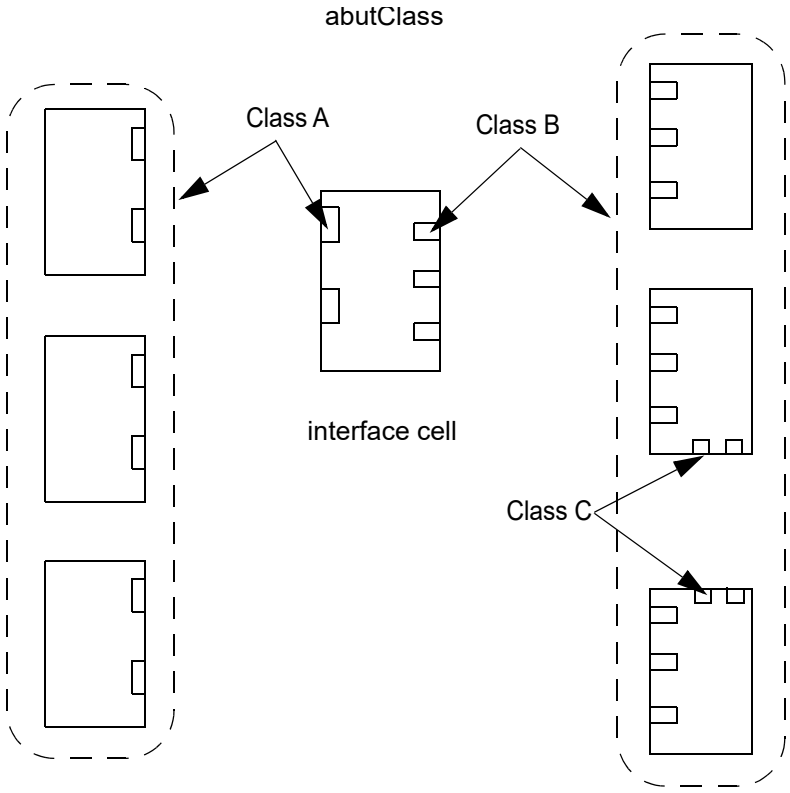
Description

Specifies that two cells can abut even if their master cellviews are not the same.

You can place this property on an individual pin or on a cellview in order to implement a global abutment class for all the pins in the cellview. The abutment function checks for the property first on the pin and then on the cellview.

When you place this property on a pin, remember that each pin targeted for abutment can belong to only one abutment class. Pins on different edges of a cell can belong to different classes.

Instances set to different `abutClass` values can be abutted if the `abutClass` values are set as equivalent using the equivalentAbutClasses environment variable.



© 2023 Cadence Design Systems, Inc.

Example

```
dbReplaceProp(leftObj~>dbId "abutClass" "string" d_layer)
```

Related Topics

abutAccessDir Property

abutCondInclusion Property

abutFunction Property

abutGateNet Property

abutMosStretchMat Property

abutStretchMat Property

abutOffset Property

Device Abutment

Creating CMOS Pcells to Use with Abutment

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

abutCondInclusion Property

```
abutCondInclusion string t_pcell_param_name
```

Description

Defines the name of the boolean property that adds or removes the source/drain contact.

You can conditionally include or exclude the contacts and the connecting metal tab. You must conditionally exclude these features if you are creating a graphical Pcell. The Pcell parameter must remove all features that are required for metal hookup and leave only the diffusion material to be stretched. For CMOS device abutment, you add this property to both drain and source sides of the device.

Example

```
dbReplaceProp(rightObj~>dbId "abutCondInclusion" "string" "rightCnt")
```

Related Topics

abutAccessDir Property

abutClass Property

abutFunction Property

abutGateNet Property

abutMosStretchMat Property

abutStretchMat Property

abutOffset Property

Device Abutment

Creating CMOS Pcells to Use with Abutment

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

abutFunction Property

`abutFunction string` *`t_function_name`*

Description

Specifies a user-defined callback to process abutment and unabutment.

The value of this property is the name of a user-defined Cadence® SKILL function to be executed before abutment takes place. Auto-abutment passes this function the following eight arguments.

1. The database ID of the cell to be abutted
2. The database ID of the cell to be abutted to
3. The database ID of the pin figure to be abutted
4. The database ID of the pin to be abutted to

If an instance is deleted, the arguments 3 and 4 above are set to `nil`.

1. An integer to indicate the **abutment access direction** of the pin being abutted where
   - 1 = top
   - 2 = bottom
   - 4 = left
   - 8 = right
2. An integer to indicate the **connection condition** where
   - 1 = pins are connected to the same net and do not connect to any other pin
   - 2 = pins are connected to the same net and the net connects to other pins
   - 3 = pins are connected to different nets. Used for nets that are not connected, so that structures like guard rings and wells can be adjusted appropriately.
3. An integer specifying the **auto-abutment event** where
   - 1 = `abutFunction` must compute and return abutment offset (in the direction of abutment)
   - 2 = `abutFunction` must adjust Pcell parameters for abutment. This is called before the offset event.
   - 3 = `abutFunction` must adjust Pcell parameters for unabutment

© 2023 Cadence Design Systems, Inc.

- 4 = `abutFunction` must compute and return spacing offset for non abutment.
- 5 = `abutFunction` must compute and update the parameters of each named abutment shape in the Pcell instance, if required, before the abutment engine computes the connection condition value to be passed to *Event* 2.

Important

If both the Pcell instances to be abutted support abutment *Event* 5, the *Event* 5 is executed before *Event* 2. If either of the instances do not support *Event* 5, abutment *Event* 5 is canceled, and the abutment of dummy shapes fails. For more information Event 5, see Abutment Event 5.

4. A database group ID (`dbGroupId`) of the abutment group, which is used in *Event* 2 and 3. The group ID is used to store information so that during unabutment, pins are returned to their original state. It is a well-accepted method to use the group attribute of a cell to hold original parameter values prior to modification by abutment.

`abutFunction` returns different values for different auto-abutment events.

- For *Event* 1, the function returns a floating-point number representing the offset needed to abut the cells in the direction of abutment.

  If it returns two floating-point numbers, the second number is the distance in user units that the abutting cell is moved in the direction perpendicular to the direction of abutment.
- For *Event* 2, the function must return a value of `true` to allow abutment and `nil` to block abutment. When it returns `nil`, it must not change any parameters in any way, including setting them to new values and then resetting them to the original values.
- For *Event* 3, the function can return anything; the value returned is not used.
- For *Event* 4, the function returns the value of the spacing property set on non-abutting instances that determines how much the instances need to move to return to their pre-abutment state. If the spacing property is not set, the instances do not need to move.
- For *Event* 5, the function must return a pair of figIDs to identify the shapes that need to be abutted. If it returns `nil`, the abutment of dummy shapes fails and *Event* 2 is called instead. For more information about *Event* 5, see Abutment Event 5.

Below is an example of an `abutFunction`.

```
procedure( abutFunction(iA iB pA pB pASide connection event @optional
   (group nil)) prog((result)
      case(event
```


© 2023 Cadence Design Systems, Inc.

```
            )

         (1  ; Compute offset
             result = getAbutmentOffset(iA iB pA pB pASide connection)
             )
         (2  ; Adjust Pcell parameters
             result = setAbutmentParams(iA iB pA pB pASide connection
                 group)
             )
         (3  ; Adjust Pcell parameters back to default
             resetAbutmentParams(group iA iB)
                result = t
             )
         (4  ; To get the spacing offset when pins are on different nets
             if(instA~>w > 3.0 && instB~>w > 3.0
             then
                 result = 3
             else
                 result = 1
             )
         (5  ; Get abut shape IDs and update Pcell parameters to remove or adjust
dummy shapes
             result = myGetInterfaceShapePairFn(iA iB pA pB)
);; end of case 5
           (t  ; Anything else return a nil
               result = nil
           )
       )
       return(result)
   )
 )
```

If each pin being abutted has a different `abutFunction` defined, auto-abutment calls only the function defined by the pin in the cell being moved.

Example

```
dbReplaceProp(obj "abutFunction" "string" "mosAbutFunc")
```

Related Topics

abutAccessDir Property

abutClass Property

abutCondInclusion Property

abutGateNet Property

abutMosStretchMat Property

abutStretchMat Property

© 2023 Cadence Design Systems, Inc.

abutOffset Property

Device Abutment

Creating CMOS Pcells to Use with Abutment

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

abutGateNet Property

```
abutGateNet string t_pcell_param_name
```

Description

Specifies the name of the net the gate shapes are on, so that the abutment engine knows where the edge of the gate is.

Example

The value "G" is the name of the gate net inside the Pcell.

```
dbReplaceProp(leftObj~>dbId "abutGateNet" "string" "G")
```

Related Topics

abutAccessDir Property

abutClass Property

abutCondInclusion Property

abutFunction Property

abutMosStretchMat Property

abutStretchMat Property

abutOffset Property

Device Abutment

Creating CMOS Pcells to Use with Abutment

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

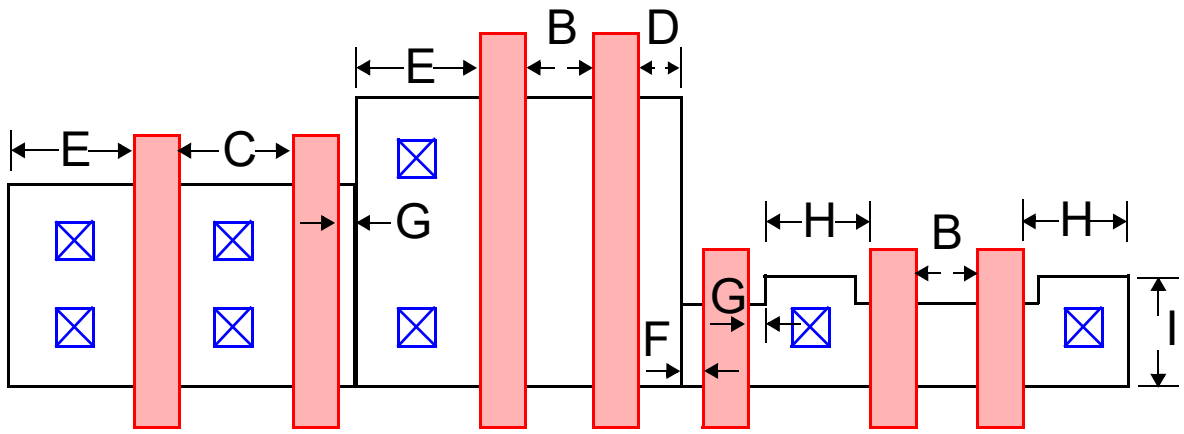
abutMosStretchMat Property

```
abutMosStretchMat list abutFlush f_value
     abutGateSpaceSeries f_value
     abutGateSpaceParallel f_value
     abutEncloseGate f_value
     abutEncloseContact f_value
     abutSmallerSeries f_value
     abutSmallerParallel f_value
     abutEncloseDogBoneContact f_value
     abutMinContactWidth f_value
```

Description

Governs how a material is stretched towards or away from a gate when an instance is altered for abutment.

The picture below illustrates the MOS diffusion extension rules for the different `abutMosStretchMat` property values.



A = abutFlush (Not shown). Applied to the pin which does not move when abutting two pins of equal width.

B = abutGateSpaceSeries. Applied to the moving pin when abutting two pins of equal width with no contact required between the gates. The `abutFlush` rule is applied to the non-moving pin.

C = abutGateSpaceParallel. Applied to the moving pin when abutting pins of equal width where a contact is required between the gates and the pin widths are greater than or equal to the `abutMinContactWidth` rule. The `abutFlush` rule is applied to the non-moving pin.

© 2023 Cadence Design Systems, Inc.

D = abutEncloseGate. Applied to the larger of two abutting pins when a contact is not required between the gates. The `abutSmallerSeries` rule is applied to the smaller of the two pins being abutted.

E = abutEncloseContact. Applied to the larger of two different-width abutting pins when a contact is required between the gates but only if the width of the larger pin is greater than or equal to the `abutMinContactWidth` rule. The `abutSmallerParallel` rule is applied to the smaller of the two pins being abutted.

F = abutSmallerSeries. Applied to the smaller of two abutting pins when a contact is not required between the gates.

G = abutSmallerParallel. Applied to the smaller of two abutting pins when a contact is required between the gates.

H = abutEncloseDogBoneContact. Applied when contacts are required and the width of the device is less than the `abutMinContactWidth` rule. It is applied to the larger of two different-width pins or to the moving pin when the two pins are of equal width.

I = abutMinContactWidth. Determines whether to use the `abutEncloseContact` rule or the `abutEncloseDogBoneContact` rule when a contact is required between gates.

Example

This example illustrates the properties being applied to pins on the left and right edges of a MOS Pcell.

- The pin rectangles were created by the `rodCreateRect` command and are referenced by two variables called `leftObj` and `rightObj` pointing to their rod object IDs.
- The Pcell parameters that control the stretching of the left and right pin extensions past the gate are called `leftSt` and `rightSt`.
- All abutment extension rule values in the `abutMosStretchMat` property list must be numeric.
- The Pcell parameters that control whether or not the left or right pins have contacts over them are `leftCnt` and `rightCnt`.
- The contact parameters must be of type boolean.

```
;; Definition of variables used in the abutRuleList calculations for
;; the abutMosStretchMat property.

;; cw  = contact width
```


© 2023 Cadence Design Systems, Inc.

```
;; doc = diffusion overlap of contact
;; pps = poly to poly space
;; pcs = poly to contact space
;; pds = poly to diffusion space

   abutRuleList = list(
      list("abutFlush" 0.0)
      list("abutGateSpaceSeries" pps)
      list("abutGateSpaceParallel" max((cw + (pcs * 2)) pps))
      list("abutEncloseGate" doc)
      list("abutEncloseContact" doc + cw + pcs)
      list("abutSmallerSeries" max((pps - doc) pds))
      list("abutSmallerParallel" max((pcs - doc) (pps - pcs - cw - doc) pds))
      list("abutEncloseDogBoneContact" cw + doc + max(pds+doc pcs))
      list("abutMinContactWidth" cw + (2 * doc))
   )

;; Definition of variables used for abutClass and autoSpacer property values.
;; d_layer = diffusion layer (used for abutClass property value)
;; dds = diffusion to diffusion space (used by autoSpacer)

;; Add the abutment and spacer properties to the left diffusion pin.

   leftObj~>dbId~>pin~>name = "leftContact"
   dbReplaceProp(leftObj~>dbId "abutCondInclusion" "string" "leftCnt")
   dbReplaceProp(leftObj~>dbId "abutMosStretchMat" "list"
   cons("leftSt" abutRuleList))
   dbReplaceProp(leftObj~>dbId "abutClass" "string" d_layer)
   dbReplaceProp(leftObj~>dbId "abutGateNet" "string" "G")
   dbReplaceProp(leftObj~>dbId "abutAccessDir" "list" list("left"))
   dbReplaceProp(leftObj~>dbId "vxlInstSpacingDir" "list" list("left"))
   dbReplaceProp(leftObj~>dbId "vxlInstSpacingRule" "float" dds)

;; Add the abutment and spacer properties to the right diffusion pin.

   rightObj~>dbId~>pin~>name = "rightContact"
   dbReplaceProp(rightObj~>dbId "abutCondInclusion" "string" "rightCnt")
   dbReplaceProp(rightObj~>dbId "abutMosStretchMat" "list"
   cons("rightSt" abutRuleList))
   dbReplaceProp(rightObj~>dbId "abutClass" "string" d_layer)
   dbReplaceProp(rightObj~>dbId "abutGateNet" "string" "G")
   dbReplaceProp(rightObj~>dbId "abutAccessDir" "list" list("right"))
   dbReplaceProp(rightObj~>dbId "vxlInstSpacingDir" "list" list("right"))
   dbReplaceProp(rightObj~>dbId "vxlInstSpacingRule" "float" dds)
```

Related Topics

abutAccessDir Property

abutClass Property

abutCondInclusion Property

abutFunction Property

abutGateNet Property

© 2023 Cadence Design Systems, Inc.

abutStretchMat Property

abutOffset Property

Device Abutment

Creating CMOS Pcells to Use with Abutment

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

abutStretchMat Property

```
abutStretchMat list drainStretch t_material_name
     abutMinExt f_value
     abutRule1Ext f_value
     abutRule2Ext f_value
     abutContactExt f_value
```

Description

Stretches material towards or away from a gate when an instance is altered for abutment.

The default value of the stretch parameter must be the distance from the edge of the material to the edge of the gate. Numeric values are in user units.

- `drainStretch` is the name of the stretchable material parameter. You name this parameter yourself to identify the material to be stretched.
- `abutMinExt` is the rule for minimum diffusion overhang from the edge of the poly gate. The argument to this element is the rule value.
- `abutRule1Ext` is the poly-layer-to-poly-layer rule. This rule is used when the net connecting the two instances does not share the net with any other pin.
- `abutRule2Ext` is the poly-layer-to-diffusion-layer rule. This rule is used when the transistor width is different for the two instances being abutted.



The Virtuoso custom digital placer does not take the abutRule2Ext into consideration, resulting in illegal overlap markers.

- `abutContactExt` is the diffusion extension value used when one of the contacts needs to be added during the abutment process. With the contacts off by default, the abutment program needs to know how close to move the selected cell to abut properly.

Example

```
abutStretchMat list("drainStretch" ilList
     list("abutMinExt" 0.5)
     list("abutRule1Ext" 0.75)
     list("abutRule2Ext" 1.0)
     list("abutContactExt" 3.0)
     )
```


© 2023 Cadence Design Systems, Inc.

Related Topics

abutAccessDir Property

abutClass Property

abutCondInclusion Property

abutFunction Property

abutGateNet Property

abutMosStretchMat Property

abutOffset Property

Device Abutment

Creating CMOS Pcells to Use with Abutment

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

abutOffset Property

`abutOffset float` *`f_value`*

Description

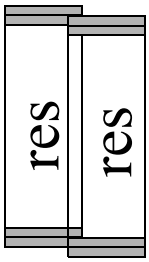
Specifies an offset based on the reference edge. The reference edge is the outside edge of the pin that triggered the abutment on each cell in the direction defined by the abutment direction.

The value number can be positive or negative and indicates a distance in user units to offset the reference edge from the outside direction of the cell. A negative number causes the pins to overlap.

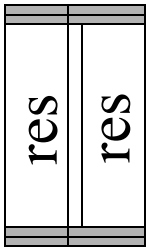
If the `abutFunction` parameter adjustment event returns `nil`, auto-abutment looks for a property on the pin called `abutOffset` and applies the value of this property for offset. If the property `abutOffset` does not exist, auto-abutment uses the outside edge of the pin with no offset.

If the pin of the device being abutted overlaps the other pin on the bottom or top, the pins are aligned by the bottom or top edges, respectively. This also applies from side to side when abutment is vertical.

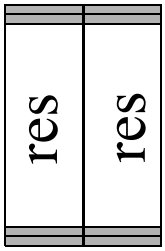
Interactive move places one cell



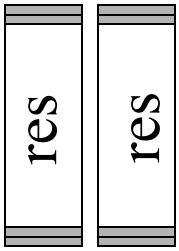
abutOffset = -5



Cells are first aligned and then the offset is applied.



abutOffset = 5



© 2023 Cadence Design Systems, Inc.

Example

```
abutOffset 5
```

Related Topics

abutAccessDir Property

abutClass Property

abutCondInclusion Property

abutFunction Property

abutGateNet Property

abutMosStretchMat Property

abutStretchMat Property

Device Abutment

Creating CMOS Pcells to Use with Abutment

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

caeIgnoreInCluster Property

```
caeIgnoreInCluster boolean { t | nil }
```

Description

Prevents constraint-aware editing from reporting a violation when a user-defined guard ring conflicts with an existing cluster in the design. You can set this property on an individual guard ring instance or supermaster.

- To ignore a a specific user-defined guard ring instance, use the following SKILL command:

  ```
  dbCreateProp(inst, "caeIgnoreInCluster" 'Boolean t)
  ```

- To ignore all instances of a specific guard ring supermaster, use the following SKILL command:

  ```
  dbCreateProp(instID~>master~>superMaster, "caeIgnoreInCluster" 'Boolean t)
  ```

You can also set this property using the layout instance Add Property form.

Layout instance Add Property form	
Name	`caeIgnoreInCluster`
Type	*boolean*
Value	`t | nil`

Related Topics

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

extractStopLevel Property

```
extractStopLevel int { 0 | 1 | 2 | ... | 32 }
```

Description

Overrides the extractStopLevel environment variable for an individual instance or instance master to decrease the effective extraction stop level.

If the value set for this property is lower than that specified by the `extractStopLevel` environment variable, the effective extraction stop level is reduced for the instances in question. For example, if the property value is set to `0` on a particular instance master, only the top level instance pin figures will be seen for instances of this master, even if the `extractStopLevel` environment variable is set to a value greater than `0`.

You set the `extractStopLevel` property in the Add Property form.

Layout instance Add Property form	
Name	`extractStopLevel`
Type	*Int*
Value	Integer value between 0 and 32 inclusive

Related Topics

extractStopLevel

Connectivity Extraction Use Model

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

extractToLevel Property

```
extractToLevel int { 0 | 1 | 2 | ... | 32 }
```

Description

Overrides the extractStopLevel environment variable for an individual instance or instance master to increase the effective extraction stop level.

If the value set for this property is greater than that specified by the `extractStopLevel` environment variable, the effective extraction stop level is increased for the instances in question. For example, if the property value is set to `1` on a particular instance master, the shapes at hierarchical depth `1` will be seen for instances of this master, even if the `extractStopLevel` environment variable is set to `0`. This is the case for Pcells also, even if the *Extract through pcells* option on the *Connectivity* tab of the Connectivity form is disabled.

You set the `extractToLevel` property in the Add Property form.

Layout instance Add Property form	
Name	`extractToLevel`
Type	*Int*
Value	Integer value between 0 and 32 inclusive

Related Topics

extractStopLevel Property

extractStopLevel (Layout L environment variable)

Connectivity Extraction Use Model

Connectivity Form

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

ignore Property

```
ignore boolean { t | nil }
```

Description

Prevents a device symbol from being generated in the layout. If the `ignore` property on the symbol is removed, the device is generated in the layout and can be bound.

You can also set the lvsIgnore Property property for layout devices if you do not want Layout XL to check them, but want the instances to be bound.

You can set the `ignore` property in either the symbol master Add Property form or the Add CDF Parameter form.

Symbol master Add Property form	
Name	`ignore`
Type	*boolean*
Value	`t | nil`

Add CDF Parameter form	
Add After Parameter	*As First Parameter*
paramType	*boolean*
name	`ignore`
defValue	`t | nil`

You can also use the Configure Physical Hierarchy Window (CPH window) to set the *ignore for generation* or *ignore for check* properties at the cell or instance level. If any updates are made to the *ignore* property settings using the CPH window, *Generate All From Source* considers the value set using the CPH window instead of the value on the schematic. For more information, see How to Ignore Components in Physical Hierarchy.

Related Topics

lvsIgnore Property

© 2023 Cadence Design Systems, Inc.

physOnly Property

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

lvsIgnore Property

```
lvsIgnore boolean { t | nil }
```

Description

Defines an instance as one that is generated, but ignored for check. Examples of such instances used in the Virtuoso flow are dummy instances and filler cells. Dummy instances and filler cells have an `lvsIgnore` property added so that the instances are ignored for check during a *Check Against Source* run, although the binder can still bind them correctly.

The `lvsIgnore` property can be set either on the schematic instance or on the layout device.

The `lvsIgnore` property on instances is also used by the Layout XL connectivity extractor to remove the connectivity on the instances terminals if the device is unabutted or moved away from another device that it was overlapping and drawing connectivity from. After the unabutment or the move, if the instance overlaps another device, Layout XL automatically reassigns the connectivity based on the new overlap. You can make the connectivity of a physical-only instance sticky by setting the lxStickyNet Property property on the instance.

You set the `lvsIgnore` property in the Add Property form.

Layout instance Add Property form	
Name	`lvsIgnore`
Type	*Boolean*
Value	`t	nil`



Tip

Use the *Add ignore check* command from the shortcut menu of an instance (or pin) to add the `lvsIgnore` property. For an instance (or pin) that already has the property added, the shortcut menu provides a corresponding option to remove the property.

Alternatively, you can use the Configure Physical Hierarchy Window (CPH window) to set the *ignore for check* properties at the cell or instance level. If any updates are made to the *ignore* property settings using the CPH window, *Generate All From Source* considers the value set using the CPH window instead of the value on the schematic. For more information, see How to Ignore Components in Physical Hierarchy.

© 2023 Cadence Design Systems, Inc.

Additional Information

By default, lvsIgnore is included in the list of propsUsedToIgnoreObjsForCheck.

Related Topics

ignore Property

physOnly Property

Instance Terminal Connectivity Considered During Extraction

How to Ignore Components in Physical Hierarchy

Additional Pins on Ignored Instances

Layout Parameters Updates

Updating Schematic Parameter Values in Preselection Mode

Updating Schematic Parameter Values in Postselection Mode

Parameters

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

lxAutoAbut Property

```
lxAutoAbut boolean { t | nil }
```

Description

Prevents abutment if either instance in the abutment pair has this property set to `nil`.

`lxAutoAbut` is a CDF-only property.

For situations when the CDF property, `lxAutoAbut`, cannot be changed to prevent abutment, adding the instances to a group can help.

Add CDF Parameter form	
Add After Parameter	*As First Parameter*
paramType	*boolean*
name	`lxAutoAbut`
defValue	`t	nil`

Related Topics

Device Abutment

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

lxAutoSpace Property

```
lxAutoSpace boolean { t | nil }
```

Description

Prevents spacing if either instance in the abutment pair has this property set to `nil`.

`lxAutoSpace` is a CDF-only property.

Add CDF Parameter form	
Add After Parameter	*As First Parameter*
paramType	*boolean*
name	`lxAutoSpace`
defValue	`t	nil`

Related Topics

Device Abutment

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

lxCombination Property

`lxCombination string` *`t_expression`*

Description

Builds a complex set of devices comprising series (`sfactor`) and parallel (`mfactor`) connections.

You typically use `lxCombination` to split a single resistor into multiple smaller devices in order to achieve maximum density in the layout by placing devices in spaces otherwise not utilized. This lets you approximate the resistance value defined in the schematic without violating the manufacturing grid.

The lxSeriesTerms Property property must be defined on the instance in the schematic in order for Layout XL to understand the connectivity of devices with sfactor >1 and instance terminals>2.

Symbol master Add Property form	
Name	`lxCombination`
Type	*string*
Value	*`t_expression`*

Add CDF Parameter form	
Add After Parameter	*As First Parameter*
paramType	*string*
name	`lxCombination`
defValue	*`t_expression`*

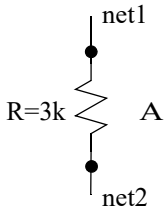
Example

In a schematic cellview, a resistor called `A` with resistance `3k` is connected to `net1` and `net2`, and has the following `lxCombination` expression defined.

```
(2k*2)/3+1500/4
```


© 2023 Cadence Design Systems, Inc.

Schematic



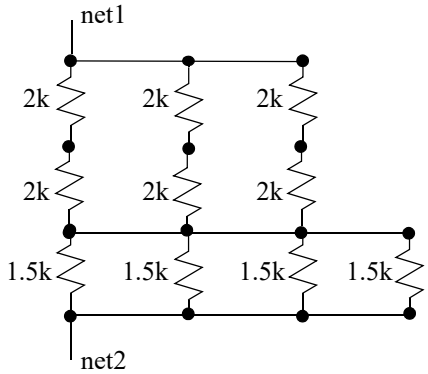
lxCombination=(2k * 2) / 3 + 1500/ 4

Where

- `2k`, `2`, `3`, `1500`, and `4` are device parameter values
- `*` and `+` are series connections (`+` can be followed by an expression; `*` must be followed by an integer)
- `/` is a parallel connection (and must be followed by an integer)

During layout generation, Layout XL reads the `lxCombination` property, ignores the `3k` value, and instead generates a total of 10 resistors: two `2k` resistors connected in series, connected in parallel three times; and four parallel-connected `1.5k` resistors connected in series to the 2k resistors.

Layout



© 2023 Cadence Design Systems, Inc.

Additional Information

- To ensure that `lxCombination` expressions are evaluated correctly, you must set the lxEvalCDFCallbacks environment variable to `t` before generating your resistors.
- You can also use `lxCombination` to define the topology for a network of capacitors. However, in this case, the property describes *only* the topology of the network; the values entered do not equate to the total capacitance of the network.
- If the `lxCombination` property is defined on the schematic symbol, the `r` parameter is ignored by the *Generate All From Source* and *Update Components And Nets* commands. The layout instances will have the correct parameters based on the `lxCombination` factor setting.
- Devices defined using `lxCombination` are named *`instName`*`.ms`*`integer`* in the layout. In our example above, if the schematic instance is called `R0`, the layout instances generated are named as follows.

Schematic	Layout
R0	|R0.ms1 |R0.ms2 |R0.ms3 |R0.ms4 |R0.ms5 |R0.ms6 |R0.ms7 |R0.ms8 |R0.ms9 |R0.ms10

Nets associated with `lxCombination` devices are named *`instName`*`.s`*`integer`*`.expandedNet`*`integer`* in the layout.

For example,

```
R0.s1.expandedNet1
```

where "`expandedNet`" signifies that the layout net has no real match in the schematic.

Related Topics

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

lxLocalAbutment Property

```
lxLocalAbutment boolean { t | nil }
```

Description

Overrides the `lxLocalAbutment` environment variable setting for individual instances.

If the `lxLocalAbutment` property is set to `t` and the environment variable set to `nil`, the environment variable behavior for `lxLocalAbutment nil` is overridden for the instance.

However, when the environment variable is set to `t` and the property is set to `nil`, there is no override. The behavior of the environment variable for `lxLocalAbutment t` stays valid.

Related Topics

lxLocalAbutment

© 2023 Cadence Design Systems, Inc.

lxMfactorSplit Property

Description

This property is automatically converted to the new Layout XL schema underlying physical hierarchy configuration.

To control whether Layout XL places a schematic device with the mfactor Property property as multiple devices in the layout,

1. From the layout window menu bar, choose *Launch – Configure Physical Hierarchy*.
2. In the *Instances* table, select the logical instance for which you want to generate multiple devices.
3. In the *Generation* tab, switch on the *Split mfactored devices* option.

   This overrides the global setting on the Generation of the Connectivity form.

   Split mfactored devices is grayed out if the selected instances do not have the `mfactor` property.

Related Topics

Layout Generation

Data Conversion to Use the Layout XL Schema

Physical Hierarchy Configuration

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

lxNetNamePrefix Property

`lxNetNamePrefix string` *`t_prefix`*

Description

Adds a unique prefix to internal nets generated in the layout using Layout XL conventions.

`lxNetNamePrefix` can be used only with devices which have the `lxCombination` property or `sfactor` parameter.

Symbol master Add Property form	
Name	`lxNetNamePrefix`
Type	*string*
Value	*`t_prefix`*

Add CDF Parameter form	
Add After Parameter	*As First Parameter*
paramType	*string*
name	`lxNetNamePrefix`
defValue	*`t_prefix`*

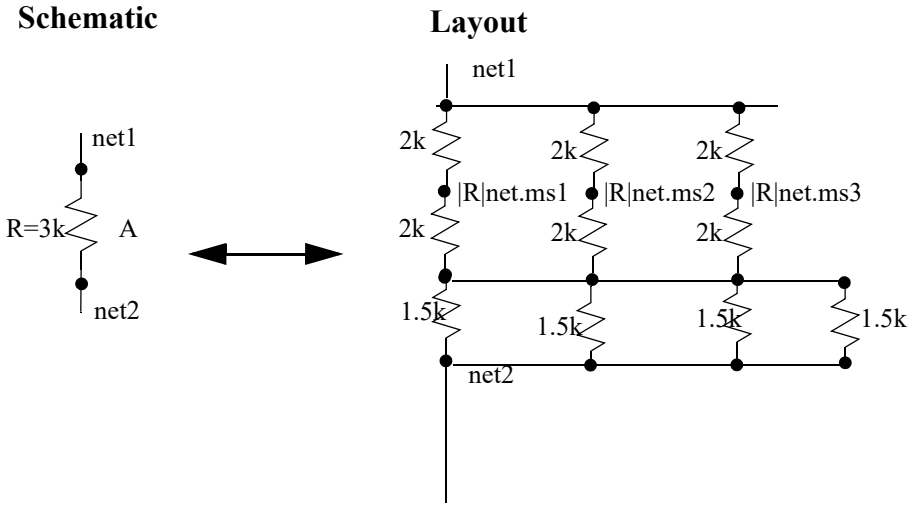
Example

Using the lxCombination Property property to generate layout devices from the `R` device in the schematic below generates three internal nets with the following names.

- `|R|net.ms1`
- `|R|net.ms2`

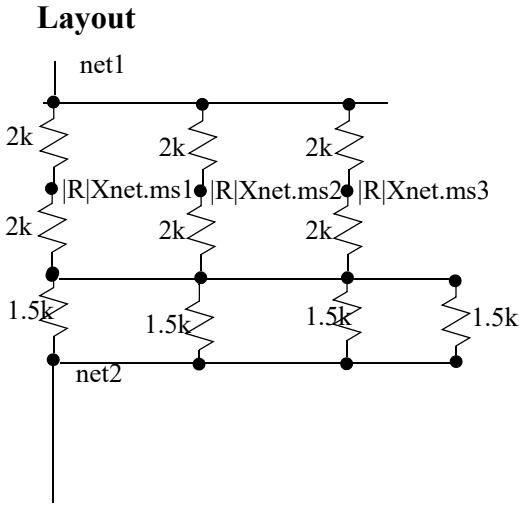
© 2023 Cadence Design Systems, Inc.

- `|R|net.ms3`



Using the `lxNetNamePrefix` property, you can add the prefix `X` to the internal net names. The names change to

- `|R|Xnet.ms1`
- `|R|Xnet.ms2`
- `|R|Xnet.ms3`



Related Topics

Layout Generation

© 2023 Cadence Design Systems, Inc.

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

lxParamsToCheck Property

`lxParamsToCheck string` *`list_of parameter_names`*

Description

Specifies the device parameters to be checked when using the *Check Against Source* command.

All other parameters are automatically ignored for check.

The `lxParamsToCheck` property can be set on an instance in the schematic or layout or on the instance master.

If the property is set in the schematic, the shell environment variable "`CPH_USE_SCHEM_PROP`" must be set to `1`.

```
setShellEnvVar("CPH_USE_SCHEM_PROP=1") or setenv CPH_USE_SCHEM_PROP 1
```

Layout instance Add Property form	
Name	`lxParamsToCheck`
Type	*string*
Value	*`list_of_parameter_names`*

Additional Information

The parameters to be checked can be specified in the physConfig at an occurrence, instance or cell level, or in the **LAM** file or using the global layout XL paramsToCheck environment variable.

Related Topics

Data Conversion to Use the Layout XL Schema

Physical Hierarchy Configuration

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

lxParamsToIgnore Property

`lxParamsToIgnore string` *`list_of parameter_names`*

Description

Specifies the device parameters to be ignored when using the following Layout XL commands.

- *Generate All From Source*
- *Generated Selected From Source*
- *Generate Clones*
- *Check Against Source*
- *Update Components And Nets*
- *Update Layout Parameters*

Important

Cadence recommends that you use `lxParamsToIgnore` instead of the `lxIgnoredParams` property.

The `lxParamsToIgnore` property can be set on an instance in the schematic or layout or on the instance master.

If the property is set in the schematic, the shell environment variable "`CPH_USE_SCHEM_PROP`" must be set to `1`.

```
setShellEnvVar("CPH_USE_SCHEM_PROP=1") or setenv CPH_USE_SCHEM_PROP 1
```

Layout instance Add Property form	
Name	`lxParamsToIgnore`
Type	*string*
Value	*`list_of_parameter_names`*

© 2023 Cadence Design Systems, Inc.

Additional Information

You can also specify the parameters to be ignored by the *Generate*, *Update*, and *Check* commands in the Configure Physical Hierarchy window. To do this,

1. From the layout window menu bar, choose *Launch – Configure Physical Hierarchy*.
2. In the *Instances* table, select the schematic instance for which you want to ignore parameters.
3. In the *Parameters* tab, list the parameters to be ignored in the *Ignore for generation* text field.

   This overrides the global setting on the Parameters form.

Related Topics

LAM file

Data Conversion to Use the Layout XL Schema

Physical Hierarchy Configuration

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

lxParamsToIgnoreForCheck Property

`lxParamsToIgnoreForCheck string` *`list_of parameter_names`*

Description

Specifies the device parameters to be ignored when using the *Check Against Source* command.



Important

Cadence recommends that you use `lxParamsToIgnore` instead of the `lxIgnoredParamsForCAS` property.

The `lxParamsToIgnoreForCheck` property can be set on an instance in the schematic or layout or on the instance master.

If the property is set in the schematic, the shell environment variable "`CPH_USE_SCHEM_PROP`" must be set to `1`.

```
setShellEnvVar("CPH_USE_SCHEM_PROP=1") or setenv CPH_USE_SCHEM_PROP 1
```

Layout instance Add Property form	
Name	`lxParamsToIgnoreForCheck`
Type	*string*
Value	*`list_of_parameter_names`*

Additional Information

You can also use the Configure Physical Hierarchy window to specify the parameters to be ignored by the *Check Against Source* command. To do this,

1. From the layout window menu bar, choose *Launch – Configure Physical Hierarchy*.
2. In the *Instances* table, select the logical instance for which you want parameter differences to be ignored.
3. In the *Parameters* tab, list the parameters to be ignored in the *Ignore for check* text field.

   This overrides the global setting on the Parameters of the Connectivity form.

© 2023 Cadence Design Systems, Inc.

Related Topics

Data Conversion to Use the Layout XL Schema

Physical Hierarchy Configuration

Connectivity Form

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

lxRemoveDevice Property

Description

The property is automatically converted to the new Layout XL schema underlying the physical hierarchy configuration.

To cause parasitic devices to be merged:

1. From the layout window menu bar, choose *Launch – Configure Physical Hierarchy*.
2. In the *Instances* table, select the logical instance in which you want to merge nets.
3. In the *Generation* tab, type the argument into the *Remove device* text field.

`lxRemoveDevice` is recognized by Layout XL if the shell environment variable, "`CPH_USE_SCHEM_PROP`", is set to `1` and the property has not already been set in the physical hierarchy configuration.

`lxRemoveDevice` can be set in the physConfig due to a previous automatic conversion or by directly editing the physConfig.

You can use the *Add ignore* command from the shortcut menu of an instance or pin to add the `lxRemoveDevice` property. For an instance (or pin) that already has the property added, the shortcut menu provides a corresponding option to remove the property.

The shell environment variable below must be set before launching the Virtuoso session.

```
setShellEnvVar("CPH_USE_SCHEM_PROP=1") or setenv CPH_USE_SCHEM_PROP 1
```

Related Topics

Ignoring Parasitic Devices

Data Conversion to Use the Layout XL Schema

Physical Hierarchy Configuration

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

lxRounding

This property is obsolete in the current release. It is automatically converted to the new Layout XL schema underlying physical hierarchy configuration.

To define how the value of a specified parameter on a cell master or schematic instance is rounded when it is evaluated by Layout XL,

1. From the layout window menu bar, choose *Launch – Configure Physical Hierarchy*.
2. In the table view, select the instance or cell in which you want to round values.
3. In the *Parameters* tab, type the required argument into the *Rounding* text field.

For more information, see

Related Topics

Rounding Parameter Values

Data Conversion to Use the Layout XL Schema

Physical Hierarchy Configuration

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

lxSeriesTerms Property

`lxSeriesTerms string` *`t_terminal_name`*

Description

Specifies the connectivity of schematic and lxCombination devices with `sfactor` greater than 1 and instance terminals greater than 2. Set this property on the schematic instance.

The *Split mfactored devices* option must be turned on in the Generation of the Connectivity form.

The environment variable does not support more than two terminals in series. To connect more than two terminals, expand them in the schematic and connect them in series or use a schematic Pcell to generate the series.

Important

Cadence recommends that this property be set in the PDK. For some reason, if you set this in the middle of a Virtuoso session, it is recommended that you set Update Connectivity Reference Form to *None* and then again set to the schematic for the property to take effect.

Symbol master Add Property form	
Name	`lxSeriesTerms`
Type	*string*
Value	*`t_terminal_names`*

Add CDF Parameter form	
Add After Parameter	*As First Parameter*
paramType	*string*
name	`lxSeriesTerms`
defValue	*`t_terminal_names`*

© 2023 Cadence Design Systems, Inc.

Example

The description below shows setting the `lxSeriesTerms` `PLUS` and `MINUS` on two instance terminals.

```
lxSeriesTerms = PLUS MINUS
```

Related Topics

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

lxSfactorSplit Property

Description

This property is automatically converted to the new Layout XL schema underlying physical hierarchy configuration.

To control whether Layout XL places a schematic device with the sfactor Property property as series-connected devices in the layout,

1. From the layout window menu bar, choose *Launch – Configure Physical Hierarchy*.
2. In the *Instances* table, select the logical instance for which you want to generate series-connected devices.
3. In the *Generation* tab, switch on the *Split sfactored devices* option.

   This overrides the global setting on the Generation of the Connectivity form.

   - *Split sfactored devices* is grayed out if the selected instances do not have the `sfactor` property.
   - The shell environment variable "CPH_USE_SCHEM_PROP=1" must be set, if CPH must read the lxSfactorSplit Property property. Else, the property is upreved to CPH and driven from there.

Related Topics

Layout Generation

Data Conversion to Use the Layout XL Schema

Physical Hierarchy Configuration

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

lxStickyNet Property

```
lxStickyNet boolean { t | nil }
```

Description

Allows a shape to retain its assigned connectivity even after it has been modified in some way.

Layout XL adds the `lxStickyNet` property automatically to a shape or via when a net is assigned to that shape using the *Add Shape To Net* command. The property remains in effect until you:

- Override it with a new net assignment
- Turn it off by setting it to `nil`
- Remove it using the *Edit – Basic – Properties* command.

Additionally:

- Choose *Edit – Basic –Properties* command to manually add the lxStickyNet property.
- For an instance with lvsIgnore Property property to have sticky connectivity, add the lxStickyNet property on the instance or on the instance master.

When the lxStickyNet property is added, it prevents reassignment of the connectivity of gate terminals and non-abutting source and drain terminals by the extractor.

For more information on `lxStickyNet`, see Interactive Wire Editing in the *Virtuoso Space-based Router User Guide*.

Layout instance Add Property form	
Name	`lxStickyNet`
Type	*Boolean*
Value	`t | nil`

Related Topics

addlxStickyNetPropOnDummy

Updating Schematic Parameter Values in Preselection Mode

© 2023 Cadence Design Systems, Inc.

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

lxStopList Property

Description

This property is obsolete in the current release. It is automatically converted to the new Layout XL schema underlying physical hierarchy configuration.

To define where Layout XL should stop when it is traversing the design hierarchy,

1. From the layout window menu bar, choose *Launch – Configure Physical Hierarchy*.
2. In the *Instances* or *Cells* table, type the view names you require into the *Inherited Stop List* field for the instance or cell in question.

To prevent Layout XL from automatically upreving `lxStopList` (and other "`lx`" properties such as `lxViewList` and `lxUseCell)`, and to continue using the schematic property instead, set the CPH shell environment variable, `CPH_USE_SCHEM_PROP`, to `1`, before launching Layout XL. If `CPH_USE_SCHEM_PROP` is set to `1`, Layout XL continues to read and honor any updates made to the properties in the schematic.

If you have already upreved the `lx` properties or made edits to the physical bindings (`lxUseCell`), physical stop list (`lxStopList`), or view list (`lxViewList`), then the value set in CPH takes precedence. To continue using the corresponding schematic properties, you must remove the edits from CPH.

Related Topics

Data Conversion to Use the Layout XL Schema

Physical Hierarchy Configuration

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

lxUseCell Property

Description

This property is obsolete in the current release. It is automatically converted to the new Layout XL schema underlying physical hierarchy configuration.

To specify which cell to use when generating a layout for a device symbol:

1. From the layout window menu bar, choose *Launch – Configure Physical Hierarchy*.
2. Do one of the following.
   - Use the *Cells* table to specify which layout cell to use for a particular logical cell.
   - Use the *Instances* table to specify which layout cell to use for a specific logical instance.

To prevent Layout XL from automatically upreving `lxUseCell` (and other "`lx`" properties such as `lxStopList` and `lxViewList)`, and to continue using the schematic property instead:

Set the CPH shell environment variable, `CPH_USE_SCHEM_PROP`, to `1`, before launching Layout XL. If `CPH_USE_SCHEM_PROP` is set to `1`, Layout XL continues to read and honor any updates made to the properties in the schematic.

If you have already upreved the `lx` properties or made edits to the physical bindings (`lxUseCell`), physical stop list (`lxStopList`), or view list (`lxViewList`), then the value set in CPH takes precedence. To continue using the corresponding schematic properties, you must remove the edits from CPH.

Related Topics

Data Conversion to Use the Layout XL Schema

Physical Hierarchy Configuration

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

lxViewList Property

Description

This property is obsolete in the current release. It is automatically converted to the new Layout XL schema underlying physical hierarchy configuration.

To tell Layout XL where to descend into the design hierarchy to find cells at a lower level,

1. From the layout window menu bar, choose *Launch – Configure Physical Hierarchy*.
2. In the *Instances* or *Cells* table, type the view names you require into the *Inherited View List* field for the instance or cell in question.

To prevent Layout XL from automatically upreving `lxViewList` (and other "`lx`" properties such as `lxStopList` and `lxUseCell)`, and to continue using the schematic property instead:

Set the CPH shell environment variable, `CPH_USE_SCHEM_PROP`, to `1`, before launching Layout XL. If `CPH_USE_SCHEM_PROP` is set to `1`, Layout XL continues to read and honor any updates made to the properties in the schematic.

If you have already upreved the `lx` properties or made edits to the physical bindings (`lxUseCell`), physical stop list (`lxStopList`), or view list (`lxViewList`), then the value set in CPH takes precedence. To continue using the corresponding schematic properties, you must remove the edits from CPH.

Related Topics

Data Conversion to Use the Layout XL Schema

Physical Hierarchy Configuration

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

mfactor Property

`mfactor integer` *`x_integer`*

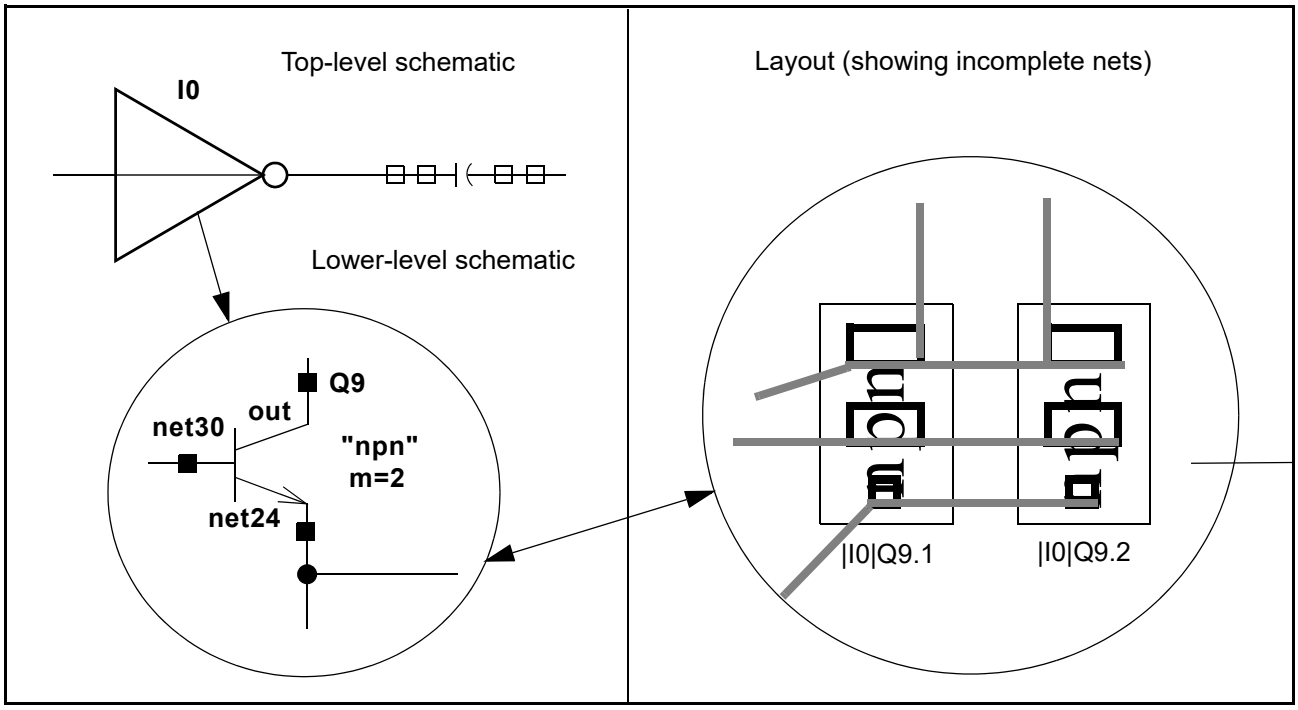
Description

Defines a one-to-many parallel relationship between a device in a schematic and multiple instances of the device in the layout.

Important

You cannot use multiplication factors with components that cannot be used in parallel, such as voltage sources.

The `mfactor` property and `mfactorSplit` environment variable both work hierarchically. Any instance in the connectivity source hierarchy can have these properties, and the multiplicity is computed from the hierarchical path.



You can use the mfactorSplit environment variable to control whether the `mfactor` produces multiple layout devices or not. You can also use the lxMfactorSplit Property property on a

© 2023 Cadence Design Systems, Inc.

given instance to override the global value given by the `mfactorSplit` environment variable.

Symbol master Add Property form	
Name	`mfactor`
Type	*int*
Value	*`x_integer`*

Add CDF Parameter form	
Add After Parameter	*As First Parameter*
paramType	*int*
name	`mfactor`
defValue	*`x_integer`*

Naming Convention

Devices specified using the `mfactor` property in the schematic (and those that are folded in the layout) are named *`instName.integer`* in the layout.

For example, if you have a schematic instance named `P0` with

```
mfactor = 3
```

the layout instances generated are named as follows.

Schematic	**Layout**

`P0`	`|P0.1`
	`|P0.2`
	`|P0.3`

© 2023 Cadence Design Systems, Inc.

Or, if the prefixLayoutInstNamesWithPipe environment variable is set to `nil`.

Schematic	Layout

`P0`	`P0.1`
	`P0.2`
	`P0.3`

Related Topics

Layout Generation

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

nonStickyTerminals Property

`nonStickyTerminals string` *`l_terminals`*

Description

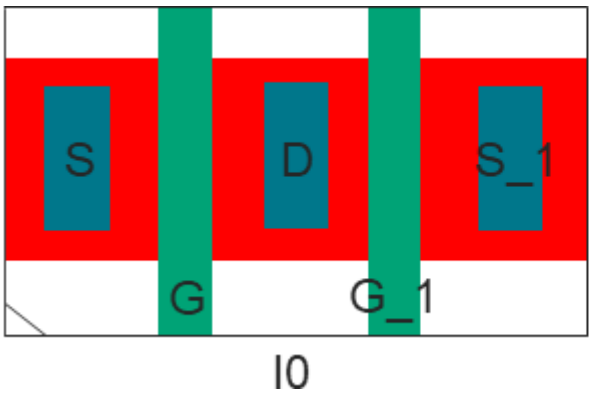
The `nonStickyTerminals` property is used for an individual instance or instance master. When the connectivity is removed from the instance terminals, the connectivity extractor does not report any open or short markers.

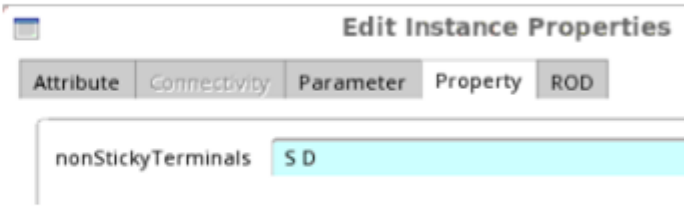
Layout instance Add Property form	
Name	`nonStickyTerminals`
Type	*string*
Value	*`l_terminals`*

Examples

In the following example:

- Instance `I0` has three terminals `S`, `D` and `G`
- `nonStickyTerminals` is set on terminal `S` and `D` of instance `I0`

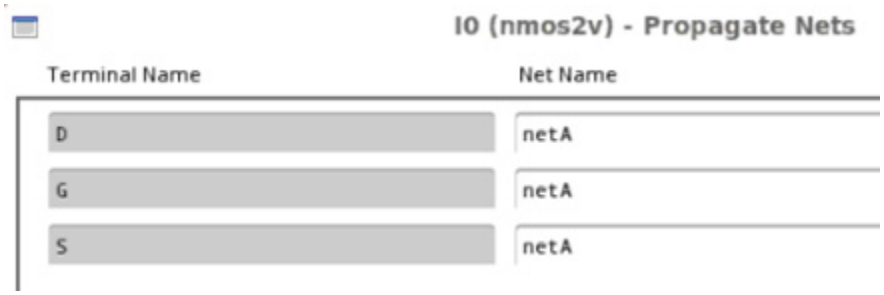




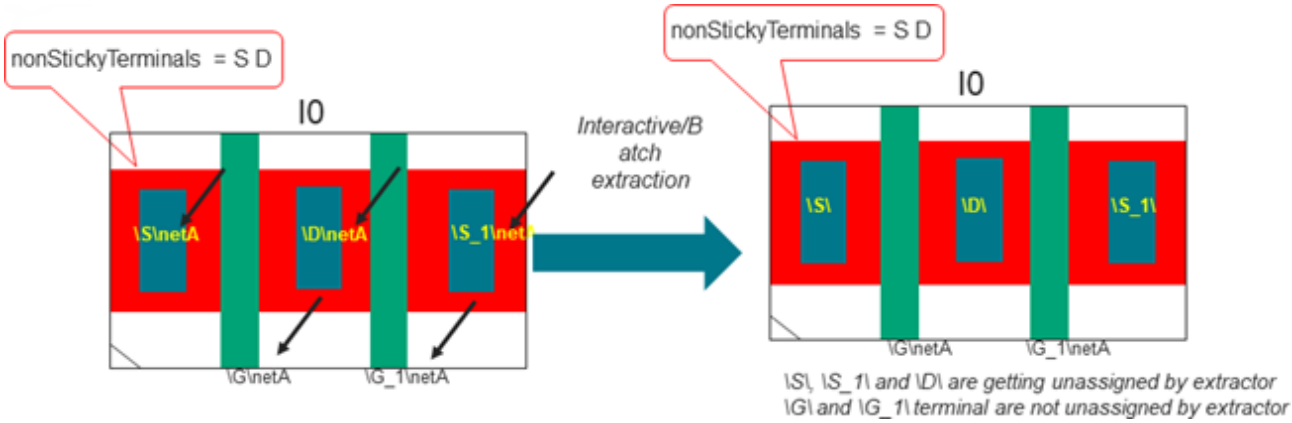
Here, `S_1` and `S`, `G` and `G_1` are the must connect terminals.

© 2023 Cadence Design Systems, Inc.

Consider that `netA` is set on three terminals using the *Connectivity – Nets – Propagate* command.



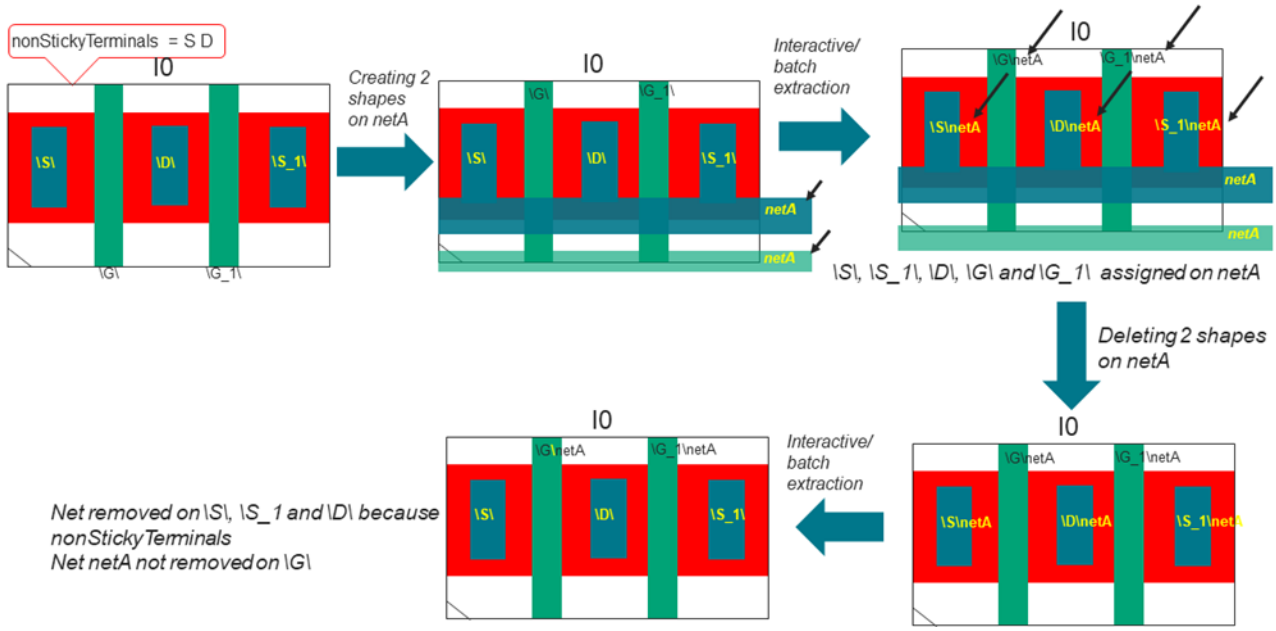
Interactive or the batch connectivity extractor removes the connectivity on `S`, `S_1`, and `D`. On terminal `G`, the connectivity is not removed.



In the following example with `nonStickyTerminals` `S` and `D`, two shapes are created on `netA`. After interactive or batch extraction, `S`, `S_1`, `D`, `G`, and `G_1` are assigned on `netA`. If the

© 2023 Cadence Design Systems, Inc.

shapes are deleted from `netA`, then after interactive or batch extraction, the net is removed from `S`, `S_1`, and `D` but not from `G`.



© 2023 Cadence Design Systems, Inc.

permuteRule Property

`permuteRule string` *`t_permuteRule`*

Description

Makes the instance pins or terminals of a device permutable.

- `(p E1 E2)` specifies that `E1` and `E2` are permutable.
- `(f E1 E2)` specifies that `E1` and `E2` are fixed (i.e., not permutable).

Alternatively, you can use the following syntax to same effect:

```
p (E1 E2)
f (E1 E2)
```

The `permuteRule` property can be defined on the layout or symbol master, in the CDF, or on the schematic or layout instance. Depending on where the `permuteRule` property is defined in a design, the rule that is used is based on the following precedence:

1. Look for the `autoPermutePins` property on the layout instance. If the property is set to `false`, do not permute.
2. Look for the permute rule in CDF to get the instance CDF for the layout instance. Search for the `permuteRule` CDF parameter. If the parameter is present and not an empty string, return the value. Else, search in `simInfo` and `auLvs`. If the parameter is present and not an empty string, return the value.
3. Find the schematic instance this layout instance is bound to and search for the `permuteRule` property on the schematic instance. If the parameter is present and not an empty string, return the value. Else, search for the `permuteRule` property on the master of the schematic instance. If the parameter is present and not an empty string, return that value. Else, search for the `permuteRule` property on the supermaster of the schematic instance. If the parameter is present and not an empty string, return the value.

Search for the `permuteRule` property on the layout instance. If the parameter is present and not an empty string, return the value. Else, search for the `permuteRule` property on the master of the layout instance. If the parameter is present and not an empty string, return that value. Else, search for the `permuteRule` property on the supermaster of the layout instance. If the parameter is present and not an empty string, return the value.

All pins that are not listed in the `permuteRule` property for a device are considered fixed. If you define the `permuteRule` property on a schematic symbol, that setting overrides any other rule defined in the layout for that instance.

© 2023 Cadence Design Systems, Inc.

You define the `permuteRule` in the Edit Component CDF form in either the *Component Parameters* section or the *Simulation Information* section. If it is defined in both sections, Layout XL uses the definition in the *Component Parameters* section.

After you add the `permuteRule` in the Edit Component CDF form, make sure that you run either of the following commands, for the permuteRule to take effect:

- *Connectivity – Check – XL Compliance* command
- *Connectivity – Update – Binding*

Alternatively, you can set the *Update Connectivity Reference* to *None* and then again set to the schematic for the property to take effect.

If you want Layout XL to permute pins automatically, switch on the *Permute pins* option in the Generation of the Connectivity form. If this option is switched off, you can only permute pins manually. Use the *Disable Permutation* constraint to prevent permutation for a selected instance. See Disabling Permutation for an Instance.

Symbol master Add Property form	
Name	`permuteRule`
Type	*string*
Value	*`t_permuteRule`*

Add CDF Parameter form	
Add After Parameter	*As First Parameter*
paramType	*string*
name	`permuteRule`
defValue	*`t_permuteRule`*

Examples

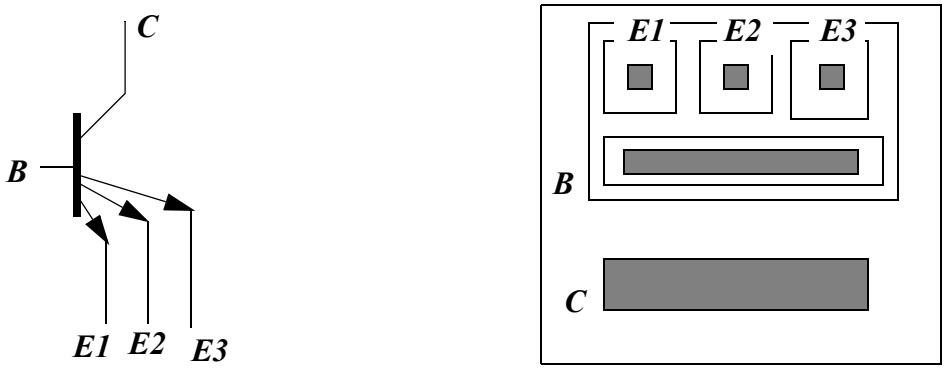
- To specify that pins `E1` and `E2` are permutable, use the following `permuteRule`.

```
(p E1 E2)
```

- To specify that pins `E1`, `E2`, and `E3` are permutable, use the following `permuteRule`.

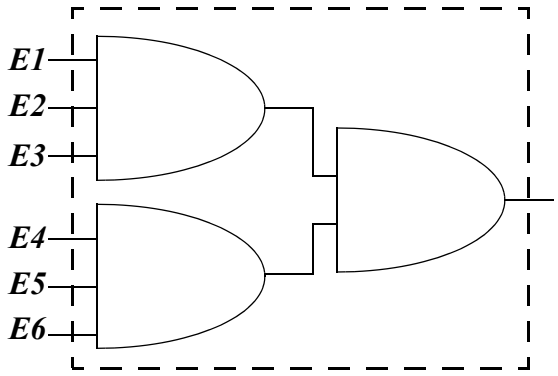
© 2023 Cadence Design Systems, Inc.

```
(p E1 E2 E3)
```



- You can also specify a hierarchical `permuteRule`. For example, the value below defines two sets of permutable pins (`E1`, `E2`, `E3` and `E4`, `E5`, `E6)` and specifies that the two sets are permutable with each other.

```
(p (p E1 E2 E3) (p E4 E5 E6))
```



- The following `permuteRule` value defines two sets of permutable pins (`E1`, `E2`, `E3` and `E4`, `E5`, `E6)` and specifies that the two sets are not permutable with each other.

```
(f (p E1 E2 E3) (p E4 E5 E6))
```

Here is a sample block of code to look for `permuteRule` for a given layout instance. If the value returned is an empty string, it implies no permute.

```
string findPermuteRule(dbInstId layInst)  {

      if (layInst->hasBoolProp("autoPermutePins", &pValue)) {
      if (pValue == false) {
        return "";
       }
     }
```


© 2023 Cadence Design Systems, Inc.

```
     cdf = getInstCDF(layInst);

     if (param = cdf->findParam(layInst, "permuteRule") {
     if (param->valueNotEmptyString()) {
          return param->value();
     }
     return ""
     } else if (param = cdf->findParam(layInst,
"simInfo/auLvs/permuteRule")) {
      // look for permuteRule property in CFD simInfo, under auLvs
      if (param->valueNotEmptyString()) {
           return param->value();
     }

     schInst = binder->findSchInst(layInst);    // find schematic instance
this layout instance is bound to

    if (schInst->hasStringProp("permuteRule", &pValue)) {
      if (pValue->valeNotEmptyString()) {
           return pValue;
      }
    }

    schMaster = schInst->master;

    if (schMaster->hasStringProp("permuteRule"), &pValue)) {
     if(pValue->valueNotEmptyString()) {
          return pValue;
     }
     }

     schSuperMaster = schMaster->superMaster;

     if (schSuperMaster->hasStringProp("permuteRule"), &pValue)) {
     if(pValue->valueNotEmptyString()) {
        return pValue;
     }
     }

      if (layInst->hasStringProp("permuteRule", &pValue)) {
      if (pValue->valeNotEmptyString()) {
```


© 2023 Cadence Design Systems, Inc.

```
            return pValue;
          }
        }

        layMaster = layInst->master;

        if (layMaster->hasStringProp("permuteRule"), &pValue)) {
         if(pValue->valueNotEmptyString()) {
              return pValue;
         }
         }

         laySuperMaster = layMaster->superMaster;

         if (laySuperMaster->hasStringProp("permuteRule"), &pValue)) {
         if(pValue->valueNotEmptyString()) {
            return pValue;
         }
         }

        return "";
    }
```

Macros

To avoid entering long lists of permutable pins for the `permuteRule` property, use the following macro notations.

- `(f ALL)`, which means that all the pins in the cell are fixed
- `(p ALL)`, which means that all the pins in the cell are permutable
- Range indications, where pins are numbered or sorted as in bus notations.

  For example, the notation

  ```
  (p A<0:3> B<4:7>)
  ```

  is equivalent to

  ```
  (p A<0> A<1> A<2> A<3> B<4> B<5> B<6> B<7>)
  ```

  The notation

  ```
  (p (f A<0:3>) (f B<4:7>))
  ```


© 2023 Cadence Design Systems, Inc.

defines two sets of fixed pins (`A<0:3>` and `B<4:7>`) which are permutable with each other

You can also express range indications in descending order

```
(p A<3:0> B<7:4>)
```

Related Topics

Pin Permutation

Disabling Permutation for an Instance

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

physOnly Property

Description

The `physOnly` or `physicalOnly` attribute is applicable for Layout XL designs that need to be interoperable with Cadence Innovus Digital Implementation System for mixed-signal design implementation.

If the `physOnly` attribute is set on a layout instance:

- The Layout XL binder ignores it and leaves the instance unbound. Likewise, the instance is also ignored during a *Check Against Source* run.
  - In Layout XL, the `physOnly` attribute is automatically set on *filler* cells during floorplanning.
  - The Innovus Digital Implementation System interprets it as an instance that needs to be implemented only in the OA block domain, not in the OA module domain.
  - The OA block domain includes the physical connectivity and the design geometry, whereas the OA module domain includes the logical connectivity and the design hierarchy. Therefore, if a layout instance has the `physOnly` attribute set to `true`, the instance will only be implemented in the block domain using Cadence Innovus Digital Implementation System.

After you set the `physOnly` attribute on an instance, you cannot remove the attribute. You will need to delete the instance and recreate it without the attribute set.

You can also set the `physOnly` attribute on instance terminals to mark them as block domain-only, like inherited connections.

Create Instance Form	
Physical Only	*True*

Related Topics

ignore Property

lvsIgnore Property

Setting Layout XL Properties

© 2023 Cadence Design Systems, Inc.

sfactor Property

`sfactor { integer | float | string | IL expression }` *`user_defined_value`*

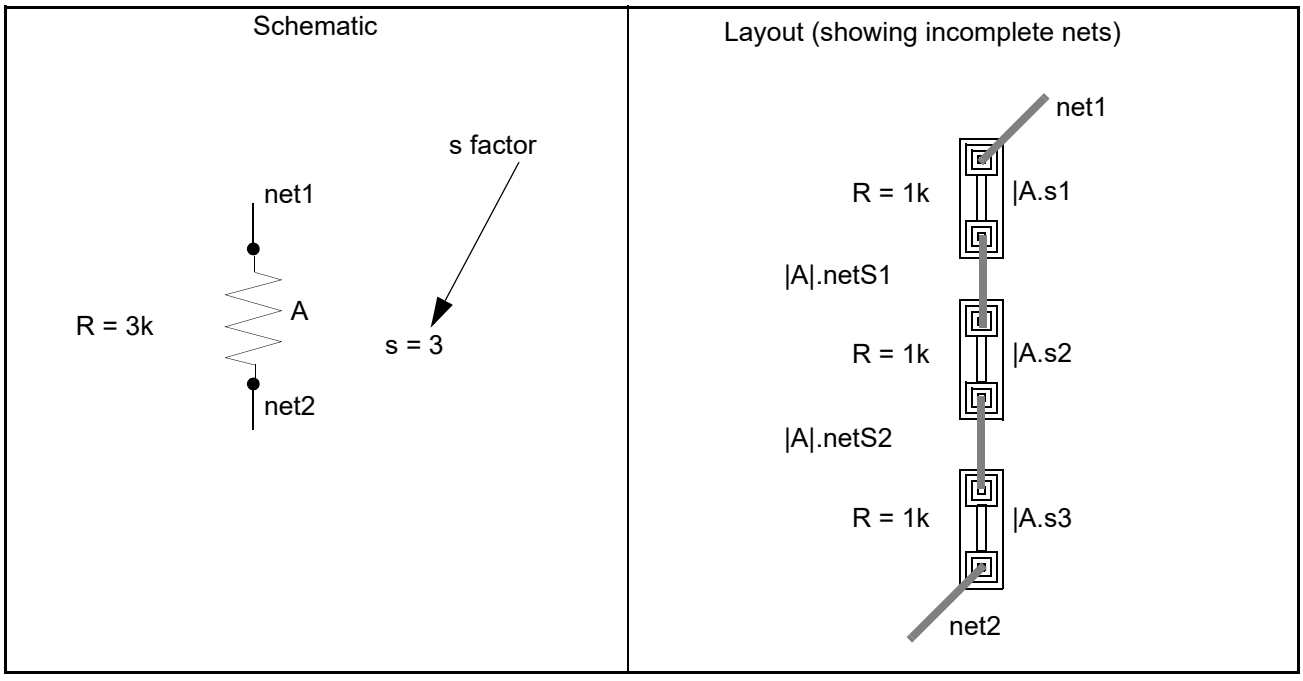
Description

Defines a one-to-many relationship between a device in a schematic and multiple instances in a series connection in the layout.

If both `mfactor` and `sfactor` are defined on the same instance, `sfactor` is ignored.

The `sfactor` property can be applied to two or three terminal resistors, capacitors, and inductors. You set the value of the `sfactor` property (called `s` or `S`, unless you change it on the General of the Connectivity form) to the number of layout devices you want to generate. When you run the *Generate All From Source* or *Update Components And Nets* command, Layout XL generates the number of layout devices specified by the property value. It creates all the devices identical in size.

The `sfactor` must set at the layout stop level. Any property above or below that level is ignored.



You can use the capacitanceParamNames, inductanceParamNames, resistanceParamNames, and transistorWidthParamNames environment variables to tell Layout XL which parameters of the device are to be split.

© 2023 Cadence Design Systems, Inc.

The lxSeriesTerms Property property must be defined on the instance in the schematic in order for Layout XL to understand the connectivity of devices with sfactor >1 and instance terminals>2.

Symbol master Add Property form	
Name	`sfactor`
Type	*int* | *float* | *string* | *ILExpr*
Value	*`user_defined_value`*

Add CDF Parameter form	
Add After Parameter	`As First Parameter`
paramType	*string* | *int* | *float*
name	`sfactor`
defValue	*`user_defined_value`*

Naming Conventions

Devices specified using the `sfactor` property in the schematic are named *`instName`*`.s`*`integer`* in the layout.

For example, if you have a schematic instance named `R0` with

```
sfactor = 3
```

the layout instances generated are named as follows.

Schematic	Layout

`R0`	`|R0.s1`
	`|R0.s2`
	`|R0.s3`

© 2023 Cadence Design Systems, Inc.

Or, if the prefixLayoutInstNamesWithPipe environment variable is set to `nil`.

Schematic	Layout

`P0`	`R0.s1`
	`R0.s2`
	`R0.s3`

Nets associated with series-connected devices are named *`instName`*`.s`*`integer`*`.expandedNet`*`integer`* in the layout.

For example,

```
R0.s1.expandedNet1
```

where "`expandedNet`" signifies that the layout net has no real match in the schematic.

Related Topics

Layout Generation

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

vxlInstSpacingDir Property

```
vxlInstSpacingDir list (left right top bottom)
```

Description

Assigns automatic spacing properties to pins of instances.

If these pins are on different nets or the pins cannot abut for any reason (for example, they are not assigned abutment properties), the software automatically separates the instances by the distance and in the direction you specify.

If one of the devices is in any type of group (including a synchronous clone), the other device must be in the same group for automatic spacing to occur.

Symbol master Add Property form	
Name	`vxlInstSpacingDir`
Type	*ILList*
Value	`left right top bottom`

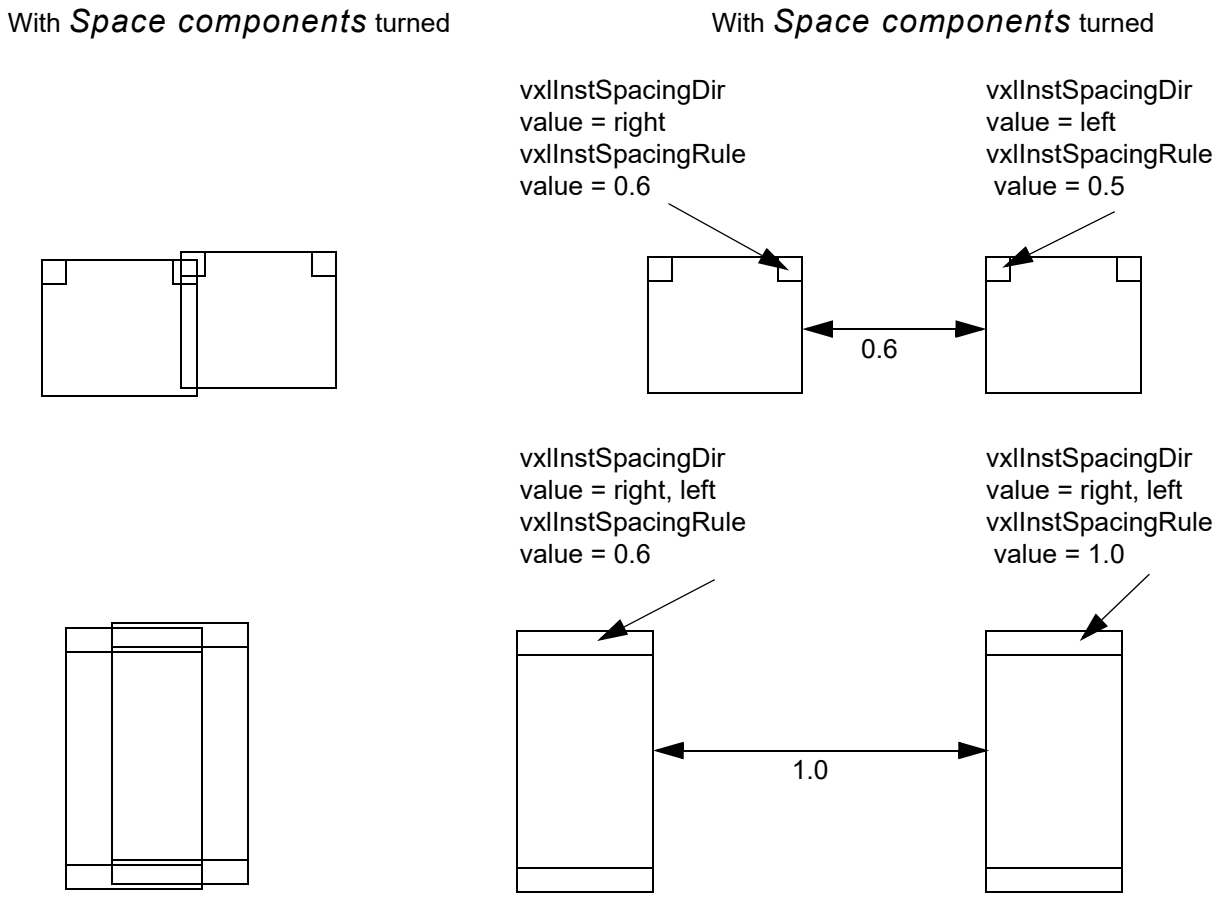
Add CDF Parameter form	
Add After Parameter	*As First Parameter*
paramType	*list*
name	`vxlInstSpacingDir`
defValue	`left right top bottom`

If you want automatic spacing to apply to Cadence® SKILL language or technology file parameterized cells (Pcells), you can enter these properties in SKILL to add the spacing properties to the pins.

If you want automatic spacing to apply to standard cells, you can enter properties on the pins.

© 2023 Cadence Design Systems, Inc.

If the vxlInstSpacingRule Property value is different for two pins affected by *Space components*, the larger of the two values is used; if the value is defined for only one of two pins affected by *Space components*, the defined value is used.



➤ To activate automatic spacing, turn on the *Space components* option in the Generation form.

Use the Layout XL user interface to enter properties on the pins in the same way that you enter the `permuteRule` properties.

To add the access direction to the pin:

```
property             value type       valid values
-----------------    ----------       ----------------------------
vxlInstSpacingDir    SKILL list       one or more of the following
                                      strings in the list:
                                      "left" "right" "top" "bottom"
```


© 2023 Cadence Design Systems, Inc.

Related Topics

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

vxlInstSpacingRule Property

`vxlInstSpacingRule float f_value`

Description

Defines a spacing offset when pins are on a different net.

Symbol master Add Property form	
Name	`vxlInstSpacingRule`
Type	*float*
Value	*`f_value`*

Add CDF Parameter form	
Add After Parameter	*As First Parameter*
paramType	*float*
name	`vxlInstSpacingRule`
defValue	*`f_value`*

A user-defined function, called `abutFunction`, can be used to define a spacing offset when pins are on a different net. When you set the `vxlInstSpacingRule` to float, two pins of the same abutment class and different nets are pushed apart a distance equal to the maximum values of the two `vxlInstSpacingRule` pins. To control the abutment spacing set the `vxlInstSpacingRule` to `abutFunction`.

`abutFunction` is not the name of your custom abutment function, but is the property that is set to the name of your abutment function.

```
dbReplaceProp(sObj~>dbId "abutFunction" "string" "myCustomAbutmentFunction")
dbReplaceProp(sObj~>dbId "vxlInstSpacingRule" "string" "abutFunction")
```

You can write your own spacing function for the two instances involved. For example, `myCustomSpacingFunc ( )` is to be called in the body of the user defined abutment function (`abutFunction`). The `myCustomSpacingFunc ( )` must return a float value to be effective.

© 2023 Cadence Design Systems, Inc.

There are two numbers passed to the abutment function, the connection condition (1, 2, or 3) and the abutment event (1, 2, 3, or 4). The connection condition (3) is for different nets. The sequence for nonabuttable pins should be the same as for abuttable. The calling of the abutment function for adjusting the Pcells (*Event* 2) and then one to calculate the spacing of the resultant reference edges (*Event* 4). This allows you to adjust the well or isolation on the Pcell and then space the devices based on the reference edges.

To get the spacing offset when pins are on different nets add the following case in your custom abutment function.

```
(4
   if(instA~>w > 3.0 && instB~>w > 3.0
   then
      result = 3
   else
      result = 1
   )
);; end of case 4
```

You can use the layout editor user interface to enter properties on the pins in the same way that you enter the `permuteRule` properties.

If you need to add a variable spacing rule to the pin, set the `vxlInstSpacingRule` to `abutFunction` as shown below:

```
dbReplaceProp(sObj~>dbId "vxlInstSpacingRule" "string" "abutFunction")
```

The properties to set automatic spacing are listed below.

Related Topics

Setting Layout XL Properties

Layout XL Properties

© 2023 Cadence Design Systems, Inc.

© 2023 Cadence Design Systems, Inc.

D

Layout XL Forms

This topic lists the Virtuoso® Layout Suite XL layout editor (Layout XL) forms.

Add Area Estimators Form

Add Soft Pin Form

Application Readiness Checker Form

Assign Layout Instance Terminals Form

Assign Nets Form

Annotation Browser Options Form

Back Annotate Dummies Form

Batch Checker Form

Block Parameters using Physical View Form

Check Against Source Form

Check XL Compliance Form

Clone Array Settings Form

Composite Waveguide Editor Form (Virtuoso Photonics Option)

Connectivity Form

Convert Libraries to Use Physical Configuration Views Form

Create Cluster Boundary Form

Create Group Array Form

Create Feed Through Terminal Block Pins Form

© 2023 Cadence Design Systems, Inc.

Create Path Form

Create Physical Configuration View Form

Custom Show Filter Form

Define Device Correspondence Form

Design Summary Form

Edit Net Attributes Form

Edit Soft Blocks Form

Extract Layout Form

Extractor Diagnostics Options Form

Generate Chained Devices Form

Generate Clones Form

Generate Folded Devices Form

Generate Layout Form

Generate Selected Components Form

Generate Selected Components - Pin Options Form

Incomplete Net Filter Form

Instance/Pin List Form

Load Traced Nets View Form

Master Variant Form

Match And Fix Form

Modify Correspondence Form

Move Cells Form

Net Name Display Options Form

Net Tracer Custom Setup Form

© 2023 Cadence Design Systems, Inc.

Net Tracer Display Options Form

Net Tracer Neighbors Form

Net Tracer Options Form

Open Physical Configuration View Form

Permutation Information Form

Save Short Locator View Form

Save Traced Nets View Form

Selection Options Form

Set Net Order Form

Set Pin Label Text Style Form

Short Locator Form

Smart Display Options Form

Snap Pins Form

Soft Block Global Options Form

Startup Option Form

Swap Components Form

Trace Manager Form

Update Binding Form

Update Clone Families Form

Update Components and Nets Form

Update Connectivity Reference Form

Update Layout Device List Form

View By Form

XL Probe Form

© 2023 Cadence Design Systems, Inc.

XL Probe Form

Related Topics

GUI Options and Corresponding Environment Variables

© 2023 Cadence Design Systems, Inc.

Add Area Estimators Form

Use the Add Area Estimators form to register an area estimator function for estimating the area enclosed by a PR boundary or a virtual hierarchy block.

Field	Description
Function name	Specifies the name of the SKILL function to be used for estimating the area of the PR boundary or a virtual hierarchy block.
Nickname	Specifies a custom name for the SKILL function. After the function is registered, the name specified in this field is used to list the SKILL function in the area estimator drop-down list.
Type	Specifies whether the area estimator function being registered is used to estimate the area enclosed by a PR boundary or a virtual hierarchy block. Choose a value from the drop-down list: ■ *PR boundary* ■ *Area boundary*
Parameters	Specifies a SKILL list of name-value pairs, where the name specifies the parameter name and the value is the parameter value to be used when estimating the area.
Run mode	Specifies whether the area estimation function is applied immediately or after all the related information is available. ■ *deferred*: Applies the area estimation function in delayed mode after all the other related information is available. This ensures the estimated area is more accurate as the area estimation calculation considers the related area values to derive the final value.

Related Topics

Generate Layout Form (PR Boundary)

Adjust Area Boundary Form

Adding an Area Estimator Function for Layout Generation

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Add A New Pin Form

Use the Add A New Pin form to create a new pin in the layout view.

Field	Description
Enter a terminal name	Specifies the name of the terminal to which the new pin is to be connected.

Related Topics

Generating All Components from Source

I/O Pin Generation

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Add Soft Pin Form

Use the Add Soft Pin form to create additional soft block pins to improve the connectivity of a design.

Field	***Description***
Soft block list	Lists all the soft blocks that were selected when you opened the form. You can create soft pins for any of the blocks that are listed in this field. The *Lib*, *Cell*, and *View* fields confirm the currently selected block.
Use	Allows you to select either *All Pins* or *Pins in Selected Set*.
Name	Specifies a name for the pin you want to create.
Filter	Lets you filter the terminal names that are displayed by typing in the text box and filtering either *All Pins* or *Pins in Selected Set*. The terminal list is updated to display only selected set of pins.
Terminal name	Specifies the name of the terminal on which you want to create a soft pin. The pins are created as strong pins by default and are listed in the table under the *Terminal name* list. Choose the terminal you want from the drop-down list, which by default, contains all the terminals in the selected block.
Name	Specifies a name for the pin you want to create.
Layer	Specifies the layer purpose on which the pin is to be created. Choose the layer purpose you want from the list of extractable layers in the design.
Width and *Height*	Specifies the width and height of the new pin. These fields are prepopulated with the `minWidth` spacing defined in the technology file for the selected pin layer.
Side	Specifies the sides of the PR boundary on which the pins can be created. If you specify more than one side, the Pin Optimizer can place the pins on any of the specified sides.
Create Strong Pin	Lets you specify the parameters using the controls to create a pin for the selected terminal.
Update	Lets you update currently selected pins with the new parameter values.
Delete	Removes the selected pins from the list.

© 2023 Cadence Design Systems, Inc.

Related Topics

Soft Block Pins

Creating a Soft Pin

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Application Readiness Checker Form

Use the Application Readiness Checker form to run the application readiness check and updates for the specified applications.

Application Readiness Checker improves the process of checking and updating the layout from the schematic and provides a one-to-one correspondence between the checks and updates. You can choose the checks you want for different applications. You can also extend the checks by registering SKILL-based check and update tests. The custom functions can return markers that can be annotated and displayed in Annotation Browser. The messages in the markers are included in the Application Readiness Checker report.

For LVS-complete designs, the tool automatically updates the schematic and layout correspondence to the best match before running the checks and updates.

Field	Description
Cellview	Name of the target library or layout cellview to check and update.
	Opens the Save Preset form that lets you save the Application Readiness Checker form data in a file at the specified path.
	Opens the Load Preset form that lets you load the file that has the Application Readiness Checker form data saved.
This section **lists the applications** to check and update. You can choose to only check or check and update.	
Application	Name of the application to check and update. *Schematic v layout* check is selected by default. Environment variable: lxcuApp
Check	Specifies whether to check the specified application. Supported applications: *Design Planner*, *Electrical Aware Design*, *Innovus*, *RF Solution*, *Schematic vs Layout*, *Voltage Dependent Rules*. The default in *Schematic vs Layout.*
Update	Specifies whether to update the specified application. Supported applications: *Design Planner*, *Electrical Aware Design*, *Innovus*, *RF Solution*, *Schematic vs Layout*, *Voltage Dependent Rules*. The default in *Schematic vs Layout.* Environment variable: lxcuUpdate

© 2023 Cadence Design Systems, Inc.

Field	Description
This section lists **the check and update options**.	
Binder	Specifies the type of binding to perform during update. ■ *Rebind* lets you reconsider the current binding and check if it is possible to get a better result by starting from scratch. ■ *Current* uses the existing binding. This is the default. ■ *LVS* rebinds using the binding information in the `svdb` directory. Environment variable: lxcuBinder
Layout State	Specifies the type of rebinding and layout update to perform. ■ *Partial* does not clear logical connectivity during rebinding. This is the default. ■ *Complete* performs hierarchical rebinding and clears all the logical connectivity. Unbound or ungenerated instances, terminals, or nets are not updated. LVS binding is allowed only when this option is specified. Environment variable: lxcuLayoutComplete
Cross Reference	Specifies the path to the instance cross-reference file (`.ixf`) that is created in the `svdb` directory of the LVS run directory. In PVS, you need to enable the `svdb` creation using *Create Quantus Input Data* option. If the LVS instance cross-reference file is specified, the binder uses the specified file to improve the bindings across the schematic and the layout. If any mismatched instances are encountered while parsing the `.ixf` file, the instances are ignored. Environment variable: lxcuSvdbPath
Netlist	Specifies the path to the LVS `.net` or `.spi` extracted netlist file from the layout that is created in the `svdb` directory of the LVS run directory. In PVS, you need to enable the `svdb` creation using *Create Quantus Input Data* option.

© 2023 Cadence Design Systems, Inc.

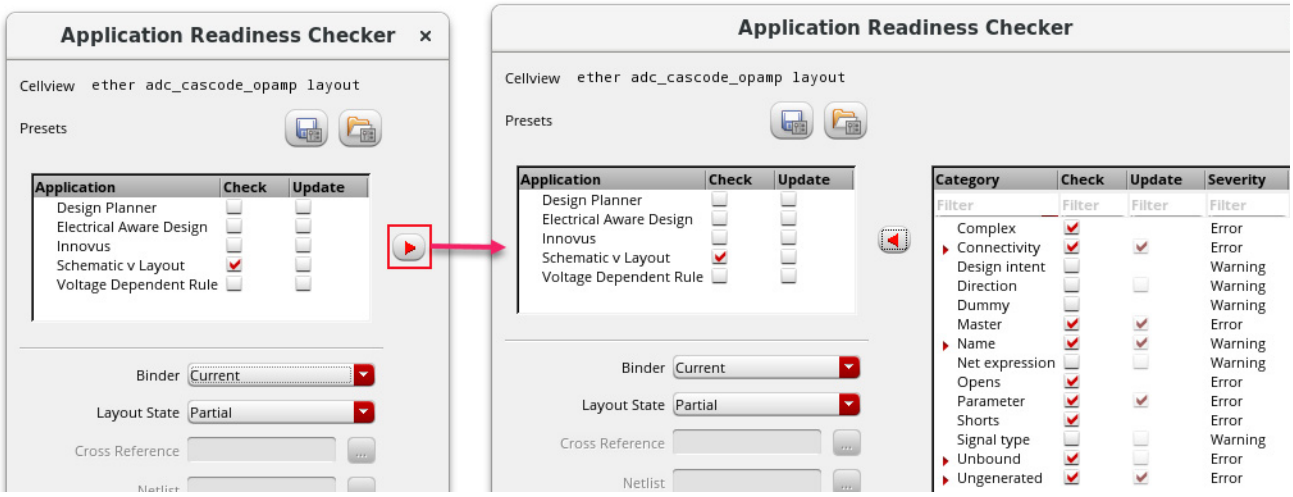
Field	Description
LVS Scale	Specifies the input scale. The rules file `input_scale` is required only if `input_scale` is different from the layout DBUPerUU. The `input_scale` value can be found in the `Gds_Map` statement of the `.gds.map` file in the `svdb` directory. This value is the key for device correspondence. Environment variable: lxcuSvdbScale
Extract	Specifies whether the layout needs to be extracted. This option is deselected by default. Environment variable: lxcuExtract
This section lets you **specify the scope** of the check and update operation.	
Scope	Specifies the scope of the check and update operation. ■ *Current cellview* runs the checks and performs an update on the current cellview. This is the default. ■ *Selected set* runs the checks and performs an update on the objects in the selected set. ■ *All cells in design* runs the checks and performs an update on all cells in the specified cellview. Environment variable: lxcuAllCellsInDesign ■ *All cells in library* runs the checks and performs an update on all cells in the specified library. Environment variable: lxcuAllCellsInLibrary
Cells without schematic	Runs the checks and performs an updates on layout cells without a schematic connectivity reference. This option is deselected by default. Environment variable: lxcuWithoutSchematic
Exclude Cells	List of cells to be excluded when the *All cells in design* or *All cells in library* option is specified. Environment variable: lxcuExcludeCells
Include Cells	List of cells to be included when the *All cells in design* or *All cells in library* option is specified. Environment variable: lxcuIncludeCells

© 2023 Cadence Design Systems, Inc.

Field	Description
Leaf Cells	List of cells to be treated as leaf cells. Only the terminals of leaf cells are checked when the *All cells in design* or *All cells in library* option is specified. Environment variable: lxcuLeafCells
Filter Messages	List of messages to be filtered from the check and update report. Environment variable: lxcuFilterMessage
This section provides reporting options.	
Open CAS workspace	Specifies whether to open the CAS workspace. This option is selected by default. See Workspaces in Layout XL. Environment variable: lxcuOpenWorkspace
Create HTML report	Specifies whether to create an HTML report. This option is selected by default. Environment variable: lxcuCreateHTMLReport
Open HTML report	Specifies whether to open the HTML report in the default browser or to issue the messages in the CIW. This option is deselected by default. Environment variable: lxcuOpenHTMLReport
Log file	Specifies whether to save the report to a text log file. This option is deselected by default. Environment variable: lxcuLogFile
Log File Directory	Specifies a directory for the log file. Environment variable: lxcuLogFileDirectory

© 2023 Cadence Design Systems, Inc.

Field	Description
	Expands the Application Readiness Checker form to display the **Check and Update pane**.



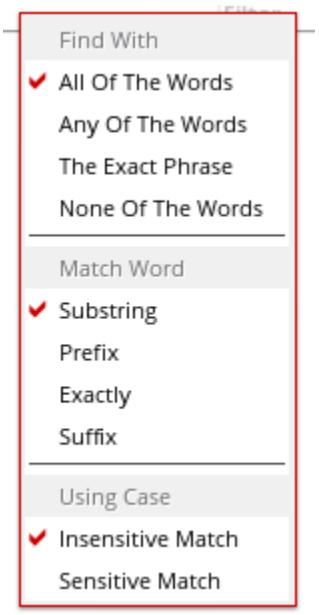
Filter

Specifies the criteria to filter the information for *Category*, *Check*, *Update*, and *Severity* columns.

You can filter by typing in the text box of the respective column. You can use the right click shortcut menu to specify the criteria for filtering. For example, you can choose to filter *All Of The Words*, *Any Of The Words*, *The Exact Phrase*, or *None Of The Words*. The following image shows all the options available to specify the criteria for filtering.

© 2023 Cadence Design Systems, Inc.

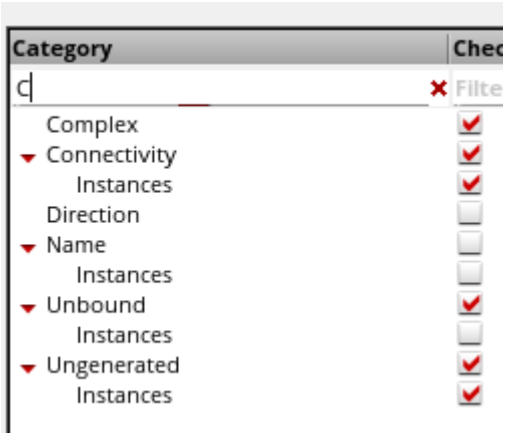
Field	Description



This option retains the history of your settings, which means that the filtering criteria you specify is retained next time you open the Application Readiness Checker form.

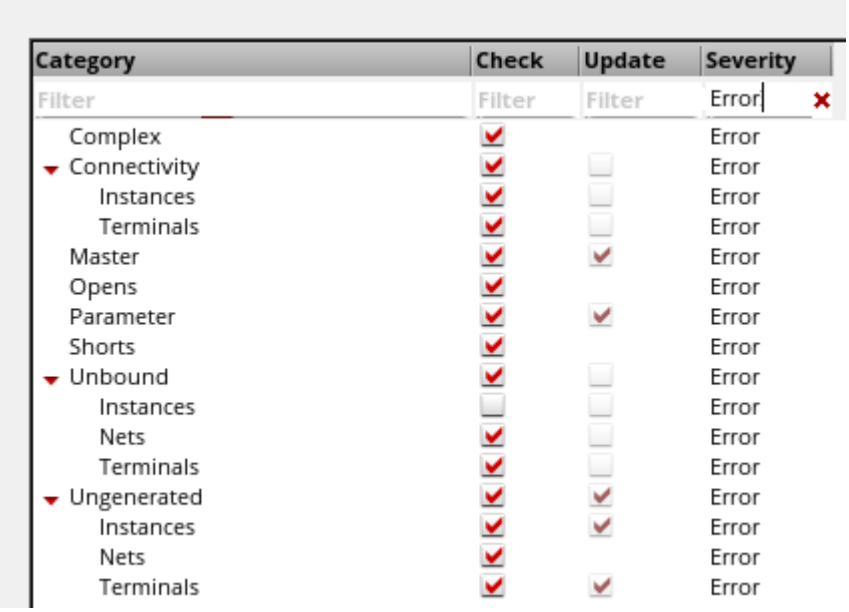
Examples

- Filter by *Category*:



© 2023 Cadence Design Systems, Inc.

Field	Description
	■ Filter by *Severity* type *Error*.



Category

Specifies the focus areas of the design that need to be checked and updated for the selected application.

Right-click in this column for the following options:

- *Expand All* to display all the categories.
- *Collapse All* to collapse all the categories.

The following checks can be run when *Schematic v Layout* is selected:

- *Complex* checks if multiple-to-one or multiple-to-multiple devices bound together in the layout are corresponding to the multiple devices in the schematic. Example of such a device is mosaic, which is considered a single device.

  If the check returns true or confirms that the layout to schematic binding for such devices exists, it means that the devices are not mapped in a one-to-one correspondence.
- *Connectivity* checks for differences in logical connectivity between the schematic and layout views.

© 2023 Cadence Design Systems, Inc.

Field	Description
	■ *Design Intent* checks if the design intent is up-to-date. ■ *Direction* checks if the terminals in the schematic and layout have the same direction (input, output, or inout). ■ *Dummy* identifies the devices as a dummy (shorted drain source and gate to the same net). ■ *Master* checks if the physical binding master specified in CPH is being used in the layout view. ■ *Name* checks for name differences between the schematic and layout views. ■ *Net Expression* checks for different or missing net expressions between the schematic and layout views. ■ *Opens* checks for the incomplete nets in the layout, which are visible on the Connectivity tab of the Annotation Browser assistant. ■ *Parameter* checks for the parameter differences between the schematic and layout devices. ■ *Shorts* checks for shorts in the layout, which are visible on the Connectivity tab of Annotation Browser. ■ *Signal Type* checks for different signal types, such as signal, analog, power, ground, tieHigh, and tieLow, between the schematic and layout nets. ■ *Unbound* checks for the extra instances or terminals in the layout that do not have correspondence in the schematic. ■ *Ungenerated* checks for the missing instances or terminals in the layout that exist in the schematic.
Check	Specifies whether to run the selected check on the application. Right-click in this column for the following options: ■ *Expand All* to display all the categories. ■ *Collapse All* to collapse all the categories. ■ *Set All On* to select all the check and update options available. ■ *Set All Off* to clear all the check and update options.

© 2023 Cadence Design Systems, Inc.

Field	Description
Update	Specifies whether to update the selected application. Right-click in this column for the following options: ■ *Expand All* to display all the categories. ■ *Collapse All* to collapse all the categories.
Severity	Specifies the severity level of any messages generated by the check. Right-click in this column for the following options: ■ *Expand All* displays all the categories. ■ *Collapse All* collapses all the categories. ■ *Set All as Error* sets severity level of all checks to error. ■ *Set All as Warning* sets severity level of all checks to warning. ■ *Set All as Info* sets severity level of all checks to Info.

Related Topics

lxCheckAndUpdate

lxCheckAndUpdateRegister

lxCheckAndUpdateRemove

lxCheckAndUpdateSet

lxcuDetailedReturn

© 2023 Cadence Design Systems, Inc.

Assign Layout Instance Terminals Form

Use the Assign layout instance terminals form to manually assign layout nets to unbound instance terminals in selected layout instances. This form is also useful when the *Define Device Correspondence* command is unable to assign all the instance terminals automatically. For example, if the instance masters in a one-to-one correspondence do not match, or if you create a one-to-many, many-to-one, or many-to-many correspondence.

Field	Description
Select layout instance terminal to assign	Lists the *Instance*, *Terminal*, and the *Net* name to which the terminal is currently connected. The instances are listed in non case-sensitive order, first ordered by instance name and then by the terminal name.
Select layout net	Lists global nets and the nets that connect to the instances being bound. You can select one or more instance terminals and assign them to a layout net. The nets are listed in non case-sensitive order.
Assign instance terminal	Lets you choose the net you want to assign to the selected instance terminals and make the assignment.

Related Topics

Device Correspondence Updates

Assigning Layout Instance Terminals to Nets

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Assign Nets Form

Use the Assign Nets form to place an instance that does not have a connectivity source. The software does not connect it to any net unless you specifically assign the instance pins to a net.

Field	Description
Create	Adds a new name to the list of net names. Type the net name you want to add in the entry field.
Delete	Deletes a net name from the list of net names. Type the net name to be deleted in the entry field or select it in the list.
Search box	Highlights in the list box the net name you type in the field.
Net name	Lets you specify the net name. Press `F3` to display a list of all the nets in the layout.

Related Topics

Assigning Instance Pins to a Net

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Annotation Browser Options Form

Use the **Annotation Browser Options** form to specify the look and behavior of the Annotation Browser. To open the form, right-click in the browser pane and choose *Options*.

Field	Description
Expand Control	This section lets you control the expansion limit of nodes.
Suppress expansion at	Specifies in 1000s the number of violations that prevents a node from being expanded. If the number of entries under a particular node exceeds the limit, the node is not expanded by default. Environment variable: suppressAbutFailMsgs If you select markers in the canvas and their parent category is not expandable in the browser pane, you can still use the *Go To The Next Selected Marker* and *Go To The Previous Selected Marker* buttons in the Annotation Browser toolbar to cycle through the selected markers. Additionally, you can override the global limit for an individual node by positioning the pointer over the node, right-clicking, and choosing *Unsuppress Expansion* from the shortcut menu.
Hierarchy Control	This section lets you control the depth level of hierarchy searched.
Hierarchy depth	Specifies the number of levels of hierarchy searched for markers. The default is 3. Environment variable: hierarchyDepth
Cross selection	Specifies whether cross selection is allowed between Annotation Browser and Navigator or Canvas. The default is selected, which means that the cross selection is allowed. Environment variable: crossSelect
Default Open	Specifies the default open behavior when you descend into a cellview.
Mode	Specifies whether the cellview is opened for *edit* or *read*. Environment variable: openMode
Open in	Specifies whether the cellview is opened in a *new tab*, the *current tab*, or a *new window*. Environment variable: openLocation

© 2023 Cadence Design Systems, Inc.

Field	Description
Marker Highlighting	This section lets you specify the marker highlighting options.
Remove highlighting when browser is closed	Removes the highlighting in the design canvas when you close the Annotation Browser assistant. The default is off, which means the markers remain highlighted in the canvas when you close the browser. Environment variable: hideMarkersWhenBrowserHidden
Automatic highlighting	Controls the highlight state of newly created markers in the design canvas. For example, when you delete a part of a completed route in the canvas, an open marker is created automatically. This option controls whether the dragline representing that marker is automatically displayed or not. ■ **Highlight new marker categories** specifies that draglines are displayed automatically in the canvas to represent newly created marker categories. ■ **Highlight new markers** specifies that draglines are displayed automatically in the canvas to represent newly created individual markers. ■ **Inherit from parent category** specifies that a newly created marker inherits the highlight state from its parent category. If the category is only partially highlighted, the dragline representing the new marker is not automatically highlighted. Environment variable: automaticHighlight

Related Topics

Annotation Browser Assistant

Custom Show Filter Form

Incomplete Net Filter Form

View By Form

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Back Annotate Dummies Form

Use the *Back Annotate Dummies* form to customize the placement of dummy instances in the schematic.

For information on updating the master cellview, see Select Master Form.

Field	Description
Generate	Specifies how instances are backannotated to the schematic. ■ *individually*: Each instance is backannotated individually. This is the default. ■ *mfactored*: Instances are backannotated as mfactored devices in the schematic. ■ *iterated*: Instances are backannotated the same way as for mfactored, but some iterated notation is used. For example, `(<0:2>)`. Specify this value when the instance you want to backannotate is not a Pcell and when no mfactored parameter is defined. Environment Variable: backAnnotateInstances
Row, Columns	Specifies the number of rows and columns of dummy instances. These values are adjusted automatically.
X, Y Spacing	Specifies the horizontal spacing. These values are adjusted automatically.
	Updates the layout of instances in a grid to be as close to a square aspect ratio as possible
 	Switches between column-priority packing and row-priority packing for the grid of instances being placed down. Environment Variable: backAnnotatePackingMethod
The following columns in the table provide information about all dummies that are not yet backannotated to the schematic. You can select and deselect instances in this table and edit the library, cell, and view information.	
Name	Name of the instance.
Schematic Lib	Name of the schematic library.
Schematic Cell	Name of the schematic cell.
Schematic View	Name of the schematic view.

© 2023 Cadence Design Systems, Inc.

Field	Description
Type	Specifies the figure type. Valid values are instances or pin.
Update Master	Opens the Select Master form that lets you specify the master Library, Cell, and View of the back annotated schematic device. See Select Master Form.
	Rotates the instance through 90 degrees counterclockwise.
	Rotates the instance through 90 degrees clockwise.
	Mirrors the instance about its y axis.
	Mirrors the instance about its x axis.

Select Master Form

The following table describes the fields available on the Select Master form..

Field	Description
Library	Specifies the library for the master cellview to be used.
Cell	Specifies the cell name for the master cellview to be used.
View	Specifies the view for the master cellview to be used.

Related Topics

Dummy Instances Backannotation

Layout Generation

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Batch Checker Form

The form contains the following tabs.

Tab	Description
Routability	Identifies the problems with your data before the router is run.
Connectivity	Checks for the current cellview, a selected set of nets, exclude selected nets, loops and dangling wires, and exclude power and ground nets.
Speciality Routing	Runs special routing checks on a design.
Placement (Advanced Node Layout EAD Only)	Checks whether the placement is DRC-correct.
Fabrication	Identifies the process antenna violations for the entire design, selected nets, or except selected nets.
WSP | SP Active	Checks shapes for width, color, and wireType conformance as compared to the active width spacing pattern (WSP).

Routability

The following table describes the fields available on the *Routability* tab of the Batch Checker form.

Field	Description
Scope	This section allows you to specify the options to control the scope and region of Pin to Trunk routing.
Current Editable Cellview	Checks shapes in the entire design for shorts and opens.
Selected Nets	Checks only the currently selected nets for shorts and opens.
Exclude Selected Nets	Checks all nets except the currently selected nets for shorts and opens.
Pin Checks	This section lets you specify the options for checks to performed on pins in the design.
Pin Accessibility	Checks for the accessibility of pins.
Pin Spacing	Checks for pin to pin spacing less than minimum pin spacing.

© 2023 Cadence Design Systems, Inc.

Field	Description
Pin Width	Checks for pin dimension less than minimum pin width.
Pin in Preferred Direction	Checks whether the pins are in the preferred routing direction specified for the pin layer. It identifies the pins that can cause the router to use wrong-way metal. This option is deselected by default.
Pin outside prBoundary	Checks for the pins that are outside the instance's place and route boundary. This option is selected by default.
Pin not on prBoundary Edge	Checks whether there are any pins on the edge of the place and route boundary. This option is deselected by default. Environment Variable: checkRoutabilityNotOnPRBoundaryEdge
On Grid	Checks for pins that are off manufacturing grid, track pattern, or routing grid. This check is based on the option that is selected in the *Snap Wires to* field in the Wire Assistant and the *Wires* field in the *Snapping* group box of the Layout Editor options form. The *Snap Wires to* field in the Wire Assistant and the *Wires* field in the *Snapping* group box of the Layout Editor options form are synchronized.
Other Checks	This section lets you specify the options to check for instances with missing prBoundary and cover obstructions.
Missing prBoundary	Checks for instances with missing place and route boundary. This option is selected by default. Environment variable: checkRoutabilityMissingPRBoundary
Missing Cover Obstruction	Checks for instances with missing blockage objects and cover obstructions. This option is deselected by default. Environment variable: checkRoutabilityMissingCoverObstruction
Filter Options	Provides options to filter the type of nets.
Exclude Types	Excludes selected nets from processing.
Power and Ground	Excludes power and ground nets from processing.
Clock	Excludes clock nets from processing.

© 2023 Cadence Design Systems, Inc.

Field	Description
Marker Limit per Type	Sets the number of markers for each violation type.

Process Rules

The following table describes the fields available on the *Process Rules* tab of the Batch Checker form.

Field	Description
DRD Options	Opens the *Batch* tab of the DRD Options form in which you can select the constraints for batch checking.

Connectivity

The following table describes the fields available on the *Connectivity* tab of the Batch Checker form**.**

Field	Description
Scope	The section allows you to specify the options to control the scope and region of nets.
Current Editable Cellview	Checks shapes in the entire design for shorts and opens.
Selected Nets	Checks only the currently selected nets for shorts and opens.
Exclude Selected Nets	Checks all nets except the currently selected nets for shorts and opens.
Check For	This section specifies the type of check to be performed.
Loops and Dangles	Checks for dangling wires and wires with loops.
Filter Options	Lets you specify the options for checks to performed on nets in the design.
Exclude Power and Ground Nets	When this option is on, power and ground nets are not checked for connectivity errors.

© 2023 Cadence Design Systems, Inc.

Speciality Routing

The batch checker only checks specialty routing constraints that are created with pathSegs and vias that are in routes.

The following table describes the fields available on the *Speciality Routing* tab of the Batch Checker form.

Field	Description
Symmetry	Checks whether a symmetry axis is defined and if so, whether its position is correctly defined. As long as one pair of symmetric pins can be identified, the axis is considered well-defined, otherwise a marker will be generated at the location of the axis with a message about the incorrectly assigned location. Only shapes on metal and poly layers are being checked and only the routed portions of the symmetric nets are checked. Geometric shapes (e.g. paths, pathSegs/vias not in a route) are not checked. A warning is issued when the checker detects geometric shapes.
Diff Pair	Checks the percentage of unpaired length for each net in the diff pair.

Placement (Advanced Node Layout EAD Only)

The following table describes the fields available on the *Placement* tab of the Batch Checker form.

Field	Description
Check For	This section lets you verify constraints and width spacing pattern (WSP)-specific settings in the design.
Constraints	Checks all elements in the design for conformance with the Alignment, Orientation, Symmetry, and Distance constraints, if set. Environment variable: constraints
Row/Grid	Checks whether instances are snapped to the correct rows or grids. Environment variable: rowOrGrid

© 2023 Cadence Design Systems, Inc.

Field	Description
Spacing	Checks instances for conformance with applicable spacing rules. Environment variable: spacing
Minimum Pin Access	Checks WSP tracks for conformance with the *Minimum Pin Access Count* specified on the *Misc* tab of the Placement Options form. Environment variable: minPinAccess
Color Aware	Checks shapes for conformance with color locks assigned to them, if any. Environment variable:colorAware
Context Aware	Verifies whether context-aware placement has been done for standard cells and whether the edge type and spacing constraints of tap cells have been honored during placement.
Reports	This section lets you specify the ways in which violations are to be reported.
Report Violations	Displays detailed error and warning messages in the CIW reporting all violations that were detected while running batch checker. Environment variable: reportViolations
Create Markers	Displays markers in the design to indicate the locations where violations are reported. Environment variable: createMarkers
Scope	This section lets you specify the scope for running the placement checks.
Selected Only	Runs batch checker on the selected instances only. Environment variable: selOnly

© 2023 Cadence Design Systems, Inc.

Fabrication

The following table describes the fields available on the *Fabrication* tab of the Batch Checker form.

Field	Description
Scope	The *Scope* group box allows you to specify the options to control the scope and region of fabrication nets.
Current Editable Cellview	Checks shapes in the entire design for process rule violations.
Selected Nets	Checks only the currently selected nets for shorts and opens.
Exclude Selected Nets	Checks all nets except the currently selected nets for shorts and opens.
Antenna Oxide Models	In this group box, you can specify which oxide models to use: *First, Second, Third*, or *Fourth*. By default, the *First* oxide model is selected.

WSP | SP Active

The following table describes the fields available on the *WSP | SP Active* tab of the Batch Checker form.

Field	Description
Scope	The *Scope* group box allows you to specify the options to control the scope and region of WSP shapes.
Current Editable Cellview	Checks the shapes in the entire design for WSP conformance.
Area	Specifies the WSP conformance check region by using one of the following methods: ■ Type the lower-left (*xLo*, *yLo*) and the upper-right (*xHi*, *yHi*) coordinates for the region. ■ Click *Select* to draw the region in the workspace by dragging a rectangle from lower-left coordinates to the upper-right coordinates.

© 2023 Cadence Design Systems, Inc.

Field	Description
Filter Options	Lets you specify the options for checks to performed on cellviews in the design.
Hierarchy Depth	Sets the hierarchy depth up to which the cellview will be checked. Environment variable: depth
Marker Limit	Sets the maximum number of violation markers. Environment variable: vioLimit
Layer	Specifies the layer for the shapes to be checked. Choose *All* or a layer from the drop-down list. Environment variable: selectLayer
Purposes	Specifies the purposes for the shapes to be checked. Choose the purposes from the list of valid purposes. Environment variable: selectPurposes
Merge Same Layer Shapes	Merges same-layer connected shapes of the same color as well as uncolored before checking. If multiple purposes are selected, merging applies to the shapes on those purposes. Environment variable: mergeShapes
Exclude Blockages	Excludes blockages while running the batch checker.
Check For	Lets you specify the options for the check to be performed.
Color	Checks the shapes for conformance to the track color of the active WSP. Environment variable: color
Wire Type	Checks the shapes for conformance with the wireTypes of the active WSP. Environment variable: wireType
Width	Checks the shapes for conformance to the track widths of the active WSP (if set to `t`), or only the centerline alignment (if set to `nil`) with the active WSP. Environment variable: width

© 2023 Cadence Design Systems, Inc.

Related Topics

Batch Checker

DRD Options Form - Batch Tab

Checking Placement Using Batch Checker

© 2023 Cadence Design Systems, Inc.

Block Parameters using Physical View Form

Use the Block Parameters using Physical View form to load predefined soft block parameters from another cellview

Field	Description
Choose Source Template View	Specifies a source OpenAccess cellview containing predefined soft block parameters which you want to copy into the *Attributes* pane of the CPH menu bar.
Library, Cell and *View Name*	Contains the information of the source cellview to be imported. When you click *OK*, the *Attributes* pane is updated with the settings from the specified cellview.

Related Topics

Loading Soft Block Parameters from Another Cellview

CPH Menu Bar

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Check Against Source Form

Use the Check Against Source form to check the differences between the schematic and the layout. Use *Connectivity – Check – Against Source* (CAS) command to invoke the form.

Field	Description
Check	This section lets you specify the types of layout versus schematic checks you want to run.
Selected only	Runs the *Check Against Source* command only on a selected set of instances. Environment variable: casSelected
Unbound instances	Reports the layout instances that are not bound to the schematic instances and the schematic instances that are not yet generated in the layout. Environment variable: casDisplayUnboundInsts
Masters	Reports the instance masters that are missing or mismatched between the schematic and layout. Environment variable: casDisplayMastersMismatch
Connectivity	Reports connectivity issues on top-level pins and global nets, mismatched or missing terminals and instance terminals, and unbound nets in the layout. Environment variable: casDisplayConnMismatch
Parameters	Reports issues with CDF parameters and properties set on the schematic and the layout views. The parameters (and properties) to be compared are set on the *Parameters* tab of the Connectivity form. If the checkParamsOnly environment variable is set to `nil`, the *Parameters* field on the form updates to *Parameters and properties*, reporting both the parameter and property differences. Environment variable: casDisplayParamsMismatch
Names	Reports mismatches between the schematic and the layout instance and net names. Environment variable: casDisplayNameMismatch

© 2023 Cadence Design Systems, Inc.

Field	Description
Dummy devices	Reports dummy instances that are unbound, which means they have no correspondence to the schematic devices and have the `ignore` property set. These are dummies instances that are good candidates for backannotation. Environment variable: casDisplayDummyMismatch
Bus terminals	Reports mismatches between the implicit bus terminals that exist in the layout cellview and the explicit bus terminals that exist in the schematic cellview. The related bus terminal markers generated in the CAS tab of the Annotation Browser assistant can be interactively fixed using the *Update All Bus Terminals* shortcut menu. Environment variable:casDisplayBusTerminalsMismatch
Design Intent	Checks if Design Intent synchronization is needed during CAS run. By default, this option is deselected.
Design planner	Reports mismatches between the virtual hierarchy in the schematic and layout views. This option is available only when Virtuoso Layout Suite EXL or higher license is checked out.
Output	This section lets you specify how the report should be displayed.
Open workspace	Opens the CAS workspace at the end of a Check Against Source run. By default, this option is ON. Environment variable: casOpenWorkspace
Display info window	Prints the report in a separate Info window. By default, the schematic versus layout differences are reported in the form of markers generated in the CAS tab of the Annotation Browser assistant. Environment variable: casInfoWindow
Overwrite log file	Clears the log file of all the information from the previous CAS run and populates it with the information from the new run. If the option is deselected, the report from a new CAS run is appended to the specified log file. Environment variable: casAppendToLogFile

© 2023 Cadence Design Systems, Inc.

Field	Description
Log File	Specifies the name of the file to which all the *Check Against Source* messages are written. You can use the browse button to locate an existing file. Environment variable: casLogFileName
Glyphs	This section lets you specify the display associated with each generated glyph.
Display	Controls whether or not the markers generated during a *Check Against Source* run are visually indicated on the layout canvas using an appropriate glyph. Environment variable: casGlyphs
Info balloon	Controls the display of the information balloon associated with each generated glyph. Choose between displaying the information balloon on click or on hover.

Related Topics

Workspaces in Layout XL

Parameters

Design Check Against Source

virtual hierarchy

ignore Property

© 2023 Cadence Design Systems, Inc.

Check XL Compliance Form

Use the Check XL Compliance form to specify the types of XL Compliance checks to run and how the report should be displayed. You can use this form to evaluate the design for device correspondence with the schematic and report information about any ungenerated and unbound devices.

Field	Description
XL Compliance	This section lets you specify the scope of the check. You can choose to check a specific cellview or all the cells in the library.
All cells in design	Checks XL compliance for all cells in the specified cellview. Annotation Browser opens in *Current Cellview To Depth* view when *All cells in design* is enabled or in *Current Cellview Only* view when *All cells in design* is disabled.
Check devices	Checks the lower-level cellviews in the design for opens and shorts. Environment variable: checkShortsAndOpens
Check bindings	Performs all binding-related checks on all cells in the library and displays a summary table showing the cells that need to be updated. Environment variable: checkBindings
Extract layout	Extracts connectivity of the nets in the layout being checked for XL compliance issues. Environment variable: checkExtractLayout
Extract stop level	Overrides the effective extraction stop level for an individual instance or instance master. Environment variable: extractStopLevel
Generate check against source markers	Runs *Check Against Source* and generates CAS markers for all cells in the library. Selecting this option automatically enables the *Check Bindings* option. Environment variable: xlComplianceCreateCasMarkers

© 2023 Cadence Design Systems, Inc.

Field	Description
Output	This section lets you generate the output of the XL Compliance report.
HTML report	Produces the XL Compliance report in HTML format. Environment variable: xlComplianceHtml
Open HTML report	Opens the XL Compliance report automatically in your default web browser. This option is grayed out if the *HTML report* checkbox is deselected. Environment variable: xlComplianceHtmlOpen
Log file	Generates the output of the XL Compliance report as separate text log files. Environment variable: xlComplianceLogs
Log file directory	Specifies the directory in which the log files generated during an XL Compliance check are saved. Environment variable: xlComplianceLogDir
Options	This section lets you specify the checks available for cellviews and messages.
Exclude cells	Specifies the cellviews (in the format `lib cell view`) to be excluded from the following checks: ■ lxHierCheck ■ lxHierCheckAgainstSource ■ lxCheckLib ■ lxHierUpdateComponentsAndNets ■ lxUpdateBinding Supports asterisks a wildcard character. For example, "(`CHK INV sch\\*`)" specifies all schematic views of cell `INV` in library `CHK`. Environment variable: checkExcludeCellViews

© 2023 Cadence Design Systems, Inc.

Field	Description
Leaf cells	Checks the terminals of the specified cellviews treated as leaf cells when running an XL Compliance check in a hierarchical design. Environment variable: checkCellsAsLeaf
Filter messages	Applies filters on the XL Compliance report to remove messages related to any design components or elements that are unwanted or to remove specific message IDs. Specify the message IDs to be filtered as a space-separated string list. The asterisk is supported as a wildcard character. For example, "(`LX-2149 LX-2150 LX-22*`)". Environment variable: xlComplianceFilter

Related Topics

Design Check for Layout XL Compliance

lxHierCheck

lxHierCheckAgainstSource

lxCheckLib

lxHierUpdateComponentsAndNets

lxUpdateBinding

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Clone Array Settings Form

Use the Clone Array Settings form to specify the number of clones that will be generated per row or column, the spacing between the generated clones, and whether or not the generated clones will be flipped by row or column or both.

Field	Description
Number of Columns or Rows	Lets you control the number of clones you want to generate in each row or column. Environment variable: cloneArrayRowOrColValue
Spacing X	Lets you edit the spacing from the rightmost edge of one clone column to the leftmost edge of the next clone column for clone array generation. Environment variable: cloneArrayXSpacing
Spacing Y	Lets you edit the spacing from the topmost edge of one clone row to the bottommost edge of the next clone row for clone array generation. Environment variable: cloneArrayYSpacing
Flip First	Lets you flip the first row or column of the generated clone array. Environment variables:cloneArrayFlipFirstColumn, cloneArrayFlipFirstRow
Flip Next From Previous	Lets you flip each row or column of the clone array from the previous row or column. Environment variables: cloneArrayFlipNextFromPreviousColumn,cloneArrayFlipNextFromPreviousRow

Related Topics

Generate Clones Form

Clone Generation

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Composite Waveguide Editor Form (Virtuoso Photonics Option)

Use the Composite Waveguide Editor form to edit a composite waveguide. A composite waveguide is a general placeholder for waveguide paths between optical components and can be generated in the layout when using the Virtuoso Photonics Solution. Using the Composite Waveguide Editor, you can edit the different building blocks of a waveguide path, such as straight waveguide and bend waveguide.

You will need the `Virtuoso_Photonics_Option` license to use the functionality discussed in this section. For information on obtaining the required license, contact your local Cadence representative.

Field	Description
Elements	Lists the various waveguide instances that together comprise the composite waveguide. For each waveguide element, the corresponding *Library*, *Cell*, and *View* information is also listed. For the selected waveguide element, the adjoining preview pane displays an image of the composite waveguide, highlighting the part represented by the selected element.
Parameters	This section lets you list the parameters corresponding to the selected waveguide element. Only those parameters that correspond to the selected waveguide element are displayed. The *Orientation* parameter is applicable to all the supported waveguide elements. Therefore, the *Orientation* parameter is always displayed.
Orientation	Lists the supported orientations for generating the selected waveguide element.
waveguide layer	Specifies the valid layer on which the optical connection is supported.
Rotation	Specifies the angle at which the selected waveguide element can rotate.
Radius	Specifies the curvature of the selected waveguide element.
Length	Specifies the length of a straight waveguide element.
Width	Specifies the width of the selected waveguide element.

© 2023 Cadence Design Systems, Inc.

Field	Description
Total Angle (deg)	Specifies the length of an arc (in degrees) for the circle whose radius is specified by the *Radius* parameter. The *Total Angle (deg)* and *Total Angle (rad)* parameters are used for bend waveguides.
Total Angle (Rad)	Specifies the length of an arc (in radians) for the circle whose radius is specified by the *Radius* parameter. This parameter is automatically calculated using the *Total Angle (deg)* value. The *Total Angle (rad)* and *Total Angle (deg)* parameters are used for bend waveguide elements.
Options	This section lets you specify alignment and zoom options.
Align elements	Specifies that when a change such as the change in rotation, total angle, or radius, is made to the selected element, all the subsequent elements are automatically rotated to maintain their alignment. For example, out of the three elements present, if you rotate the middle waveguide element, the last waveguide element is also automatically rotated to stay in alignment. However, due to this re-alignment, the original alignment between the middle element and the first element is broken. If you add an element at the end of the current elements, the new element is also automatically rotated to correctly align.
Show updates on canvas	Disables the dynamic zooming in and panning that is performed on clicking *Apply*.
Apply	Zooms into and pans to display on the canvas the waveguide element selected for editing. The zoom and pan functionality is controlled by the dynamic zoom and pan functionality supported by the Display Options form. By default, the selected waveguide element is panned into and displayed on the canvas, keeping the zoom settings unchanged.

Related Topics

Generation of RF TLine Chains (Virtuoso MultiTech Framework)

© 2023 Cadence Design Systems, Inc.

Connectivity Form

Use the Connectivity form to set Layout XL options either for the current cellview or Layout XL session. The form is split into the following tabs.

Tab	Description
General	Defines the scope of the options settings you make in the form, and loads and saves the settings from and to an options file.
Display	Specifies how the Layout XL windows are arranged on your desktop, how draglines are displayed in the layout window, and to enable and disable cross-selection and net probing.
Connectivity	Sets the Layout XL connectivity extraction parameters for the current design.
Generation	Specifies how certain components are handled when they are generated in the layout view.
Parameters	Specifies which parameters are to be ignored by the generation and check command and how parameters are compared during *Check Against Source*.
Configure	Lets you view the presets that reflect the current state of the associated environment variables in the `.cdsenv` file, or the state of the associated form fields.

General

Use the *General* tab to define the scope of the options settings you make in the form and to load and save settings from and to an options file.

Field	Description
Scope	This section lets you specify whether the option settings apply for the current cellview only or globally for the current cellview *and* all the cellviews opened subsequently in Layout XL.
Global	Applies the settings to the current cellview and to all cellviews opened subsequently in Layout XL.
Cellview	Applies the settings only to the current cellview. Choosing this option grays out the *Connectivity*, *Generation*, and *Parameters* tabs, along with certain options on the *Display* and *General* tabs.

© 2023 Cadence Design Systems, Inc.

Field	Description
Load/Save	This section lets you load, save or delete values to or from a cellview, library, technology library, or options file.
Cellview	Updates the options for the cellview from which you opened the form.
Library	Updates the library that contains the cellview from which you opened the form.
Tech library	Updates the technology library of the cellview from which you opened the form.
File	Updates the `.cdsenv` file (the default) or another file specified in the text field.
Constraint-Aware Editing	This section provides options for constraint-aware editing.
Constraint-aware editing	Ensures that the *Copy*, *Move*, *Stretch*, and *Rotate* commands honor the following constraints: *Alignment*, *Fixed*, *Locked*, *Matched Parameters*, *Orientation*, *Matched Orientation*, and *Symmetry*. This is a system-wide setting that applies to all existing and new windows opened in the current Virtuoso session. See Constraint-Aware Editing. Environment variable: constraintAwareEditing

© 2023 Cadence Design Systems, Inc.

Field	Description
Connectivity Reference	This section specifies the connectivity reference for the schematic window.
Open with	Specifies whether the Virtuoso Schematic Editor is launched when you launch Layout XL or a higher tier and, if the schematic editor is started, which of its tiers opens. You can choose from: *Schematic L, Schematic XL, No Schematic*. *No Schematic* implies that the Virtuoso Schematic Editor is not launched. The default is *Schematic XL*. Environment variable: lxSchematicDefaultApp The schematic editor tier level does not change when you launch Layout XL from the Virtuoso Schematic Editor XL nor when you switch between layout editor tier levels after Layout XL or GXL has been launched. If you choose *No Schematic*, the *Open connectivity reference during Edit In Place/Descend* field is disabled.
Open connectivity reference during Edit In Place/Descend	Specifies whether the source cellview is opened when you use the *Edit In Place* or *Descend* commands. When switched off, the layout cellview remains open in Virtuoso Layout XL mode, but with no connectivity reference. The field is disabled if you choose *No Schematic* for the connectivity reference to *Open with*. If you choose to open the connectivity reference, you can choose to open the schematic cellview in a new window or a new tab. By default, the schematic cellview opens in a new window. Environment variables: openConnRef, openConnRefTab

© 2023 Cadence Design Systems, Inc.

Field	Description
Close schematic when exiting layout	Closes the schematic window associated with the layout window that is closed, if the layout was opened first in Virtuoso Layout XL or a higher and the schematic window was automatically opened. If the schematic was opened first and then a Virtuoso Layout XL or higher window was launched from the schematic, the schematic window is not automatically closed when the layout window is closed. Environment variable: closeSchWindow
Power/Ground Nets	This section provides options to specify the list of power and ground net names.
Power net names	Specifies the list of power net names used to achieve optimized chaining results in Layout XL device level schematics. Environment variable: lxSupplyNetNames
Ground net names	Specifies the list of ground net names to help you achieve optimized chaining results in Layout XL device-level schematics. Both lists are also used by the Virtuoso Custom Digital placer to exclude power and ground nets during wire length optimization; and to determine the power nets that are hidden when the hideDraglinesForGlobalNets environment variable is switched on.

© 2023 Cadence Design Systems, Inc.

Display

Use the *Display* tab to specify how the Layout XL windows are arranged on the desktop, how draglines are displayed in the layout window, and to control cross-selection and net probing.

Field	Description
Initial Display - Auto arrange windows	Controls whether Layout XL automatically rearranges its four windows on your desktop when you launch the application.
	The default is on, meaning that the windows are positioned based on the values specified for the following environment variables: ciwWindow, lamReadFileMessage, lxAllowPseudoParallelNets, and schematicWindow. If those environment variables are not set, or they are set to the default value `"((0.0 0.0) (0.0 0.0))"`, Layout XL tiles the three windows based on the height and width of your screen. This option is not available in *Cellview* mode. See *General* tab. Environment variable: autoArrange
Draglines - Display Draglines	Toggles the display of draglines when using the *Generate Selected From Source*, *Generate Clones*, *Move*, and *Stretch* commands. If the option is OFF, the options in this section are disabled.
	By default, the *Display Draglines* option is ON. Environment variable: flightLineEnable The option is not available in the *Cellview* mode. See *General* tab.
Show distant connections	Displays draglines for all the connections to the instance that is being manipulated. When switched OFF, draglines are shown only for the connections closest to the current instance. Environment variable: showDraglinesForDistantConns
Hide global, power, and ground nets	Hides draglines for global nets and power and ground nets specified using the lxGroundNetNames and lxSupplyNetNames environment variables. See hideDraglinesForGlobalNets.

© 2023 Cadence Design Systems, Inc.

Field	Description
Use colors	Shows each dragline in a different color. Displaying many draglines during interactive commands can impair the performance of Layout XL. To mitigate these effects, use the `maxDragFig` environment variable to limit the number of figures that can be in a dragset. To set `maxDragFig` interactively so that it takes effect in the current session, type the following in the CIW, where *`win`* is the window to which the limit applies. *`win`*`~>maxDragFig = 1000` To set a default value for `maxDragFig`, which will be used in all future Layout XL sessions, add the following line in your `.cdsenv` file. graphic maxDragFig int 1000 nil Environment variable: colorDraglines
Cross-selection	Enables cross-selection of components between schematic and layout. When you select a component in the layout, the corresponding component is selected in the schematic and vice versa, unless the instance is ignored in either view. Environment variable: crossSelect
Connectivity - Probe nets during object creation	Controls whether nets that are tapped during interactive editing are highlighted in the layout canvas. Environment variable: probeDuringCreate
Messages - Redirect Layout XL messages to separate Info window	Shows the messages issued by the *Check Against Source*, *Update Layout Parameters*, and *Update Schematic Parameters* commands in a separate info window instead of in the CIW. Environment variable: infoWindow

© 2023 Cadence Design Systems, Inc.

Connectivity

Use the options on the *Connectivity* tab to set the Layout XL parameters for the current design. If you change any of the options on this tab, the new settings are taken into account on the next interactive editing, if interactive verification is enabled. Else, the new settings are considered for the whole design if you extract the design by using the Extract Layout command.

This tab is not available in *Cellview* mode. See the General tab.

The *Connectivity* tab is further split into the following tabs:

Tab	Description
Controls	Specifies the verification controls for the layers in the design ans the connectivity assignment of the shapes and instance pins.
Annotations	Sets annotation types for incomplete nets, illegal connections, and necessary connections.
Limits	Specifies annotation limit values.

© 2023 Cadence Design Systems, Inc.

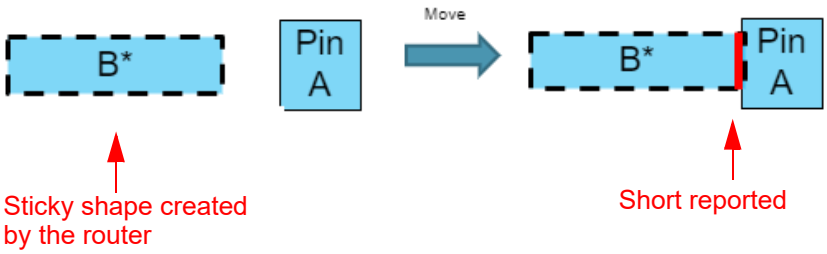
Controls

Field	Description
Verification Controls	This section specifies the verification controls for the layers in the design.
Derive extractable layers from constraint group	Sets the default constraint group from which Layout XL derives the extractable layers in the design. See Prerequisites for Using the Layout XL Connectivity Extractor.
Diagnostics Report	Displays extraction-related information about the selected constraint group and appropriate warning messages to indicate if the technology file has been correctly set up for extraction. You can display the report in a separate information window or in the CIW. See Extractor Diagnostics Options Form
Extract connectivity to level	Specifies how much of the design hierarchy the extractor considers when extracting the top-level design. See Extracting a Top-Level Design. Whatever value you set for *Extract connectivity to level*, the extractor only extracts the top level; i.e., it only changes the connectivity of objects or creates markers at the top level of the design. To extract hierarchical cellviews, set the value of this option to a value greater than `0` and use the *Extract Layout* command.
Update connectivity information when design is modified	Enables incremental connectivity extraction, which updates the connectivity model whenever the design is modified. When switched off, the connectivity model is not updated automatically and no new short or open violation markers are generated in the layout window. Even when incremental extraction is switched off, you can run connectivity extraction at any time using the *Connectivity – Update – Extract Layout* command. Environment variable: extractEnabled
Enable substrate connections and set well layers as soft	Allows connections to the substrate and sets the well layer connections as soft connections in the current cellview.

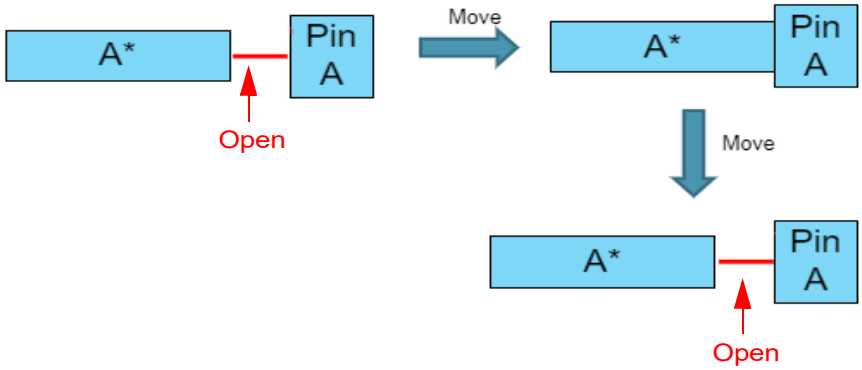
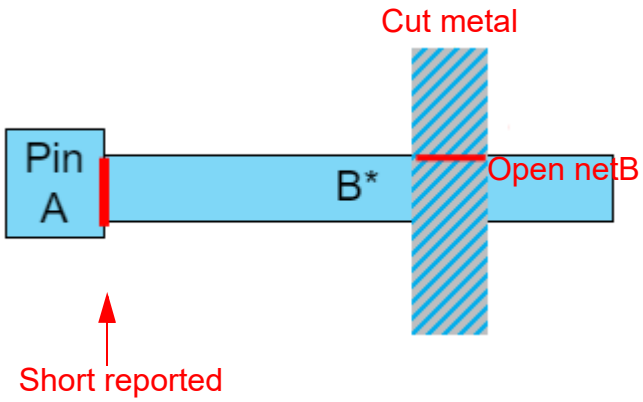
© 2023 Cadence Design Systems, Inc.

Field	Description
Use (identifier) to define nets as virtually connected	Indicates the identifier that when placed after the net name in the label text means that the corresponding set of physically connected shapes are virtually-connected. By default, colon (:) is used as a virtual-connect identifier. The connectivity extractor does not report open markers between virtually connected shapes. However, after extraction, an INFO message is issued in the CIW, which indicates the number of virtual connections. Environment variables: extractEnableVirtualConnections, extractVirtualConnectionSeparator If virtual connections are enabled, a colon (:) at the end of a terminal name is not considered as a name mismatch. See Enabling Virtual Connections.
Extract through pcells	Specifies that the connectivity of non-pin shapes in Pcells is extracted.
Check shorted terminals in pcells	Checks and prevents incorrect shorts due to incorrectly assigned shapes in Pcells. By default, the option is deselected. This option is available only if *Extract through pcells* is selected.
Stop Layers	This section specifies the controls on stop layers during extraction.
Promote all stop shapes	Specifies whether stop shapes at the `extractStopLevel + 1` level are visible to the extractor. Environment variable: extractPromoteAllStopShapes
Ignore color lock state	Specifies whether lock state of shapes should be ignored during extraction. Environment variable: extractIgnoreColorLockState
Connectivity Assignment	This section specifies the connectivity assignment of the shapes and instance pins.
Keep shape assignment	Preserves the connectivity of a design by keeping the shapes assigned. By default, the option is deselected. Environment variable: extractKeepShapeAssignment

© 2023 Cadence Design Systems, Inc.

Field	Description
Remove lxStickyNet property on non-floating shapes	Ensures that shapes that are connected to a connectivity source, such as a pin, an instance terminal, or a label, automatically have their lxStickyNet Property property removed to avoid any shorts or illegal must connections. By default, the option is OFF. Environment variable: extractRemoveStickyProperty During a *Copy* or *Repeat Copy* run with *Copy Connectivity* selected, the option automatically removes the lxStickyNet Property property on floating sticky shapes to avoid any shorts or illegal must ‘connections.  lxStickyNet property removed from the copied via Copied via reassigned to netB, no short reported The lxStickyNet Property property on a shape is not removed if the sticky shape is created by a router. 

© 2023 Cadence Design Systems, Inc.

Field	Description
	The sticky shape and the pin are on the same net.  Shape not assigned after the move, open reported The sticky shape is stopped by another layer. 
Connect unassigned instance pins	Ensures that an instance pin is automatically connected to the net assigned to the overlapping shape. By default, the option is ON. For example, consider an unassigned instance terminal that overlaps a shape connected to a net. If the option is ON, the instance terminal is assigned to the net. If the option is OFF, an illegal connection marker is created in the Annotation Browser and the instance terminal stays unassigned. Environment variable: extractConnectUnassignedInstTerms

© 2023 Cadence Design Systems, Inc.

Field	Description
Assign shapes from attached labels	Ensures that a top-level shape with an attached label is automatically connected to the net referenced by the label text. By default, the option is OFF. See Connectivity from Attached Labels to Consider During Extraction.
Assign shapes from overlapping labels as defined by 'stampLabelLayers' rule	Ensures that a shape overlapping the origin of a label is assigned to the net referenced by the label text. By default, the option is OFF. See Connectivity from stampLabels During Extraction.

Diagnostics Report Displays extraction-related information about the selected constraint group and appropriate warning messages to indicate if the technology file has been correctly set up for extraction. You can display the report in a separate information window or in the CIW (see Extractor Diagnostics Options Form for more information).

```
*** Constraint Group Diagnostics ***

-----------------
Techfile library: gpdk090
Constraint group: virtuosoDefaultExtractorSetup
-----------------

WARNING (LCE-2011): Cannot determine the function of layers 'Oxide Bondpad' in constrai

Extractable layers:
  Layer      Function     Mask
  (Oxide)    undefined    1
  (Nwell)    n-type well  2
  (Poly)     poly         3
  (Cont)     cut          6
  (Metal1)   metal        7
  (Via1)     cut          8
  (Metal2)   metal        9
  (Via2)     cut          10
  (Metal3)   metal        11
  (Via3)     cut          30
  (Metal4)   metal        31
  (Via4)     cut          32
  (Metal5)   metal        33
  (Via5)     cut          34
  (Metal6)   metal        35
  (Bondpad)  undefined    36
  (Via6)     cut          37
  (Metal7)   metal        38
  (Via7)     cut          39
  (Metal8)   metal        40
  (Via8)     cut          41
  (Metal9)   metal        42
```


© 2023 Cadence Design Systems, Inc.

The report, as displayed above, contains the following information:

- Details about the extractable layers, equivalent layers, and other similar extraction-related information generated by the `lcePrintExtractLayers()` SKILL function.
- Details about the standard via and custom via definitions generated by the `lcePrintExtractVias()` SKILL function.
- Warning messages generated by the extractor to indicate errors in the technology file setup for the selected constraint group.
- Details about the options selected on the *Connectivity* tab that determine the way the selected constraint group is parsed for extraction.

  For example, depending on the options selected on the *Connectivity Assignment* tab, the extractor may parse the selected constraint group to verify connectivity from stamp labels and to substrate or well layers.

  The details about the various sections in the Diagnostics Report and the information these sections provide about the extracted design is tabulated below:

Section	Purpose
Extractable Layers	Lists the physical layers that have been selected for extraction by using the `validLayers` constraint. For more information, see validLayers.
Equivalent Layers	Lists the layers that have been defined as electrically equivalent. For more information, Connectivity Rules.
Electrically-Connected Layers	Lists the layers that are electrically-connected. For more information, see validLayers.
Bulk Area Layers	Lists the layers that can be used to define the bulk area of shapeless terminals. For more information, see validLayers – Bulk Area Layers Specification and Bulk Area Connectivity Extraction.
Stop Layers	Lists the layers that are defined as “stopped” other layers. For more information, see validLayers – Stop Layers Specification.

© 2023 Cadence Design Systems, Inc.

Stamp_Label Layers	Lists the layers that can be used to create labels that stamp other layers. For more information, see stampLabelLayers.
Standard Via Definitions	Lists the valid standard via definitions that have been parsed by the extractor. For more information, see Via Definitions.
Custom Via Definitions	Lists the valid custom via definitions that have been parsed by the extractor. For more information, see Via Definitions.
Derived Layers in `validLayers` constraint	Lists the derived layers that have been parsed by the extractor for the `'validLayers'` constraint. For more information, see Derived Layers.
Excluded LPPs	Lists the layer purpose pairs that are not recognized by the extractor.

© 2023 Cadence Design Systems, Inc.

Annotations

Field	Description
Incomplete nets (opens)	Reports open violations in the design. When switched OFF, none of the dependent options is honored by Layout XL. Environment variable: extractVerifyOpenViolations ■ *Unimplemented instance pins:* Reports violations to indicate that the extractor should consider instance terminals with pin shapes that are not extractable (including instance terminals with no pin shapes at all). ■ *Shapes on non-routing layers*: Controls if open markers should be created with instance pin figures on a non-routing layers. ■ *Signal types*: Specifies the signal types in the design for which open markers should be reported. To report open markers for all the signals, click the *All* button. To specify that none of the signals be reported for open or incomplete nets, click the *None* button.
Incomplete must Connect All Pins	Reports the open net connections for pins that have the `mustJoinAllPins` property set.
Illegal weak connections	Reports the weak-connect violation markers for the current cellview in Annotation Browser. Environment variable: extractVerifyWeakMarkers
Illegal must connections	Reports the must-connect violation markers for the current cellview in Annotation Browser. Environment variable: extractVerifyMustConnectMarkers
Illegal hierarchical connection violations	Reports the illegal hierarchical connections in the current cellview. Environment variable: extractVerifyIllegalHierConnections
Ignore unassigned hierarchical shapes	Ignores the unassigned hierarchical shapes when checking illegal hierarchical connections. Environment variable: extractVerifyIllegalHierConnectionsIgnoreUnassignedHierShapes

© 2023 Cadence Design Systems, Inc.

Field	Description
Illegal overlaps for stopped colored shapes	Reports illegal overlap violations for color-dependent stopping of shapes. By default this option is deselected. See Support for Color-Dependent Shape Trimming. Environment variable: extractVerifyIllegalOverlapsForStoppedColoredShapes
Unnecessary pseudo-parallel connections	Reports the overlap between two sub-nets of a pseudo-parallel net. Environment variable: extractVerifyUnnecessaryPseudoParallelConnections

Limits

Field	Description
Maximum number of opens	Specifies the highest number of open violation markers that can be generated by the extractor for a particular net or cellview. By default, the extractor can generate a maximum of 100 open violation markers per net and 5000 open violation markers per cellview. Environment variables: extractCellviewOpenViolationLimit, extractNetOpenViolationLimit
Maximum number of shorts	Specifies the highest number of short violation markers that can be generated by the extractor for a particular net or cellview. By default, the extractor can generate a maximum of 100 short violation markers per net and 1000 short violation markers per cellview. Environment variables: extractCellviewShortViolationLimit, extractNetShortViolationLimit
Maximum number of illegal connections	Specifies the highest number of illegal connections that can be reported by the extractor for a particular design. By default, the extractor can report a maximum of 1000 illegal connections per design. Environment variable: extractCellviewIllegalConnectionLimit

© 2023 Cadence Design Systems, Inc.

Field	Description
Maximum number of unnecessary pseudo-parallel connections	Specifies the highest number of unnecessary pseudo-parallel connections that can be reported by the extractor for a particular cellview. By default, the extractor can report a maximum of 1000 such connections per cellview. Environment variable: extractCellviewUnnecessaryPseudoParallelConnectionLimit
Maximum number of invalid connections	Specifies the highest number of invalid connection markers that can be generated by the extractor for a particular net or cellview. By default, the extractor can generate a maximum of 100 invalid connection markers per net and 1000 invalid connection markers per cellview. (Virtuoso MultiTech Framework)

© 2023 Cadence Design Systems, Inc.

Generation

Use options on the *Generation* tab to specify how certain components are handled when they are generated in the layout view. This tab is not available in *Cellview* mode.

See General tab.

Field	Description
Generation	This section lets you specify the options available in *Generation* section.
Split mfactored devices	Controls whether Layout XL places schematic devices with the mfactor Property property as multiple devices in the layout. See Splitting Mfactored Devices. Environment variable: mfactorSplit
Split sfactored devices	Controls whether Layout XL places schematic devices with the sfactor Property property as multiple devices in the layout. See Splitting Sfactored Devices. Environment variable: sfactorSplitsfactorSplit
Split fingered devices	Controls whether each finger of a schematic device will be placed in the layout as a separate device. See Splitting Fingered Devices. Environment variable: fingerSplit
Create pseudoparallel connections	Enables the detection of pseudoparallel nets during chaining. A pseudoparallel net connects nodes that are always the same voltage, so the current does not pass through the net. See Generating a Layout without Chaining and Folding on Pseudoparallel Nets and Defining a Pseudoparallel Connection. Environment variable: setPPConn

© 2023 Cadence Design Systems, Inc.

Field	Description
Create implicit bus terminals	Creates, checks, and updates implicit bus terminals in the layout cellview based on the explicit bus terminals in the schematic cellview. This avoids the need to run verilogAnnotate on the layout cellview before referencing the cellview for: ■ Running verilog2oa ■ Importing another verilog design into Cadence Innovus Digital Implementation System. The *Create implicit bus terminals* option is OFF by default. Environment variable: createImplicitBusTerminals
Device orientation	Controls the orientation of Layout XL layout devices generated from the schematic. Devices generated with a particular orientation may be further transformed if either the *Chain, Fold,* or *Chain Folds* options are switched on. Environment variable: lxGenerationOrientation
PR boundary snap	Controls the snapping of the PR boundary and the area boundaries when running the Generate All From Source or the Adjust Boundary command. ■ *None* implies no boundary snapping takes place. ■ *Lower Left* creates the PR boundary of the specified size and snaps the lower-left point of the boundary bbox to the grid. ■ *All Points* snaps all the points of the PR boundary bbox to the grid. This may result in the boundary bbox being larger than expected. Environment variable: prBoundarySnap
Auto Adjustment	This section lets you specify the auto adjustment options.

© 2023 Cadence Design Systems, Inc.

Field	Description
Abut devices	Turns on automatic abutment, which abuts prepared devices so that they can share pins. Abutment is switched on for both automatic and interactive layout generation and during automatic placement using the Virtuoso Custom Digital Placer. Environment variable: autoAbutment ■ *Align abutted devices* snaps an instance in the direction perpendicular to the direction of abutment in order to align the instances ■ *Preserve terminal contacts* specifies whether pin properties are preserved when an instance is abutted. Switching this option on means that the terminal contacts will be preserved when you abut devices in Layout XL. Environment variable: preserveTerminalContacts ■ *Perform auto abutment only after a move or stretch* restricts abutment to occur only after interactive edits using the *Move* and *Stretch* commands.
Space components	Turns on automatic spacing, which allows components with the properties vxlInstSpacingDir Property and vxlInstSpacingRule Property to be spaced automatically according to the values specified in the properties. If one of the devices is in any type of group (including a synchronous clone), the other device must be in the same group for automatic spacing to occur. Environment variable: autoSpace
Permute pins	Turns on automatic pin permutation during manual routing or editing. See permuteRule Property.

© 2023 Cadence Design Systems, Inc.

Field	Description
Mirror devices	Specifies that if a short violation is created during abutment, then Layout XL first mirrors the device to resolve the short. If this is unsuccessful, Layout XL attempts to resolve the short by permuting the pins. If you switch the option off, Layout XL uses only pin permutation when attempting to resolve shorts. If one of the devices is in any type of group (including a synchronous clone), the other device must be in the same group for automatic mirroring to occur. Environment variable: autoMirror
Process all violations during batch extraction	Specifies if the XL fixer engines, such as *auto-permute*, *auto-mirror, auto-spacing*, and *auto-abutment*, should process the violations during batch extraction. Environment variable: processBatchViolations
Device Folding	This section lets you specify the device folding options.
Generate minimal folding	Creates the minimum number of folded devices. When switched off, Layout XL generates an odd number of folded devices. For example, it adds one fold if the device width divided by number of folds yields an even number. Environment variable: lxStackMinimalFolding
Retain device orientation	Preserves the original device orientation when running the *Generate Folded Devices* command or the *Generate All From Source* command with the *Fold* option switched on. If the *Chain* option is also switched on, device orientation is not preserved. Environment variable: lxRetainFoldOrient

© 2023 Cadence Design Systems, Inc.

Field	Description

Update width parameter method	Specifies how the *Update Layout Parameters* command updates folded devices when the width value is changed in the schematic.
	
	■ *No Change*: Does not update folded devices.
	
	■ *Equalize*: Creates folds of equal widths in the layout instance. The number of folds does not change, only (potentially) the width of each fold changes.
	
	For example, the total width in the schematic is 12 and is distributed as follows in the layout.
	
	`layout widths I1.1 w = 2`
	`I1.2 w = 4`
	`I1.3 w = 6`
	
	If you change the width in the schematic to 15 and run *Update – Layout Parameters* with this option set, the new widths are as follows.
	
	`layout widths I1.1 w = 5`
	`I1.2 w = 5`
	`I1.3 w = 5`
	
	■ *Distribute*: Distributes the additional (or reduced) width equally amongst all the folds in the layout instance. Again, the number of folds does not change, only the width of each fold changes.
	
	For example, the total width in the schematic is 12 and is distributed as follows in the layout.
	
	`layout widths I1.1 w = 2`
	`I1.2 w = 4`
	`I1.3 w = 6`
	
	If you change the width in the schematic to 18 and run *Update – Layout Parameters* with this option set, the new widths are as follows.
	
	`layout widths I1.1 w = 4`
	`I1.2 w = 6`
	`I1.3 w = 8`
	
	An extra 2 units of width is added to each layout instance.

© 2023 Cadence Design Systems, Inc.

Field	Description
	The *Update Layout Parameters* command applies `lxRounding` on the folded widths before setting to maintain the widths as rounded during folding. The *Check Against Source* command will continue to report the total width, which may not match in this case. Environment variable: lxUpdateFoldedWidth
Device Chaining	This section lets you specify the device chaining options.
Create interdigitated chains	Automatically identifies nodes that qualify as pseudoparallel connections and defines them during chaining and abutment. A pseudoparallel net connects nodes that are always the same voltage, so the current does not pass through the net. Contacts are dropped automatically, if appropriate. However, if the *Preserve terminal contacts* option is switched on, the contacts are preserved. This option applies only when chaining is turned on. It does not control the automatic identification and creation of pseudoparallel nets during manual abutment. To do that, switch on *Create pseudoparallel connections*. Environment variable: lxAllowPseudoParallelNets
Maximum number of devices in a chain	Sets the maximum number of devices permitted in a chain.
Search levels up for abutments	Controls how far up the hierarchy Layout XL looks for opportunities to chain devices. A value of `0` means that chaining can look only in the same hierarchical level. A value of `N` means that chaining can look up `N` levels of hierarchy. This option is always considered by the *Generate All From Source* and *Generate Selected From Source* commands. However, the *Generate Chained Devices* command considers it only if there are 100 instances or more in the selected set. Environment variable: lxStackPartitionParameters

© 2023 Cadence Design Systems, Inc.

Field	Description
Align P Devices	Controls the alignment of p-device chains. The default alignment for p-device chains is “Top*”* but you can choose to align a chain to “Bottom*”* or “Center*”*. Environment variable: lxChainAlignPMOS
Align N Devices	Controls the alignment of n-device chains. The default alignment for n-device chains is “Bottom” but you can choose to align a chain to “Top*”* or “Center”. Environment variable: lxChainAlignNMOS
Chain left net	Controls whether source or drain nets are optimized to the left of the generated chains. The default is *Source*, which means a chain is optimized so that one of its source nets is on the left-hand side (where possible). Set to *Either* if you have no preference (and to maintain the default behavior from previous releases). *Source* and *Drain* here refer to schematic source and drain nets, and not the layout source and drain nets, which may have been permuted. Environment variable: chainLeftNet
Chain effort	Controls the effort applied during chaining. Increased effort can improve the quality of results, but it can be time-consuming. The default is *Nominal*. Environment variable: chainEffort
Chain optimize	Optimizes the chains by minimizing the number tracks or maximizing the number abutments or both. The default value is *Both*. Environment variable: chainOptimize

© 2023 Cadence Design Systems, Inc.

Parameters

Use options in the Parameters tab to specify which parameters are to be ignored by the generation and check command and how parameters are compared during *Check Against Source.*This tab is not available in Cellview mode. See General tab.

Field	Description
Ignore/Check	This section lets you specify the names of parameters to be ignored or checked by the check and update commands and the names of properties that cause objects to be ignored during generation and check.
Add	Adds to the list a parameter or property name you have typed into the text field.
Remove	Removes the selected parameters or properties from the list.
Edit	This field list the parameters and properties that need to be checked.
Parameters to check	Lists only those parameters that are checked by the *Check Against Source* command. Cadence recommends that you use the LAM file to set the parameters or provide an exhaustive list of parameters to be checked. Environment variable: paramsToCheck
Parameters to ignore during check	Lists the parameters that are ignored by the *Check Against Source* command. Mismatches for any of the listed parameters are not reported by the check. The list inherits all the parameter names from *Parameters to ignore during generation*. Environment variables: paramsToIgnore, paramsToIgnoreForCheck

© 2023 Cadence Design Systems, Inc.

Field	Description
Parameters to ignore during generation and update	Lists the parameters that are ignored by the following *Connectivity* commands: *Generate – All From Source, Generate – Selected From Source*, *Generate – Clones*, *Update – Components And Nets*, *Update – Layout Parameters*, *Update – Schematic Parameters*. The following parameters are always ignored: `instancesLastChanged`, `instNamePrefix`, `lxIgnoreParamForCAS`, `lxIgnoredParams`, `lxMFactorNum`, `lxParamsToIgnore`, `lxParamsToIgnoreForCheck`, `lxPlacementStatus`, `lxRounding`, `lxStopList`, `lxTimeStamp`, `lxUseCell`, `pin`, `posi` Environment variable: paramsToIgnore
Properties used to ignore objects during check	Lists the properties that cause pins and instances to be ignored by the *Check Against Source* command. Any object with one of the listed properties set to `t` is ignored during these operations. The list inherits all the property names from *Properties used to ignore objects during generation* and also includes lvsIgnore Property by default. Environment variable: propsUsedToIgnoreObjsForCheck
Properties used to ignore objects during generation and update	Lists the properties that cause pins and instances to be ignored during the generate, check, and update commands listed above. Any object with one of the listed properties set to `t` is ignored during these operations. By default, the list contains the following properties. `lxRemoveDevice ignore nlAction` Environment variable: propsUsedToIgnoreObjs
Inherited from Generation	Lists the parameters or properties that are inherited from the list defined for generation. Parameters and properties ignored for generation are always ignored for check.
Schematic Parameter Names	This section lets you specify the names of special parameters used in the schematic. Select the parameter you want to edit from the cyclic field and type the names in the text field.

© 2023 Cadence Design Systems, Inc.

Field	Description
Parallel-connected factor	Lists the names of the schematic parameters used to specify the multiplication factor (`mfactor`) for transistors. The default is `m M`. Layout XL checks each schematic instance for a parameter matching one of the names on the list and uses the value of that parameter to generate the appropriate number of parallel-connected devices in the layout. Environment variable: mfactorNames **Note:** For PDKs that use `simM` as the schematic multiplier, the `mfactorNames` environment variable should be set to `simM`.
Series-connected factor	Lists the names of the schematic parameters used to specify the number of series-connected (`sfactor`) devices to be generated in the layout. The default is `s S`. Layout XL checks each schematic instance for a parameter matching one of the names on the list and uses the value of that parameter to generate the appropriate number of series-connected devices in the layout. Environment variable: sfactorNames
Transistor	Lists the names of schematic parameters used to specify transistor width. The default is `w` and the value must be split among the generated parallel-connected devices during the *Generate All From Source*, *Check Against Source*, or *Update Layout Parameters* commands. Layout XL checks each schematic instance for one of the listed parameter names and updates the width values of the matching layout parameters such that `number of mfactored instances * mfactorSplit value = source parameter value` Environment variable: transistorWidthParamNames

© 2023 Cadence Design Systems, Inc.

Field	Description
Capacitance	Lists the names of the schematic parameters that are used to specify capacitance. The default is `c C`. The capacitance value must be split among the generated series-connected devices during the *Generate All From Source*, *Check Against Source*, or *Update Layout Parameters* commands. Layout XL checks each schematic instance for one of the listed parameter names and updates the capacitance values of the matching layout parameters accordingly. Environment variable: capacitanceParamNames
Resistance	Lists the names of the schematic parameters that are used to specify resistance. The default is `r R`. The resistance value for a device must be split among generated series-connected devices during the *Generate All From Source*, *Check Against Source*, or *Update Layout Parameters* commands. Layout XL checks each schematic instance for one of the listed names and updates the values of the matching layout parameters accordingly. Environment variable: resistanceParamNames
Inductance	Lists the names of the schematic parameters that are used to specify inductance. The default is `l L`. The inductance value for a device must be split among generated series-connected devices during the *Generate All From Source*, *Check Against Source*, or *Update Layout Parameters* commands. Layout XL checks each schematic instance for one of the listed names and updates the values of the matching layout parameters accordingly. Environment variable: inductanceParamNames
Fingers	Lists the schematic parameters that are recognized as finger values when generating the layout. Layout XL checks each schematic instance for one of the listed parameter names and generates the device in layout with as many fingers, if applicable, based on the finger value associated with that parameter name. However, if fingerSplit is set to `t`, the fingers are split into separate devices when generating the layout.

© 2023 Cadence Design Systems, Inc.

Field	Description
Segments	Lists the schematic parameters that are set to indicate a single instance of a resistor with one or many segments. The name of each segment is unique and can be set by using the segmentParamNames environment variable. The default is `segments`.
Parameter Comparison, Generation and Update	This section lets you defines how CDF parameters and properties are compared during the *Check Against Source*, *Update Layout Parameters*, and *Update Schematic Parameters*.
Consider parameters only	Compares only CDF parameters in the schematic against the parameters and properties in the layout and reports (or updates) values that do not match, parameters that cannot be checked, and parameters that are missing from the layout view. By default, parameters that are present in the schematic but missing from the layout are ignored. To propagate the missing parameters to the layout, make sure the *Ignore missing parameters or properties* check box is OFF. Environment variable: checkParamsOnly
Consider parameters and properties	Checks CDF parameters and cell and instance properties in the schematic against the parameters and properties in the layout and reports (or updates) values that do not match, parameters that cannot be checked, and parameters that are missing from the layout view. Use this option to check or update user-defined properties in the schematic and layout or when you know that a particular instance has certain properties that you are interested in comparing or updating. By default, parameters and properties that are present in the schematic but missing from the layout are ignored. To propagate the missing parameters and properties to the layout, make sure the *Ignore missing parameters or properties* check box is OFF. Environment variable: checkParamsOnly

© 2023 Cadence Design Systems, Inc.

Field	Description
Ignore missing parameters or properties	Ignores parameters (and properties, if *Consider parameters and properties* is checked) that are present in one view but missing from the other. Switch off this option if, for example, your layout has additional parameters or properties that you want to propagate to the schematic using the *Update Schematic Parameters* command. Environment variable: checkMissingParamsOrProps
Tolerance	Specifies the relative tolerance used when comparing values between the schematic and layout. If `a` is the source parameter value, `b` is the layout parameter value, and `e` is the *Tolerance* then parameter values are considered equal if `| (a/b) - 1.0 | < e` Environment variable: paramTolerance
Always consider properties for generation	Transfers properties from schematic instances to layout instances during the *Generate All From Source* or *Generated Selected From Source* commands. Properties are transfered in addition to the CDF parameters, which are always transferred.
Update based on parameters to ignore for	Updates instance parameters, ignoring the parameters that are defined as parameters to be ignored for generation or the parameters to be ignored for check. See Ignore Parameters and Terminals. By default, the option updates instance parameters based on the parameters to check or parameters to ignore for check. In this case, only the parameters marked for checking are updated if they report a parameter or property mismatch during Check Against Source. The parameters that are ignored during the check do not report a mismatch and are not updated. Environment variable: updateParamsForCheck

© 2023 Cadence Design Systems, Inc.

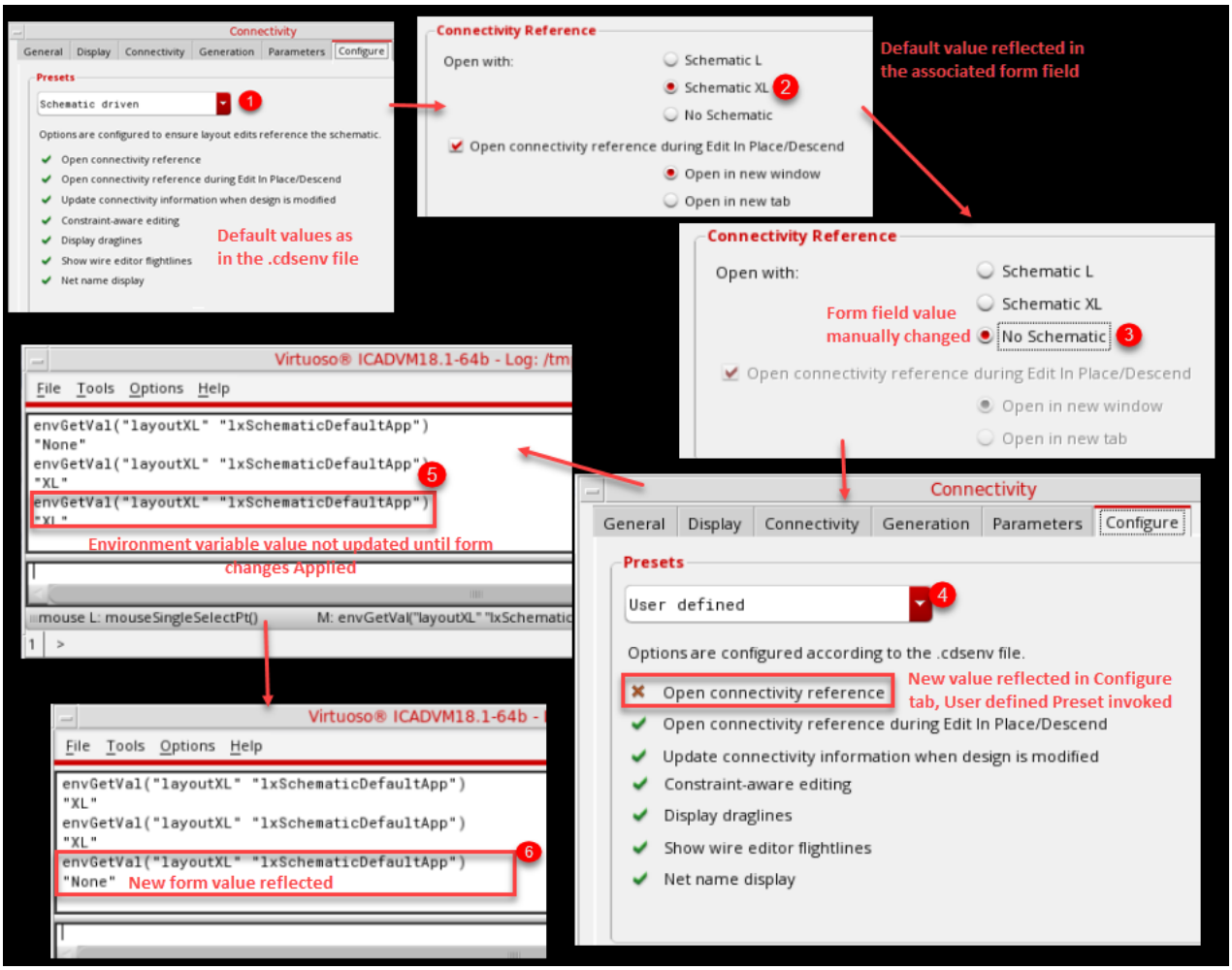
Field	Description
Update schematic parameter expressions with evaluated value	Updates the value of schematic parameters that are defined as expressions with the evaluated layout values, when the parameter values across the two views are different. Environment variable: updateSchParamWithEvalValue By default, the option is ON, which implies that all the schematic parameter values, including that of schematic parameters that are expressed as AEL or NLP expressions, are updated with the layout parameter values, when the evaluated schematic and layout parameter values are found to be different. For schematic parameter values that are expressions, the automatic update can overwrite the expression parameter values with the fixed value of the evaluated layout parameter. To avoid the automatic update, you can set updateSchParamWithEvalValue to `nil`.
Component Description Format	This section lets you specify the format of component description.
Evaluate CDF callbacks during generation and update	Causes all SKILL callbacks defined on CDF parameters to be evaluated by default when you run the *Generate* and *Update* commands in Virtuoso Layout Suite XL. Callbacks defined on buttons are not evaluated. Environment variable: lxEvalCDFCallbacks
Use significant digits when calculating parameter values	Controls the number of significant digits used by Layout XL when calculating the value of layout CDF parameters. When the option is selected, the value it specifies overrides the setting of `aelGetSignifDigits` for calculations in the layout cellview. This lets you specify a different precision to be used in layout cellview calculations; for example, those involving Pcells. The *Use significant digits when calculating parameter values* check box is available only if the shell environment variable CDS_Netlisting_Mode is set to Analog. See Setting the CDS_Netlisting_Mode Environment Variable.

Configure

Use options on the *Configure* tab to know the values that environment variables that are associated with the Preset options are set to when the Connectivity form is launched.

© 2023 Cadence Design Systems, Inc.

Similarly, for form fields that are manually updated to use a different value, the Configure options reflect the current state of the updated form fields.



Environment variable: displayConfigureTab

Field	Description
Presets	This section lets you specify the preset options.
Schematic driven	Reflects that all the form fields associated with the Preset options have their value set to ON, implying the option is selected.
Non schematic driven	Reflects that all the form fields associated with the Preset options have their value set to OFF, implying the option is deselected.

© 2023 Cadence Design Systems, Inc.

Field	Description
User defined	Reflects that the form fields associated with the Preset options are neither Schematic driven (all fields set to ON) nor Non schematic driven (all fields set to OFF). If the Connectivity form is already open, the *Configure* tab is switched to *User defined* Preset mode. The associated environment variable is also set to the new form field value, after the form field value is applied. If the *Connectivity* form is opened again in the same Virtuoso session, it opens the *Configure* tab in *User defined* Preset mode, using the new form field value.

© 2023 Cadence Design Systems, Inc.

Field	Description
Options are configured to ensure layout edits reference the schematic	■ *Open connectivity reference:* Lets you confirm whether or not the Virtuoso Schematic Editor is started when you launch Layout XL. Use the *Connectivity Reference – Open* field in the *General* tab of the Connectivity form to changes the preset value. Environment variable: lxSchematicDefaultApp ■ *Open connectivity reference during Edit In Place/ Descend*: Lets you confirm whether or not the source cellview is opened when you use the *Edit In Place* or *Descend* commands. When switched off, the layout cellview is still opened in Layout XL mode, but with no connectivity reference. Use the *Connectivity Reference – Open connectivity reference during Edit In Place/Descend* field in the *General* tab of the Connectivity form to change the preset value. Environment variables: openConnRef, openConnRefTab ■ *Update connectivity information when design is modified:* Lets you confirm that the incremental connectivity extraction will be enabled, which updates the connectivity model whenever the design is modified. Use the *Controls – Update connectivity information when design is modified* to change the preset value in the *Connectivity* tab of the Connectivity form. Environment variable: extractEnabled ■ *Constraint-aware editing:* Confirms that the *Copy*, *Move*, *Stretch*, and *Rotate* commands will honor the following constraints: *Alignment*, *Fixed*, *Locked*, *Matched Parameters*, *Orientation*, *Matched Orientation*, and *Symmetry*. Use the *Constraint-aware editing* to change the preset value in the *General* tab of the Connectivity form. Environment variable: constraintAwareEditing

© 2023 Cadence Design Systems, Inc.

Field	Description
	■ *Display draglines*: Confirms whether or not draglines will be displayed when using the *Generate Selected From Source*, *Generate Clones*, *Move*, and *Stretch* commands. Use the *Display draglines* field in the Display Tab of the Connectivity form to change the preset value. Environment variable: flightLineEnable ■ *Show wire editor flightlines:* Lets you confirm whether or not flightlines will be displayed during *Wire*, *Bus*, and *Guided Routing* commands. Use the *Show flightlines* to change the preset value in the Interactive Section of the Wire Assistant GUI. Environment variable: autoUpdateFlightlines ■ *Net name display:* Lets you confirm that net name labels will be displayed in the layout for the shapes that have associated net name labels. Use the *Display net name labels* to change the preset value in the Net Name Display Options form. Environment variable: netNameDisplayEnabled

Related Topics

Connectivity Form

Net Name Display Options Form

Constraint-Aware Editing

Generating All Components from Source

Placement Modes for Generating Selected Components from Source

Rapid Adoption Kit (*Introduction to Connectivity-Driven Design in VLS XL*)

Components and Nets Updates

© 2023 Cadence Design Systems, Inc.

Design Check Against Source

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Convert Libraries to Use Physical Configuration Views Form

Use the Convert Libraries to Use Physical Configuration Views form to specify which schematic libraries and designs are converted to use the Layout XL schema and what base name is used for the physical configuration views that are created.

Field	Description
Libraries not to convert	Lists the libraries referenced in the current library definitions (`cds.lib`) file that will not be converted.
Libraries to convert	Lists the libraries that will be converted when you click *OK*. The software automatically moves all the valid libraries defined in the current library definitions (`cds.lib`) file into this field. Reference libraries shipped with Cadence software are not shown in either list. If you do not have permission to write to a library, that library is not converted.
Arrow buttons	Lets you move library names from *Libraries to convert* to *Libraries not to convert* pane.

Configuration view base name	Specifies the base name of the physical configuration views to be generated. This lets the system automatically add the physical configuration view names to the *Logical switch view list*. The default is `physConfig`.
	
	If there are multiple schematic views to be converted, the base name is perpended to each schematic view name.
	
	For example, if a cell has the following views before conversion:
	`schematic`
	`schematic1`
	`schematic2`
	
	It will have the following views after conversion:
	`physConfig`
	`physConfig_schematic1`
	`physConfig_schematic2`

Related Topics

Physical Hierarchy Configuration

© 2023 Cadence Design Systems, Inc.

Layout Design Conversion

Data Conversion to Use the Layout XL Schema

Layout XL Forms

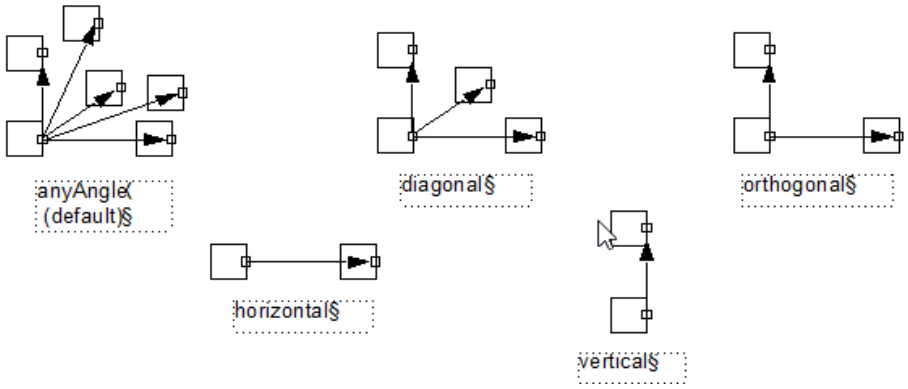
© 2023 Cadence Design Systems, Inc.

Create Cluster Boundary Form

Use the Create Cluster Boundary form to create a cluster boundary around one or more cluster objects.

Field	Description
Cluster Name	Lists the clusters in the design and lets you create a cluster using the *Create – P&R Objects – Clusters* command. You can define clusters as *inclusive*, which means that you can place them in the cluster boundary, or *exclusive*, which means that they are excluded from and cannot be placed in the cluster boundary.
Cluster Boundary Name	Specifies a unique name for the cluster boundary. By default, the system assigns a name of the form `CB_integer`, where `integer` is incremented each time a cluster boundary is created.
Fit Member Instances Into Cluster Boundary	Places all the components in a cluster inside the specified cluster boundary. Components are placed according to their size from left to right and bottom to top. That means that the largest object is placed in the bottom left corner and the smallest object in the top right corner. If there is not enough space, the components in the top row are overlapped.
Cross-Highlight Boundaries	Highlights in the layout window the cluster boundary associated with the cluster selected in the form. When generating new cluster boundaries, use this option to associate existing clusters with their boundaries.
Create Method	This section lets you specify whether boundary creation is *Manual* or *Automatic*.
Manual	Lets you specify the cluster boundary by clicking with your mouse in the layout canvas.
Automatic	Specifies that the system creates a boundary based on the origin, aspect ratio and area information you specify.
Rectangle	Lets you create a rectangular boundary by clicking twice in the canvas.
Polygon	Lets you create a polygonal boundary by clicking once for each vertex in the canvas. The shape you draw must have at least three points and must not be self-intersecting. To close the polygon, double-click.

© 2023 Cadence Design Systems, Inc.

Field	Description
Snap Mode	Lets you select the types of shapes you can draw. Valid modes are: 
Aspect Ratio (W/H)	Sets the width-to-height ratio of the cluster boundary. For example, a value of 1 specifies a square boundary, 2 specifies a boundary twice as high as it is long, and 0.5 specifies a boundary half as high as it is long. The default is 1. Environment variable: clusterBoundaryAspectRatio
Origin X and *Origin Y*	Sets the origin point.
Area Information	This section lets you specify a target area for the cluster boundary or informs the system to estimate the area for you.
Area Estimator	Lets you estimate the target area based on the PR boundary, the bounding box of the components contained in the cluster, or using a user-defined estimator you have registered in the system. The *Estimate* button is grayed out unless there is a registered estimator. Use the `leRegClusterBdyEstimator` SKILL function to register an estimator.
Estimated Area For Cluster	Displays the estimated area required to accommodate the components in the cluster depending on the other settings in the form.
Utilization (%)	Specifies the percentage of area within the cluster boundary that you want to fill with components. The lower the utilization value, the more free space is available.
Target Area	Lets you specify a target area for the cluster boundary.

© 2023 Cadence Design Systems, Inc.

Related Topics

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Create Group Array Form

Use the Create Group Array form to create a group array with the specified parameters.

Field	Description
Include Clone Source	Includes the clone source in the group array. The clone source is the lower-left cell of the group array. The clone targets the other cells of the group array. If the clone source is already a synchronous clone or contains a synchronous clone, it cannot be included in the group array. In this case, the *Include Clone Source* option is disabled.
Number of Clones in the array	Specifies the number clones that are included in the group array.
Mode	Controls the type of spacing for the group array. ■ *Pitch*: Lets you specify the distance that is measured between the same-edge corners of cells in the *Delta X,Y* fields. For example, the distance between the left edges of two cells. ■ *Spacing*: Lets you specify the distance that is measured between the adjacent or closest edges of cells in the *Spacing X,Y* fields. For example, the distance between the top edge of a cell and the lower edge of another cell.
Spacing X,Y	Specifies the spacing in the Spacing X,Y fields
Reference LPP	Specifies the layer-purpose pair used for computing the bounding box of the cells in a group array. The combined bounding box of the geometries on the specified reference layer-purpose pair is used for calculating the spacing between cells.
Array Pattern	Displays the Array Pattern form to let you specify the orientation pattern for the group array.

Related Topics

Array Pattern Form

Creating a Group Array when Generating Clones

© 2023 Cadence Design Systems, Inc.

Specifying Group Array Orientation Using the Array Pattern Form

Predefined Orientation Patterns for a Group Array

© 2023 Cadence Design Systems, Inc.

Create Feed Through Terminal Block Pins Form

Use the Create Feed Through Terminal Block Pins form to create additional feed through pins to improve the connectivity of your design.

Field	Description
Soft block list	Lists all the soft blocks that were selected when you opened the form. You can create soft pins for any of the blocks that are listed this field. The *Lib*, *Cell*, and *View* fields confirm the currently selected block.
Select Net/Bundle	This section provides options to select nets or net bundles. To assist you in defining constraints, the canvas shows flight lines to represent the nets between the terminals you select.
Search for	Filters the design objects that are searched. Choose one of *Nets*, *Bus*, or *Net Class*.
Name	Lets you type the search expression. Click *Filter* to see the results.
Define TTB pin parameters	This section provides options to specify in detail the pins you want to create.
User defined terminal name	Specifies the name of the terminal on which the pins will be generated. If you do not specify a name, the system assigns one for you.
Layer	Specifies the layer purpose on which the pin is to be created. Choose the layer purpose you want from the list of extractable layers in the design.
Width and Height	Specifies the width and height of the new pin. These fields are prepopulated with the `minWidth` spacing defined in the technology file for the selected pin layer.
Num	Specifies the number of pins to create on the specified terminal. The default is `1`.
Side	Specifies on which sides of the PR boundary the pins can be created. If you specify more than one side for a pin, the pin optimizer can place the pins on any of the specified sides.
Align Pin	Helps you to align the pins on a particular layer purpose.
Add	Adds the feed through pin parameters to the table.
Update	Updates the pin table with the pins and parameters you provide.

© 2023 Cadence Design Systems, Inc.

Field	Description
Delete	Deletes a feed through terminal pin.

Related Topics

Soft Block Pins

Creating a Feedthrough Terminal Pin

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Create Path Form

Use the Create Path form to specify how a path is created in a design.

Field	Description
Width	Specifies the path width in user units. The path will initially default to the minimum width defined for the layer on which you create the path unless you enter a new value in this field. Once you have entered a new value in the field, each time you create a path on the layer whose value you have changed, the value is reused until you enter a new value in the field. To return to the minimum width, click the *Defaults* button.
Fixed Width	Uses the width you specify in the *Width* field for all segments of the path when turned on. When turned OFF, the path width remains set at the last setting until you click the *Defaults* button. The *Defaults* setting uses the `minWidth` defined in the technology file for the current layer.
Justification	Controls which edge of the path you create: *left*, *center*, or *right*. If you set a path *Offset*, *Justification* controls which edge of the path is offset from the line you create.
Offset	Offsets the line you use to create the path from the final path. To control which edge of the path is offset, set the *Justification* cyclic field.
End Type	This drop-down menu provides options to control how the path ends are created. ■ *truncate*: Specifies that the path ends and path points end at the same points. ■ *extend*: Specifies that the path ends extend from the path points by one half the path width. ■ *round*: Specifies that the path ends extend from the path points by one half the path width, symmetrically in four points at each end. ■ *variable*: Specifies that the path ends extend from the path points by whatever value you enter in the *Begin Extension* and *End Extension* fields.

© 2023 Cadence Design Systems, Inc.

Field	Description
Begin Extension and End Extension	When *End Type* is set to *variable*, lets you enter a beginning and an ending extension in user units.
Net Name	Assigns the path to a specific net.
Create as ROD Object	Creates the path as a relative object design (ROD) object. ROD *Name* assigns a name to the new path in the current layout cellview if *Create as ROD Object* is on. The name must be unique in the cellview. If you do not edit this field, the system assigns a unique name consisting of the prefix `path`, followed by a number. For example, the first ROD path would be named `path0`, the second, `path1`, and so on.
Enable Metal Slotting	Enables you to create a path with slots. If selected, it enables the *Slot Options* tab on which you can specify the settings for creating a path with slots.
Snap Mode	This drop-down menu provides options to control how path segments snap to the grid. ■ *anyAngle*: Creates lines at any angle. ■ *diagonal*: Creates lines parallel to the X or Y axis or at a 45-degree angle to the axes. ■ *orthogonal*: Creates lines parallel to the X or Y axis. ■ *L90XFirst*: Creates orthogonal two-segment lines and creates the first line in the X direction. ■ *L90YFirst*: Creates orthogonal two-segment lines and creates the first line in the Y direction.NNTo toggle between *L90XFirst* and *L90YFirst* while you are creating a path, click right. ■ *Acute Angle*: Lets you create acute angle paths when set to on (available only when *Snap Mode* is set to *anyAngle*). When set to off, you can create paths only from 90 degree to 180 degree angles.
Connectivity	This section provides an option to highlight nets.
Probe Nets	Highlights nets while creating interconnect.

© 2023 Cadence Design Systems, Inc.

Related Topics

Slot Options

Creating Objects

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Create Physical Configuration View Form

Use the Create Physical Configuration View form to create a new physical configuration view to be used in a Layout XL session.

Field	Description
Library Name	Specifies the name of the library in which the physical configuration view resides.
Cell Name	Specifies the name of the cell to which the physical configuration view relates.
View Name	Specifies the name of the physical configuration view.
Use Template	Specifies whether a template or file containing predefined values is to be used for the new physical configuration view.
Template	Lets you select a predefined template file from the drop-down list.
From File	Lets you specify a template file stored in another location. If you do not specify a template or file, Layout XL uses the default template and any overriding values from the corresponding environment variables.
Open CPH	Opens the Configure Physical Hierarchy Window automatically after you click *OK* in this form. Environment variable: cphLayoutSync

Related Topics

Physical Hierarchy Configuration

Layout XL Launched from a Schematic View

Starting Layout XL with a New Physical Configuration View

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Custom Show Filter Form

Use the **Custom Show Filter** form to define more complex filtering criteria, which reduce the number of annotations displayed in the Annotation Browser. To open the form, right-click the header of the column for which you want to define the custom filter. Then, choose *Show – Custom* from the shortcut menu.

The *Custom Filter* option is not supported by the following columns: *Tree*, *Set Highlight State*, *Set Highlight Color*, and *Set Marker Check State*.

Field	Description
Add criteria	Adds the filtering criteria to the list.
Match all of the following	Species whether filtering should match all of the specified criteria.
Match any of the following criteria	Species whether filtering should match any of the specified criteria.
	Specifies the operator for the criteria: ■ < Less than ■ <= Less than or equal to ■ == Exactly equal to ■ >= Greater than or equal to ■ >>br>Greater than ■ != Not equal to
	Specifies the column content by which you want to filter. The content of this list is determined automatically by the system.
	Removes the corresponding criterion from the list.

© 2023 Cadence Design Systems, Inc.

Related Topics

Annotation Browser Assistant

Annotation Browser Options Form

Incomplete Net Filter Form

View By Form

© 2023 Cadence Design Systems, Inc.

Define Device Correspondence Form

Use the Define Device Correspondence form to view and change the device correspondence for the instances and terminals in your design. The Define Device Correspondence form is also available in the Read mode in Layout XL. However, in this mode, the *Bind* and *Unbind* buttons are inactive.

Device correspondence defined using the *Define Device Correspondence* form is persistent between Layout XL sessions, and can be preserved during *Generate All From Source* and *Update Components And Nets* by selecting the *Preserve User-Defined Bindings* option in the Generate of the Generate Layout form.

Important

Check Against Source does not report any issues found in user-defined bindings that are created by using the **Define Device Correspondence** form.

The form is dynamically updated to reflect the selections and changes you make. For performance reasons, the form does not update dynamically if you select more than 200 objects from the design canvas. To define device correspondence for a large number of objects, select them in the form itself.

Field	Description
Filter	Lists the devices in the *Schematic* and the *Layout* trees. Choose between *All Instances*, *All Terminals*, *Unbound Instances*, and *Unbound Terminals*. ■ *All Terminals* and *Unbound Terminals*: Displays the terminals in the design, not the pin figures that typically represent them in the layout window. ■ *Unbound Instances:* Displays only the unbound instances in the *Filter* cyclic field. If you delete a pin figure in the layout window, the corresponding terminal is removed from the *Define Device Correspondence* form only if the net connected to this terminal has no other connections. If the net has other connections, the terminal is not removed. Similarly, if you delete an instance, the terminals connected to the instance terminal's nets are deleted only if they are no longer connected in the design.

© 2023 Cadence Design Systems, Inc.

Field	Description
Schematic	Lists all the devices in the schematic view apart from those that are ignored for generation. An object or pin icon next to the device name indicates that the device is bound to a device in the layout. A red check mark indicates that it is unbound. If you select a bound device, the corresponding devices in the *Layout* pane is also selected.
Layout	Lists all the devices in the layout view apart from those that are ignored for generation. An object or pin icon next to the device name indicates that the device is bound to a device in the schematic. A red check mark indicates that it is unbound. If you select a bound device, the corresponding device in the *Schematic* pane are also selected.
Update Layout Instance	The *Update Layout Instance* options are available only when defining a one-to-one or one-to-many device correspondence. When you have made your selections, use the *Bind*, *Unbind*, and *Deselect All* buttons to perform the required action.
Name	Updates the layout instance name to match the corresponding schematic instance. Environment variable: bindUpdateName
Parameters	Updates the layout instance parameters to match the corresponding schematic instance.
Bind	Creates a correspondence between the devices selected in the *Schematic* pane and the devices selected in the *Layout* pane. This button is enabled only if all the selected devices are unbound.
Unbind	Deletes the existing correspondence between bound devices selected in the *Schematic* and *Layout* panes. The devices that are unbound have all their instance terminal connectivity removed.
Deselect All	Clears the selected set in both the panes.

Related Topics

Device Correspondence Updates

© 2023 Cadence Design Systems, Inc.

Showing Best Matches

Rapid Adoption Kit (*Working with the Binder to stay XL compliant*)

Defining a One-to-One Device Correspondence

Defining a One-to-Many Device Correspondence

Defining a Many-to-One Device Correspondence

Defining a Many-to-Many Device Correspondence

Deleting a Device Correspondence

Define Device Correspondence SKILL Function

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Design Summary Form

Use the Design Summary form to specify the types of statistics you want to view in the design summary report of the current cellview.

Field	Description
Select All	Selects all the options in the Design Summary form. When all options are selected, the design summary report displays the basic summary for the current cellview and the statistics based on nets, signal types, and layers.
Basic Summary	Displays the basic summary for the current cellview, which includes details such as statistics related to layer objects, routing grids, and instances.
Net Statistics	Displays in the design summary the number of terminals, instance terminals, guides, and steiners for each net in the current cellview. For each constituent figure of a net, the report lists the layers, wire length, horizontal and vertical wire length, vias, routes, and wire segments.
Connectivity Statistics	Displays in the design summary the connectivity statistics for the current cellview based on signal types, such as signal, power, ground, and clock.
Routing Statistics	Displays in the design summary the routing statistics for the current cellview based on layers.

Related Topics

Summary Window

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Edit Net Attributes Form

Use the Edit Net Attributes form to modify the criticality and signal types of existing nets in the design.

Field	Description
Net Name	Shows you which net you are currently editing. If you select multiple nets, use the *Previous* and *Next* buttons to cycle through the selected nets
Common	Specifies that the current settings are to be applied to all the selected nets. When switched off, the settings are applied only to the net in the *Net Name* field.
Criticality	Sets the criticality of the specified net (or all nets). The default value is `10`. The specified value must be in the range `-128` to `128`.
Signal Type	Sets the signal type of the specified net. Choose one from *signal, supply, ground, clock, tieOff, tieLo, tieHi, analog, scan* and *reset*.

Related Topics

Editing Net Attributes and Properties

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Edit Soft Blocks Form

Use the Edit Soft Blocks form to modify the type of soft block that will be created; the shape and size of the PR boundary; the shape, size, and signal type of soft pins; and the routing and placement obstructions inside the block.

The form includes certain common options, which you can use to control information display. The following table describes the common options.

Field	Description
Soft Block List	This section lets you select the soft blocks you want to edit. A soft blocks that is selected in the layout window when you opened the form is highlighted. If there were no soft blocks selected, the first block in the list is highlighted.
Select All	Selects all the soft blocks in the design.
Update Selected Soft Block Attributes	This section lets you to modify the attributes of the selected soft blocks. This list represents only selected soft blocks unlike the *Soft Block List*, which contains all the soft blocks present in the design. This list is affected by modifications to the *Soft Block List* section.
Library Name, *Cell Name*, *View Name*, *Cell Type*	You can modify soft blocks individually. The *Library Name*, *Cell Name*, *View Name*, *Cell Type*, and *Block Type* fields display the soft block being modified. You can also use Common mode to modify the attributes of multiple soft blocks simultaneously. In this case the *Library Name*, *Cell Name, View Name*, and *Cell Type* fields are grayed out.
Block Type	Specifies whether the block is *custom* or *digital*.

The Edit Properties form contains the following tabs.

Tab	Description
Boundary	Lets you update the place and route boundary of the selected soft blocks.
I/O Pins	Lets you update the layer, width, height, number, criticality, and signal type of existing pins.

© 2023 Cadence Design Systems, Inc.

Obstructions	Lets you change routing and placement obstructions implemented using layer and placement halos.

Related Topics

Soft Block Editing

Changing the Block Type

Changing the Boundary of Soft Blocks

Changing I/O Pin Attributes

Changing Obstructions

Layout XL Forms

Boundary

The *Boundary* tab has two modes: Fixed and Non-fixed.

The table below lists the valid combinations of the options in fixed mode.

Aspect ratio (W/H)	*Boundary Height*
Aspect ratio (W/H)	*Boundary Width*
Boundary Width	*Boundary Height*

Non-fixed mode is where you have not specified the width and height of the blocks and you want to estimate the area. The table below lists the valid combinations in non-fixed mode.

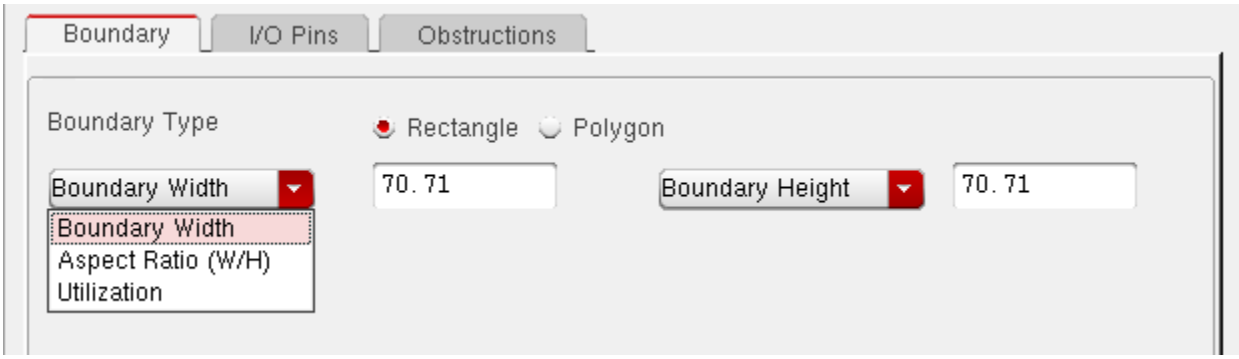
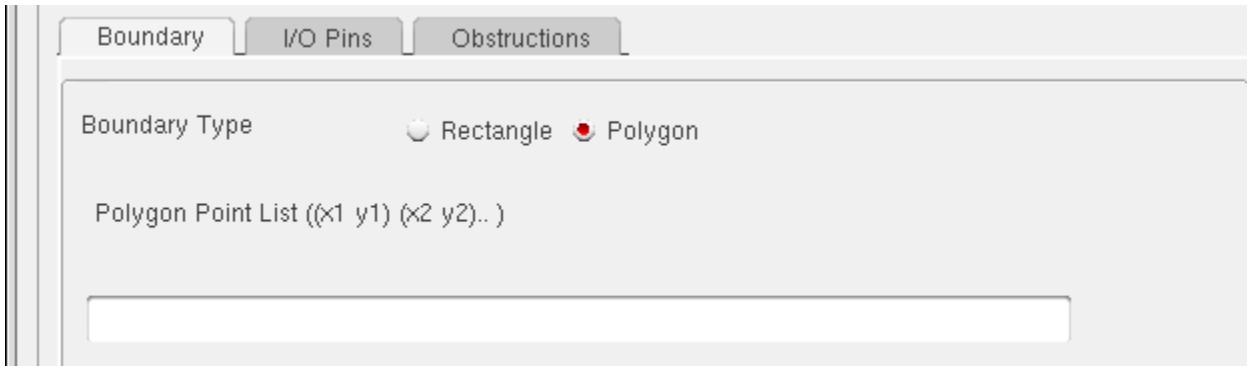
Utilization	*Height*
Utilization	*Width*
Utilization	*Aspect ratio (W/H)*

In non-fixed mode, the *Area Calculation* section becomes available, where you can select how the area of the place and route boundary is calculated. The methods available depend on the cell type of the soft blocks.

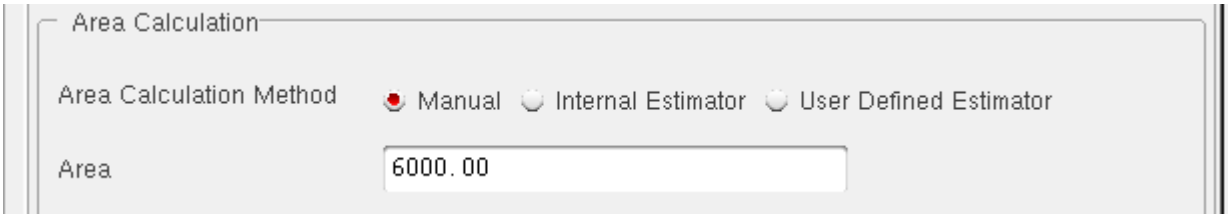
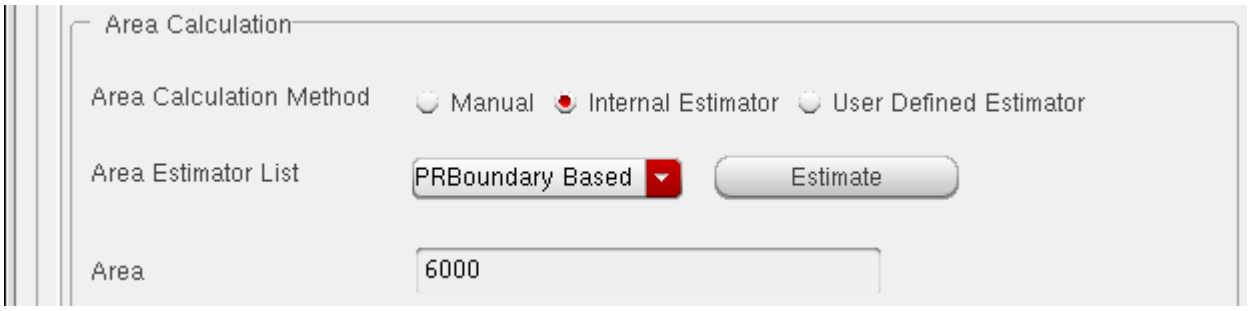
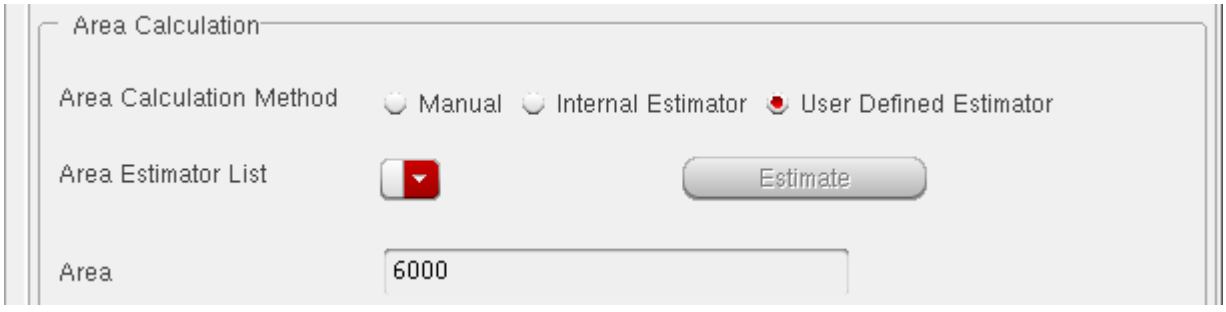
© 2023 Cadence Design Systems, Inc.

If the block has type *softMacro*, you can use the *Manual*, *Internal Estimator*, or *User Defined Estimator* methods; if the block has type *blockBlackBox*, you can use the *Manual*, *Area Per Gate*, or *User Defined Estimator* methods.

The following table describes the fields available on the Boundary tab of the Edit Soft Blocks form:

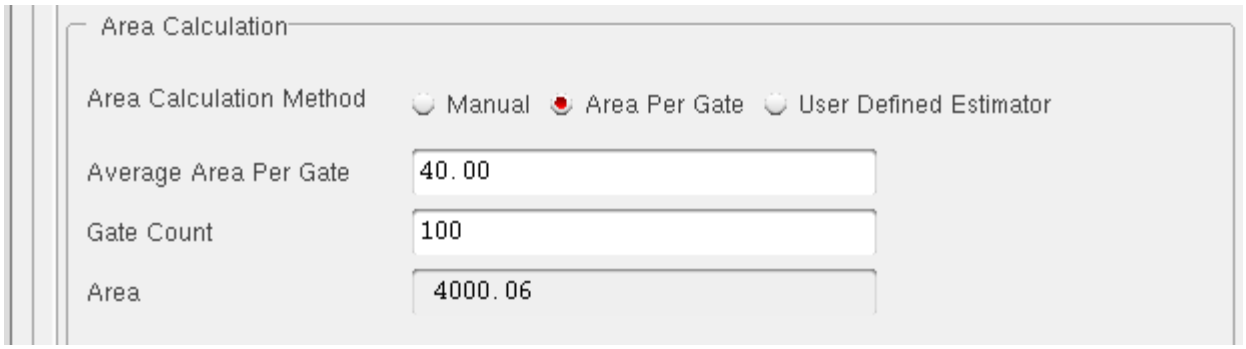
Field	Description
Rectangle	Lets you create a rectangular boundary. Use the two cyclic fields below the *Rectangle* radio button to specify the dimensions of a soft block.
	
Polygon	Lets you specify the points of a polygonal boundary.
	
Manual	Lets you specify the area in the field provided. Use this method if you already know the area of the soft block. Type the area into the field provided.

© 2023 Cadence Design Systems, Inc.

Field	Description
	
Internal Estimator	Calculates the area based on either the place and route boundary or the bounding box of the components contained in the soft block. Choose either *PRBoundary Based* or *BBox Based* from the cyclic field and click *Estimate* to calculate the area.
	
User Defined Estimator	Lets you specify your own area calculation function using the framework. See Creating and Registering a User-Defined Area Estimation Function. The picture below shows no user-defined function registered.
	

© 2023 Cadence Design Systems, Inc.

Field	Description
Area Per Gate	Lets you enter the area per gate and the gate count for a digital block. The system calculates the area based on the following formula. Area = Area Per Gate * Gate Count



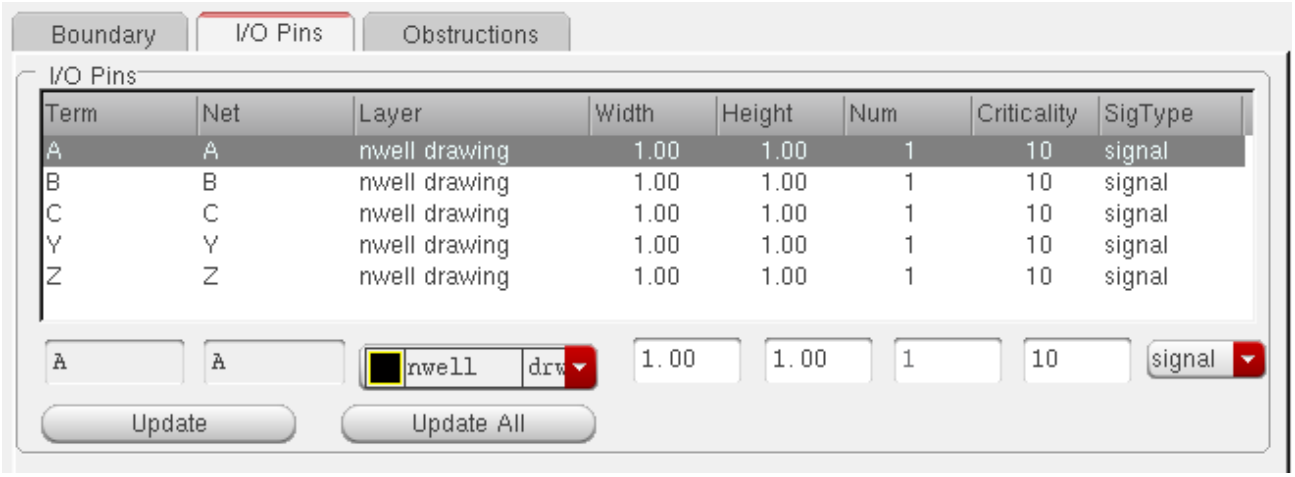
Related Topics

Changing the Boundary of Soft Blocks

Edit Soft Blocks Form

I/O Pins

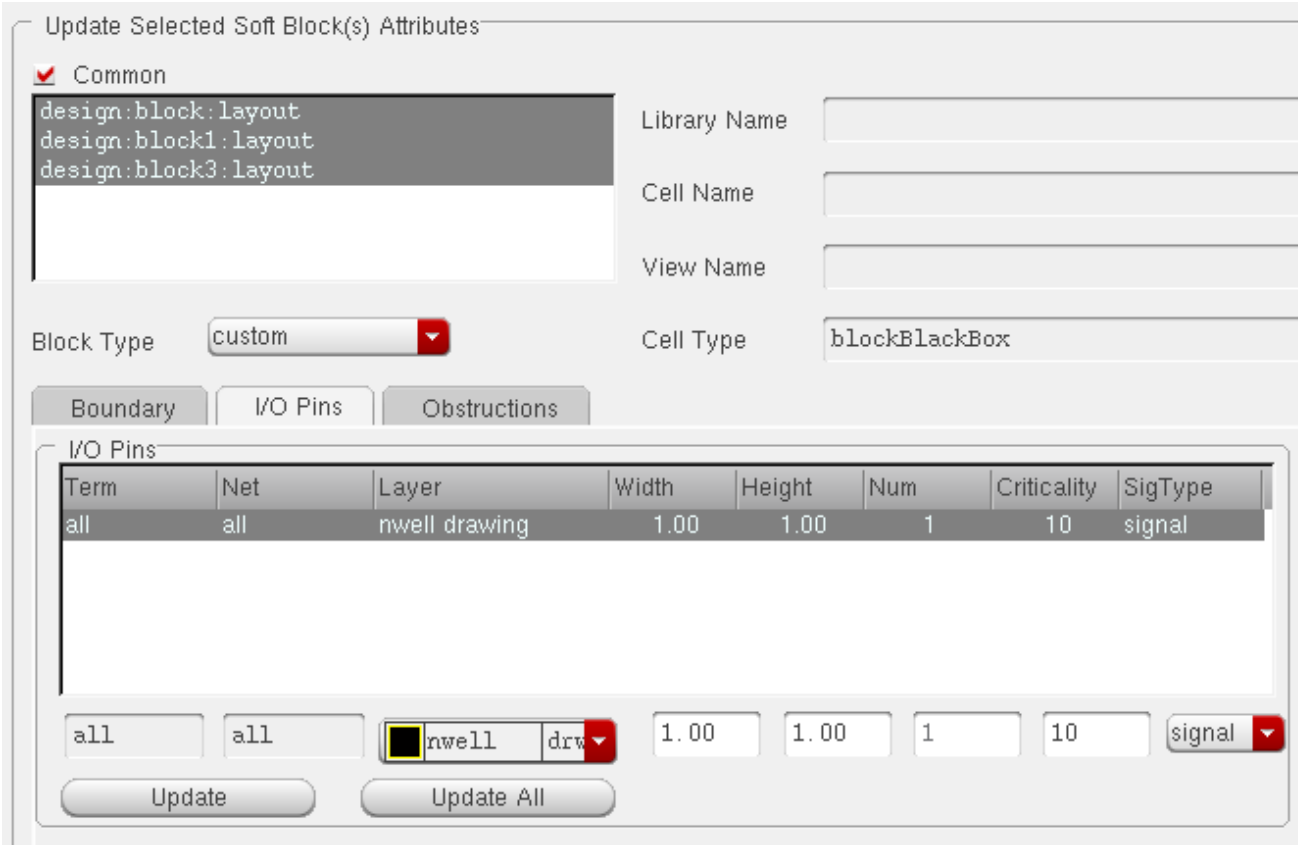
This table is initialized with *Term* and *Net names* from the symbol view of the selected soft block.



© 2023 Cadence Design Systems, Inc.

Change the parameters as required and use the *Update* button to update the selected pins, or use the *Update All* button to update all the pins in the block with the current settings.

If you select multiple blocks the *I/O Pins* tab changes as shown below.



Change the parameters as required and use the *Update* button to update all the pins in all the selected blocks.

Related Topics

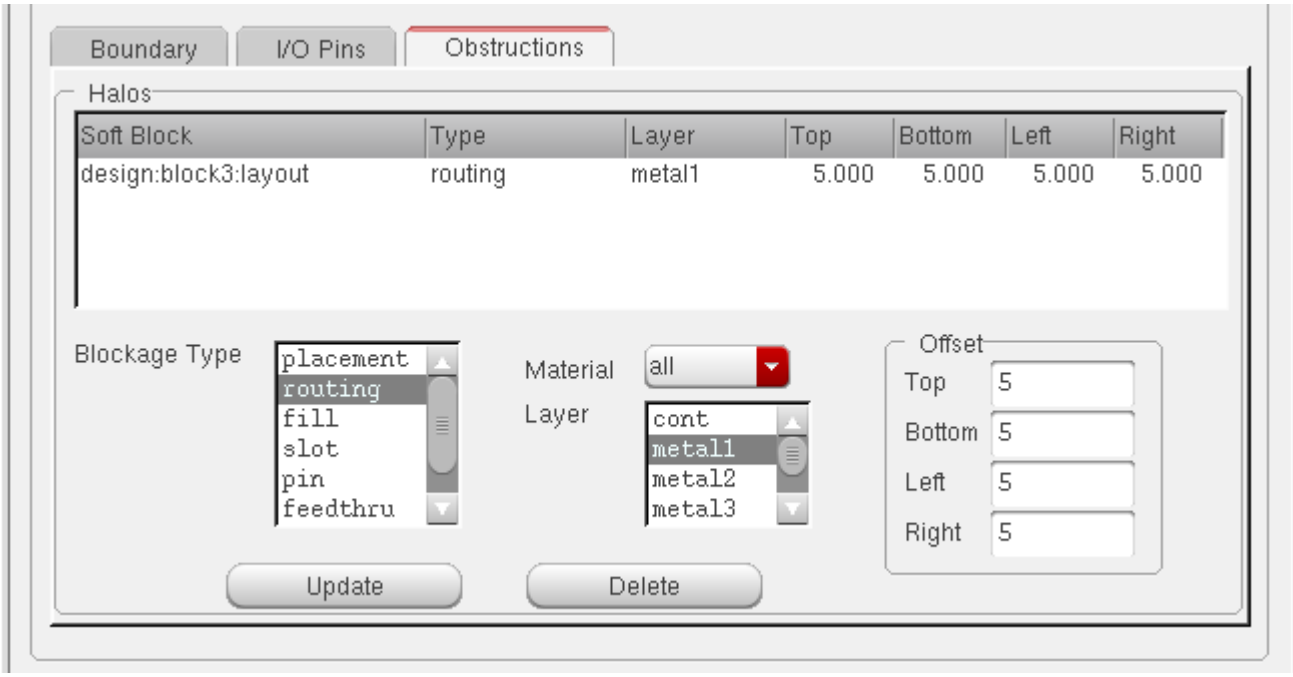
Changing I/O Pin Attributes

Edit Soft Blocks Form

© 2023 Cadence Design Systems, Inc.

Obstructions

The following table describes the fields available on the *Obstructions* tab of the Edit Soft Blocks form:



Field	Description
Halos	Lists the halos or obstructions in the selected soft blocks.
Blockage Type	Lists the various types of halos that you can create.
Material	Filters the layers based on the functions defined in the technology file.
Layer	Lists the layers in the technology file that you can use to create halos.
Offset	Lets you specify the top, bottom, left, and right offsets for the halos. To define a layer halo, you need to set the *Blockage Type*, the *Layer* and *Offsets*. There is also a *Material* filter which you can use to select layers based on the function defined in the technology file. To define a placement halo, you need set only the *Offsets*.

© 2023 Cadence Design Systems, Inc.

Related Topics

Changing Obstructions

Edit Soft Blocks Form

© 2023 Cadence Design Systems, Inc.

Extract Layout Form

Use the Extract Layout form to run connectivity extraction on the layout design.

Field	Description
Options	Opens the *Connectivity* tab of the Connectivity form, where you can set the extraction parameters for the design.
Scope	This section lets you choose how much of the design is extracted.
Current Cellview	Extracts only the top level of the current design. Layout XL also supports extraction of read-only cellviews.
Current Cellview and Cellviews in Hierarchy	Extracts the selected cellviews – including real-only cellviews – in the hierarchy along with the current top-level cellview. The extraction is performed bottom-up, starting with the cellview lowest in the hierarchy and ending with the top-level cellview. You can select the cellviews to extract in the Cellviews in Hierarchy pane.
Area	Specifies the area of the design to be extracted. Specify the coordinates of the bounding box of the area to be extracted in the *Bounding Box* field, either by typing the coordinates directly into the field or by clicking *Select Area* and drawing the bounding box on the canvas.
Unverified Areas	Extracts connectivity only from areas that were not interactively verified by the connectivity extractor.
Nets	Extracts connectivity from the nets specified in the adjacent text field. The net names typed in the text field must be space-separated.
Selected Nets	Extracts connectivity from all the selected nets. When *Selected Nets* is selected, any net names specified for extraction are ignored. Only those nets that are selected are extracted, irrespective of their being specified for extraction in the *Nets* field.
Report Type	This section lets you choose the scope of the extraction report to display.
None	Displays no report.
Simple	Displays a basic extraction report indicating the number of opens, shorts, and violations.

© 2023 Cadence Design Systems, Inc.

Field	Description
Full	Displays a detailed extraction report covering details about the extracted cellview, extracted nets, types of annotations displayed and the time taken to extract each value. In addition, the report displays the basic information displayed in a simple report.
Cellviews in Hierarchy	This section lets you specify which hierarchical cellviews are extracted. The controls in the *Cellviews in Hierarchy* group box are enabled only when the extraction scope is set to *Current Cellview and Cellviews in Hierarchy*.
From Selected Instances	Displays only those hierarchical cellviews that belong to the selected instances from the current, edited cellview.
Edited Only	Displays only those hierarchical cellviews that belong to the cellviews that have been edited since the last extraction run.
Search Depth	Specifies how far down the hierarchy to look for the cells to extract.
Select By	Lets you filter the entries in the list using the library, cell, or view names. Type in the first characters of the names you want to select. The filter mechanism has no wildcard capability. To select a cell named ‘`resistor`’, you must type ‘`r`’, ‘`re`’, ‘`res`’, and so on. If you type ‘`sis`’, the cell is not selected. You can click *Select All* to select all the cellviews listed or click *Deselect All* to deselect all the cellviews listed.
Save Extracted Cellviews	Saves the extracted cellviews automatically.

Related Topics

Connectivity Extraction Use Model

Extracting a Top-Level Design

Extracting Hierarchical Cellviews

Extracting Connectivity by Area

Connectivity

© 2023 Cadence Design Systems, Inc.

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Extractor Diagnostics Options Form

Use the Extractor Diagnostics Options form to control the output of the extractor diagnostics report.

Field	Description
Output Control	This section lets you control the output of the Extractor diagnostics report.
Display report in a separate window	Prints the report in a separate Info window, otherwise the report is printed in the CIW. The report contains the following information: ■ Details about the extractable layers, equivalent layers, and other similar extraction-related information generated by the `lcePrintExtractLayers()` SKILL function. ■ Details about the standard via and custom via definitions generated by the `lcePrintExtractVias()` SKILL function. ■ Warning messages generated by the extractor to indicate errors in the technology file setup for the selected constraint group. ■ Details about the options selected on the *Connectivity* tab that determine the way the selected constraint group is parsed for extraction. For example, depending on the options selected on the *Connectivity Assignment* tab, the extractor may parse the selected constraint group to verify connectivity from stamp labels and to substrate or well layers.
Log file name	Specifies the name of a file in which all the extractor diagnostics information is written. You can use the browse button to locate an existing file.
Append to log file	Appends the results of the current run to the specified log file instead of overwriting it.
Do not show me this dialog again	Lets you re-run the extractor diagnostics report with the current settings without the dialog being displayed.

© 2023 Cadence Design Systems, Inc.

The details about the various sections in the Diagnostics Report and the information these sections provide about the extracted design is tabulated below:

Section	Purpose
Extractable Layers	Lists the physical layers that are selected for extraction using the `validLayers` constraint. See validLayers.
Equivalent Layers	Lists the layers that have been defined as electrically equivalent. See Connectivity Rules.
Electrically-Connected Layers	Lists the layers that are electrically-connected.
Bulk Area Layers	Lists the layers that can be used to define the bulk area of shapeless terminals.
Stop Layers	Lists the layers that are defined as “stopped”.
Stamp_Label Layers	Lists the layers that can be used to create labels that stamp other layers. See stampLabelLayers.
Standard Via Definitions	Lists the valid standard via definitions that have been parsed by the extractor.
Custom Via Definitions	Lists the valid custom via definitions that have been parsed by the extractor.
Derived Layers in validLayers' constraint	Lists the derived layers that have been parsed by the extractor for the ‘`validLayers`’ constraint.
Excluded LPPs	Lists the layer purpose pairs that are not recognized by the extractor.

Related Topics

Controls tab (in the *Connectivity* tab of the Connectivity form)

Extract Layout Form

Via Definitions

Derived Layers

© 2023 Cadence Design Systems, Inc.

validLayers – Bulk Area Layers Specification

validLayers – Stop Layers Specification

Bulk Area Connectivity Extraction

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Generate Chained Devices Form

Use the Generate Chained Devices form to select and chain specific devices in your design. Select the devices you want to chain before launching the form. If the majority of the selected devices are oriented vertically, the generated chain is also oriented vertically. The devices to be chained need not be bound to a schematic instance.

Field	Description
Preserve Existing Chains	Ensures that any existing chains in the design are not broken during interactive chaining. Environment variable: chainPreserveExistingChains
Use Device Order	Maintains the relative horizontal starting positions of the specified instances when forming the chain. Instances are sorted by the x-coordinates of their origins (from lowest to highest) and the resultant list used to abut the devices from right to left. This argument does not always preserve device orientations; for example, it may be necessary to mirror an instance to abut it with its neighbor. If *Use Device Order* is selected, abutment is supported for instances that have different values set for the abutClass property but have their abutClass values defined as equivalent. Dummy, unconnected, and part-connected devices are chained, where possible. The device nets are assigned to enable device abutment in the order defined by *Use Device Order*. If *Use Device Order* is deselected, these equivalentAbutClasses are ignored. Dummy and unconnected devices are not chained. These devices are placed down separately in the order: chains, dummies, and then unconnected devices. Part-connected devices are chained, where possible, with nets assigned to enable device abutment. If *Use Device Order* is selected, advanced node devices are chained even if their S/D pins are on different nets. In this case, the devices are abutted on dummy pins. Environment variable: chainUseDeviceOrder

© 2023 Cadence Design Systems, Inc.

Field	Description
Extend Selection	Always selects the entire chain when you click an individual instance, mfactored device, or folded device within the chain. When switched off, you can selected individual devices, mfactors, or folded legs within a chain without selecting the entire chain. Environment variable: chainExtendSelection
Interdigitate Chains	Automatically identifies nodes that qualify as pseudoparallel connections and defines them during chaining and abutment. A pseudoparallel net connects nodes that are always the same voltage, so current does not pass through the net. Environment variable: lxAllowPseudoParallelNets
Mirror	Automatically mirrors the instances in a chain. Environment variables: chainMirror, chainPermutePins
Permute	Automatically permutes the instances in a chain. Environment variables: chainPermute, chainPermutePins
Create In Place	Creates the device chains on layout canvas at the same location as the left-most instance in a row region, if at least one of the instances selected for chaining is on a row region. If none of the selected instances are on a row region, the chain is created at the same location as the left-most selected instance. If the original instance is at an orientation, such as `R90`, that can place the chain vertically, the *Create In Place* option can be used to create the new chain above or below the original instance. Choose between *Increasing* or *Decreasing* to specify how the new chain is positioned. Environment variables: chainInPlace, chainInPlaceDirection

© 2023 Cadence Design Systems, Inc.

Field	Description
Align PMOS	Controls the alignment of PMOS chains. The default alignment for PMOS chains is *Top* but you can choose to align a chain to *Center* or *Bottom*. Environment variable: lxChainAlignPMOS If you are using a `FinFET` design, the option name will be displayed as *Align PFIN*, which can be used to control the alignment of `PFIN` device chains. Depending on the `PDK` that you are using, the abutment function being used might trigger a different alignment compared to the alignment that you specify. In this case, the user-defined alignment is disregarded and a message indicating the same is issued.
Align NMOS	Controls the alignment of NMOS chains. The default alignment for NMOS chains is *Bottom* but you can choose to align a chain to *Top* or *Center*. Environment variable: lxChainAlignNMOS If you are using a `FinFET` design, the option name will be displayed as *Align NFIN*, which can be used to control the alignment of `NFIN` device chains. Depending on the `PDK` that you are using, the abutment function being used might trigger a different alignment compared to the alignment that you specify. In this case, the user-defined alignment is disregarded and a message indicating the same is issued.
Show Device List	Expands the form to list the transistors available for chaining. If there is nothing preselected in the layout canvas, the form lists all the transistors in the design. If there were transistors selected in the layout canvas before you choose the *Generate Chained Devices* command, only the selected transistors are listed as highlighted. Select the transistors to chain from the list and click *Apply* to chain them.

Related Topics

Manual Device Abutment

© 2023 Cadence Design Systems, Inc.

Chaining Devices Interactively

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Generate Clones Form

Use the Generate Clones form to replicate the section of the layout that is associated with a section of the schematic in such a way that the new piece of layout material can be placed at more than one location, with each part preserving the hierarchical structure of the design.

Field	Description
Clone Source	This section shows the schematic and layout devices to be used as the clone source. You must specify a clone source to enable the other sections of the form. Only bound instances are recognized as clone sources by the cloning engine.
Layout Cellview	Lists the layout cellviews opened in Layout XL that have a connectivity reference available, including those opened in read-only mode**.**
Schematic Instances	Shows the names of the schematic clone source devices that are bound to the instances in the layout clone source.
Layout Instances	Shows the names of the layout clone source instances to be cloned. If instances and shapes are already selected in the current layout cellview when you open the Generate Clones form, and if the selection forms a valid clone source, the fields are already filled in when the form opens.
Update From Selected	Lets you use the source components from another open layout cellview, change the source *Layout Cellview* to the appropriate cellview, select the clone source in this cellview, when you click *Update From Selected*.
Search	This section lets you control the scope and criteria used when searching for target structures that match the clone source structure. Set the options you want and click *Search* to find potential clones. ■ *In Schematic*: Searches the entire schematic to find matching target structures. If *Selected Set Only* is enabled, the cloning engine searches for clone targets only within the selected set in the schematic. ■ *In Layout*: Searches the layout cellview for matching target structures that exist outside the PR Boundary. Environment variable: cloningSearchIn

© 2023 Cadence Design Systems, Inc.

Field	Description
Selected Set Only	Defines the scope of the clone search. ■ When searching in schematic, if *Selected Set Only* is enabled, the cloning engine searches for clone targets only within the selected set in the schematic. ■ When searching in layout: ❑ If *Selected Set Only* is enabled, the cloning engine searches for clone targets only within the selected set in the layout. ❑ If *Selected Set Only* is disabled, all the layout instances are searched, excluding those already placed within the PR Boundary. Environment variable: cloningSearchInSelSetOnly
Constraint Only	Searches for matching target structures only within the other clone definitions that belong to the same Clone Family constraint as the source. The option is enabled only when the clone source belongs to a Clone Def. The Clone Defs found during the search are ordered in the same manner as they exist in the constraint. See Constraint Only. Environment variable: cloningSearchInConstraintOnly
With Allowed Permutation	Considers pin permutability when searching for matching target structures. If an appropriate permute rule is defined for the device in question, and if permuting the pins on a device results in a match, then that match is reported (but only if no matching target structures were found without permutation). Related Property: permuteRule Property

© 2023 Cadence Design Systems, Inc.

Field	Description
With Exact Parameters	Requires that the parameter values on the components in a target structure match the parameter values on the components in the clone source exactly. When switched off, Layout XL also reports target structures that have different parameters and values. If required, you can update the parameters and values when you generate a clone by checking the *Update Layout Parameters to Match Schematic* option. To prevent any layout-only parameters to be updated, add them to the parameters to be ignored list. See Exact and Non-Exact Matches for Cloning. Environment variable: cloningDoExactMatch

© 2023 Cadence Design Systems, Inc.

Field	Description
With Exact Connectivity	Requires that the connectivity of the components in a target structure matches the connectivity of the components in the clone source exactly. When switched off, Layout XL also reports target structures where the set of instances is the same, but they are connected differently. Searching for non-exact connectivity matches takes significantly longer than searching for exact matches. **Note:** For large designs, Cadence recommends that you create a clone constraint in the schematic so that the clone mapping is done inside the constraint and there is no need to run the search in the cloning interface. Turn off *Exact Connectivity* and turn on the *Constraint Only* option.

© 2023 Cadence Design Systems, Inc.

Field	Description
More Options	Opens the Non-Exact Connectivity Matches form, where you can control the quality and number of non-exact connectivity matches reported. ■ *Maximum partial net matches:* Sets the maximum number of source nets that can be unmatched in a target structure. If a target structure has more than the specified number of unmatched nets, it is not reported. If you set the value too high, it might prevent the software from finding better target structures with fewer partial nets. To avoid this, either lower the value or limit the search to a selected set of instances. ■ *Allowed partial net names:* Specifies the names of source nets that are permitted to be unmatched in the target structure. You can use regular expressions to specify multiple net names with common elements. If a source net name is not listed (or does not match one of the regular expressions specified), then the net must be matched in the target structure exactly. ■ *Maximum found matches:* Sets the maximum number of non-exact target matches to be reported.

© 2023 Cadence Design Systems, Inc.

Field	Description
Create Options	This section lets you specify the types of clones that are generated from the target structures found. This section is enabled only if there are target structures reported in the *Clones Found* pane.
Create Clones as	This drop-down menu lets you specify the type of clone source. ■ *Synchronized Family*: Generates the clone source and each individual clone as a group in the layout view and links all the groups together as members of the same synchronized family. Each group contains all the physical shapes, vias, instances, and groups in the layout implementation of the respective clone. If you make a change to one member of a synchronized family, all the other members are automatically changed in the same way. ■ *Grouped Objects*: Generates the clone source and each individual clone as a group in the layout view but does not link the groups to each other in a synchronized family. Each group contains all the physical shapes, vias, instances, and groups in the layout implementation of the respective clone. Clones that are generated as groups are constrained in the ways they can be moved and rotated. ■ *Free Objects*: Generates the clones as ungrouped objects in the layout view. ■ *Update Constraint*: Creates a new Clone Family constraint and the associated Clone Def in the Constraint Manager, based on the selected synchronous clone figGroups. If a Clone Family constraint already exists, *Update Constraint* either leaves the constraint as is, or adds a new Clone Def, if appropriate. Environment variable: cloningUpdateConstraint
Update Layout Parameters to Match Schematic	Automatically updates the parameters and parameter values on the devices in the generated clone to match their counterparts in the schematic clone target. When switched off, the parameters and values are taken from the layout clone source. Environment variable: cloneJustify

© 2023 Cadence Design Systems, Inc.

Field	Description
Display Draglines	Controls whether draglines are shown while generating clones. Draglines indicate connections from the pins of the object you are moving to pins of the nearest objects. For information on how to change the draglines, see the *Draglines* options on the *Display* tab of the Connectivity form. Environment variable: flightLineEnable
Place Clones as Array	Places the generated clones in an array arrangement based on the array settings specified using the Clone Array Settings form. See Clone Placement as an Array.
Clones Found	This section lets you list the target structures that can be generated as clones in the layout cellview. Select one of the structures and move your cursor into the layout canvas to generate a clone for that structure.
Create Group Array	Displays the Create Group Array form, where you can create a group array with the specified parameters.
Edit	Opens the Modify Correspondence Form, where you can change the correspondence between the instances in source and target structures before you generate a clone. ■ Rotate to rotate the clone through 90 degrees counterclockwise. ■ Flip Horizontal to mirror the clone about its y axis. ■ Flip Vertical to mirror the clone about its x axis.
Orientation	This section lets you change the orientation of the selected clone instance.
Rotate Right	Rotates the clone through 90 degrees clockwise.
Rotate Left	Rotates the clone through 90 degrees counterclockwise.
Flip Horizontal	Mirrors the clone about its y axis.
Flip Vertical	Mirrors the clone about its x axis.

© 2023 Cadence Design Systems, Inc.

Field	Description
Transform	This section lets you use the Transform toolbar buttons to specify the clone transformation (*Rotate Left, Rotate Right, Flip Vertical, or Flip Horizontal*). When a toolbar button is clicked, the resulting transform is shown in the *transform* combo box. The transformation can also be set directly in the *transform* combo box.
Select All	Selects all the clones displayed in the *Clones Found* list.
Justification	Lets you choose the alignment of the clone you want to generate before you place it in the layout view. There are five alignments to choose from: *Center, Bottom left, Bottom right, Top left, and Top right.*
Transform From	Lets you choose the reference for transformation. By default, the reference for transformation is the clone source. **Note:** When the *Place Clone as Array* check box is selected, the *Transform From* option has no effect.

Related Topics

Clone Array Settings Form

Clone Generation

Clone Source Selection and Target Search and Generation

Generating Synchronous Clones

Types of Clones

Mfactored Component Cloning

Generating a Family of Synchronous Clones

Update Constraint

Specifying Transform From Settings

Modify Correspondence Form

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Generate Folded Devices Form

Use the Generate Folded Devices form to control interactive folding. You can divide the following into two or more folds:

- an individual transistor or a chain of transistors
- an already folded transistor or a chain of folded transistors
- an individual FinFET device or a chain of devices
- a multi-fingered device or transistor.

See Folding a Multi-fingered Device to Create Split Fingers.

Generate Folded Devices form also enables dividing an individual FinFET device or a chain of devices into two or more folds. Folding lets you change the aspect ratios of the devices in your design while retaining the original orientation. The devices to be folded need not be bound to a schematic instance. If the disableFolding environment variable is set and the folding threshold for the component type is set to `0`, folding is disabled.

If FinFET devices are driven for folding by *Number of fins*, the Generate Folded Devices form displays *Transistor Fins.* If folding for the selected FinFET devices is driven by the *Width parameter*, the form displays *Transistor Width*.

Field	Description
Transistor Name	Shows the name of the transistor that will be folded. If you have selected more than one transistor, use the *Next* and *Previous* buttons to move through the selected set.
Transistor Width	Specifies the value of the width property of the selected device, indicating also whether the value is taken from the schematic or the layout view or is user-defined. When *Ignore mfactor* is ON, the value shown is the effective width in the schematic, which is the product of mfactor and schematic gate width. If required, you can type in a value to override the effective schematic width. Alternatively, you can use the bindkey *Ctrl + Scroll Up* or *Down* operation to increase or decrease the transistor width value. **Note:** The *Transistor Width* field is displayed when the devices selected for folding are MOS transistors or FinFET devices that have the *Width parameter* set.

© 2023 Cadence Design Systems, Inc.

Field	Description
Transistor Fins	Shows the value of the *Number of fins* parameter of the selected FinFET device, also indicating whether the value is taken from the schematic or the layout view. **Note:** The *Transistor Fins* field is displayed only for FinFET devices that have the *Number of fins* parameter set.

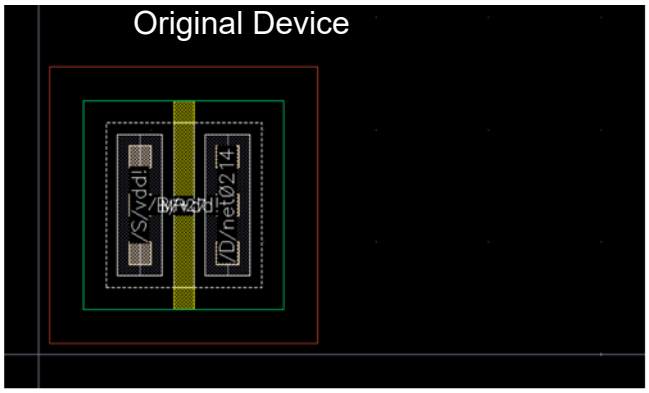
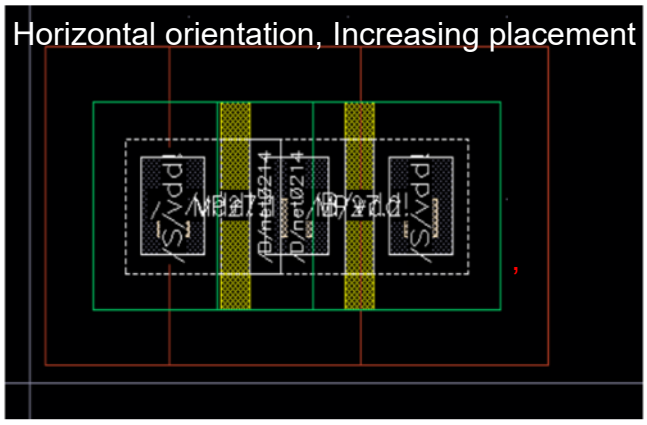
© 2023 Cadence Design Systems, Inc.

Field	Description
Number of Folds	Lets you type in the number of folds to be generated for the selected device, which may be already folded. Alternatively, use the bindkey *Shift + Scroll Up/Down* operation to increase or decrease the number of folds. When refolding, the number of generated folds may change based on the new number of folds you specify. Specifying a *Number of Folds* value greater than one displays the *Distribute* button, which evenly distributes the transistor width across the available folds. Alternatively, you can manually modify the width of each fold. **Note:** Such a distribution is also applicable for FinFET devices that are set for folding based on the *Width parameter.* For example, if the total device width is 3u, clicking *Distribute* splits the width evenly across two folds such that the width of each fold is 1.5u. If you specify multiple folds to be generated for the device, the width value displayed for each fold is controlled by the environment variable, lxGetSignifDigits. If the environment variable is not set, the width of each fold defaults to six significant digits. For example, if the transistor width is 4u and you specify that three folds be created, then depending on the state of the lxGetSignifDigits environment variable, the width of the folds varies as follows: ■ If the lxGetSignifDigits environment variable is set to four significant digits, the width of each fold is set to 1.333u. ■ If the lxGetSignifDigits environment variable is not set, the fold width defaults to 6 significant digits, resulting in a width of 1.33333u for each fold. **Note:** If you specify more than 500 folds to be added and click *Distribute*, a warning message displays prompting for confirmation to proceed with the folding. Environment variable: foldsLimit

© 2023 Cadence Design Systems, Inc.

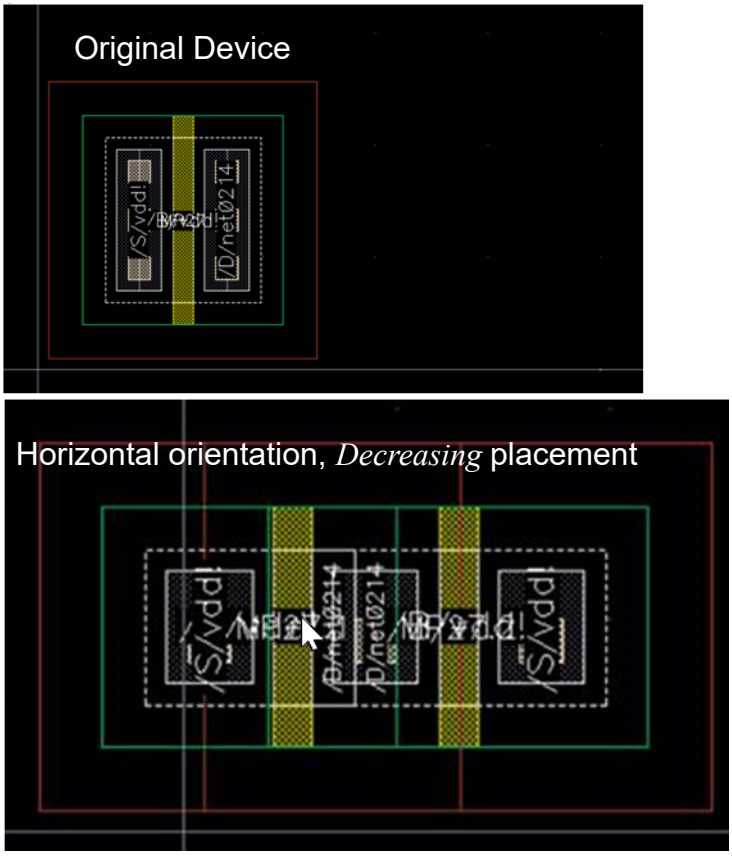
Field	Description
	Note: For FinFET devices, folding is driven by the *Number of fins* value. Clicking *Distribute* divides the total fins across all the folds such that each fold gets a whole number of fins. The *Fins* field accepts only integer values for fins. If you type in a non-integer value, Layout XL automatically reverts to the last integer value.
Use Threshold	Specifies if the *Folding Threshold* value should be used to fold the selected devices in the layout. When the option is selected, the *Number of Folds* value is calculated based on the transistor width or the folding threshold value. Otherwise, the *Number of Folds* value is specified by the user.
Folding Threshold	Specifies the maximum width beyond which the selected devices are automatically folded.
Ignore mfactor	Ignores the schematic multiplication factor when generating folded devices. The *Transistor Width* is set to the effective width in the schematic; that is, the product of mfactor and schematic gate width.
Chain Folds	Specifies Layout XL to chain the resultant folds together, where possible. The command deletes all existing folds except the ones that are selected. The first fold from the retained set is then used to create the new folds. Environment variable: chainFolds
Align	Controls the alignment of the device selected for folding. You can choose one of the following values: *Top*, *Center*, or *Bottom*. ■ If the selected device is a PMOS, the value of the *Align* field is the current value of the lxChainAlignPMOS environment variable. ■ If the selected device is an NMOS, the value of the *Align* field is the current value of the lxChainAlignNMOS environment variable.

© 2023 Cadence Design Systems, Inc.

Field	Description
Create In Place	Creates the device folds at the same location in the layout canvas as the original device in a row region when at least one of the selected instances is on a row region. If the original instance is at an orientation, such as R90, that can place the folds vertically, the *Create In Place* option can be used to create the new folds above or below the original instance. Choose between *Increasing* or *Decreasing* to specify how the new folds are positioned. The positioning of the folds also varies depending on the placement orientation - horizontal or vertical. ■ For horizontal orientation, the new folds start at the lower left of the selected row region when positioning the folds in an *Increasing* placement, as shown below:  

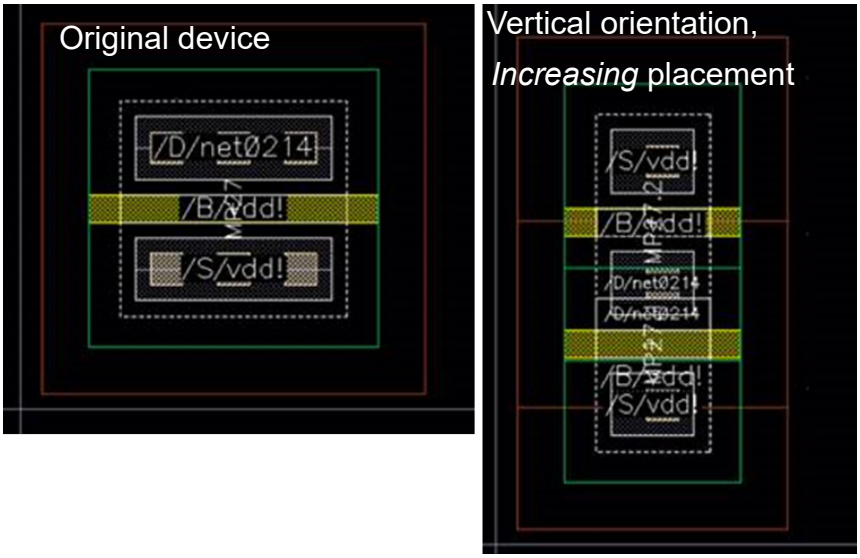
© 2023 Cadence Design Systems, Inc.

Field	Description
	■ For horizontal orientation, the new folds start at the lower right of the selected row region when positioning the folds in a *Decreasing* placement, as shown below:

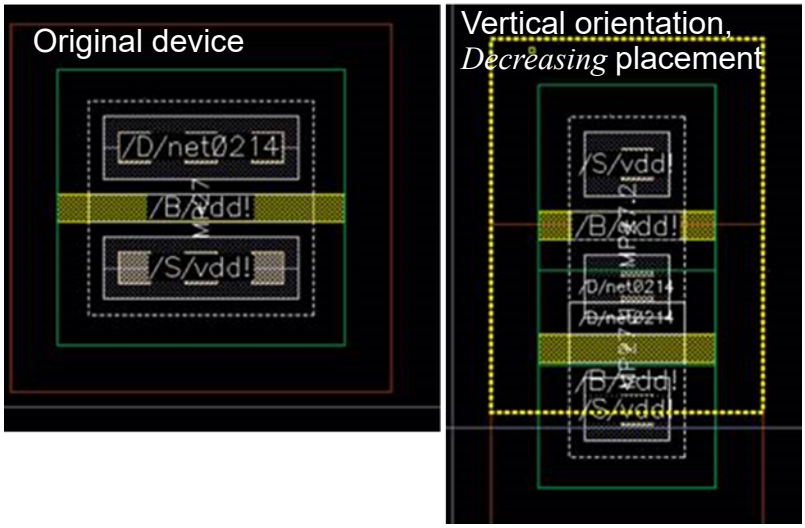


© 2023 Cadence Design Systems, Inc.

Field	Description
	■ For vertical orientation, the new folds start at the lower left of the selected row region when positioning the folds in an *Increasing* placement, as shown below:



■ For vertical orientation, the new folds start at the upper left of the selected row region when positioning the folds in a *Decreasing* placement, as shown below:



© 2023 Cadence Design Systems, Inc.

Field	Description
Total Width	Shows the total value of all the fold widths displayed in the *Width* fields at the bottom of the form. ■ *Add Fold*: Adds an additional fold to the device and provides a field where you can specify the *Width* for the new fold. ■ *Distribute*: Divides the total width of the transistor evenly by the number of folds and sets the resultant *Width* as the value for each fold. ■ *Delete*: Removes the corresponding fold from the device. If you set a width with no unit, the unit is automatically updated with the default value of the CDF width parameter. For example, if the default CDF value is `3u` and you enter a width of `6`, the width is set to `6u`. If the default CDF value is `3` and you enter a width of `6`, the width remains set to `6`. As in the case of individual fold widths, the value displayed in the *Total Width* field is controlled by the lxGetSignifDigits environment variable. If, for example, the environment variable is set to 4 significant digits, the value in the *Total Width* field displays up to four significant digits. If the environment variable is not set, the *Total Width* field displays up to six significant digits. Irrespective of the original width of the transistor, the *Total Width* field displays the sum of the width values of the various folds. For example, if the original width of the transistor is 4u and the sum of the fold widths is calculated at 3.999u, the *Total Width* field displays the width as 3.999u instead of rounding off the value to 4. If the difference between the *Total Width* and the original transistor width exceeds the tolerance value set by using the paramTolerance environment variable, a warning message is displayed indicating that the sum of the folded widths is not equal to the original transistor width.

© 2023 Cadence Design Systems, Inc.

Field	Description
	For FinFET devices that are set to fold based on *Number of fins*, the *Total Width* field is replaced by *Total Fins*. *Total Fins* is the sum of the fins in each fold of a FinFET device. The values considered for *Add Fold*, *Distribute*, and *Delete* are also specified in terms of number of fins.

Related Topics

Additional Information on Interactive Folding

Ignoring Schematic Mfactor during Folding

Generating All Components from Source

Placement Modes for Generating Selected Components from Source

Unit Identifiers in Layout XL

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Generate Layout Form

Use the Generate Layout form to generate layout representations of schematic design components.

The form is split into four tabs.

Tab	Description
Generate	Lets you select which design objects are generated in the layout view.
I/O Pins	Lets you specify default attribute values for pins, select and update the attribute values of pins generated in the layout, and specify the type of label generated when you create a pin.
PR Boundary	Lets you specify the shape of the place and route boundary, how system calculates the shape and size of a rectangular boundary, and how the area boundary of a virtual hierarchy is estimated.
Floorplan	Lets you choose the floorplanning objects to be preserved in the generated layout.

© 2023 Cadence Design Systems, Inc.

Generate

The following table describes the fields available on the *Generate* tab of the Generate Layout form.

Field	Description
Generate	This section lets you choose which design objects are generated in the layout view.
Instances	Lets you generate all the instances in the schematic that do not have one of the ignore properties attached to them. Environment variable: initCreateInstances ■ *Chain*: Abuts MOS transistors automatically into chains during layout generation. Environment variable: initDoStacking ■ *Fold*: Divides devices into folds automatically to prevent the gate width from exceeding a specified size. If the `disableFolding` environment variable is set and the folding threshold for the component type is set to `0`, folding is disabled. You can change the folding threshold using CPH Component Types mode or LAM. Environment variables: initDoFolding, disableFolding ■ *Chain Folds*: Automatically chains the individual folds of a transistor. For the *Chain Folds* option to be available, only the *Fold* check box must be selected. If you select the *Chain* check box as well, *Chain Folds* is deactivated. Environment variable: chainFolds

© 2023 Cadence Design Systems, Inc.

Field	Description
I/O Pins	Generates all the pins listed on the *I/O Pins* tab. The generated pins are automatically snapped to the placement grid. Environment variable: initCreatePins ■ *Except Global Pins*: Stops Layout XL generating layout pins for the global nets in the schematic. Environment variable: initGlobalNetPins ■ *Except Pad Pins*: Stops Layout XL from generating layout pins for schematic pins that are connected to I/O pads (cells of type `pad`, `padSpacer`, `padAreaIO`, or `coverBump`). When unchecked, the software generates pads, coverBumps, and pins. Environment variable: initCreatePadPins
PR Boundary	Lets you generate a PR boundary based on the settings on the *PR Boundary* tab. All placements and estimations are based on the generated PR boundary. Environment variable: initCreateBoundary ■ *Snap Boundary*: Generates a rectangular snap boundary that encloses the generated PR boundary. You can generate a snap boundary only if *PR Boundary* is selected. Environment variable: initCreateSnapBoundary
Position	This section lets you specify the positions of instances.
Minimum Separation	Positions the instances in the layout at a minimum separation based on the value you specify. You can also control this option by using the lxPositionMinSep environment variable. The minimum separation value is honored even if the lxGenerateInBoundary environment variable is set. Therefore, with lxGenerateInBoundary set, if the minimum separation value you specify is large, the instances may still get positioned outside of the design boundary. Environment variables: lxPositionMinSep, lxGenerateInBoundary, lxGenerateInBoundary.

© 2023 Cadence Design Systems, Inc.

Field	Description
In Boundary	Generates layout representations within the design boundary when the *Generate All From Source command is run.* Environment variable: lxGenerateInBoundary
Device Correspondence - Preserve User-Defined Bindings	Preserves user-defined bindings of devices between the schematic and the layout. This option preserves only user-defined one-to-one, many-to-many, many-to-one, and one-to-many device correspondence defined in the Define Device Correspondence Form. It does not report missing devices or shapes within a bound group. Environment variable: initCreateMTM
Connectivity Extraction - Extract Connectivity After Generation	Runs connectivity extraction as part of the layout generation process. Check this option to see the incomplete nets in the design immediately after layout generation has run.
Design Planning	This group box is available only when you have the Layout EXL or higher license checked out.
Virtual Hierarchy	Generates a virtual hierarchy using the area boundary options selected on the *PR Boundary* tab, or updates an existing virtual hierarchy to match the schematic hierarchy. When updating an existing virtual hierarchy, the area boundary of the virtual hierarchy is not updated. Environment variable: generateVirtualHierarchy
Auto Generate Soft Blocks	Generates soft blocks for top-level virtual hierarchy blocks that have no schematics available or have schematics with only pins but no instances or physical binding. For the virtual hierarchy blocks that have no schematic available, soft block generation uses the bound symbol view to generate pins. Soft block boundary is generated using the options defined on the *PR Boundary* tab. Environment variable: generateSoftBlocks

© 2023 Cadence Design Systems, Inc.

I/O Pins

The following table describes the fields available on the *I/O Pins* tab of the Generate Layout form.

Field	Description
Default Values	This section lets you specify attribute values and *Apply* them to all the pins shown in the list box.
Type	Specifies the type of pin. Valid values are *Electrical* and *Optical*. This option is available only when both electrical and photonic data is present in the schematic. The following options are available, if you choose pin type as *Electrical*.
Layer	Specifies the layer-purpose on which the pins are generated. The cyclic field lists only the conducting layers. You can use the initIOPinLayer environment variable to specify the layer-purpose pair that you want to use for generating the pins. The default is the current layer selected in the Layer Assistant, provided it is a valid layer. Else, it is the first extractable layer in the technology file that has a `pin` purpose. If an extractable layer with a `pin` purpose does not exist, the first extractable layer with a `drawing` purpose is selected as the default. If there are no extractable layers, the cyclic field lists all the valid layout layers. The layers are extracted from the validLayers constraint if they have a purpose defined. However, the purposes are not used to restrict the displayed layer-purpose pairs. To filter the purposes, use the initIOPinPurposeNames environment variable. Environment variables: initIOPinLayer, initIOPinPurposeNames, validLayers
Width	Specifies the width for each pin. The default is the `minWidth` value set for the current layer in the technology file. Any change to the value is applied only if the new value is greater than the default value. Environment variable: initPinWidth

© 2023 Cadence Design Systems, Inc.

Field	Description
Height	Specifies the height for each pin. The default is the `minWidth` value set for the current layer in the technology file. Any change to the value is applied only if the new value is greater than the default value. Environment variable: initPinHeight
Num	Specifies how many instances of this pin to generate. If you type 0, the pin is not generated. Environment variable: initPinMultiplicity
Apply	Applies the *Layer*, *Width*, *Height*, *Num*, and *Create* settings for all the listed pins. The following options are available, if you choose pin type as *Optical*. You need to check out the `Virtuoso_Multitech` and `Virtuoso_Photonics_Option` licenses to create optical pins for the design. (Virtuoso Photonics Option) You can specify the default pin values, such as *Width*, *Radius*, and *Input Angle* for the optical pins using the *I/O Pins* tab.
Layer	Specifies the layer-purpose on which the optical pins are generated. The default layer is the determined from the value set for the `phoPinLayer` environment variable. If no value is set for the environment variable, the first photonic layer in the `validLayer` list is used. Environment variable: phoPinLayer (Virtuoso Photonics Option)
Width	Specifies the width of the waveguide associated with the photonic port. Environment variable: phoPinWidth (Virtuoso Photonics Option)
Radius	Specifies the curvature of the waveguide associated with the photonic port. Environment variable: phoPinRadius (Virtuoso Photonics Option)

© 2023 Cadence Design Systems, Inc.

Field	Description
Input Angle	Sets the pin angle for all the input pins in the table. The other pin angles are then derived from this value. ■ Input pin - Sets to the specified value. ■ Output pin - Sets to the 180 degree complementary angle of the input angle. ■ All other pins - Not set and must be set individually by selecting the pin and using the field below the pin table. Environment variable: phoPinInputAngle (Virtuoso Photonics Option)
Apply	Applies the *Layer*, *Width*, *Radius*, and *Input Angle*, settings for all the listed optical pins.
Specify Pins To Be Generated	This section lets you select pins from the list box and update the attribute values used when those pins are generated in the layout. Click *Update* to update the *Layer*, *Width*, *Height*, and *Num* settings for the currently selected pins.
Term Name	Specifies the schematic terminal name. You cannot change this value.
Net Name	Specifies the net associated with the pin in the layout. You cannot change this value. If terminal and net names differ in the schematic, Layout XL generates a pin with the same name as the schematic terminal and a net with the same name as the net attached to the terminal in the schematic. If there is no explicit net label in the schematic, both the pin and the net names in the layout are the same as the schematic terminal. This is the default behavior.

© 2023 Cadence Design Systems, Inc.

Field	Description
Layer	Specifies the layer-purpose on which the pins are generated. The cyclic field offers only the conducting layers. You can use the initIOPinLayer environment variable to specify the layer-purpose pair that you want to use for generating the pins. The default is the current layer selected in the Layer Assistant, provided it is a valid layer. Else, it is the first extractable layer in the technology file that has a `pin` purpose. If an extractable layer with a `pin` purpose does not exist, the first extractable layer with a `drawing` purpose is selected as the default. If there are no extractable layers, the cyclic field lists all the valid layout layers. See validLayers.
Width	Specifies the width for each selected pin. The default is the `minWidth` value set for the current layer in the technology file. Any change to the value is applied only if the new value is greater than the default value. Environment variable: initPinWidth
Height	Specifies the height for each selected pin. The default is the `minWidth` value set for the current layer in the technology file. Any change to the value is applied only if the new value is greater than the default value. Environment variable: initPinHeight
Radius	Curvature of the waveguide associated with the photonic port. By default, the radius is not defined. (Virtuoso Photonics Option) This column is visible only when there are optical pins in the design.
Angle	Access direction of the photonic port. (Virtuoso Photonics Option) This column is visible only when there are optical pins in the design.

© 2023 Cadence Design Systems, Inc.

Field	Description
Num	Specifies how many instances of each selected pin are generated. If you type 0, the selected pin is not generated. To limit the number of pins that are displayed in the pin table, set generatePinsDisplayLimit to the required display limit. If the design includes more pins than the number allowed to display in the form, the additional pins are suppressed and the form is updated to mention the number of suppressed pins, as shown below. You can right-click the suppressed pin notification, and choose *Un-suppress* to display all the pins. Environment variable: initPinMultiplicity
Add New Pin	Opens the Add A New Pin Form dialog, where you can specify the name of a terminal for which to generate a new pin. To view this option, right-click in the pins table.
Pin Label	This section lets you specify the type of label generated when you create a pin. This setting is honored by the *Generate All From Source* and *Generate Selected From Source* commands.
Create Label As	Specifies whether a pin label is created when the pin is generated. You can choose to create either a *Label* object or a *Text Display*. Environment variable: initIOLabelType.
Options	Accesses the Set Pin Label Text Style Form, where you can set the size, font, style, justification and orientation of the label lettering, and the drawing or pin layer on which the labels are displayed.If your pin labels are not visible in the canvas, turn on the *Pin Names* option in the Display Options form. For this setting to take effect, you must also set the createPinLabel environment variable to `t`.

© 2023 Cadence Design Systems, Inc.

PR Boundary

The following table describes the fields available on the *PR Boundary* tab of the Generate Layout form.

Field	Description
Shape	This section lets you specify whether the place and route boundary is a rectangle or a polygon.
Rectangle	Specifies a rectangular place and route boundary. Use the *Area Estimation* group box to specify how the size of the boundary is calculated.
Origin	Specifies the coordinates of the boundary's origin. The default is (0.0 0.0).
Polygon	Specifies a polygonal boundary. Use the *Points List* to specify the coordinates of each of the vertices of the polygon.
Area Estimation	This section lets you specify how the system calculates the shape and size of a rectangular boundary. Area estimation comprises two parts: the first specifies the aspect ratio and utilization of the boundary; the second estimates the size of boundary required to accommodate the components to be generated. To specify the aspect ratio and utilization, set two of the following four parameters. Choose one of the parameters from the first cyclic list and any one of the remaining valid choices from the second cyclic list.
Width	Specifies the width of the design boundary. The default is the size of the last boundary or `10`.
Height	Specifies the height of the design boundary. The default is the size of the last boundary or `10`.
Utilization (%)	Specifies the percentage of area within the cell boundary that you want to fill. The default is `25`. Environment variable: initUtilization When the *Chain, Fold, or Chain Folds* options are switched on, the *Utilization* value is applied only after chaining and folding is complete so that the size of the boundary is calculated accurately.

© 2023 Cadence Design Systems, Inc.

Field	Description
Aspect Ratio (W/H)	Specifies the width-to-height ratio of the design boundary. A value of 1 specifies a square boundary; 0.5 specifies a boundary twice as high as it is wide; and 2 specifies a boundary twice as wide as it is high. The default is `1`. Environment variables: initAspectRatio and initAspectRatioOption
Estimator	Lets you choose the type of area estimator to be used. ■ *PR Boundary Based*: Calculates the area based on the sum of the areas enclosed by all the place and route boundaries of the components to be generated. Environment variable: initAreaFuncName ■ *BBox Based*: Calculates the area based on the sum of the areas enclosed by all the bounding boxes of the components to be generated. Environment variable: initAreaFuncName

© 2023 Cadence Design Systems, Inc.

Field	Description
Register	Opens the Add Area Estimators form, which lets you register a user-defined SKILL function to estimate the area enclosed by a PR boundary or by the area boundary of a virtual hierarchy block. ■ The SKILL function registered for estimating the PR boundary appears in the *Estimator* drop-down list and can be used to estimate the area of the top-level boundary when generating a layout. ■ The SKILL function registered for estimating the area boundary appears in the *Soft Block – Area* drop-down list and can be used to estimate the area of the virtual hierarchies when generating a layout. If you specified deferred when you registered the PR boundary area estimator and are now choosing to generate virtual hierarchies with area boundaries, the PR boundary estimation will be deferred until the area boundary of the virtual hierarchy is created. This allows the area estimation function to use the virtual hierarchy area boundary values to determine the area for the PR boundary. See Creating and Registering a User-Defined Area Estimation Function.
Virtual Hierarchy Area Boundary	This section lets you specify how the area boundary of a virtual hierarchy is estimated. This group box is available only when you have the Layout EXL or higher license checked out
Enclose by	Specifies the distance from the objects inside the virtual hierarchy at which the area boundary is created.
Utilization (%)	Specifies the acceptable area utilization percentage for deriving the size of the area boundary for the virtual hierarchy. Environment variable: areaBoundaryUtilization
Top Level	Specifies that the selected area boundary settings be used for generating virtual hierarchy area boundaries only at the top level.

© 2023 Cadence Design Systems, Inc.

Field	Description
All Levels	Specifies that the selected area boundary settings be used for generating virtual hierarchy area boundaries at all levels in the virtual hierarchy of the top-level design.
None	Specifies that the virtual hierarchies for the design are generated without an area boundary.
Soft Block	Specifies the area of a soft block to be created. This group box is available only when you have the Layout EXL or higher license checked out.
Area	Specifies the acceptable area value of the soft blocks to be created. The value specified using the *Area* field is applied by default to all the soft blocks that are generated. To override this value and to customize the area of each generated soft block, you can specify a cellview property area on the Symbol Generation Options form when creating a new symbol. Alternatively, you can update the property for an existing symbol using the Edit Cellview Properties form. In either case, a floating point value is specified for the area property. If a SKILL function has been registered to estimate the area of a virtual hierarchy, the function appears in the drop-down list associated with the *Area* field. The same area estimator function can be accessed using the Adjust Area Boundary form in the Virtuoso Design Planner. Environment variable: softBlockArea

Floorplan

The following table describes the fields available on the *Floorplan* tab of the Generate Layout form.

Field	Description
Preserve Floorplanning Objects	This section lets you choose the floorplanning objects to be preserved in the generated layout.
All	Selects all the items in the list.

© 2023 Cadence Design Systems, Inc.

Field	Description
None	Deselects all the items in the list, or selects one or more floorplanning objects to preserve them in the generated layout.
Rows and Custom Placement Areas	Preserves any existing rows and custom placement areas in the layout. Also preserves `rowRegionSpec` and row regions in the layout when *Generate All From Source* is run. Cellviews containing row regions open in Layout XL in read-only mode. For editing such cellviews, the layout must be opened using *Launch – Layout EAD*. Environment variable: preserveRows
Blockages	Applies only to standalone blockages that have no parent object; a blockage with a parent object is preserved automatically along with its parent. Environment variable: preserveBlockages
Area Boundaries	Preserves all existing area boundaries. Environment variable: preserveAreaBoundary
Track Patterns	Preserves all existing track patterns. Environment variable: preserveTrackPattern
Clusters	Preserves any existing clusters. Environment variable: preserveClusters
Cluster Boundaries	Preserves cluster boundaries only if clusters are also being preserved. Environment variable: preserveClusterBoundaries
PR Boundary	Preserves any existing PR boundaries. Environment variable: preservePRBoundary
Snap Boundary	Preserves snap boundaries only if PR boundaries are also being preserved. Environment variable: preserveSnapBoundary
All	Selects all the listed floorplanning objects and preserves them during layout generation.

© 2023 Cadence Design Systems, Inc.

Field	Description
None	Deselects all the listed floorplanning objects, marking them as not to be preserved during layout generation.

Related Topics

Add Area Estimators Form

Symbol Generation Options

Adjust Area Boundary

Edit Cellview Properties

Layout Generation

Generating All Components from Source

Rapid Adoption Kit (*Introduction to Connectivity-Driven Design in VLS XL*)

Add A New Pin Form

Set Pin Label Text Style Form

Generation of RF TLine Chains (Virtuoso MultiTech Framework)

Define Device Correspondence Form

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Generate Selected Components Form

Use the Generate Selected Components form to generate selected schematic instances and pins directly in the layout view.

Field	Description
Placement Mode	This section lets you specify the placement mode of schematic instances and components.
As In Schematic	Generates the selected schematic instances and pins together in the layout view. By default, the components are placed in the same relative positions as in the schematic. However, you can change the alignment, spacing, and orientation of the components by using the options in the form. You can chain and fold mfactored components, both generated and partially generated. You cannot change the instance properties or pin attributes in this mode.
Array	Generates the selected components as ungrouped objects, a Modgen, a mosaic, or a group array. Irrespective of their type, the arrays that you generate are XL-compliant. By default, the row-column spacing is DRC correct, but you can choose to modify the row, column, or spacing values for an array, as appropriate. For Modgens, mosaic,and group arrays, you can modify these values even after the array is generated. But, for instance arrays, if the row, column, or spacing values are modified, the array must be regenerated. Array placement mode cannot be used to place pins. Environment variable: generateModgens
Individually	Generates each of the selected schematic components one at a time in the layout view. When generating instances, the form shows the name and master of the instance currently being generated; when generating pins, the form shows the name of the pin currently being generated and lets you change the attributes of that pin. The *Chain* option is not available in *Individually* mode. To chain multiple devices, select the *Chain* option in *As In Schematic* mode or use the *Generate Chained Devices* command.

© 2023 Cadence Design Systems, Inc.

Field	Description
Chaining and Folding	This section lets you specify the options for chaining and folding of transistors.
Fold	Automatically divides MOS transistors into folds to optimize utilization. If the `disableFolding` environment variable is set and the folding threshold for the component type is set to `0`, folding is disabled. Environment variables: initDoFolding, disableFolding
Chain Folds	Automatically chains the individual folds of a transistor. For the *Chain Folds* option to be available, only the *Fold* check box must be selected. If you select the *Chain* check box as well, *Chain Folds* is disabled.
NMOS Fold Threshold	Specifies the maximum size of the folded NMOS transistor. This value defaults to the folding threshold value set on the connectivity source component. Changing this value overrides any folding threshold value defined in the component type.
PMOS Fold Threshold	Specifies the maximum size of the folded PMOS transistor. This value defaults to the folding threshold value set on the connectivity source component. Changing this value overrides any folding threshold value defined in the component type. The *NMOS Fold Threshold* and the *PMOS Fold Threshold* options are available only when the *Fold* check box is selected. The *NMOS Fold Threshold* and the *PMOS Fold Threshold* fields are disabled if the fold threshold value for all the component types is set to `0` and the disableFolding environment variable is set. Folding is prevented and a warning issued if the `disableFolding` environment variable is set and the component type fold threshold value is set to `0`, but the fold threshold value on the form is > `0`.
Chain	Automatically abuts the selected set of MOS transistors to form a chain. The *Chain* option is available only for instances in *As In Schematic* mode. The transistors to be abutted must be set up for abutment first.

© 2023 Cadence Design Systems, Inc.

Field	Description
Interdigitate Chains	Automatically identifies nodes that qualify as pseudoparallel connections and defines them during chaining and abutment. A pseudoparallel net connects nodes that are always at the same voltage, so that current does not pass through the net. This option does not control the automatic identification and creation of pseudoparallel nets during manual abutment. To do that, select *Create pseudoparallel connections* on *Generation* tab of the Connectivity form. Environment variable: lxAllowPseudoParallelNets
Alignment	This section lets you change the alignment and spacing between the selected components before you place them in the layout view. See Aligning Objects by using the Align Toolbar.
Align Left	Aligns the left edges of the selected components.
Align Vertical	Aligns the vertical center lines of the selected components.
Align Right	Aligns the right edges of the selected components.
Align Top	Aligns the top edges of the selected objects.
Align Horizontal	Aligns the horizontal center lines of the selected objects.
Align Bottom	Aligns the bottom edges of the selected objects.
Align Spacing	Spaces the components evenly, separated by the distance specified in the text field. If you switch off this option, the components are aligned but retain their original spacing.
Orthogonal Spacing	Additionally spaces the components perpendicular to the align direction, with each successive component offset by the distance specified in the text field.

© 2023 Cadence Design Systems, Inc.

Field	Description
Design Planning	This group box is available only when you have the Layout EXL or higher license checked out.
Virtual Hierarchy	Lets you generate a virtual hierarchy for the selected schematic instance, which can be accessed using the DESIGN PLANNING category in the Navigator assistant. By default, the option is OFF. When set to `nil`, the selected schematic instance is generated as a regular layout instance.
Unplaced	Opens the Instance/Pin List, which lists each component in the schematic for which there is no corresponding instance or pin in the layout. When you select a device from the list, the form displays properties set for that device. When you select a pin, the form shows the options described in Generate Selected Components - Pin Options Form.
Orientation buttons	This section lets you change the orientation of the selected instance.
Rotate	Rotates the component 90 degrees counterclockwise.
Sideways	Mirrors the component on the y axis (flips it horizontally).
Upside Down	Mirrors the component on the x axis (flips it vertically).

Related Topics

Instance/Pin List Form

Layout Generation

Pcell Set Up for Abutment

Placement Modes for Generating Selected Components from Source

Generate Selected Components - Pin Options Form

Generating Components As In Schematic

Configure Physical Hierarchy Window

© 2023 Cadence Design Systems, Inc.

Generating Individual Pins

Listing Unplaced Components after Layout Generation

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Generate Selected Components - Pin Options Form

Use the *Generate Selected From Source* form to generate a pin in Place Individually mode, the Generate Selected Components form expands to show the attributes of the pin currently being generated. Any changes you make to the attributes before clicking in the layout view are applied to the current pin and also become the new defaults for all subsequent pins in the selected set.

Field	Description
Terminal	Shows the name of the current terminal.
Layer	Specifies the layer-purpose pair for the pin.
Width	Specifies the width of the pin.
Height	Specifies the height of the pin.
Create Label As	Specifies whether a pin label is created when the pin is generated. You can choose to create either a *Label* object or a *Text Display*. *Label* drives connectivity by providing connectivity-related information. They can be copied to different cellviews. However, they can become outdated if the connectivity changes on the shapes to which they are attached. *Text Display* is driven by connectivity and they show current connectivity. This is why, when you change a net on the associated shape or pin, the display updates with the new net, indicating the shapes connectivity. Text displays cannot copied to different cellviews.
Options	Opens the Soft Block Global Options form where you can set the size, font, style, justification and orientation of the label lettering, and the drawing or pin layer on which the labels are displayed. Environment variable: initIOLabelType
I/O Type	Assigns a property used by routers to identify the direction of the signal into or out of the pin. The signal can be *input, output, inputOutput* (bidirectional), *switch* (carries data either in or out but not simultaneously), *jumper* (passes data through the pin), *tristate*, or *unused*.
Access Direction	Assigns a property used to identify the part of the pin to which routers can connect routing. The access direction can be *Top*, *Bottom*, *Left*, *Right*, *Any*, or *None*.

© 2023 Cadence Design Systems, Inc.

Related Topics

Soft Block Global Options Form

Layout Generation

Placement Modes for Generating Selected Components from Source

Generating Individual Pins

Listing Unplaced Components after Layout Generation

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Incomplete Net Filter Form

Use the **Incomplete Net Filter** form to control the display of incomplete net markers for the specified nets.

To open the form, right-click in the browser pane and choose *Incomplete Net Filter*.

When a filter has been applied, the *Incomplete Nets* category label in the Annotation Browser is displayed in blue text.

Field	Description
Filter incomplete nets	Specifies whether the incomplete net filter needs to be set, based on the values of the incompleteNetFilterDefaultList and incompleteNetFilterMode environment variables, unless the required filter settings are already saved in the design. Environment variable: incompleteNetFilter
Filter Mode	Enables you to choose whether to show (or hide) the incomplete net markers for the specified net (or nets).
Net Names	Lets you specify a net (or nets) on which the selected *Filter Mode* must apply. ■ If the selected *Filter Mode* is set to *Show*, only the specified net names have their labels showing or displaying in the layout canvas. ■ If the selected *Filter Mode* is set to *Hide*, only the specified net names have their labels hidden in the layout canvas.
List of Nets	Populate the list of nets using the buttons on the right of the form.
Add To List	Adds the specified nets to a list on which the selected filter mode applies.
Remove From List	Removes a selected net (or nets) from the existing filter list.
Remove All	Removes the entire filter list.
Add Selected Nets	Lets you add nets to the filter list by selecting the nets in the Navigator assistant or from the layout canvas. This option can be used when a large number of nets need to be added to the filter list. Typing the names to add to the list can be time consuming when the number of nets to be added are too many.

© 2023 Cadence Design Systems, Inc.

Field	Description
Save To Cellview	Saves the current filtering options to the cellview such that the same settings can be reloaded later when the cellview is relaunched. Environment variable: incompleteNetFilter The *Save To Cellview* option is enabled only when the current cellview is editable.
Reload From Cellview	Enables you to revert to the filtering options previously saved for the cellview even when the cellview is opened in a different window. This option is enabled only when some information was previously saved into the cellview.
Delete From Cellview	Enables you to delete the net name filter settings already saved for the cellview. This option is enabled only when the current cellview is editable.
Defaults	Reverts the saved settings and sets the incomplete net filter based on the default settings that are driven by the environment variables. ■ incompleteNetFilterDefaultList ■ incompleteNetFilterMode

Related Topics

Annotation Browser Assistant

Annotation Browser Options Form

Custom Show Filter Form

View By Form

© 2023 Cadence Design Systems, Inc.

Instance/Pin List Form

The Instance/Pin List form lets you see all the components in the schematic that are not yet placed in the layout. Use it when generating selected components from source.

Field	Description
List of instances	Displays a list of components available in the schematic. Wires, labels, text, instances with an ignore property, and instances that have already been placed in the layout are not considered available components and are not shown in the list.
Apply	Selects the components in the schematic for the specified instances and generates pin.

Related Topics

Listing Unplaced Components after Layout Generation

Generating Components As In Schematic

Click in the layout canvas to place the selected instances at the required location.

Generating Individual Pins

Placement Modes for Generating Selected Components from Source

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Load Physical View Form

Use the Load Physical View form to import cellview information from an existing OpenAccess cellview called the source physical cellview. The *Load Physical View* menu item is available only in edit mode.

Field	Description
Choose Source Physical View	Specifies the source cellview from which values are to be loaded into the target cellview.
Library Name	Specifies the library of the source cellview. If you are accessing the form for the first time, you will see the target library and cell names appear in the *Choose Source Physical View* section. If you have accessed the form earlier and specified the library and cell name to be loaded from the source, you will see the same until you change the values.
Cell Name	Specifies the cell name of the source cellview.
View Name	Specifies the view name of the source cellview.
Browse	Opens the Library Browser to select the source cellview. These text fields are case-sensitive.
Choose Mapping File	Layout XL or a higher tier, defines the mapping between instances and pin layers in the source and target cellviews.
Mapping File	Layout XL or a higher tier, specifies a file containing the instance and pin layer mapping to be applied when the *Load Physical View* command is run.
Select All	Selects all options in the Load Physical View form.
None	Clears all selections in the Load Physical View form.
Update Instances	Specifies the instances to be loaded into the target cellview.
Using Instance Name	Updates instances using their names. This option is useful for updating instances that have the same name in the source and target cellviews. When updating instances, only those instance attributes are loaded that do not impact the connectivity or the constraints of the target cellview.

© 2023 Cadence Design Systems, Inc.

Field	Description
Using Master Name	Updates instances irrespective of their names. This option allows the PAD type IO instances to be updated based on their labels. If the source cellview has IO instances that have the same label text as the instTerm associated with the target IO instance, the target IO instance is updated with the source IO instance. During this update, the label is also copied to the target cellview instance. *Using Master Name* allows updating only the *IO Pads* type instances. To update the other type of instances, select *Using Instance Name*.
Standard Cells	Updates standard cell instances in the target cellview. The source and target master for each standard cell instance must be the same.
Custom Cells	Updates custom cell instances in the target cellview. The source and target master for each custom cell instance must be the same.
Macros	Updates macros in the target cellview. The source and target master for each macro must be the same.
IO Pads	Updates IO pads in the target cellview.
Add physical only instances	Adds physical-only instances that exist only in the physical domain and lack a corresponding instance in the logical domain, for example, filler cells.
Update Design	Specifies the parameters to be updated in the target cellview
Update Cell Type	Updates cell type in the current design based on the source cellview. In addition to the cell type, the CLASS and CORNER properties are also copied from the source cellview to the target cellview. Use the Set Cell Type form to verify the changes.
Update Block Type	Updates block types in the current design based on the source cellview. Use the Edit Properties form to verify changes.
Update Symmetry	Updates symmetry information in the current design based on the source cellview. Symmetry defines the orientation of the cellview. The symmetry of a cellview can be set to a string value: `X, Y, XY, any,` or `none`. Here, `X` aligns the cellview about the x-axis and `Y` aligns the cellview about the y-axis.

© 2023 Cadence Design Systems, Inc.

Field	Description
Update Design Site Pattern	Updates the `sitePattern` of the current design based on the source cellview. Use the Create Row form to verify the changes.
Update Design Track Pattern	Updates design track patterns in the current design based on the target cellview. Track patterns may be user-defined or defined in the technology file. Only the user-defined track patterns are updated when the *Update Design Track Pattern* check box is selected. Use the Edit Track Pattern form to verify the changes.
Update Virtual Hierarchies	Layout XL or a higher tier, transfers virtual hierarchies from the source to the target cellview only when the virtual hierarchies have matching schematic instance masters. With this option selected, virtual pin definitions are also copied.
Load Pins and Update Nets	This section lets you specify different ways in which you can load pins and update nets.
Pins	Loads pins from the source cellview into the target cellview. When loading pins from the source cellview, information about the connectivity model assigned to the pins is also loaded into the target cellview.
Mode	Specifies the mode to load the existing pins in the target with the pins from the source - *Replace* or *Update*. By default, the pins are replaced.
Update Sig Type of Nets	Updates signal-type information on nets when the net names in the source and target cellviews are the same.
Update Unattached Labels	Copies the unattached labels present in the source cellview to the target cellview.
Add Geometries	Imports shapes from the source to the target cellview and ensures that the imported shapes appear at exactly the same location in the target cellview as in the source. The *Add Geometries* functionality checks out four Layout GXL tokens.

© 2023 Cadence Design Systems, Inc.

Field	Description
Delete Shapes, Wires and Vias in Destination	Deletes existing shapes and wires (except Modgens and pins) before adding the same from the source. This option is useful when the target cellview has a shape or wire present at the same location as that in the source cellview. Copying shapes and wires from the source view can lead to an overlap of the components in the target cellview. This option is available only for updating the existing geometries and is therefore enabled only after you select a shape or wire to be updated.
Shapes	Imports all shapes from the source cellview to the target cellview. At advanced nodes, *Add Geometries* copies snap patterns (SP) and width spacing patterns (WSP) from the source to the target cellview along with other shapes. *Add Geometries* also copies pattern flipping-related attributes from the source to the target cellview. Pattern flipping refers to flipping an existing WSP pattern in alternate periods in a region. WSP pattern flipping can be specified at two levels: ■ On a WSP region ■ On a WSP global grid
Wires and Vias	Imports all wires (pathSegs and vias) and their topologies from the source cellview to the target cellview.
Filter Pins and Geometries	Allows you to selectively transfer pins, routes, wires, and shapes on the specified metal layers.
Add Pins and Geometries for Selected Nets	Transfers pins, shapes, and wires on the selected nets when the selection is done from the target cellview.
Add Pins and Geometries for Specified Layers	Specifies the metal and poly layers on which you want to place the pins from the source cellview to the target cellview.

© 2023 Cadence Design Systems, Inc.

Field	Description
Update Boundaries	Imports the boundary information from the source cellview to the target cellview. If a boundary you chose to import already exists in the source cellview, the points of the target boundary are updated. Otherwise, the boundary from the source cellview is replicated at the target.
PR Boundary	Imports the PR boundary from the source cellview to the target cellview.
Snap Boundary	Imports the snap boundary from the source cellview to the target cellview.
Replace Rows	Replaces row information in the target design with the row information from the source view.
Standard Rows	Replaces standard row information in the target design with the row information from the source view.
Custom Rows	Replaces custom row information in the target design with the row information from the source view.
Row Regions	(Advanced Node Layout EAD Only) Replaces row regions in the target cellview with those in the source cellview.
Replace Obstruction	Replaces an existing blockage or halo from the target cellview with an appropriate obstruction from the source cellview. Before any new obstructions are loaded, the system looks for existing obstructions in the target cellview. Depending on the presence of the obstruction owner, which may be an instance, a cluster, or a PR boundary, the obstruction may or may not be updated with the one from the source cellview. Existing obstructions in the target cellview are automatically deleted before new obstructions are loaded.
Blockages	Replaces all existing blockages.
Halos	Replaces all existing halos.
Type	Specifies the type of obstructions to be replaced for blockages or halos.

© 2023 Cadence Design Systems, Inc.

Field	Description
Transfer Constraints	Transfers relationship constraints, such as the alignment constraint, process rule overrides, and constraint groups, from the source cellview to the target cellview. If none of the check boxes are selected, the *Replace* and *Update* buttons for *Mode* are disabled.
All Constraints and Process Overrides	Transfers all constraints, pin group guides (including the pins, guide members, and area boundaries), process rule overrides, and constraint groups from the source cellview to the target cellview. Constraints can be transferred in one of the following modes: ■ *Replace*: Deletes all constraints, process rule overrides, and floating constraint groups from the target cellview and copies those from the source cellview to the target cellview. **Note:** Floating constraint groups are the constraint groups that are either not assigned to an owner or do not have a built-in constraint group. ■ *Update*: Copies constraints, process rule overrides, and constraint groups from the source cellview to the target cellview. ❑ The mapping criteria for constraint and process rule override updates is found by matching member pins in the source and target cellviews. A given source constraint is mapped to a target constraint if the number of matching member pins is highest. Load Physical View updates the member pins and parameters of the mapped target constraint from those in the source constraint. ❑ The mapping criteria for constraint groups are the constraint group names. If a match is found, the contents of the constraint group or the process rule overrides are replaced with that in the source. The default constraint group contents are transferred in *Replace* and *Update* modes.

© 2023 Cadence Design Systems, Inc.

Field	Description
	When transferring area boundaries, the `validRoutingLayers` and layer priority order, if set, are copied to the target area boundary. ■ If the area boundary is located in the same pin group guide in both cellviews, the area boundary in the target cellview is deleted and is replaced with the one in the source cellview. ■ If the area boundary is located in different pin group guide in both cellviews, the area boundary in the target cellview is retained. The area boundary from the source cellview is renamed and copied to the target cellview.
Only Constraint Groups	Transfers only constraint groups. Choose one of the following modes: ■ *Replace*: Deletes all floating constraint groups from the target cellview, and copies those from the source cellview to the target cellview. ■ *Update*: Copies new floating constraint groups from the source cellview to the target cellview. For constraint groups with matching names in the source and target cellviews, the content of the target constraint group is replaced with that of the source constraint group. The default constraint group contents are transferred in both modes. The *Transfer Constraints* functionality checks out 4 Layout GXL tokens.

Related Topics

The Load Physical View Command

Preparing Data for Floorplanning

Edit Soft Blocks Form

Rows

Creating a Track Pattern

Pattern Flips

© 2023 Cadence Design Systems, Inc.

Layout Support for Row Regions

Working With Pin Connectivity Models

© 2023 Cadence Design Systems, Inc.

Load Traced Nets View Form

Use the Load Traced Nets View form to load the previously saved traces from a design cellview to your current layout cellview. The Load Traced Nets View form is available through the *Connectivity – Net Tracer – Load Physical Traces* command.

Field	Description
Library	Displays the name of the design library from which the saved traces will be loaded.
Cell	Displays the name of the cell from which the saved traces will be loaded.
View	Displays the view name from which the saved traces will be loaded. The loaded traces are added along the already displayed traces in the current cellview.
Replace Traced Nets	Replaces the already displaying traces in the current layout cellview with the traces from the saved cellview.

Related Topics

Loading Physical Traces

Save Traced Nets View Form

Trace Manager Form

Net Tracer Options Form

Net Tracer Display Options Form

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Master Variant Form

Use the Master Variant form in the shortcut menu of non-parameterized cells and mosaics to generate a new master variant view for layout instances.

Field	Description
Library	Specifies the library in which the new master variant view for the selected layout instances will be generated. By default, the *Library* field displays the name of the current selected instance master library.
Cell	Specifies the cell in which the master variant view for the selected layout instances will be generated. By default, the *Cell* field displays the name of the current selected cell.
View	Specifies the name of the new master variant view to be generated for the selected instances. By default, the variant view name for the first variant is specified as `layout_variant_1`, with each new variant name automatically incrementing to `layout_variant_2`, and so on. You can specify any name for the new variant, instead of choosing the default name. If the specified variant view already exists, a thumbnail image of the same is also displayed alongside the *View* field. In this case, a new variant is not created and saved on disk, but the layout instance is remastered.
Hierarchy	This section provides options for editing the copied layout after it is remastered. You can choose to either *Edit In Place* the remastered instances or descend into the hierarchy and edit them.
Edit in Place	Edits the first instance from the set of instances that have been remastered.
Descend Edit	Descends into the hierarchy of the remastered instances to enable editing.

Related Topics

Generating Master Variants

© 2023 Cadence Design Systems, Inc.

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Match And Fix Form

Use the Match And Fix form to identify layout patterns in a design and to fix them according to the fixing rules or guidelines associated with each pattern.

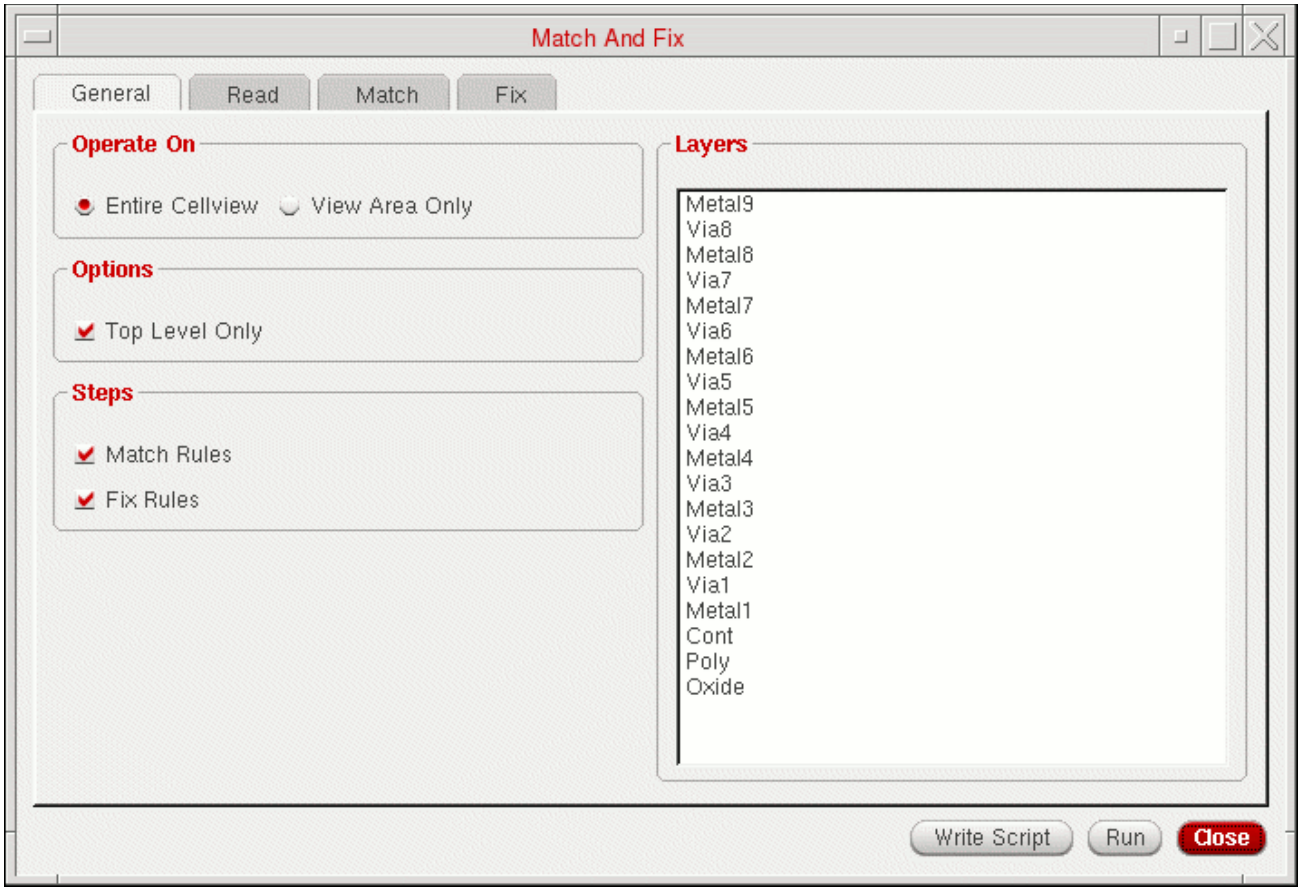
This form has the following tabs.

Tab	Description
General	Lets you define the scope of the options to be used for matching and fixing the layout patterns
Read	Lets you set one or more rule deck files and specify the set of rules to be applied to match and fix the layout pattern
Match	Lets you specify the options for matching the layout pattern.
Fix	Lets you specify the options for fixing the layout pattern.

© 2023 Cadence Design Systems, Inc.

General

Use the *General* tab to define the scope of the options to be used for matching and fixing the layout patterns. The following table describes the fields available on the *General* tab of the Match and Fix form.



Field	Description
Operate On	This section lets you specify the scope for running the *Entire Cellview* and *View Area Only* command.
Entire Cellview	Specifies the matching and fixing of the layout patterns be done on the entire cellview.
View Area Only	Specifies the matching and fixing of the layout patterns be done on the view area only.

© 2023 Cadence Design Systems, Inc.

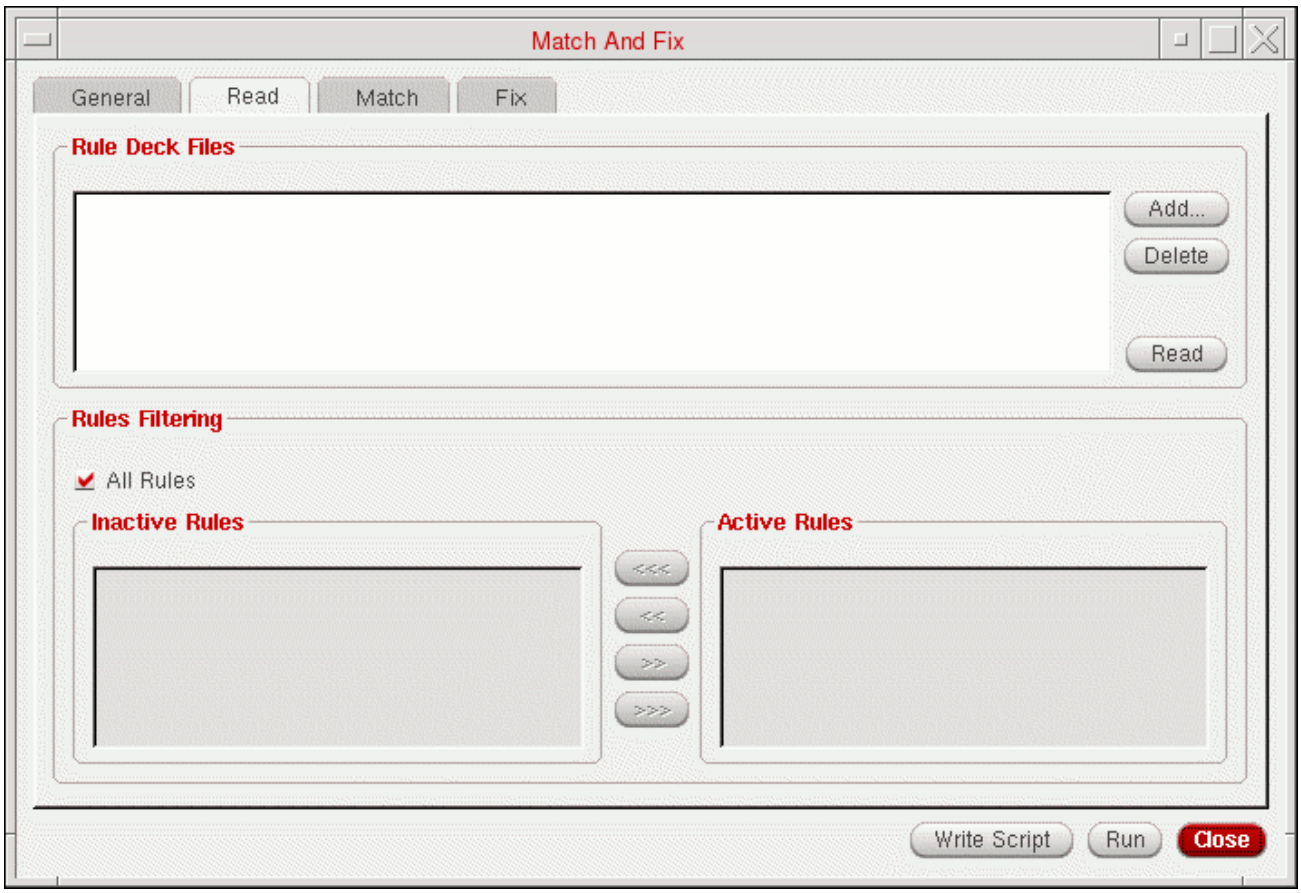
Field	Description
Options	This section lets you specify whether or not the matching and fixing of the layout patterns is done only on the shapes at the top level of the open design.
Top Level Only	Specifies matching and fixing of the layout patterns be done only on the shapes at the top level of the open design. Fixing of layout patterns can only be done at the top level of the open design.
Steps	This section lets you control whether to perform the match or to perform the automatic fixing of matched patterns or violations. Enabling either of these flows also enables modification of the corresponding tabs in the form.
Match Rules	Performs matching of layout patterns and enables modification of the options on the *Match* tab.
Fix Rules	Performs automatic fixing of the matched layout patterns and enables modification of the options on the *Fix* tab.
Layers	Lets you choose the layers on which the matching and fixing should be applied.

Read

Use the *Read* tab to set one or more rule deck files and specify the active rules to be applied to match and fix the layout pattern.

© 2023 Cadence Design Systems, Inc.

A rule deck file defines pattern-based manufacturability checks. These match and fix rules are composed of a pattern definition and corresponding fixing rules. Consult with your foundry for a compatible rule deck file.



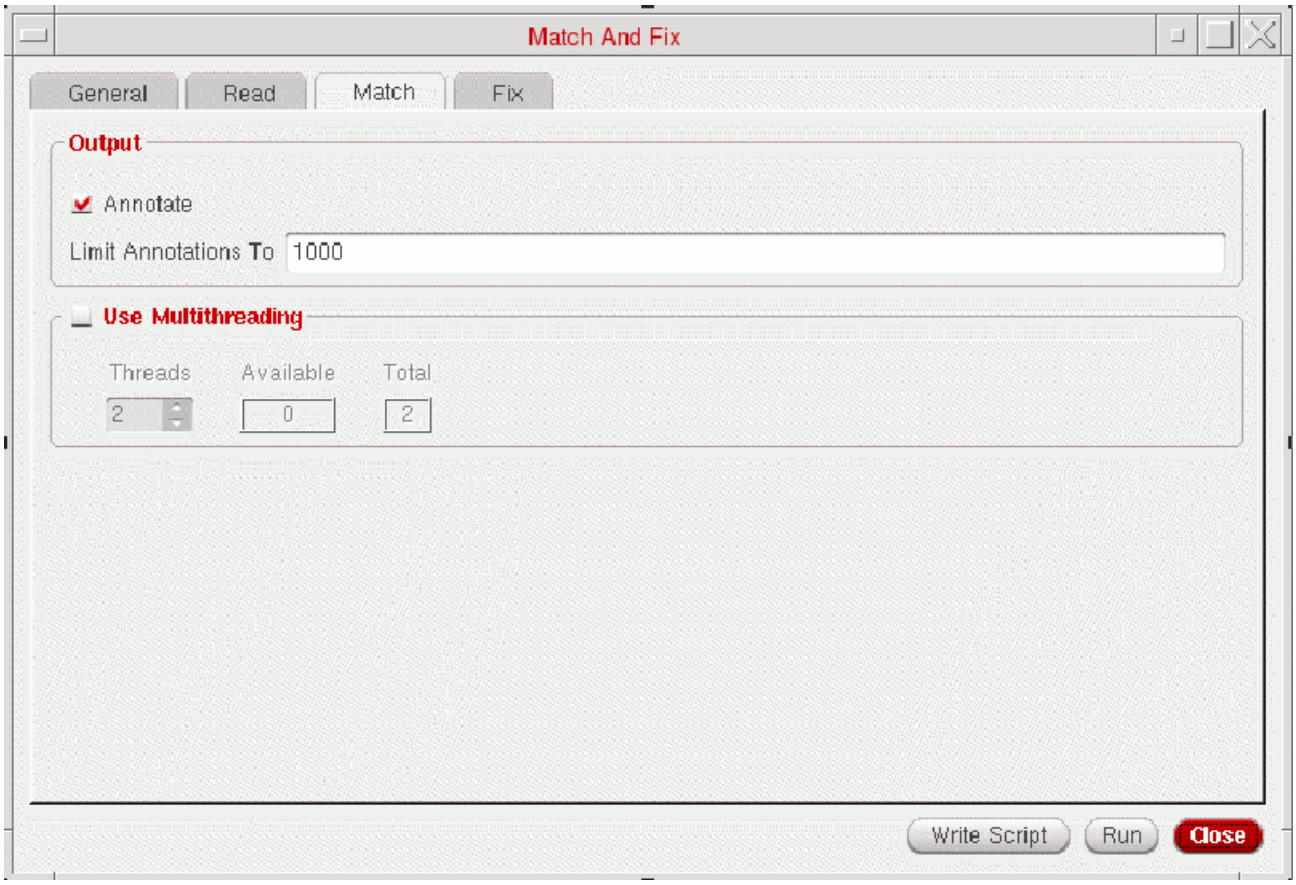
Field	Description
Rule Deck Files	Enables you to load the layout patterns from one or more rule deck files.
Add	Add the files.
Read	Reads the rules specified in the selected rule deck files and import the active rules to be applied to match and fix the layout pattern.
Delete	Deletes a file.
Rules Filtering	Specifies the set of rules to be applied from the loaded rule deck files.

© 2023 Cadence Design Systems, Inc.

Field	Description
All Rules	Specifies that all rules from the selected rule deck files be applied to match and fix layout patterns. By default, the option is ON.
Inactive Rules	Specifies the list of rules that should not be applied to match and fix layout patterns.
Active Rules	Specifies the list of rules that should be applied to match and fix the layout patterns.

Match

Use the *Match* tab to specify the options for matching the layout pattern.



© 2023 Cadence Design Systems, Inc.

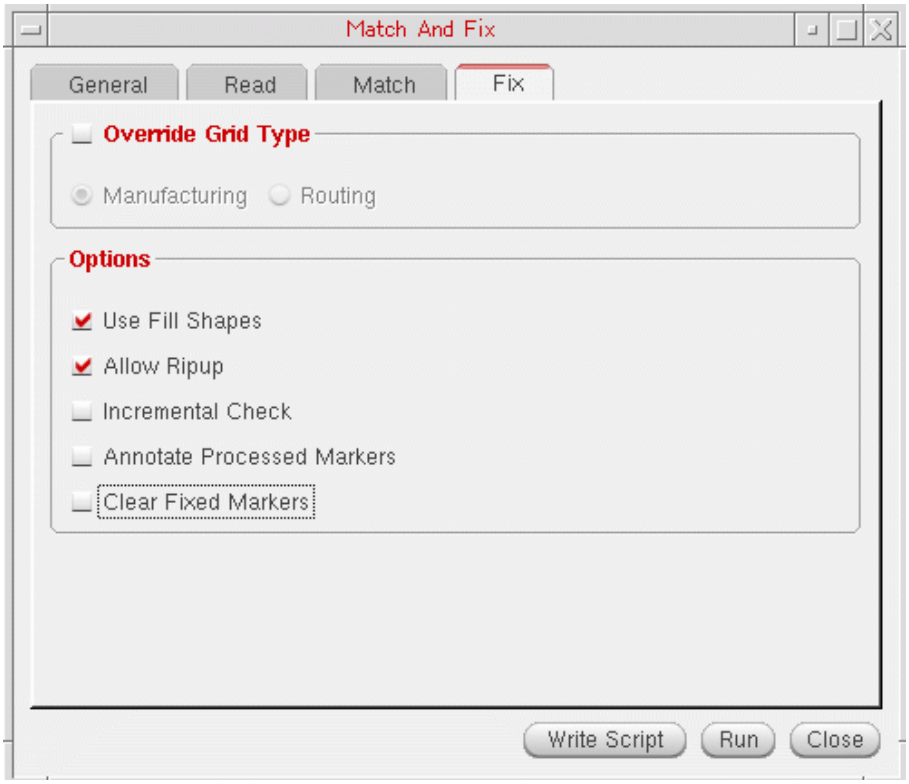
The following table describes the fields available on the *Match* tab of the Match And Fix form.

Field	Description
Output	This section lets you controls whether or not to return an annotation for each match to the annotation browser and specifies the limit of the number of matched returned.
Annotate	Controls whether or not to return an annotation for each match to the annotation browser.
Limit Annotations	Specifies the limit of the number of matches returned. By default, the number of matches returned are limited to 1000.
Use Multithreading	This section lets you specify whether to enable multi-threaded search for additional efficiency while performing the match. By default, the *Use Multithreading* option is OFF.
Threads	Specifies the number of threads to used for matching the layout pattern.
Total	Specifies the number of processors available on the station.
Available	Specifies the difference between the total and used processors on the station.

© 2023 Cadence Design Systems, Inc.

Fix

Use the *Fix* tab to specify the options for fixing the layout pattern.



The following table describes the fields available on the *Fix* tab of the Match And Fix form.

Field	Description
Override Grid Type	**This section enables you to override the default grid type being used in the layout.**
Manufacturing	Specifies that the manufacturing grid be used and all shapes in the layout be snapped to this grid.
Routing	Specifies that the routing grid be used and all shapes in the layout be snapped to this grid.
Options	**This section lets you specify the parameters for fixing the layout pattern.**
Use Fill Shapes	Specifies use of fill shapes for fixing the layout pattern.
Allow Ripup	Specifies local ripup and reroute fixing be allowed for fixing the layout pattern.

© 2023 Cadence Design Systems, Inc.

Field	Description
Incremental Check	Controls the fixing engine to incrementally call the matching engine to check whether or not an active layout pattern has been introduced while fixing the layout pattern.
Annotate Processed Markers	Controls whether additional markers be added to the annotation browser to show the region of the layout that has been modified while fixing the layout pattern.
Clear Fixed Markers	Controls display of fixed matches or violations in the annotation browser. If this option is selected, all fixed markers are cleared from the Annotation Browser, otherwise, the fixing step does not clear any of the markers from the matching step.

Related Topics

Match and Fix Flow

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Modify Correspondence Form

Use the Modify Correspondence form to change the relative placement of instances in the target structure manually by changing the correspondence of the individual instances in the source and the target.

Field	Description
Connectivity Source	Lists the instances in the source structure. The list is read-only.
Connectivity Target	Lists the instances in the target structure. When you select an instance in the target structure, the corresponding instance in the source structure is highlighted in the *Connectivity Source* list. You change the correspondence of an instance by selecting it in the *Connectivity Target* list and moving it up or down in the list using the arrow buttons.
Swap	Swaps the correspondence of a pair of target instances.
OK	Checks that connectivity is correct before accepting the new correspondence.

Related Topics

Clone Generation

Clone Source Selection and Target Search and Generation

Editing a Target Structure Before Generation

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Move Cells Form

Use the Move Cells form to move selected cells from one component type to another or to the *No component type* folder.

Field	Description
Select new component type	Lets you choose which component type should be set for the selected cells. Select the component type from the pull-down list and click *OK*.
No component type	Moves the selected cells to the *No component type* folder, which means that the selected cells are not assigned to any component type.

Related Topics

Component Types Mode in CPH

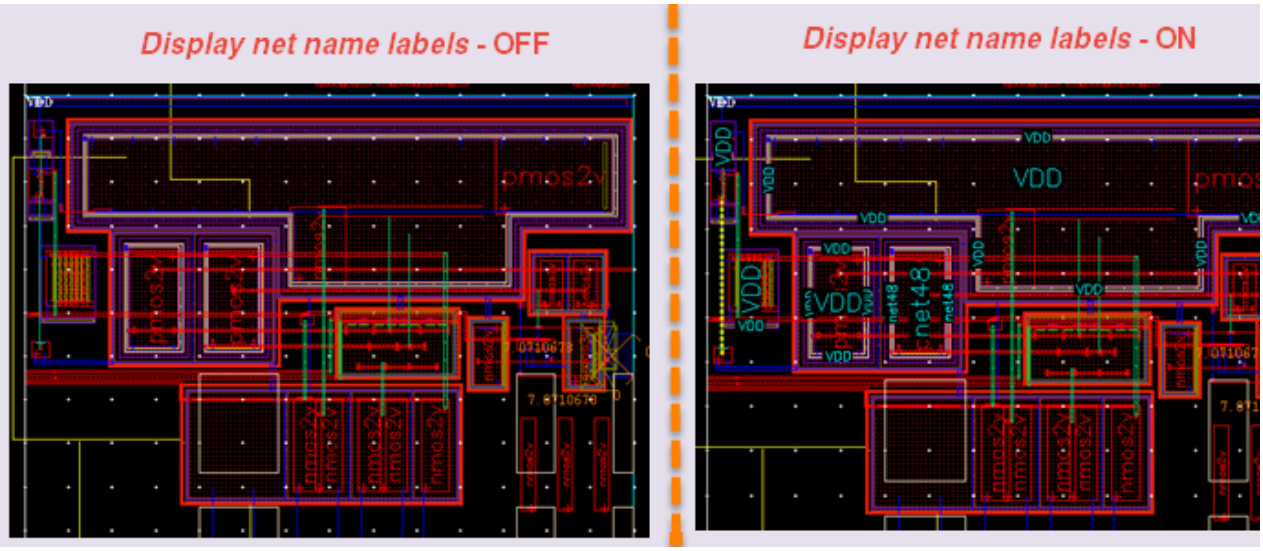
Defining a Design-Level Component Type

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Net Name Display Options Form

Use the Net Name Display Options form to control the display of net name labels on layout shapes. The net name labels are displayed in the layout canvas for those shapes that are big enough to display these labels.



Field	Description
Enable	Controls whether net name labels are displayed in the layout canvas. Environment variable: netNameDisplayEnabled
Hierarchy Range	Sets how many levels of the hierarchy are the nets displayed in. ■ *Current cellview*: Displays the nets in the current cellview. ■ *Top to bottom*: Displays the nets in the entire hierarchy of the cellview. Environment variable: netNameDisplayScope
Drag net name	Displays the net name of an object when you drag the object during the move, copy, or stretch operation. Environment variable: dragDisplayNetName
Net Name	Controls the display of net names on shapes, vias, instance pins and Pcell pins.

© 2023 Cadence Design Systems, Inc.

Field	Description
Show	Controls the display of net names on shapes. By default, the option is ON, which means net names are displayed in the layout canvas. Environment variable: netNameDisplayNetShow
On Vias	Controls the display of net names on vias. Environment variable: netNameDisplaySkipVias
On Instance pins	Controls the display of net names on instance pins. Environment variable: netNameDisplaySkipInstances
On Pcell pins	Controls the display of net names on Pcell pins. Environment variable: netNameDisplaySkipPcells
Terminal Name	Controls the display of terminal names.
Show	Controls the display of terminal names on shapes. Environment variable: netNameDisplayTermShow
On Instance pins	Controls the display of terminal names on instance pins. Environment variable: netNameDisplayTermSkipInstances
On Pcell pins	Controls the display of terminal names on Pcell pins. Environment variable: netNameDisplayTermSkipPcells
Label Options	Specifies the available label options.
Draw on top	Displays the net names on top of all the shapes, which can help make the labels more visible. The labels are displayed on top of the shapes after the design is first fully drawn. This ensures that the labels are actually drawn on top of the shapes, making them more prominent. Environment variable: netNameDisplayDrawOnTop

© 2023 Cadence Design Systems, Inc.

Field	Description
Draw background	Displays the label background along with the label text. Displaying the label background makes the label appear prominently on the canvas. The color of the background is controlled by the `netNameBackground` packet. When the `netNameBackground` packet is not specified, the background color is set to black. Environment variable: netNameDisplayDrawBackground
Color	Selects a color for the label from the drop-down color palette. Environment variable: netNameDisplayUserColor
Same as shape	Displays each net name label in the same color as the shape with which it is associated. Environment variable: netNameDisplayUseShapeColor
User	Displays each net name label in a user-specified color. You can select the color from the drop-down list.
Size	Specifies the size of the label on the net. The slider lets you choose the size of the label on the net. Environment Variable: netNameDisplayMaxLabelSize
Full size	Creates a label with the maximum size that can cover the net. Environment Variable: netNameDisplayMaxLabelSizeFull
Thickness	Chooses the thickness of the label on the net. You can choose from *Thin*, *Medium*, *Thick*, or *Custom*. When you select *Custom*, you can choose the thickness from the drop-down list.
Name Filtering	This section lets you selectively display the net name labels. You can use an appropriate filtering mode to selectively display or hide net name labels. By default, name filtering is OFF. But when enabled, the *Options* toolbar icon that controls the net name display functionality, looks slightly different,  , also displaying a filter symbol.

© 2023 Cadence Design Systems, Inc.

Field	Description
Filter net names	Enables you to be selective in displaying (or hiding) the nets for which name labels are displayed in the canvas. By default, all shapes that are big enough to display a label, have a net name label displayed. But, you can select the *Filter net names* option to display (or hide) the net name labels only for a specified net (or nets). Environment variables: netNameDisplayFilter, netNameDisplayFilterDefaultList
Filter Mode	Enables you to choose whether to show (or hide) the net name labels for the specified net (or nets). Environment variable: netNameDisplayFilterMode
Net Names	Specifies a net (or nets) on which the selected *Filter Mode* must apply. If the selected *Filter Mode* is *Show*, only the specified net names have their labels showing or displaying in the layout canvas.If the selected *Filter Mode* is *Hide*, only the specified net names have their labels hidden in the layout canvas.
Add To List	Adds the specified nets to a list on which the selected filter mode applies.
Remove From List	Removes a selected net (or nets) from the existing filter list.
Remove All	Removes the entire filter list.NNEven with the *Filter net names* option selected, if the entire filter list is removed; all net name labels are displayed in the canvas because there are no net names available for filtering.
Add Selected Nets	Adds nets to the filter list by selecting the nets in the Navigator assistant or from the layout canvas. This option can be used when a large number of nets need to be added to the filter list. Typing in the names to add to the list can be time consuming when the number of nets to be added are too many.
Save To Cellview	Saves the current filtering options to the cellview such that the same settings can be reloaded later when the cellview is relaunched. The *Save To Cellview* option is enabled only when the current cellview is editable.

© 2023 Cadence Design Systems, Inc.

Field	Description
Load From Cellview	Enables you to revert to the filtering options previously saved for the cellview even when the cellview is opened in a different window.
Delete From Cellview	Enables you to delete the net name filter settings already saved for the cellview.
Layer Filtering	Lets you selectively hide the net name for specific layers.
Hide Layers	Enables you to be selective in specifying the layers for which net name is not displayed. Environment variable: netNameDisplayFilterLayers
Layer Names	Lets you specify the name of layers for which net name is not displayed. You can specify multiple layer names by separating them with a space. Environment variable: netNameDisplaySkipLayers
Defaults	Reverts the saved settings to the values defined in the associated environment variables: ■ netNameDisplayBackground ■ netNameDisplayDrawOnTop ■ netNameDisplayFontStyle ■ netNameDisplayUseShapeColor ■ netNameDisplayFilterMode ■ netNameDisplaySkipVias ■ netNameDisplayFilterDefaultList

Related Topics

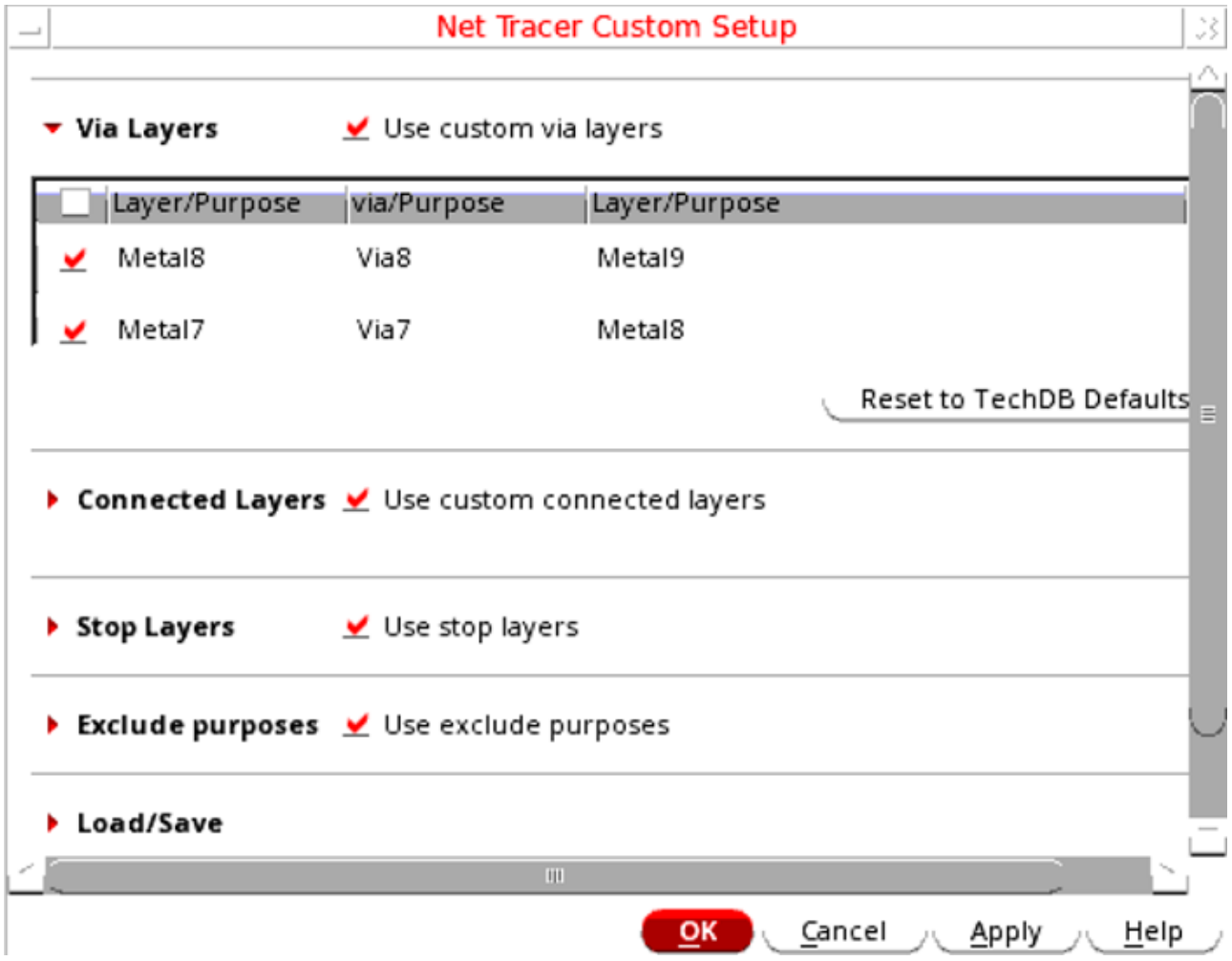
Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Net Tracer Custom Setup Form

Use the Net Tracer Custom Setup Options form to customize the source constraint group to be used for creating traces.

To display the Net Tracer Custom Setup Options form, access the Net Tracer Options form and choose *Connection Setup* – *Source* – *Custom* and click the adjoining *Setup* button.



Field	Description
Via Layers	This section specifies the via layers.

© 2023 Cadence Design Systems, Inc.

Field	Description
Use custom via layers	Lets you specify the via layers to be used for tracing. When this option is selected, the *layer1-via-layer2* lists displayed in the table below are enabled. The table lists all the via layers from the technology file and any via layers that you have added. Select the check boxes associated with the individual via layers that you want to use for tracing.
Reset to TechDB Defaults	Lets you use the via layers defined in the technology file. Clicking this button populates the list box with the via layers from the technology file and deletes any via layers that you may have added.
Connected Layers	This section specifies the connected layers.
Use custom connected layers	Lets you specify the connected layers to be used for tracing. When this option is selected, you can define connected layers in the *(layer1 purpose) (layer2 purpose)* format. The connected layers map to the equivalent layers defined in the technology file.If the option is deselected, connected layers are not used for tracing.
Reset to TechDB Defaults	Lets you use the connected layers defined in the technology file. Clicking this button populates the list box with the connected layers from the technology file and deletes any connected layers that you may have added.
Stop Layers	This section specifies the stop layers.
Use stop layers	Lets you define the layers on which the *Net Tracer* command must stop tracing a net. The stop criterion is specified in the *layer stopLayer* format, where both layers can be specified as LPPs. The *Net Tracer* command stops tracing the connected shapes on the *layer* LPP, if it finds an overlapping shape on the *stopLayer* LPP. If the *Use stop layers* option is deselected, all the connected shapes selected in the *Connected Layers* widget are used for tracing.
Layer/Purpose and Stop Layer/Purpose pair	Displays the stop layers that you have defined. The *Stop Layers* table displays additional *Color* and *Lock* columns that specify the mask color and lock status to be considered for the stop layers used for tracing. If a non-colored stop layer is specified, the *Color* and *Lock* fields are grayed out.”

© 2023 Cadence Design Systems, Inc.

Field	Description
Exclude purposes	*Use exclude purposes:* Lets you specify the layer purposes to be excluded for tracing.
Load/Save	This section specifies the options to load and save a file.
File	Specifies the name of the file: ■ to which you can save the customized layer settings. ■ from which you can load the customized layer settings. The *File* field displays a default filename. But, you can specify a filename, as required.
Browse	Opens the UNIX Browser window. Use this window to specify the name and location of the file ■ to which you want to save the customized layer settings. ■ from which you want to load the customized layer settings. If you specify a file from which to load the customized layer settings; then all the via, connected, and stop layers are read from this file at the start of a new Virtuoso session.
Via layers	Lets you save (or load) via layer settings to (or from) the file specified in the *Browse* field.
Connected layers	Lets you save (or load) connected layer settings to (or from) the file specified in the *Browse* field.
Stop layers	Lets you save (or load) via layer settings to (or from) the file specified in the *Browse* field.
Exclude Purposes	Lets you save (or load) exclude purposes settings to (or from) the file specified in the *Browse* field.By default, settings for all the three layers — via, connected, and stop — are saved (and loaded) to (or from) the file specified in the *Browse* field.
Save	Saves the layer settings to the file specified in the *Browse* field.
Load	Populates the file specified in the *Browse* field with the via, connected, and stop layer settings to be used for tracing.

Related Topics

Net Tracer Options Form

© 2023 Cadence Design Systems, Inc.

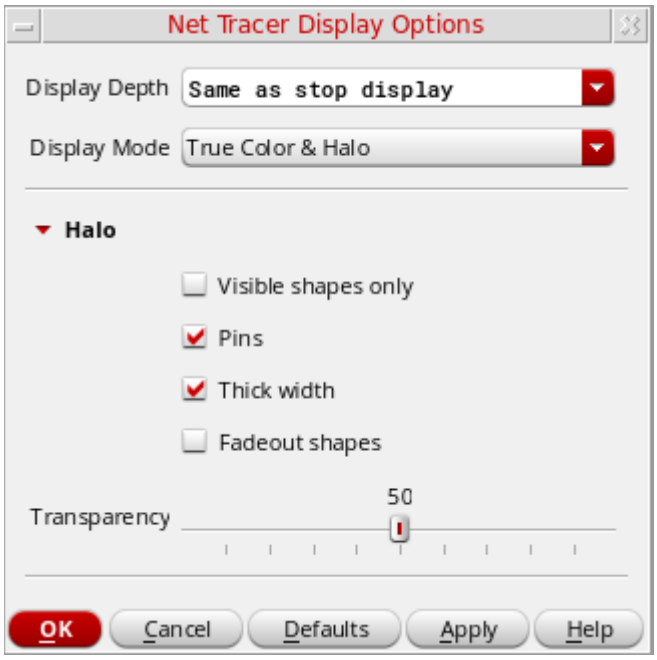
Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Net Tracer Display Options Form

Use the Net Tracer Display Options form to control how traces are displayed on the canvas.

The Net Tracer Display Options form, displayed below, is available through the *Options – Net Tracer* command.

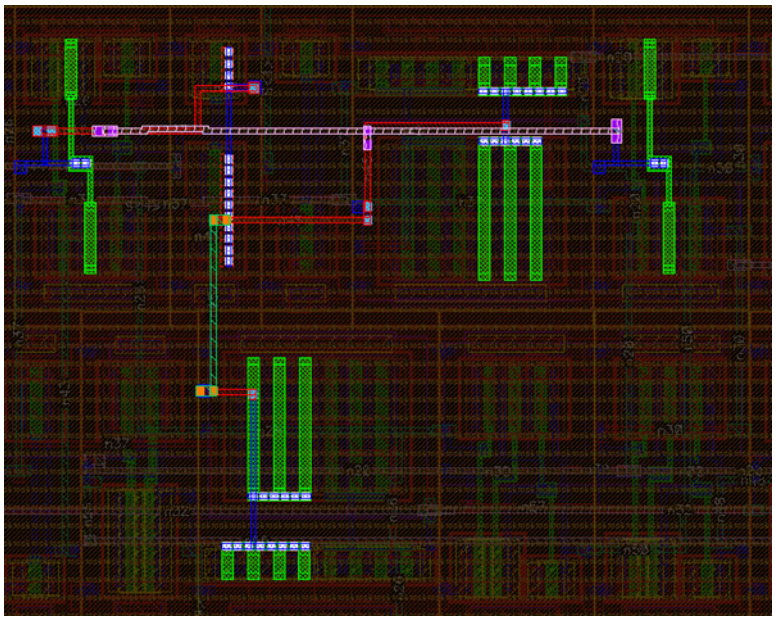


Alternatively, use the options in the Removing All Traces to control some of the trace display options.

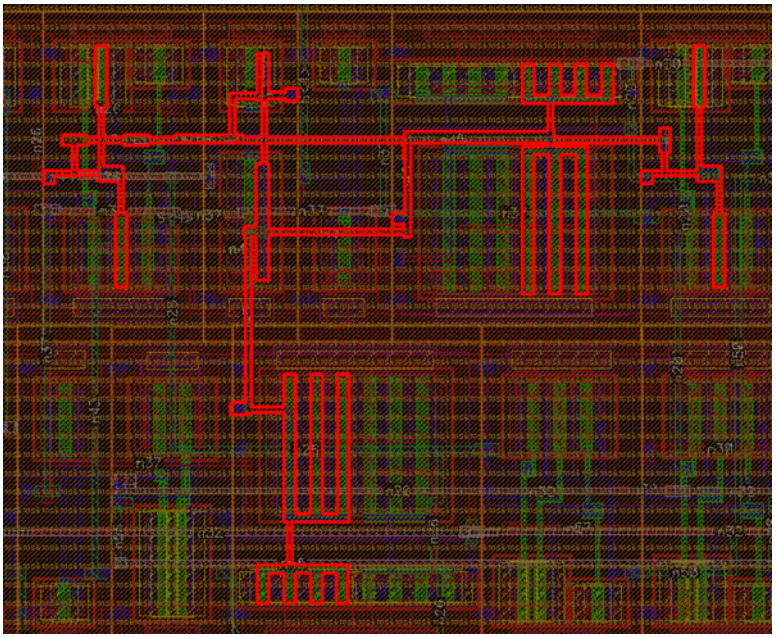
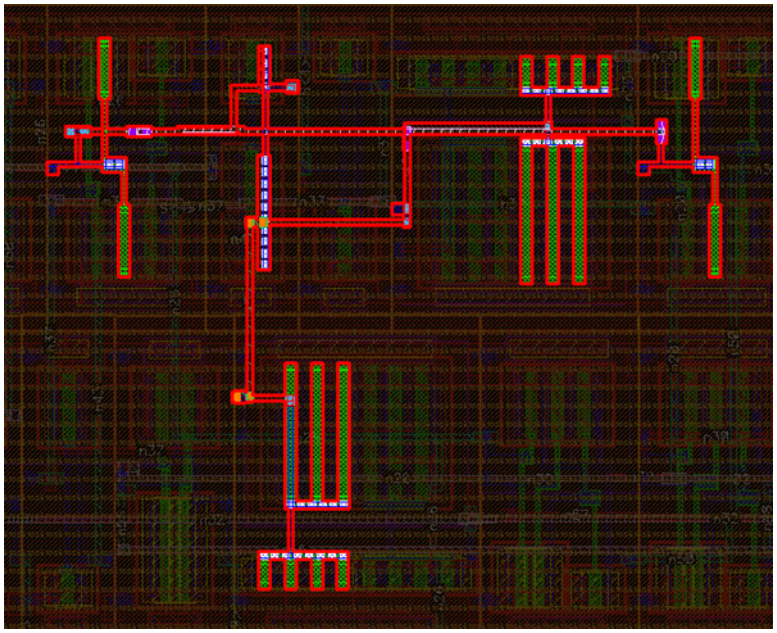
Field	Description
Display Depth	Lets you specify the depth up to which the shapes for a traced net are displayed. By default, *Display Depth* is set to *Same as stop display.* This means that in the Display Options form, if the *Display Levels – Stop* value is set to 0, only the shapes belonging to the trace on level 0 are displayed. If you set the *Display Depth* to say 32 and *Display Levels – Stop* value in the Display Options form is still set to 0, all the shapes in the hierarchy belonging to the trace are displayed. Environment variable: netTracerDisplayDepth

© 2023 Cadence Design Systems, Inc.

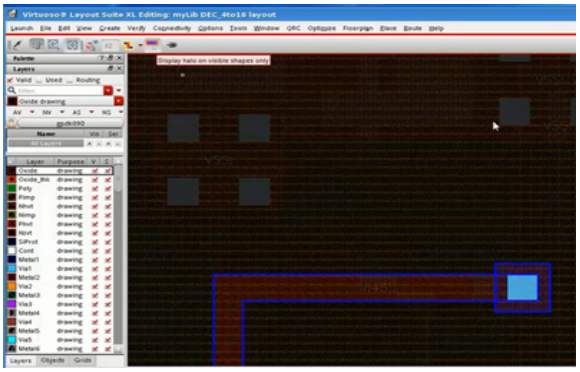
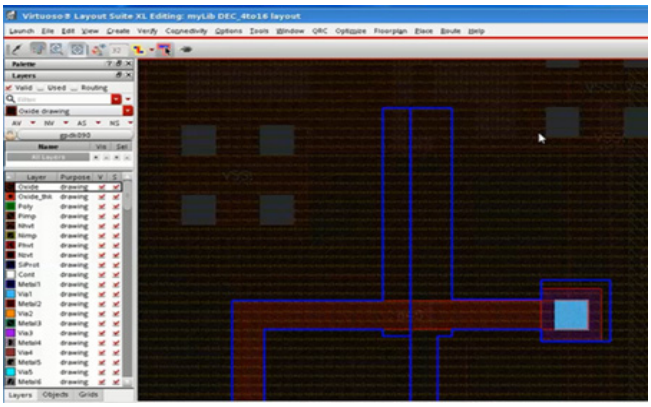
Field	Description
Display Mode	This section lets you control whether the traces over shapes, pins, and other objects associated with a traced net are displayed in true color, and whether a trace displays a halo. By default, traces are displayed in true color. Environment variable: netTracerDisplayMode
True Color	Displays the traces in true shape color.



© 2023 Cadence Design Systems, Inc.

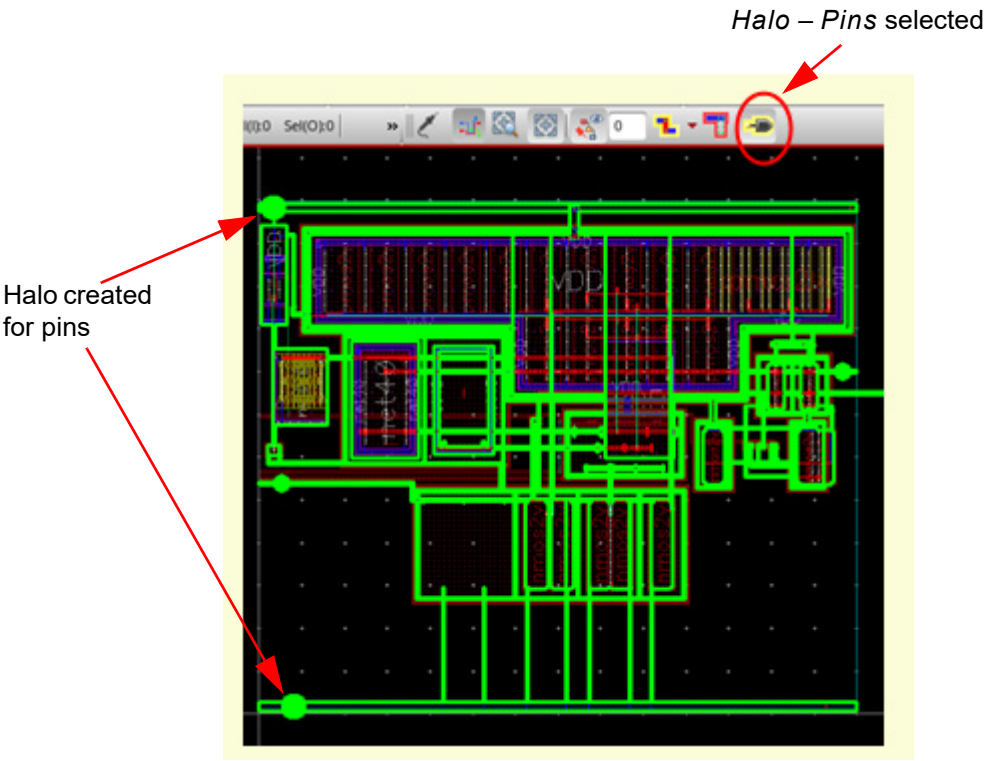
Field	Description
Halo	Displays only a halo around the shape. 
True Color & Halo	Displays the traces in true shape color and displays a halo around the shape. 

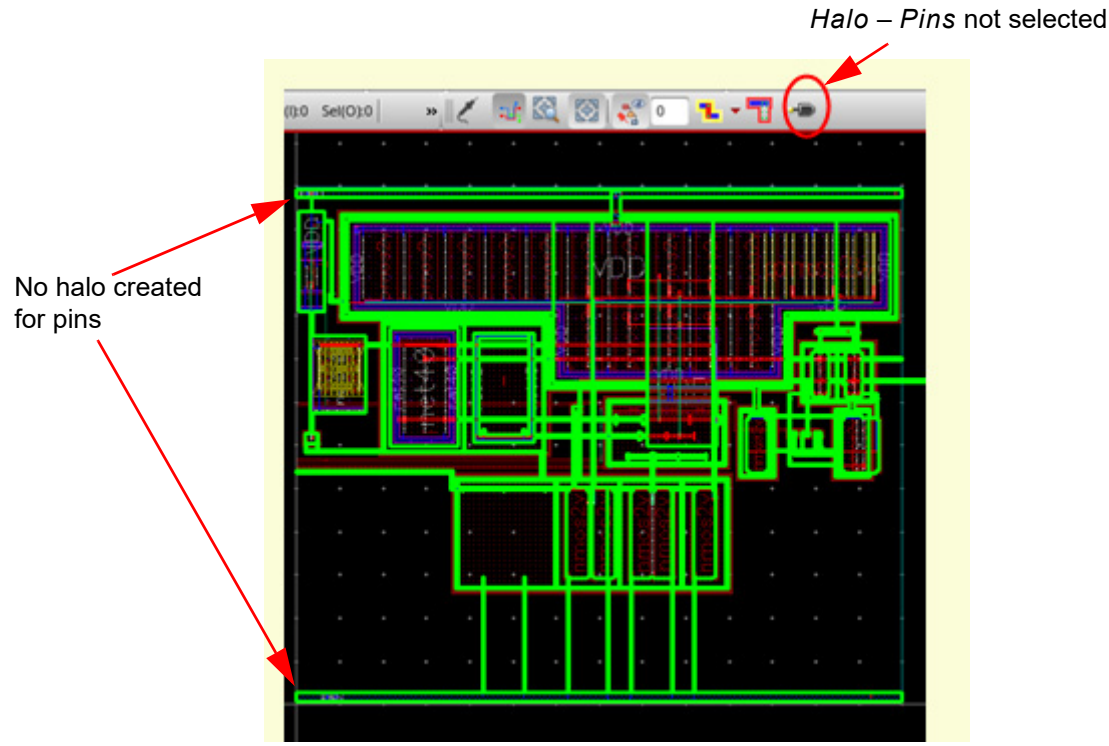
© 2023 Cadence Design Systems, Inc.

Field	Description
Halo	This section lets you specify the type of halo.
Visible shapes only	Displays the halo only around shapes that are on visible layers. Shapes on layers that are not displayed are not haloed. When the option is OFF, even those shapes that are not on the visible layers have their contours highlighted, as displayed in the figure below. By default, halos are created for all shapes. Environment variable: netTracerHaloForShapes  Only shapes on Shapes that are not on visible 
Pins	Creates halos around pin shapes. By default, halos are displayed for pins. Environment variable: netTracerHaloForPins

© 2023 Cadence Design Systems, Inc.

Field	Description

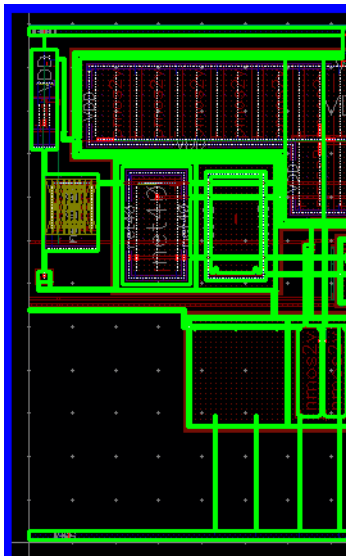




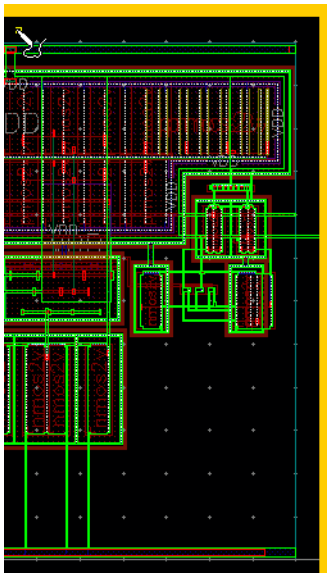
© 2023 Cadence Design Systems, Inc.

Field	Description
Thick width	Creates a halo with a thicker width, making the halo more prominent. By default, halos are displayed with a thick width. Environment variable: netTracerHaloStyleThick

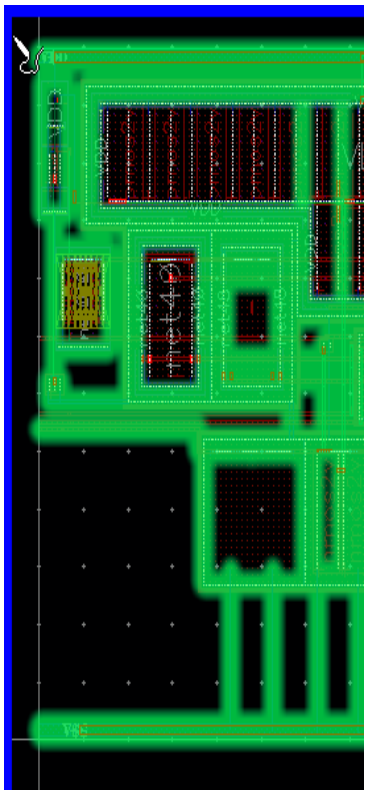
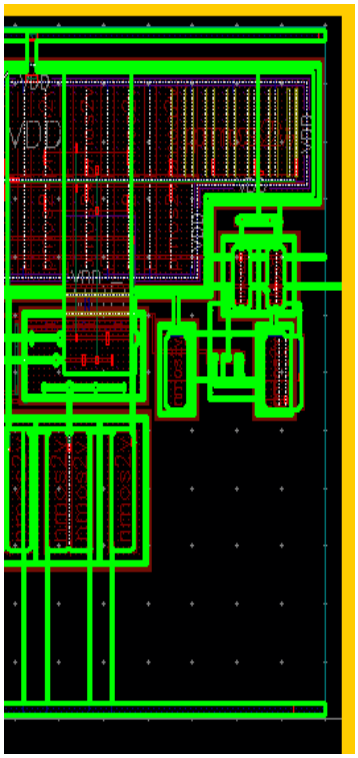
Halo – Thick width ON, halos displayed are thicker



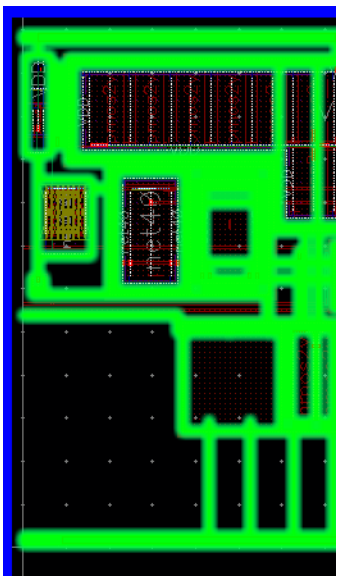
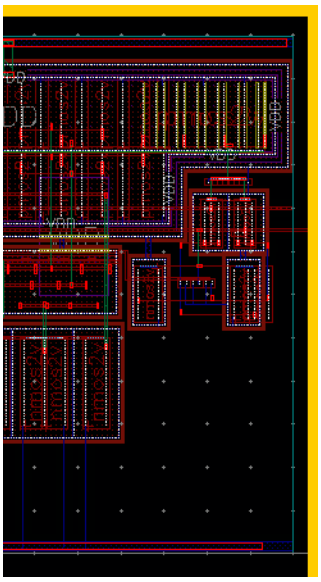
Halo – Thick width OFF, halos displayed are thinner



© 2023 Cadence Design Systems, Inc.

Field	Description
Fadeout shapes	Creates a halo that is faded out. By default, halos are not faded out. You can use the *Transparency* slider at the bottom of the form to control the transparency of the displayed halo. Environment variable: netTracerHaloFadeoutShapes *Halo – Fadeout shapes* ON  *Halo – Fadeout* 

© 2023 Cadence Design Systems, Inc.

Field	Description
Transparency	Enables you to control how transparent the displayed halo can be. The effect of this option is visible when the *Fadeout shapes* option is selected. By default, halos are set to be *50* percent transparent. But, you can move the slider from left to right to vary the halo transparency from *0* percent to *100* percent, as displayed in the figure below. Halo transparency set to 0 percent  Halo transparency set to 100 percent 

Related Topics

Display Options

Net Tracer Options Form

Net Tracer Custom Setup Form

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Net Tracer Neighbors Form

Use the Net Tracer Neighbors form to specify the settings that the Net Tracer uses for finding the neighboring shapes of a traced shape.

The form is available through the *Connectivity – Net Tracer – Neighbors* command.

Field	Description
Neighbors	Specifies the layer on which the Net Tracer finds the neighboring shapes of a traced shape. ■ *On same layer as traced shape*: Finds neighboring shapes on the same layer as the traced shape. ■ *On any extractable layer*: Finds neighboring shapes on any extractable layer. Neighboring shapes need not exist on the same purpose as the traced shape. Environment variable: netTracerNeighborsOnLayer
Distance	Specifies the distance from a traced shape within which the Net Tracer finds the neighboring shapes of a traced shape.
Compute net name	Layout EXL or a higher tier, enables net name computation for the neighboring shapes of a traced shape. Environment variable: netTracerComputeNeighborNet

© 2023 Cadence Design Systems, Inc.

Field	Description
Compute From	Layout EXL or a higher tier, specifies the method used for computing net names for the neighboring shapes of a traced shape. ■ *Logical*: If a neighboring shape already has a net name, the same net name is used. In a hierarchy, the top-most net is used as the net name for a neighboring shape. If a neighboring shape does not already have a net name, no net name is assigned to the shape. ■ *Logical and Physical*: If a neighboring shape already has a net name assigned, the same net name is used. Otherwise, the neighboring shape is physically traced until a shape with a net name is found, and that net name is used for the neighboring shape. Environment variable: netTracerComputeNeighborNetMethod

Related Topics

Finding Neighboring Shapes of a Traced Shape

Net Tracer Options Form

Net Tracer Display Options Form

Layout XL Forms

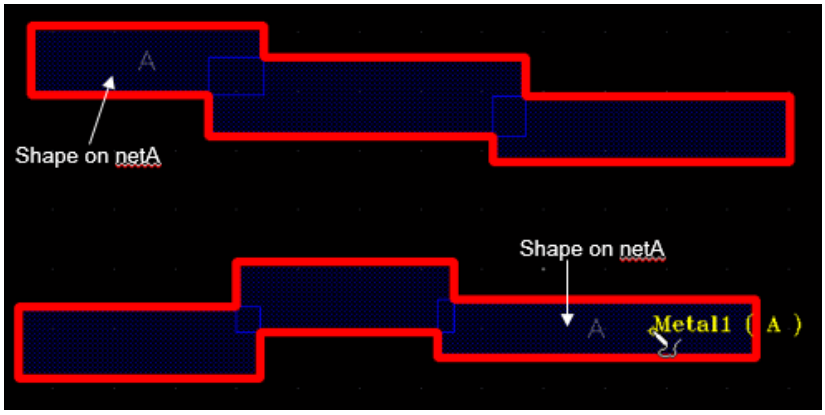
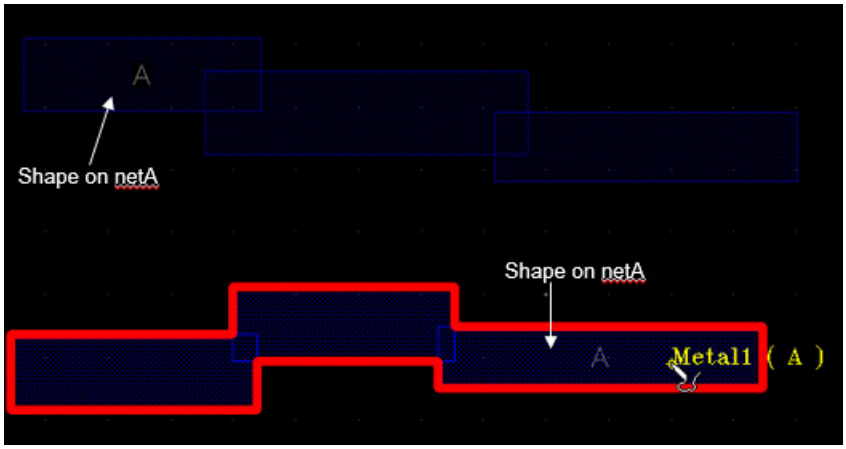
© 2023 Cadence Design Systems, Inc.

Net Tracer Options Form

Use the Net Tracer Options form to control the mode of tracing, the cellview region, and the Layout XL constraint group for creating the net traces, and to specify your preview preferences.

Field	Description
Tracing Mode	This section lets you specify tracing mode.
Physical	Lets you add traces to highlight how the nets are physically connected in a design.
Logical	Lets you add traces to highlight how the nets are connected with the source schematic.
Cross-highlight net in schematic	Cross-highlights a net in the schematic cellview when that net is traced in the layout cellview. Environment variable: netTracerCrossHighlightNetInSchematic
Step mode	Lets you progressively highlight a trace in the layout canvas, each step tracing an additional set of shapes. You can choose to navigate back and forth along a step trace, and the canvas display of the trace changes accordingly. If both *Step mode* and *From net* are selected, *Step mode* is set as the preferred tracing mode. Environment variable: netTracerStepMode
From net	Lets you add traces to highlight all the shapes that are physically connected on the same net. Even with *From net* selected, the Net Tracer may not highlight all the islands on a net, if the traced shape does not contain the required net information. Before you click the shape to generate the trace, move the cursor over the shape to see if the information balloon displays both the layer name and the net name. If only the layer name is displayed, the shape in question may not have the net information available. So, tracing will be performed assuming *From net* is deselected. If both *Step mode* and *From net* are selected, *Step mode* is set as the preferred tracing mode. Environment variable: netTracerFromNet

© 2023 Cadence Design Systems, Inc.

Field	Description
	Net trace with *From net* - ON, Preview displays all the islands on netA  Net trace with *From net* - OFF, Preview displays only 1 island on netA 
Trace through pcells	Allows traces to pass through Pcells. This option is enabled only when Physical trace mode is selected. By default, the option is ON.
Highlight incrementally	Highlights the traced nets for connected shapes in increments of 10K shapes. For example, if a net has 50K connected shapes, the trace will be formed incrementally five times with 10K shapes at a time, until all the shapes are traced. Environment variable: netTracerUpdateIncrementally
Preview	This section lets you specify options for preview.

© 2023 Cadence Design Systems, Inc.

Field	Description
Pop up selection window	Displays the *Choose LPP to Trace* pop-up box when you click at multiple overlapping shapes. The pop-up box allows you to choose the layer-purpose pair for creating the trace. Environment variable: netTracerPopupSelectionWindow
Show preview	Displays a highlight dynamically around the shape over which the pointer is hovered. In addition, the preview displays the layer and net name on which the trace will be created, if the shape being hovered over is clicked. Irrespective of the value set in the *Cut Shapes Extraction* field, for performance reasons, a small number of cut shapes is chased and displayed in the preview. To cycle through the layers being hovered, use the *Spacebar* bindkey. Environment variable: netTracerShowPreview
Color	Lets you display each new trace in a color from the palette. If you choose Cycle, the first trace gets the color at the top of the palette. Any subsequent traces that are added get the next color in the sequence. Alternatively, you can choose a specific color for all the traces. Environment variable: netTracerColor
Region	This section lets you control the region in which nets are traced.
Hierarchy Range	Controls how deep in the hierarchy a trace is created. Choose from one of the predefined ranges, or choose *User defined* to set a hierarchy start and stop level. Environment variable: netTracerRange After a trace is created, you can use the *Display Depth* field in the Net Tracer Display Options Form to control the depth up to which the shapes belonging to the trace will be highlighted.
Top to bottom	Creates a trace on the selected net that passes through the entire hierarchy of the cellview.

© 2023 Cadence Design Systems, Inc.

Field	Description
Current to bottom	Creates a trace on the selected net that passes from the current level of hierarchy to the level specified in the *Display Level To* field in the Display Options form (stop level).
Top to current	Creates a trace on the selected net that passes from the level specified in the *Display Level From* field in the Display Options Form (start level) to the current level of hierarchy.
Top to stop level	Creates a trace on the selected net that passes from the top level in the hierarchy to the level specified in the *Display Levels Stop* field in the Display Options form (stop level)**.**
Current to stop level	Creates a trace on the selected net that passes from the current level in the hierarchy to the level specified in the *Display Levels Stop* field in the Display Options Form (stop level).
User defined	Enables the *Start* and *Stop* fields. Use these fields to set the hierarchy start and stop levels for tracing nets.
Start	Sets the hierarchy level at which the tracing starts. This field is editable only when *Hierarchy Range* is set to *User defined*. Environment variable: netTracerStartLevel
Stop	Sets the level at which the tracing stops. This field is editable only when *Hierarchy Range* is set to *User defined*. Environment variable: netTracerStopLevel
Region	Controls the area in which nets are traced. Environment variable: netTracerRegion
Entire cellview	Traces nets in the entire cellview. This option is selected by default.
Viewing area	Traces nets in the portion of the cellview that is displayed in the canvas at that instant.
Custom	Lets you specify a rectangular or polygonal area for tracing nets. Click the appropriate button to specify the required area shape—rectangular or polygonal— and then click in the canvas to specify the vertices of the selected shape.
Cut Shapes Extraction	Controls the number of cut shapes that are traced per overlap. Use the slider to choose the number of cut shapes that are traced.

© 2023 Cadence Design Systems, Inc.

Field	Description
Low	Traces 100 cut shapes.
Medium	Traces 1000 cut shapes. This is the default.
High	Traces 10000 cut shapes.
Full	Traces all the cut shapes. Environment variable: netTracerMaxCutShapes
Connection Setup	This section lets you specify the connection setup for the layer.
Source	Lets you choose a constraint group that defines the via layer, connected layer, and stop layers to be used for creating the traces. By default, the `virtuosoDefaultExtractorSetup` constraint group is used as the source.Alternatively, you can choose *Custom* to manually specify the source settings. See Net Tracer Custom Setup Form. Environment variable: netTracerConstraintGroup
Stack Layers	Lets you specify a start layer and an end layer to narrow down to a smaller set of extractable layers to be used for tracing. The same layer can be used to define the start and end layer for tracing. This option is useful when creating a big trace, such as for a power or ground net.

Related Topics

Display Options Form

Net Tracer Custom Setup Form

Net Tracer Display Options Form

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Open Physical Configuration View Form

Use the Open Physical Configuration View form to open an existing physical configuration view to be used in a Layout XL session.

Field	Description
Library Name	Specifies the name of the library in which the physical configuration view resides.
Cell Name	Specifies the name of the cell to which the physical configuration view relates.
View Name	Specifies the name of the physical configuration view.
Open CPH	Opens the Configure Physical Hierarchy window automatically after you click *OK* in this form. Environment variable: cphLayoutSync

Related Topics

Physical Hierarchy Configuration

Layout XL Launched from a Schematic View

Starting Layout XL with an Existing Physical Configuration View

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Permutation Information Form

The Permutation Information form specifies you what changed as a result of using the *Permute Pins* command. Display it by pressing `F3` after you have run the command.

Related Topics

Pin Permutation

Checking Permutation Information

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Save Short Locator View Form

Lets you specify the view name to use for saving the Short Locator view. This is useful to resume the net analysis from where you left it last:

- Any virtual labels that you added are saved and restored on load.
- Because the net is saved, for a large net this allows resuming work much faster as no shape-chasing is required on load.

Related Topics

Locating Shorts Using Short Locator

Short Locator Form

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Save Traced Nets View Form

Use the Save Traced Nets View form to save all the visible traces and their locked mask colors, where available, to a layout cellview. To mark the traces as visible, use the Trace Manager form.

Field	Description
Library	Specifies the name of the design library to which the visible traces in the current layout library cellview will be saved.
Cell	Specifies the name of the cell to which the visible traces in the current layout library cellview will be saved.
View	Specifies the name of the view to which all the visible traces in the current layout library cellview will be saved.
Method	Lets you specify the method used for saving traces. ■ *One view per trace*: Saves each visible trace as a separate view. This helps to optimize the performance in case of a large trace that has lots of figures. ■ *One view for all traces*: Saves the visible traces into a single view. ■ *Auto*: Lets the system to automatically determine the fastest method to be used for saving traces. ❑ If only one trace is visible, the *One view per trace* method is used. ❑ If at least one visible trace is large, the *One view per trace* method is used. ❑ If all visible traces are small, the *Saving one view for all traces* method is used. Environment variable: netTracerSaveMethod

Related Topics

Saving All Visible Traces

Load Traced Nets View Form

Trace Manager Form

© 2023 Cadence Design Systems, Inc.

Net Tracer Options Form

Net Tracer Display Options Form

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Selection Options Form

Use the Selection Options form to control how selection is performed in the layout window.

You can choose between full and partial section modes, control the behavior of point-and-click and area selection functionality, specify automatic selection expansion for routing objects, and control the size and number of selected sets that can be saved in the system for later reuse.

Field	Description
Selection Controls	This section lets you specify options for selection control.
Mode	Defines whether selection is full or partial. ■ *Full*: Selects objects fully inside the area selection box. ■ *Partial*: Selects objects fully if you completely enclose them with the area selection box, and partially selects objects if you partially enclose them with the area selection box. In *Partial* selection mode, the *Routing Object Granularity* cyclic field is not available.

© 2023 Cadence Design Systems, Inc.

Field	Description
Routing Object Granularity	Helps select routing objects by automatically expanding the selection based on the modes described below. You can also set these modes from the right mouse button shortcut menu in the layout canvas. Alternatively, you can choose an appropriate mode by clicking the *Selection Granularity* icon  in the Options toolbar. ■ *Shapes or Vias*: Selects a via or shape (either a path, path segment, rectangle, or polygon) used to make a wire. In *Partial* selection mode, this is the only granularity that is available. When zoomed out, a single click selects an individual shape or via. Two clicks extends the selection to include all the connected pathSegs in the wire until a T-junction or via is encountered. When zoomed in, clicking in the middle of the shape selects only that shape; clicking on the boundary of the shape selects all the connected pathSegs in the wire until a T-junction or via is encountered. ■ *Entire Wire*: Expands the selection automatically to include all the objects forming the wire when you select a shape or via. ■ *Connected Shapes*: Expands the selection automatically to include all the objects with a physical connection to a selected shape or via, except labels. The *Connected Shapes* mode does not consider net assignments on shapes and vias. The expansion is abandoned if more than 1000 objects are found – in this case, only the object under the mouse is selected. ■ *All Shapes on Net*: Expands the selection automatically to include all the objects assigned to the same net as a selected shape or via. If the selected shape or via has no net assigned, then the selection is not expanded. The expansion is abandoned if there are more than 20,000 objects assigned to the net – in this case, only the object under the mouse is selected.

© 2023 Cadence Design Systems, Inc.

Field	Description
	■ *Net*: Lets you select nets as well as instances from the canvas. When a net is selected on the canvas, the corresponding net in the Navigator assistant is also selected. The selected net is probed in the layout window according to the options set in the Highlight Options Form. The *Net* granularity behaves similar to *All shapes on Net* except that in this case the net object is selected instead of the shape. In fact, there is no limit to the number of shapes that can be highlighted. However, if a shape has no assigned net, no shapes are highlighted. For performance reasons only *Shapes or Vias*, *All Shapes on Net*, and *Net* modes are supported for area selection. If you perform an area selection when the routing granularity is set to either *Entire Wire* or *Connected Shapes*, the selection defaults to Shapes or Vias mode.
Spine	Selects all contiguous pathsegs automatically with the same width as the pathseg on which you click. It excludes pathsegs that branch off from the main spine at a T junction This option is switched on by default; if you switch it off, only the pathSeg you click is selected. This option is enabled only when *Routing Object Granularity* is set to *Shapes or Vias*.
Point Selection Controls	This section lets you specify options for point selection control.
Overlap Mode	Lets you select overlapped objects one by one by using the left mouse button to select any object and continuing to select them by clicking on the original object until you reach the largest overlapped object.

© 2023 Cadence Design Systems, Inc.

Field	Description
Toggle mode	Lets you change selection between two objects. **1.** Click using the left mouse button on one object and the software toggles back and forth between the smaller object and the larger object. **2.** Add objects to the selected set by holding down the `Shift` key and selecting additional objects. **3.** Delete selections by holding down the `Ctrl` key and clicking the left mouse button on the original selected object. **4.** Objects are deselected in the reverse order to which they were selected. **5.** Once you have selected the objects you want in the set you can also click the selected edge of one of the objects. This deletes all the selections except the one you have clicked on. This does not delete the objects themselves.

© 2023 Cadence Design Systems, Inc.

Field	Description
Cycle mode	Use *Cycle* mode when three or more objects overlap at the same point. Cycle mode always cycles from the smallest object to the largest object. If you continue to click the original object after reaching the largest object, the cycle returns to the smallest object you first clicked on. The cycle repeats by re-selecting the smallest object you clicked on and selecting outwards again to the largest object. If you start by selecting the largest object in the overlapping stack the selection will not cycle back to the smaller objects. You must select a smaller object so that each selected point selects larger and larger objects. Multiple objects are not retained in the selected set. The final selected object is the selection that is retained. You can deselect by clicking a point outside of all the overlapping objects. Add objects to the selected set by holding down the `Shift` key and selecting additional objects. Delete selections by holding down the `Ctrl` key and clicking the left mouse button on the original selected object. Objects are deselected in the reverse order to which they were selected. Once you have selected the objects you want in the set you can also click the selected edge of one of the objects. This deletes all the selections except the one you have clicked on. This does not delete the objects themselves. The objects are selected according to the selection filter set for the Objects panel in the Palette assistant. Cycle mode is window-based.

© 2023 Cadence Design Systems, Inc.

Field	Description
User Aperture Mode	Enables the point selection aperture function and lets you specify the *Size* of the aperture in user units. This helps you select objects that are not directly under the cursor but close to it (within the distance specified by the aperture). ■ *off*: Specifies that point selection does not use the specified aperture. ■ *on*: Specifies that point selection uses the specified aperture. ■ *visible*: Specifies that point selection uses the specified aperture and that the aperture is displayed on the canvas.
Size	Defines a box around the initial point. You can select multiple points within this aperture range and the software will treat it as one point.
Area Selection Controls	This section lets you specify options for area selection control.
Full Mode	Controls area selection in Full selection mode. ■ *Enclosed Figures*: Selects the objects you completely enclose with the area selection box. ■ *Enclosed and Crossed Figures*: Selects the objects you completely or partially enclose with the area selection box. ■ *Crossed Figures*: Selects objects you partially enclose with the area selection box, but does not select any fully-enclosed objects.

© 2023 Cadence Design Systems, Inc.

Field	Description
Partial Mode	Controls area selection in Partial selection mode. ■ *Vertex*: Selects all the objects you completely enclose with the area selection box, vertices of any objects you partially select with the area selection box, and edges of any objects you enclose with the area selection box. ■ *Enclosed Edges*: Selects all the objects you completely enclose with the area selection box, and the edges of any objects you partially enclose with the area selection box. ■ *Crossed Edges*: Selects the edges of any objects you enclose with the area selection box, and does not select fully-enclosed objects. ■ *Enclosed and Crossed Edges*: Selects all the objects you completely enclose with the area selection box, the vertices of any objects you partially enclose with the area selection box, and the edges of any objects you enclose with the area selection box.
Auto Store Selection Controls	Stores multiple selected sets you have created. You can reload a previously-selected set into the cellview without having to manually reselect the objects in the canvas.

© 2023 Cadence Design Systems, Inc.

Field	Description
Auto Store	Enables the auto store functionality. The default is off. ■ The selected set consists of full or partially selected figures. ■ The selected set is a cellview property. The *Edit – Select – Save/Restore* is not cumulative.Once the cellview is closed the recorded information is not retained. ■ You can iterate through the sets by selecting the sets in the in the Save/Restore Selection Set form or by using the *Edit – Select – Next* or *Edit – Select – Previous* commands. ■ Deleted or modified objects are removed from a saved selection set. If all objects are removed from the selection set, the set is removed from the Save/Restore Selection Set form. ■ Restore is independent of the object selectability of the *Objects* panel in the Palette assistant. ■ The command is not compatible with the *Undo* or *Redo* commands. ■ The command does not work with the *Edit In Place* or *Descend* commands if the lower-level cellview is already open.
Max number of Stored sets	Limits the number of sets saved in the Save/Restore Selection Set form. The default is 10, which is also the maximum number that can be stored. When the limit is reached, the oldest saved selection set is removed from the list.
Max number of selected elements within a set	Limits the number of objects you can save in a single selection set. The default is 100, which is also the maximum number that can be saved.

Related Topics

Object Editing in Layout XL

Highlight Options Form

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Set Net Order Form

Use the Set Net Order form to create an ordered net constraint on a group of nets. An ordered net constraint is honored by the pin optimizer engine, which optimizes the nets while maintaining the order specified by the constraint.

Field	Description
Net Order Name	Specifies a name for the ordered net constraint. If you do not specify a name, the system generates one for you.
Selected Nets	Lists the nets that are currently selected in the layout canvas.
Nets Name	Finds nets in the list of selected nets. Type in a string and click *Filter* to see only the nets you specified.
Ordered Net Members	Lists the names of the nets in the constraint.
Add	Adds one or more selected nets to the constraint.
Remove	Moves one or more selected nets out of the constraint.

Related Topics

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Set Pin Label Text Style Form

Use the Set Pin Label Text Style form to set the style of the labels used to annotate pins in the design window.

Field	Description
Height	Sets the height of the label.
Font	Sets the text style of the label.
Text Options	■ *Drafting:* Prevents the label from being rotated more than 90 degrees. ■ *Overbar:* Determines how text strings containing underscore characters are displayed in a layout window. The default is off, which means that the software displays underscore characters (_) as part of the text string. When switched on, the software interprets underscore characters in the text string name as toggle switches that control where overbars begin and end. Overbars appear above the text string, as shown in the examples below.
Layer Name	Sets the layer on which the labels are generated. Choose a layer from the cyclic field or select *Same As Pin* to create the label on the same layer as the pin with which it is associated. Environment variables: pinTextSameLayer,pinTextLayerNames

Text string	Appears in design window as
_abcde	abcde
ab_cde	abcde
_abc_de	abcde
ab_cd_ef_gh_ij	abcdefghij

© 2023 Cadence Design Systems, Inc.

Field	Description
Layer Purpose	Sets the layer purpose on which the labels appear. Choose a purpose from the cyclic field or select *Same As Pin* to create the label on the same layer purpose as the pin with which it is associated. Environment variables: pinTextSamePurposed, pinTextPurposeNames
Justification	Sets the location of the label origin. The origin appears as a small square on the label when you place or select it. Choose from *lowerLeft*, *centerLeft*, *upperLeft*, *lowerCenter*, *centerCenter*, *upperCenter*, *lowerRight*, *centerRight*, and *upperRight*.
Orientation	Lets you specify the orientation of the labels. Choose orientation from: ■ `R0`: No rotation ■ `R90`: Rotate 90 degrees clockwise ■ `R180`: Rotate 180 degrees clockwise ■ `R270`: Rotate 270 degrees clockwise ■ `MY`: Mirror over the Y axis ■ `MYR90`: Mirror over the Y axis and rotate 90 degrees clockwise ■ `MX`: Mirror over the X axis ■ `MXR90`: Mirror over the X axis and rotate 90 degrees clockwise Environment variable: pinTextOrientation

Related Topics

Generating All Components from Source

I/O Pin Generation

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Short Locator Form

Use the Short Locator form to locate the error that is causing a short.

The form contains two tabs.

Tab	Description
General	Displays the shorted net names and lets you specify how the shapes are displayed and lets you add virtual labels.
Preferences	Lets you specify the display preferences such as the color for shorted nets and for the common and other shapes.

General

The following table describes the fields available on the *General* tab of the Short Locator form.

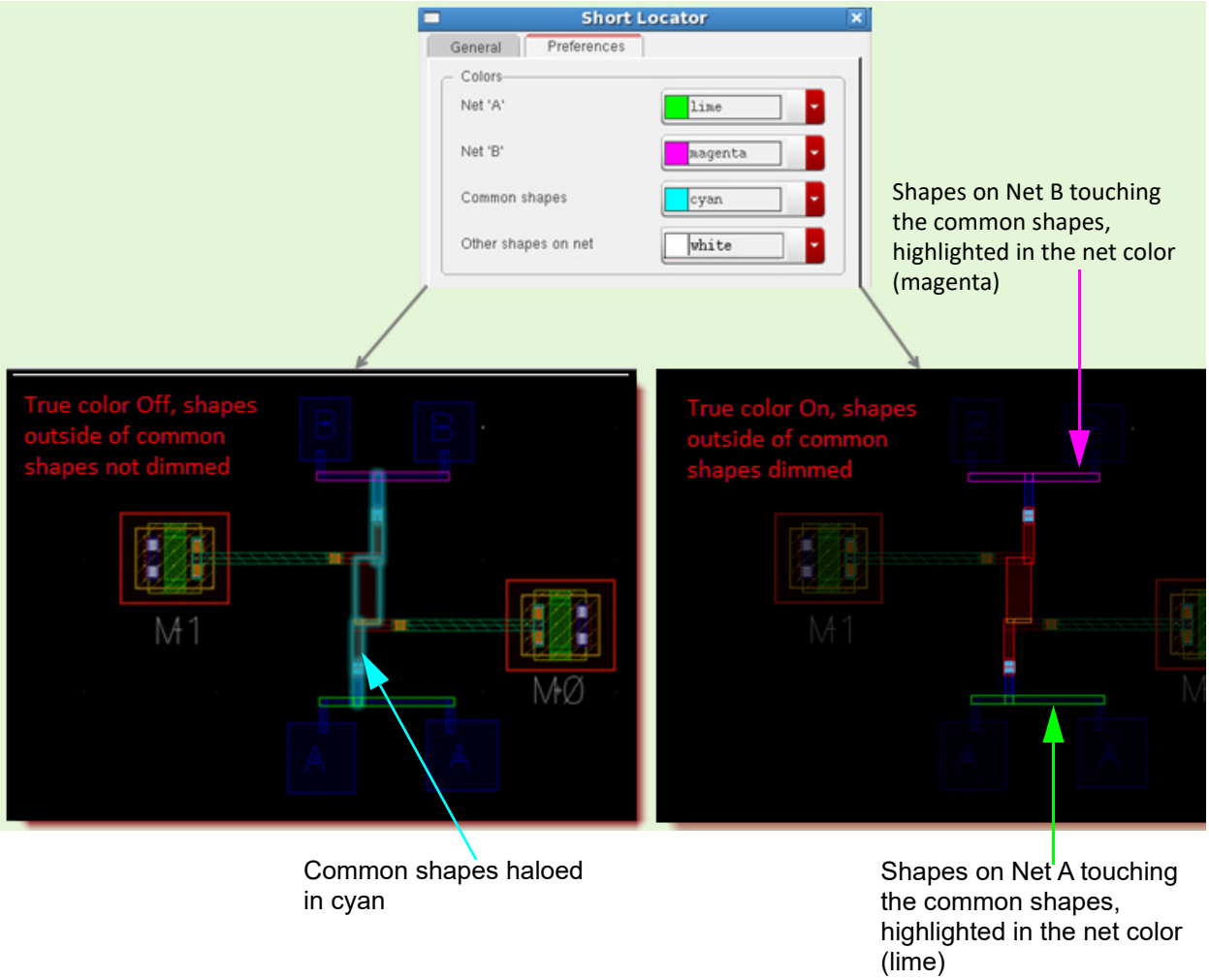
Field	Description
Area	Lets you define an area in which the Short Locator is to operate.
Run	Lets you run the Short Locator command on the net of the clicked shape.
Select Area	Lets you select the area on the canvas. When you specify an area, its coordinates are added to the text box.
Shorted Nets	This section shows the name of the shorted nets and the color used to mark the shapes on each net.For each shorted net, the text field below the net name indicates the number of associated pins (P), labels (L), virtual labels (VL), and instance pinFigs (IP), as displayed in the figure below.
Use instance pins	Lets instance pinFigs be considered as sticky points for the shorted nets, in addition to pins, labels, and virtual labels. When the connectivity of devices is known to be correct, using instance pinFigs as sticky points can help locate shorts faster.

© 2023 Cadence Design Systems, Inc.

Field	Description
Virtual Labels	■ *Add:* Lets you add a virtual label on one of the shorted nets. Adding virtual labels helps narrow down the common shapes, which eventually helps identify the shape that is causing the short.After each virtual label is added, the Short Locator recomputes the number of common shapes. ■ *Add New Name:* Lets you add a virtual label on a new net name. ■ *Delete*: Lets you delete the virtual label from a specific net. ■ *Delete All*: Lets you delete all the virtual labels.
Display	This section lets you select the mode in which the shapes are displayed in the layout canvas. You can also choose to display a diamond shape in the layout canvas for each pin, label, and virtual label in the layout. Environment variable: extractShortLocatorDisplayMode
Common shapes displays	Common shapes haloed in the color selected for common shapes. Shapes on a shorted net that are touching the common shapes highlighted in the color associated with that net. If *True color* is selected, common shapes are displayed in their true color and the other shapes are dimmed, as illustrated in the figure below.

© 2023 Cadence Design Systems, Inc.

Field	Description



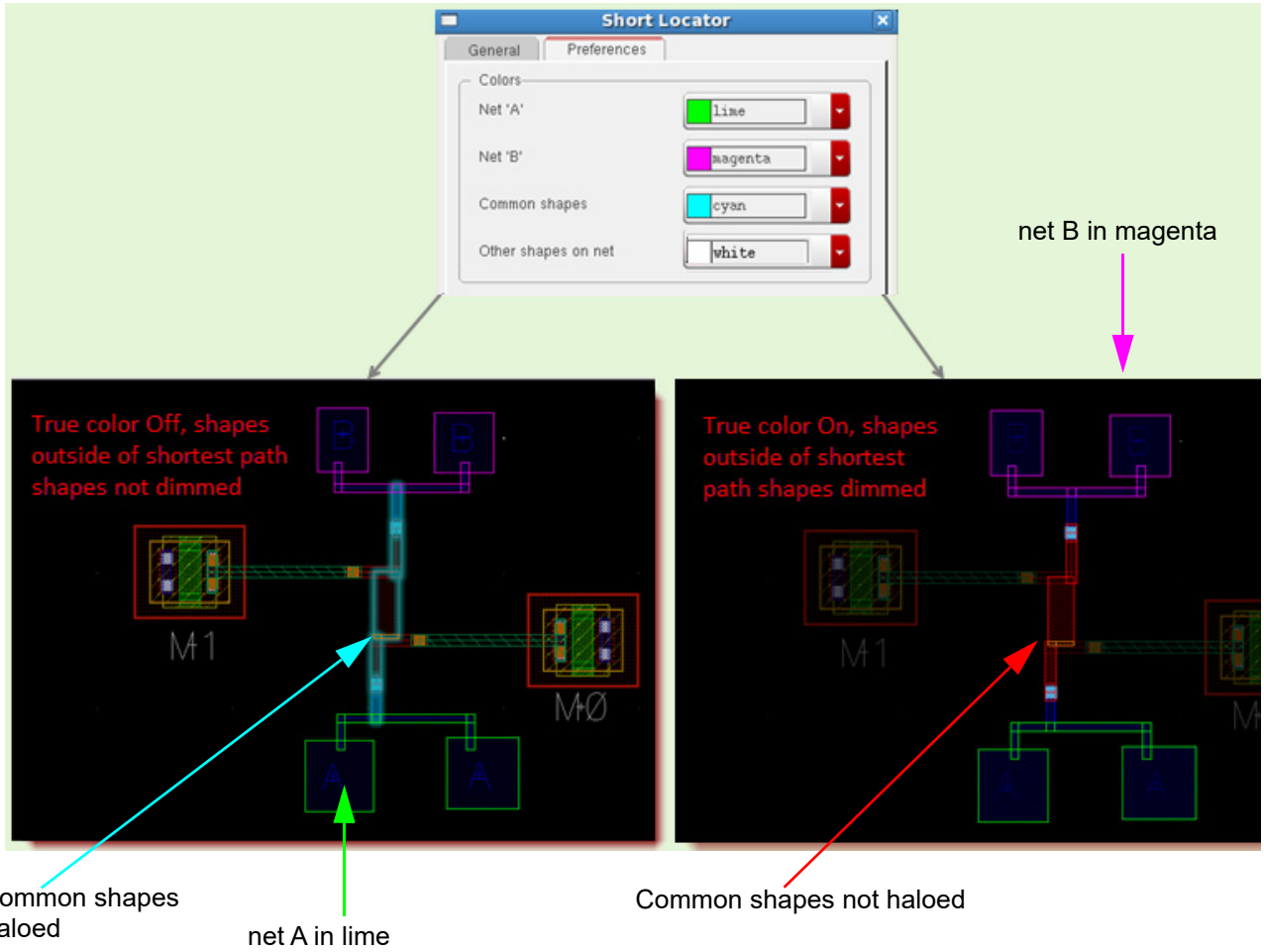
All shortest paths displays

- Common shapes haloed in the color selected for common shapes.
- Shapes on a shorted net highlighted in the color associated with that net.

If *True color* is selected, shapes on the shortest paths are displayed in their true color and the other shapes are dimmed, as illustrated in the figure below.

© 2023 Cadence Design Systems, Inc.

Field	Description



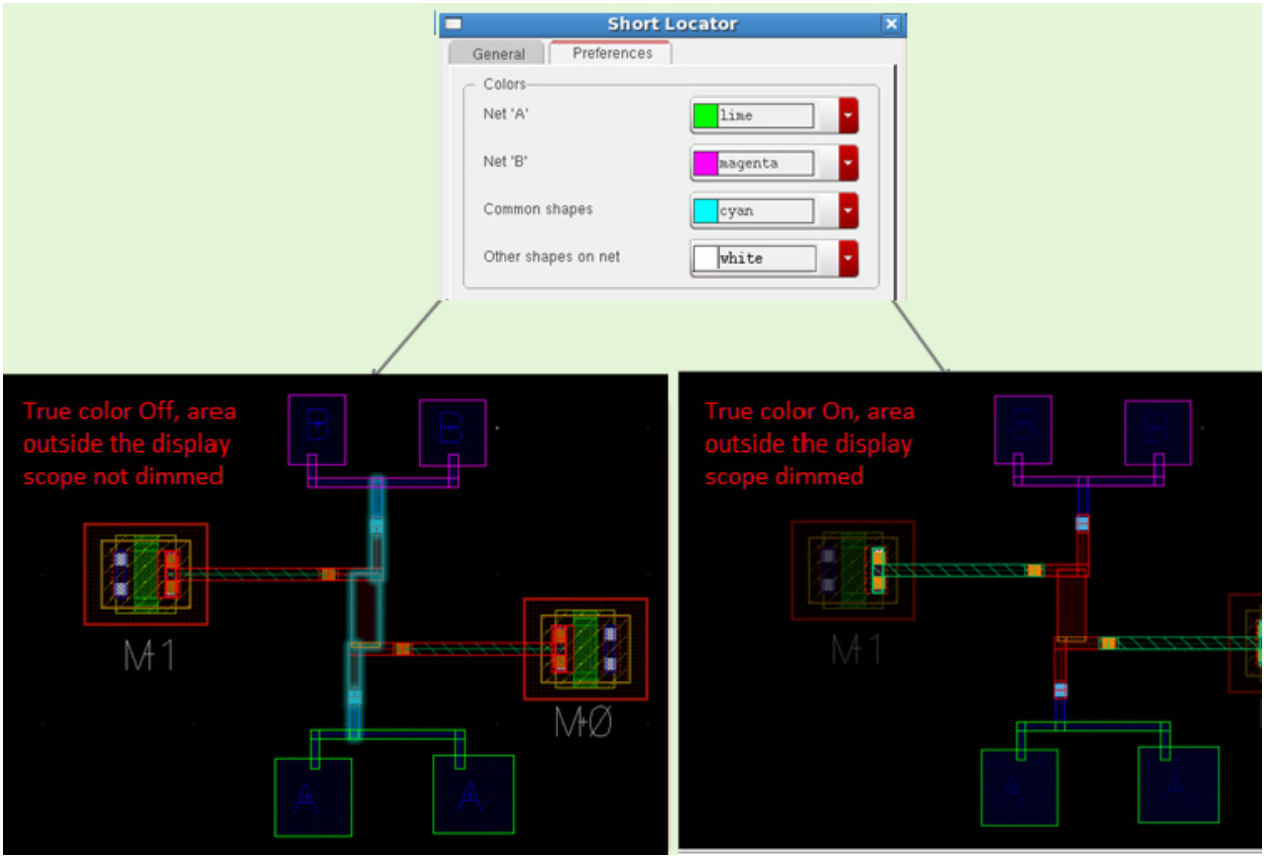
Entire net

Common shapes haloed in the color selected for common shapes. Shapes on the shortest paths highlighted in the color selected for the corresponding shorted net. Other shapes on the net highlighted in the color selected for other shapes.

If *True color* is selected, all shapes on the net are displayed in their true color and shapes not on the net are dimmed, as illustrated in the figure below.

© 2023 Cadence Design Systems, Inc.

Field	Description



Compute	Calculates the shortest paths and common path. This button is enabled only when `Ctrl+C` is pressed to stop chasing.
Toggle	Enables you to quickly switch the layout display between the *Entire net* display mode and one of the other two modes — *Common shapes* or *All shortest paths*, which ever was last used.
Zoom to Middle	Zooms to the middle common shape. Zooming to middle can help perform a dichotomy to locate the error causing the short for a design that has a large number of common shapes.
Zoom to Fit	Zooms to fit all the shapes in the selected mode—common shapes, shapes on all the shortest paths, or shapes on the entire net—in the available layout window.

© 2023 Cadence Design Systems, Inc.

Field	Description
True color	Displays shapes corresponding to the selected display mode in their true color. The *True color* view makes it easy to see the shapes of interest as all the other shapes are dimmed. Environment variable: extractShortLocatorShowPinsAndLabels
Show pins and labels	Displays a diamond shape above each pin and label. Environment variable: extractShortLocatorShowPinsAndLabels
Show instance pins	Displays a diamond shape for instance pinFigs. Environment variable: extractShortLocatorOtherShapesColor

Preferences

The following table describes the fields available on the *Preference* tab of the Short Locator form.

Field	Description
Connection Setup	Specifies the constraint group that defines the connectivity stack to be used. By default, the constraint group used by the connectivity extractor is used.
Stack layers	Specifies a start layer and an end layer to narrow down to a smaller set of extractable layers to be used for shape chasing. The same layer can be used to define the start and end layer for shape chasing.

© 2023 Cadence Design Systems, Inc.

Field	Description
Colors	Specifies the highlight colors for the following: ■ The two shorted nets Environment variables: extractShortLocatorNetColor1, extractShortLocatorNetColor2 ■ Other shapes on the net Environment variable: extractShortLocatorOtherShapesColor ■ Common shapes: For common shapes, the section specifies the color to use for creating the halo around the common shapes causing the shorts. Environment variable: extractShortLocatorCommonShapesColor
Auto zoom to fit	Zooms the layout automatically to fit all the shapes in the selected display mode. Environment variable: extractShortLocatorAutoZoomToFit
Show net for virtual labels	Displays in layout canvas a label with the name of the net associated with the virtual labels. Environment variable: extractShortLocatorShowVirtualLabelNetOnShapes
Font size	Specifies the font size for the net used to display the virtual labels. Environment variable: extractShortLocatorVirtualLabelNetOnShapesFontHeight
Highlight limit	Specifies the maximum number of shapes that will be highlighted for a large net. Environment variable: extractShortLocatorHighlightLimit
Save	Opens the Save Short Locator View form that lets you specify the view name to use for saving the Short Locator view.
Load	Opens the Load Short Locator View form that lets you load a previously saved Short Locator view.

© 2023 Cadence Design Systems, Inc.

Related Topics

Locating Shorts Using Short Locator

Layout XL Forms

Save Short Locator View Form

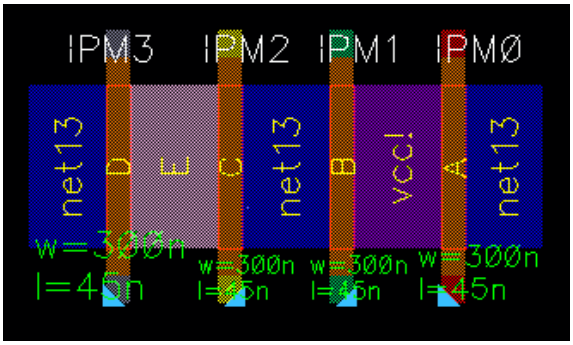
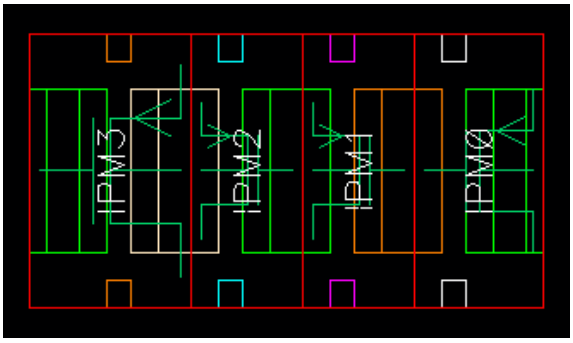
© 2023 Cadence Design Systems, Inc.

Smart Display Options Form

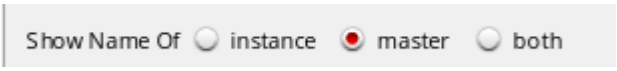
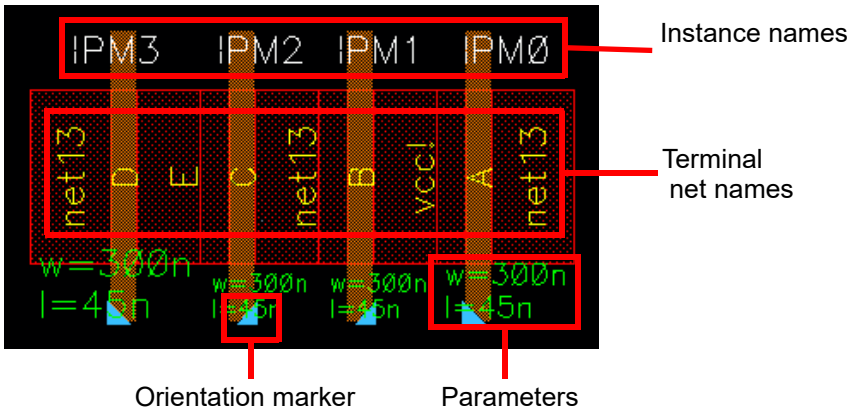
Use the Smart Display Options form to specify the options for symbolic representation of the layout.

Field	Description
Display Mode	Lets you turn off Smart Display or switch between Smart Display modes. The available options are: ■ *Off*: Turns off Smart Display. ■ *Schematic*: Turns on *Schematic* mode. In this mode, devices are represented using their schematic symbols and connections are displayed using symbolic flight lines. The displayed devices could be representing virtual hierarchies, if the design is viewed using Layout XL or a higher tier. ■ *Abstracted*: Turns on *Abstracted* mode for the entire cellview or for the objects selected in the canvas. In this mode, you see an abstract representation of devices.

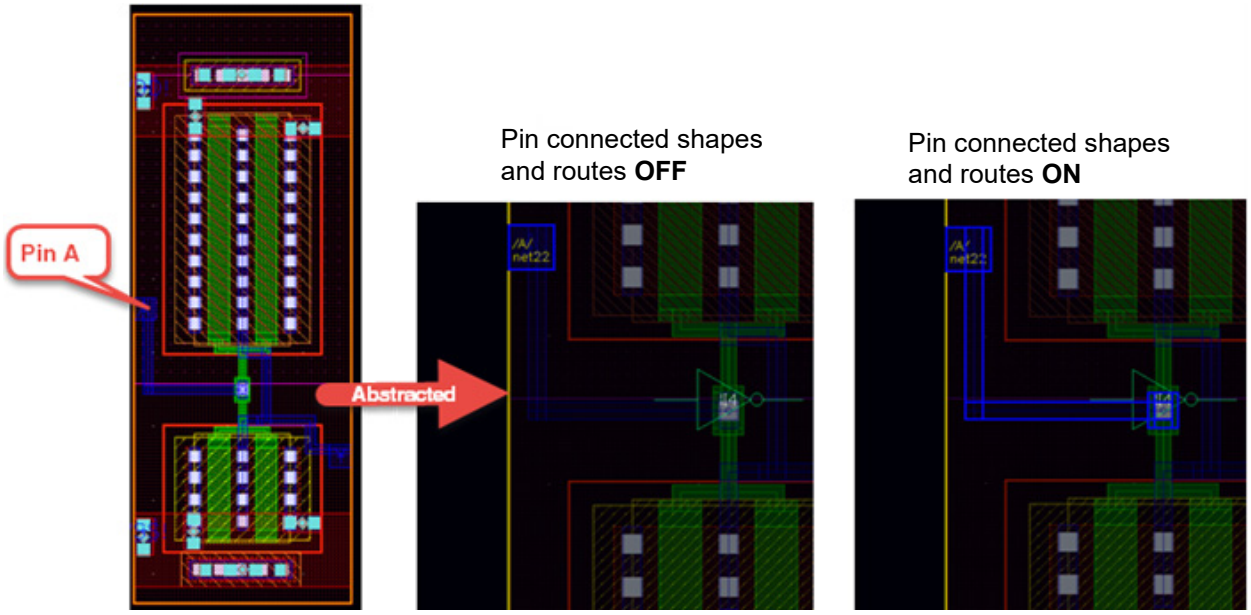
© 2023 Cadence Design Systems, Inc.

Field	Description
Surround Display	This section determines how surrounding objects are displayed. The available options are: ■ *normal*: Displays surrounding objects with dimming intensity at 0 percent. ■ *dim*: Displays surrounding objects with dimming intensity set to the value specified in the Display Options form. ■ *hide*: Displays surrounding objects with dimming intensity at 100 percent. ■ *pins*: Displays all pins on the same net with the same color, as shown below. Surrounding objects are displayed with dimming intensity at 100 percent.  Abstracted mode  Schematic mode

© 2023 Cadence Design Systems, Inc.

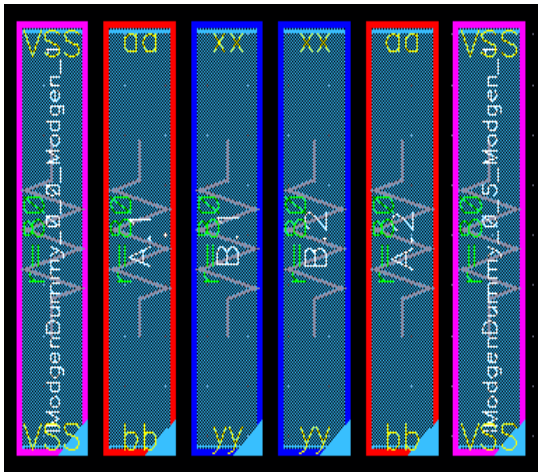
Field	Description
Instance Display	Determines the information to be displayed for instances.
Instance name	Displays instance names. For hierarchical blocks in the layout, when the display level > 0, the instance name always displays. When the display level = 0, the instance name displays only when the *Show Name Of* field in the Display Options form is set to *maste*r. 
Instance terminal net name	Displays the terminal net names for instances.
Orientation marker	Displays the orientation and end markers for devices.
Parameters	Displays the width and length values (parameters) for devices. 
Pin connected shapes and routes	Highlights the pins and same-layer shapes and routes that overlap with the pins. Also displays the associated net labels. The value of this option also impacts the value of *Options – Display Pins – Instance* when displaying the instance pins at level 0. The instance display values described above comprise the smart labels. The size of the smart labels is automatically adjusted when you zoom in or zoom out the design.

© 2023 Cadence Design Systems, Inc.

Field	Description
	
Schematic symbol	Displays the schematic symbol view overlaying the layout instance blocks in *Abstracted* display mode. This option is turned on by default and is grayed out for *Schematic* display mode.
Instance BBox	This section determines how the instance bounding box is displayed.
Colored for mfactored devices	Displays a colored bounding box for mfactored instances with mfactor > 1. The bounding boxes of instances with matching patterns are displayed using the same color.

© 2023 Cadence Design Systems, Inc.

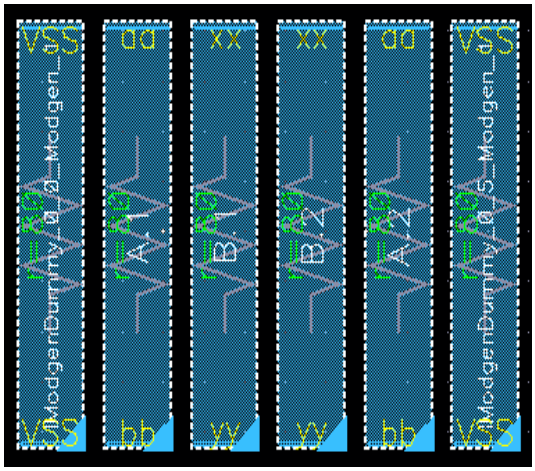
Field	Description
Colored for dummies	Displays a colored (pink) bounding box for dummy devices.



Dummy instance

Mfactored

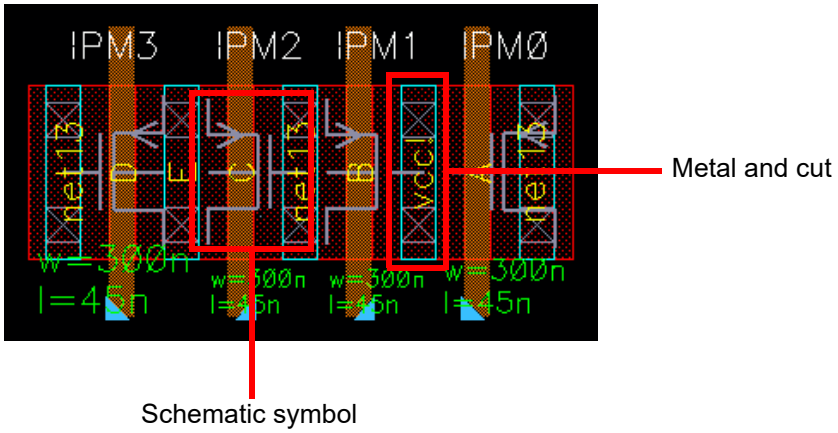
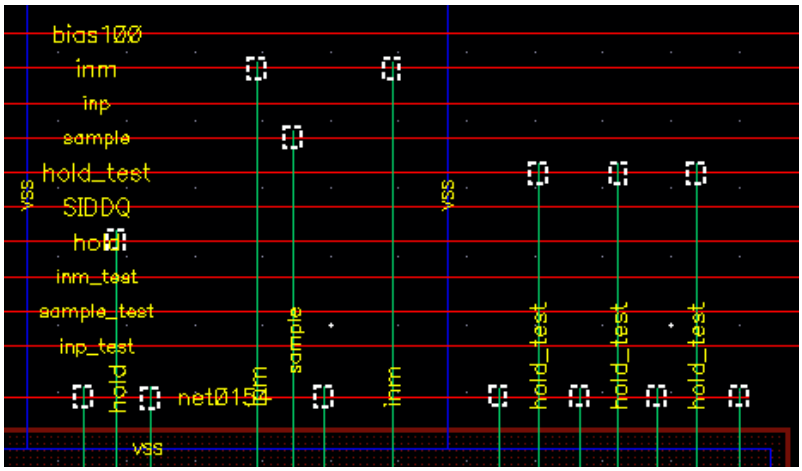
Only visible layers	Displays a bounding box enclosing the device.



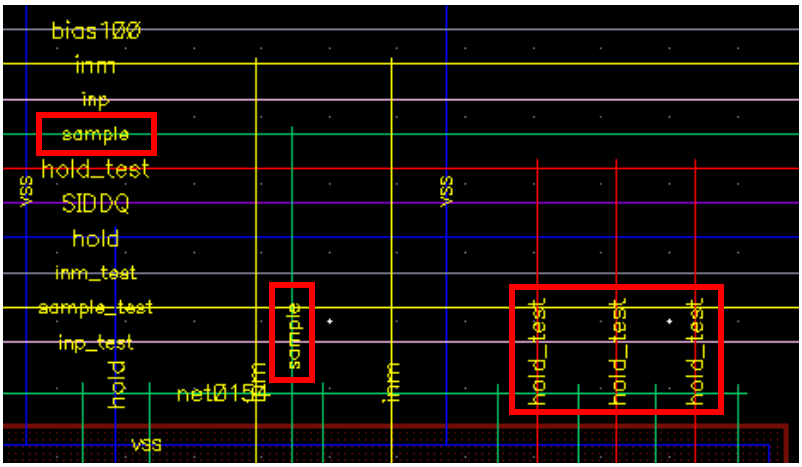
Bounding box
on visible layer

MOS Only	This section applies only to MOS devices.
Schematic symbol	Displays the schematic symbol for MOS devices in *Abstracted* mode.

© 2023 Cadence Design Systems, Inc.

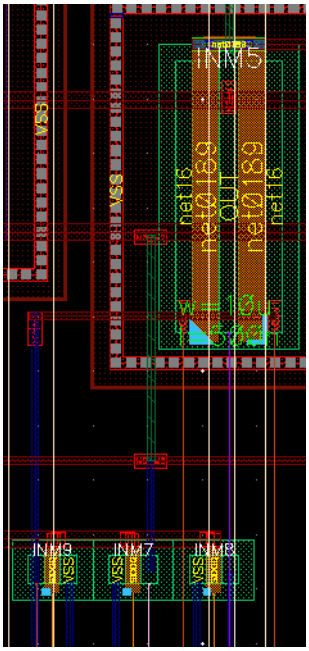
Field	Description
Metal and cut layers	Displays details of objects drawn on metal and cut layers. 
Connectivity	This section determines the connectivity information to be displayed.
Abstract via	Displays vias with a white dotted line, as shown below. Wires are displayed with only their center lines visible. 

© 2023 Cadence Design Systems, Inc.

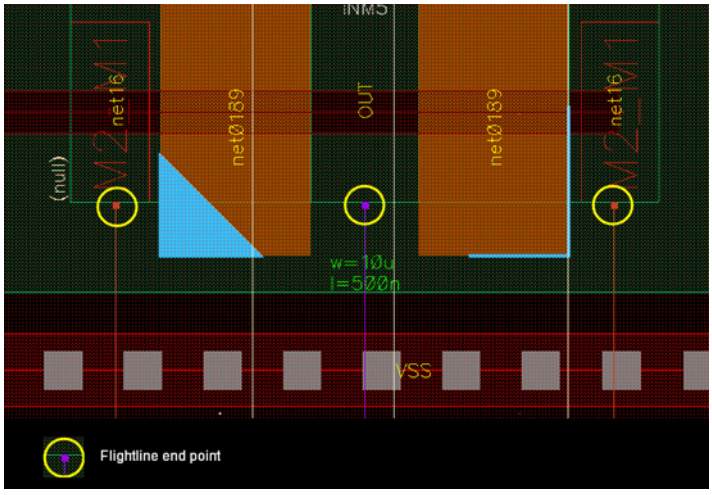
Field	Description
Abstract wire	Displays wires with only their center lines visible. All wires that are on the same net are displayed using the same color. Wires that are not on a net are displayed in gray. Vias are hidden from view. For example, in the figure below, all wires on net `hold_test` are displayed in red and all wires on net `sample` are displayed in green. 

© 2023 Cadence Design Systems, Inc.

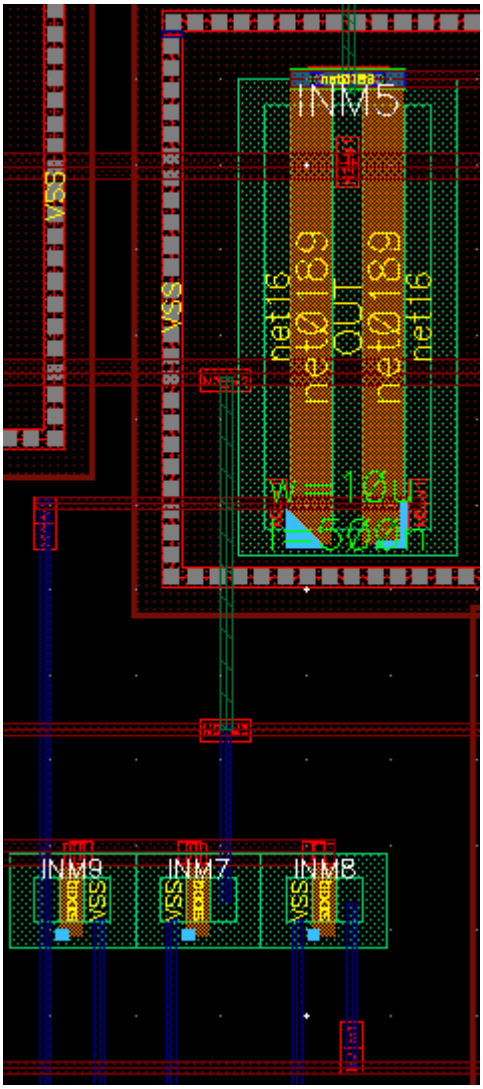
Field	Description
Flight lines	Displays orthogonal flight lines.



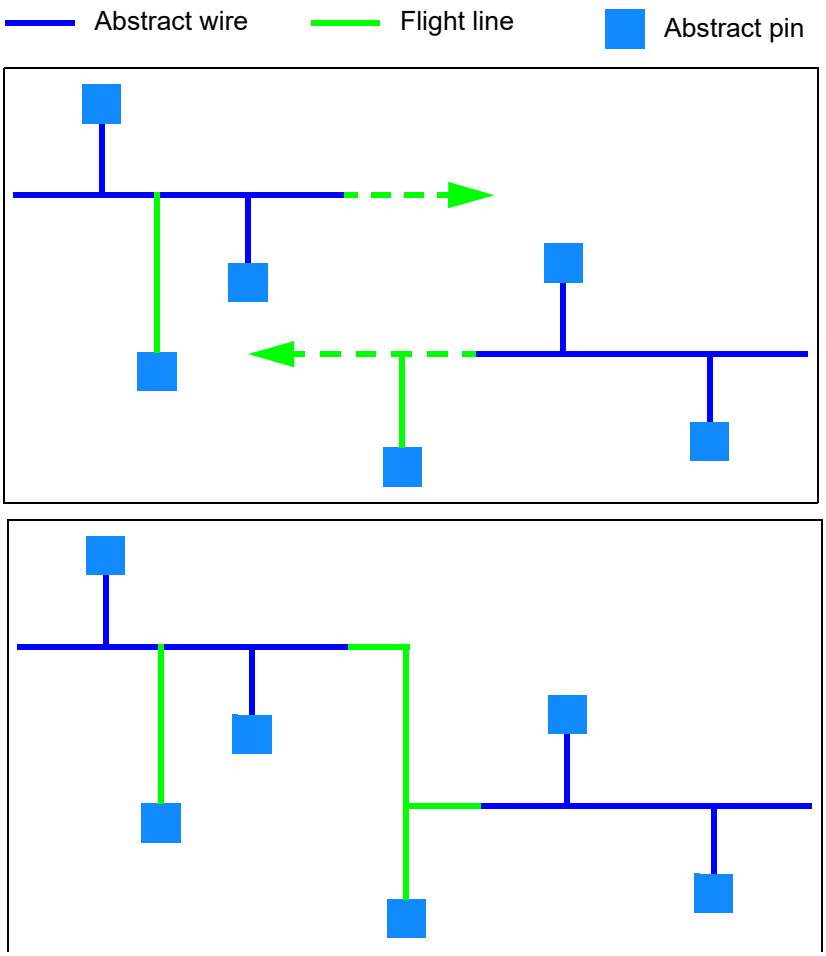
The end points of flightlines are represented using small rectangles as illustrated in the figure below.



© 2023 Cadence Design Systems, Inc.

Field	Description
Flight lines none	Hides orthogonal flight lines. 
Ignore power and ground nets	Prevents the display of flight lines representing power and ground nets, when selected. This option is selected by default and applies only when connectivity is set to *Flight Lines*.

© 2023 Cadence Design Systems, Inc.

Field	Description
Extend existing trunks	Uses flight lines to extend existing trunks on the same net to display connectivity. The trunks are extended in either direction and unconnected pins are connected to these trunks with orthogonal flight lines to complete connectivity. This option is deselected by default and applies only when connectivity is set to *Flight Lines*. As shown in the diagram below, for an incomplete net, Net A, the two horizontal trunks are extended at the ends and the nearest pins are connected to the trunks with flight lines (vertical twigs). The two trunks are also connected with orthogonal flight lines to complete connectivity. 

© 2023 Cadence Design Systems, Inc.

Related Topics

Smart Display

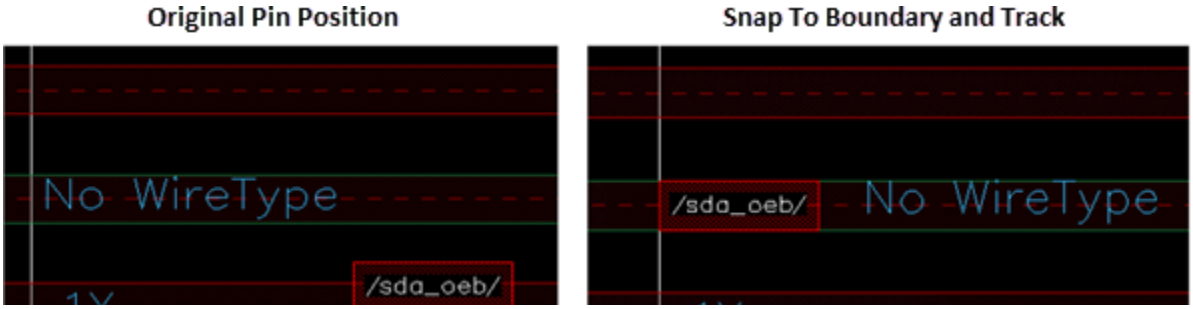
Symbol Overlay

Layout XL Forms

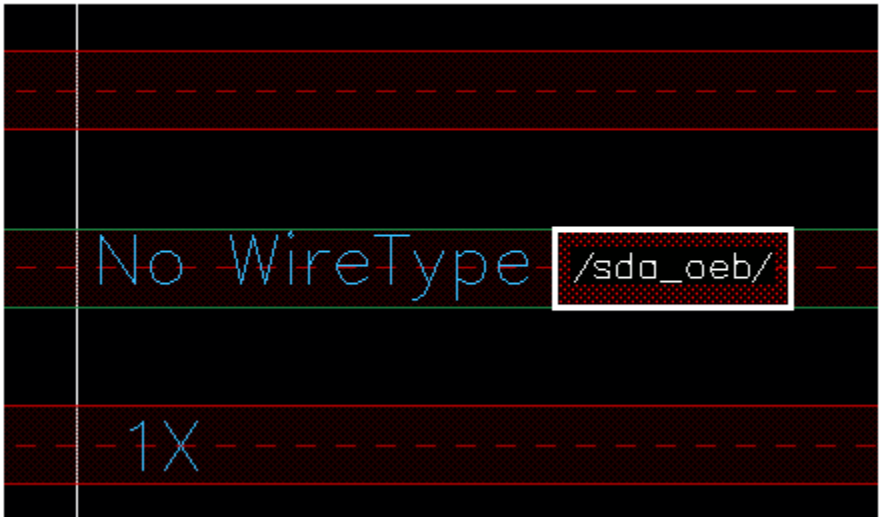
© 2023 Cadence Design Systems, Inc.

Snap Pins Form

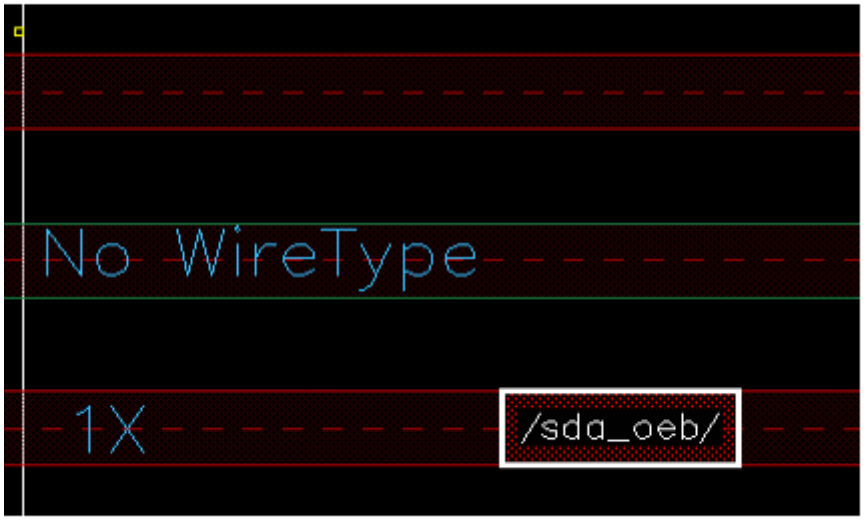
Use the Snap Pins form to snap top-level and level-1 soft block pins to the nearest grids. This command works on both preselected and post selected pins. At advanced nodes, pins are snapped to nearest WSP tracks. The options in the *Layer Awareness*, *Width Awareness*, and *Length Awareness* sections, available only at advanced nodes, enable you control pin snapping.

Field	Description
Snapping Mode	This section lets you select the pins to be snapped by setting them to one of the following values.
All	Snaps all pins in the design.
Selected	Snaps only the selected pins.
Inside PR Boundary Offset	Prevents snapping of pins that are away from the PR boundary offset and edges.
Snap To Boundary And Track	Snaps pins to the PR boundary and to the nearest WSP track. This option can be run on all pins, selected pins, or pins within a certain offset from the PR boundary. 
Snap To Track	Snaps pins to the nearest WSP track without snapping them to the PR boundary. In this case, the *Layer Awareness* and *Width Awareness* options are disabled.

© 2023 Cadence Design Systems, Inc.

Field	Description
Advance Node Options	This section lets you specify the advanced node options.
Constraint Aware	Honors the following constraints during pin snapping: Wire type, width, order, bus, side, edge, pin group guide, layer per side, blockages, and pre placed pins. When this option is selected, pins are snapped to the nearest WSP track honoring the wire type constraints as shown in the following figure. **Snap To Track - Constraint Aware mode**  Wire type constraints are read in the following sequence, based on their availability: **1.** Constraint system **2.** Environment variables: powerWireTypes, groundWireTypes, and signalWireTypes **Note:** These environment variables are valid only in *Snap To Track - Constraint Aware* mode.

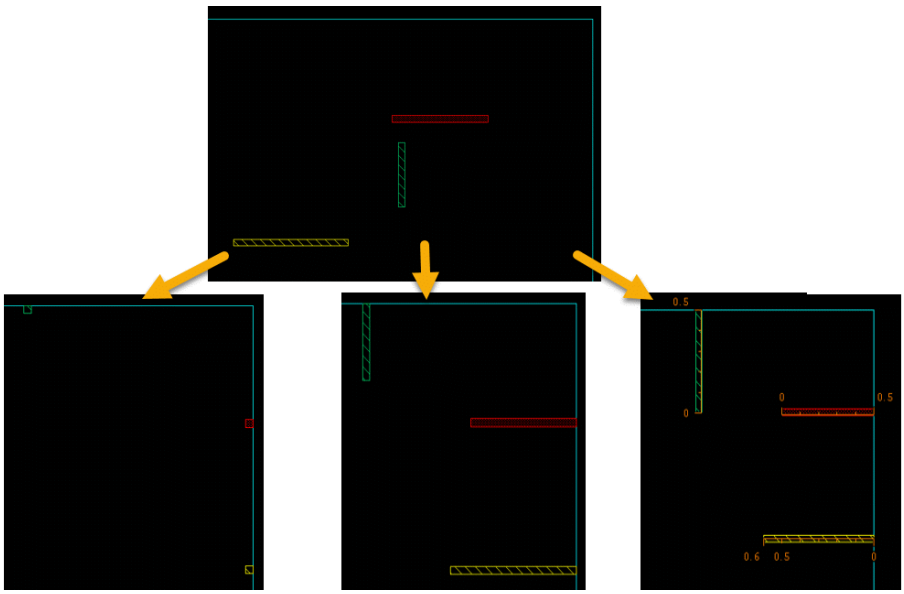
© 2023 Cadence Design Systems, Inc.

Field	Description
	In non-constraint-aware mode, pins are snapped to the nearest WSP track. Wire type checking is not done while snapping pins to the nearest track as shown in the following figure. Snap To Track - Constraint unaware mode  The *Length Awareness* options are honored in both constraint-aware and non-constraint-aware modes.
Layer Awareness	Specifies the layers on which the pins are to be placed after they are snapped.
Retain Pin Layer	Snaps pins to the nearest WSP tracks, while retaining the pins in their original layers and honoring all applicable pin constraints.

© 2023 Cadence Design Systems, Inc.

Field	Description
Auto Assign Pin Layer	Assigns pin layers to the sides of the block. During snapping, pins are moved to the nearest soft block edges and are re-layered as per your specifications. Use the two list boxes in this section to make the required pin layer assignments. When this option is selected, the default pin layer and side constraint values are loaded based on the `snapGridVertical` and `snapGridHorizontal` constraints defined in the technology file. ■ WSSPDefs with the `snapGridVertical` constraint have tracks intersecting with the left and right edges. ■ WSSPDefs with the `snapGridHorizontal` constraint have tracks intersecting with the top and bottom edges.
Update Pin Layers	Displays the selected pin layer in the *Pin Layers* column in the first box.
Clear Pin Layers	Clears the pin layers of the selected side.
Width Awareness	This section lets you control the width of the snapped pins.
Fixed Pin Width	Snaps pins to the nearest WSP tracks with matching or higher pin widths.
Length Awareness	This section lets you control the length of the snapped pins.
Square pin	Pins are converted to square shapes after snapping.
Retain Pin Length	The pin length is retained after snapping to WSP tracks.

© 2023 Cadence Design Systems, Inc.

Field	Description
Honor Minimum Length	The length of the snapped pins is set to the minimum pin length specified in the technology file. When *Snap To Track* is selected and the pin length is less than the minimum length specified in the constraint, the pin is resized to a length equal to the specified value while snapping to nearest WSP track. The pin width is the same as the WSP track width. 
Snap Pins	Snaps all the top-level and level-1 pins to appropriate grids. The pins with placement status as `Fixed` are also snapped, and their status is retained as `Fixed`.

Related Topics

Snapping Soft Block Pins to Grid

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Soft Block Global Options Form

Use the Soft Block Global Options form to specify that labels are to be created for all the pins in all the soft blocks in your design.

Field	Description
Create Label	Specifies that pin labels are to be created for all the pins in all the soft blocks in the design.
Pin Label Text Style	This section sets the style of the labels used to annotate pins in the design window.
Height	Sets the height of the labels.
Font	Sets the text style of the labels.
Text Options	■ *Drafting:* Prevents the label from being rotated more than 90 degrees. ■ *Overbar:* Determines how text strings containing underscore characters are displayed in a layout window. The default is off, which means that the software displays underscore characters (_) as part of the text string. When switched on, the software interprets underscore characters in the text string name as toggle switches that control where overbars begin and end. Overbars appear above the text string, as shown in the examples below.

Text string	Appears in design window as
_abcde	abcde
ab_cde	abcde
_abc_de	abcde
ab_cd_ef_gh_ij	abcdefghij

© 2023 Cadence Design Systems, Inc.

Field	Description
Layer Name	Sets the layer on which the labels are generated. Choose a layer from the cyclic field or select *Same As Pin* to create the label on the same layer as the pin with which it is associated. ■ *Same As Pin Layer*: Creates the label on the same layer as the pin. ■ *Same As Pin Layer Purpose*: Creates the label on the same layer purpose pair as the pin.
Justification	Sets the location of the label origin. The origin appears as a small square on the label when you place or select it. Choose from *lowerLeft*, *centerLeft*, *upperLeft*, *lowerCenter*, *centerCenter*, *upperCenter*, *lowerRight*, *centerRight*, and *upperRight*.
Orientation	Lets you specify the orientation of the labels. Choose orientation from: ■ `R0`: No rotation ■ `R90`: Rotate 90 degrees clockwise ■ `R180`: Rotate 180 degrees clockwise ■ `R270`: Rotate 270 degrees clockwise ■ `MY`: Mirror over the Y axis ■ `MYR90`: Mirror over the Y axis and rotate 90 degrees clockwise ■ `MX`: Mirror over the X axis ■ `MXR90`: Mirror over the X axis and rotate 90 degrees clockwise Environment variable: pinTextOrientation

Related Topics

Specifying I/O Pins

© 2023 Cadence Design Systems, Inc.

Creating Pin Labels During Layout Generation

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Startup Option Form

Use the Startup Option form to control the Layout XL launch operation. Depending on what you specify, you will be asked to complete different forms as Layout XL is launched.

Layout	Configuration	You see the following dialogs...
Create New	*Create New*	New File Create Physical Configuration View
Create New	*Use Existing*	New File Open Physical Configuration View
Create New	*Automatic*	New File
Use Existing	*New*	Open File Create Physical Configuration View
Use Existing	*Use Existing*	Open File Open Physical Configuration View
Use Existing	*Automatic*	New File

Field	Description
Layout	This section lets you control whether the system creates a new layout view or opens an existing one for the schematic.
Create New	Opens the New File form where you type the name of a new layout view for your Layout XL design.
Open Existing	Opens the Open File form where you can choose an existing layout view.
Configuration	This section lets you control the physical configuration view used for your design.
Create New	Opens the Create Physical Configuration View form where you can specify a new physical configuration view for the design. This happens after you have specified the layout view.

© 2023 Cadence Design Systems, Inc.

Field	Description
Open Existing	Opens the Open Physical Configuration View form where you can choose an existing physical configuration view. This happens after you have specified the layout view.
Automatic	Notifies Layout XL to handle all operations related to the physical configuration view automatically. The system creates a physical configuration view (or opens it, if one already exists) containing the minimum information required for Layout XL to operate correctly, and manages this information automatically throughout the session. If you need to modify the hierarchy configuration, component type definitions, or soft blocks at a later time, use the *Launch – Configure Physical Hierarchy* command.

Related Topics

Managing Libraries

Physical Hierarchy Configuration

Working with Cellviews

Create Physical Configuration View Form

Open Physical Configuration View Form

Starting the Design Environment for Layout XL

Layout XL Launched from a Schematic View

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Swap Components Form

Use the Swap Components form to control how two components in the layout can exchange their position in the canvas.

Field	Description
Swap Origin	Lets you specify whether the swapping is triggered at instance origin or at one of the specified orientations. The default is *instOrigin*, which is the point of origin of the instance. Environment variable: swapOrigin
Resolve overlaps	Lets you choose whether Layout XL should resolve any components that overlap after the swap.
Fix swapped objects during overlap resolution	Lets you control whether components are automatically fixed after swapping to resolve any overlaps.

Related Topics

Swapping Components

© 2023 Cadence Design Systems, Inc.

Trace Manager Form

Use the Trace Manager form to control the display of traces for various nets. Also use the form to change the trace color for a net, or to delete a trace.

Field	Description
Search net name	Allows you to search for a net name or choose a net from the associated drop-down list to create a trace for the net.
Trace	Creates a trace for the selected net and displays the traced net and the associated traced shapes in a hierarchical structure in the Trace Manager window. If you type a net name that does not exist in the cellview and click the *Trace* button, no trace is created.
Net	Displays the name of the traced net and all the corresponding island trace IDs in a tree structure. Selecting a net in the Trace Manager halos the net in the layout canvas even when the trace is set as *not visible* in the Trace Manager using the *Visibility* icon. ■ The Net Tracer creates one physical trace per island, whether or not the *From net* option on the Net Tracer Options Form is selected. ■ If the *Auto zoom to selected traces*  button is clicked in the Net Tracer Toolbar, the Net Tracer automatically zooms to the selected trace.
Visibility icon 	Controls the display of traces in a design. Click the icon in the column header to display (or hide) all the traces in one go. Alternatively, select the check box corresponding to each net to display (or hide) the trace for the net.
Color	Lets you choose a trace color for the selected net.
Type	Displays the icon specific to the type of trace created: ■ Physical –  ■ Logical –  ■ Step – 

© 2023 Cadence Design Systems, Inc.

Field	Description
Remove all traces	Lets you delete all or specific net traces. Click the icon in the column header to delete all traces. Alternatively, click the delete icon associated with specific nets to delete the corresponding traces.
Expand All 	Expands the hierarchy under the traced nets to display the names of all the shapes on each traced net.
Collapse All 	Collapses the hierarchy under the traced nets to display only the traced net names.

Related Topics

Removing All Traces

Net Tracer Options Form

Net Tracer Custom Setup Form

Net Tracer Display Options Form

Net Tracer Options Form

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Update Binding Form

Use the Update Binding form to increase Layout XL-compliance by improving the binding between the schematic and the layout instances at the current level or within the hierarchy.

If your design is largely LVS clean but the bindings are incorrect, you can clear the logical connectivity and re-extract the design. If you are sure that any user-defined bindings you created using the Define Device Correspondence Form form are correct, you can choose to preserve existing user-defined bindings. Alternatively, you can bind the instances based on a binding file.

Field	Description
Binding	This section lets you specify the type of binding to be applied.
Name only	Binds the schematic and layout instances that have matching names. Environment variable: bindNameOnly
All cells in design	Runs the Update Binding command at the current level for each layout and schematic cellview pair found in the design hierarchy. At the end of the command, the XL status is displayed for each layout cellview with connectivity reference, and a list of layout cellviews with no connectivity reference is also displayed. Environment variable: bindAllCellsInDesign For backward compatibility, the ?hierarchical argument is also supported.

© 2023 Cadence Design Systems, Inc.

Field	Description
Connectivity	This section lets you specify the type of Connectivity for the design.
Clear	Deletes all logical connectivity from the design and re-extracts the layout to derive the physical connectivity so that the binder can bind using the correct connectivity. Physical connectivity is the connectivity derived from overlapping physical objects (or shapes) in the layout. It is recommended to use physical connectivity for updating bindings if the design is LVS clean but the layout was created outside Layout XL. If the logical connectivity is cleared, the *Extract to level* must be specified for the re-extraction. If the *Clear* option is deselected, the *Update Binding* command uses both logical and physical connectivity to improve the bindings. To clear all the bindings before attempting to rebind, deselect both the *Preserve* binding options. Environment variable: bindClearConn
Extract to level	Extracts the layout down to the specified level. By default, the command is set to extract only the top level. However, you can update the field to choose extracting up to a specific hierarchical level. For example, if the design has route cells and the extraction level is set to 0, the binder cannot improve the bindings unless the extraction depth is manually increased and the layout is extracted again. When extracting hierarchically, it is recommended that you define stopLayers in your technology file before running *Update Binding* so that the extractor ignores the layers that are being used to cut other layers and any undesirable violations can be avoided. Environment variable: bindExtract

© 2023 Cadence Design Systems, Inc.

Field	Description
Create - Ungenerated pins from labels	Creates pins from labels in Layout XL, if the label text or text display matches an ungenerated schematic terminal name. Pins are also created for terminals that already have pins but labels are swapped in the layout. Environment variable: createPinsFromLabels
Remove - Duplicate objects	Removes duplicate instances, shapes, and vias from the layout cellview. Environment variable: removeDuplicateObjects
Use - Physical bindings	Binds schematic instances with those layout instances that have the master defined using the Configure Physical Hierarchy Window. By default, the option is ON. The *Physical bindings* option is disabled in the *Hierarchical* mode when binding based on a *Binding* file or an *Instance Cross Reference* file. By default, the bindCorrectMaster environment variable needs to be `nil` for initial binding and `t` for *Update Binding*. Therefore, the *Physical bindings* field on the Update Binding form is by default selected even when the environment variable is set to nil. In other words, the Update Binding form binds with the correct master, irrespective of the default value of the bindCorrectMaster environment variable. The *Physical bindings* field cannot be used to change the value of the `bindCorrectMaster` environment variable. This implies that if the value of the bindCorrectMaster environment variable needs to be set to nil to support initial binding, the update needs to be saved in the .cdsinit file. Environment variable: updateBindUsePhysicalBinding
Update - Physical bindings	Updates the physical bindings in CPH if the current physical binding does not match the bound layout instance. The *Update – Physical bindings* option is disabled when the *Use – Physical bindings* option is selected. Environment variable: updateBindPhysicalBinding
Ignore	This section specifies the instances to be ignored.

© 2023 Cadence Design Systems, Inc.

Field	Description
Route cells	Adds an *ignore* property on the layout instances that are identified as route cells to ignore the instances for binding. Route cells are identified as cells that have no devices using extract stop layers. For cases when no extract stop layers are defined, route cells are identified as the ones that have no terminals but only metal or cut shapes. By default, the option is OFF. Environment variable: bindIgnoreRouteCells If the route cells exist within a transparent layout hierarchy, the ignore property cannot be added during an *Update Binding* run. Adding the ignore property on route cells within a transparent hierarchy requires editing the transparent instance master. The *Update Binding* command does not support such edits that require modifying the layout cellviews other than at the top level.
Dummies	Ignores any unbound, potential dummy instances that are identified during a binding run. By default, the option is OFF. Alternatively, you can ignore the unbound instances by selecting the instances and choosing *Add Ignore* from the shortcut menu. Environment variable: bindIgnoreDummies
Preserve	This section lets you specify the preserve options for bindings.
User-defined bindings	Preserves the user-defined bindings of devices between the schematic and the layout. This option preserves user-defined one-to-one, many-to-many, many-to-one, and one-to-many device correspondences defined in the Define Device Correspondence Form form. If the option is deselected, the *Update Binding* command deletes any existing device correspondence before processing the bindings. Environment variable: bindPreserveUserBindings

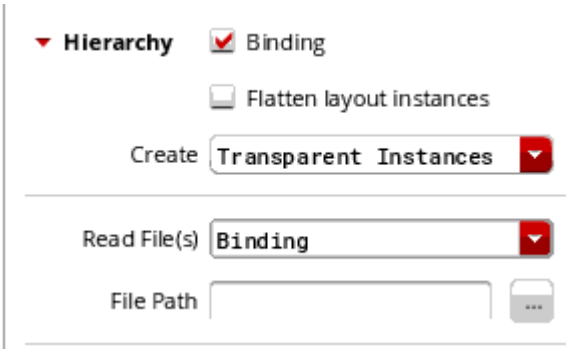
© 2023 Cadence Design Systems, Inc.

Field	Description
Existing bindings	Preserves not only the user-defined bindings but also any other existing bindings between the schematic and the layout devices. If the option is deselected, the *Update Binding* command recreates all bindings. Environment variable: bindPreserveExistingBindings
Hierarchy	This section lets you specify the options in hierarchy.
Binding	Traverses through the hierarchy, binding the schematic and the layout instances along the way.
Flatten layout instances	Flattens the layout instances, if required, when binding hierarchically and creates leaf-level bindings. Since the leaf-level bindings have a one-to-one correspondence, full Engineering Change Order (ECO) capability is available for the design. If the *Flatten layout instances* option is deselected, the layout hierarchy is preserved and you can choose to create either complex bindings to resolve hierarchy mismatches between the schematic and the layout or create transparent instances. If you choose to create complex bindings, ECO capability of the design is restricted. Environment variable: bindPreserveLayoutHierarchy (determines whether the layout hierarchy will be preserved or the layout instances will be flattened to create leaf-level bindings). The *Flatten layout instances* option is enabled only when *Hierarchy – Binding* is selected. Depending on the value of the *Create* option, the flattened instances can exist as free objects or can be formed into a synchronized family or a regular figure group.

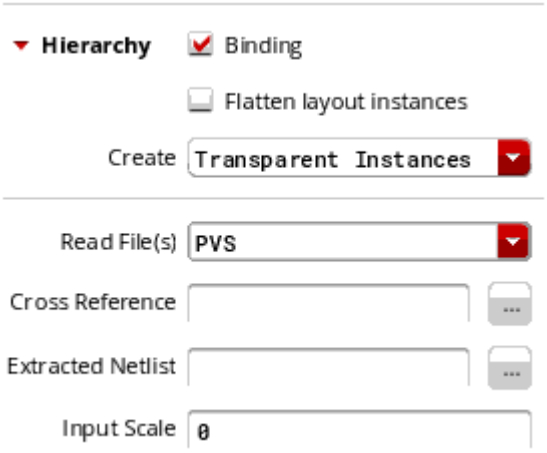
© 2023 Cadence Design Systems, Inc.

Field	Description
Create	Generates transparent instances or complex bindings during hierarchical binding when *Flatten layout instances* is OFF. Alternatively, the *Create* option can create a synchronized family, a figure group, or let the flattened instances to exist as free objects, when *Flatten layout instances i*s ON. By default, transparent instances are created when binding hierarchically. The options that you see in the *Create* field depend on the value of the *Flatten layout instances* option. If *Flatten layout instances* is OFF, *Create* displays the following options: *Transparent Instances* `and` *Complex Bindings*. The default is to create transparent instances. If *Flatten layout instances* is ON, *Create* displays the following options: *Synchronized Family* , *Grouped Objects*, and *Free Objects*. The default is to create *Synchronized Family* . Environment variable: bindFlattenLayoutCreate
Create – Transparent Instances	Converts layout instances into transparent instances such that the binder can bind the instances within the transparent instance to the corresponding schematic instances. By allowing such one-to-one bindings, the original layout hierarchy is preserved. The *Create – Transparent Instances* option is only enabled in the *Hierarchical* mode when the *Flatten layout instances* option is disabled and the Create option is set to *Transparent Instances.* In other words, the *Create – Transparent Instances* option can help improve binding by allowing the layout instances to be made transparent instead of flattening them. If *Update Binding* is run at the current level and the layout design already has a transparent hierarchy, the *Update Binding* command considers the transparent layout hierarchy for binding. This means, wherever possible, the *Update Binding* command binds to the layout instances that exist within the transparent hierarchy in the design.

© 2023 Cadence Design Systems, Inc.

Field	Description
Read File	This section lets you specify options available in the *Read File* drop-down menu.
None	Specifies that the binder performs an automatic hierarchy match and creates complex bindings to improve the schematic versus layout bindings. Environment variables: bindFile, bindCrossRefFile, and bindExtractedNetlistFile If all the environment variables are set to "", *Read File* sets to *None*. If bindFile is set to a valid value, and the environment variable takes precedence over the other associated environment variables, *Read File* sets to *Binding*. Else, if bindCrossRefFile or bindExtractedNetlistFile are set, *Read File* sets to *PVS*.
Binding	Provides the path to the binding file to be used for improving the schematic versus layout bindings.  If the *Binding* option is selected and a binding file is selected in the *File Path* field, the binder uses the binding file to improve the bindings across the schematic and the layout. However, if the *Binding* option is selected but no binding file is specified, the *Update Binding* run does not proceed. Environment variable: bindFile

© 2023 Cadence Design Systems, Inc.

Field	Description
PVS	Specifies that the input from a PVS LVS run will be used for running *Update Binding*. 
Cross Reference	Provides path to the PVS LVS `.ixf` instance cross-reference file that is created in the `svdb` directory of the LVS run directory by using the *Create Quantus Input Data* option. If the PVS LVS instance cross reference file is specified, the binder uses the specified file to improve the bindings across the schematic and the layout. If any mismatched instances are encountered while parsing the.ixf file, they are ignored. If the PVS LVS instance cross reference file is not specified, *Update Binding* run does not proceed. Environment variable: bindCrossRefFile
Extracted Netlist	Provides the path to the PVS LVS `.net` or `.spi` extracted net list file that is created in the `svdb` directory of the LVS run directory by using the *Create Quantus Input Data* option. If the path to the instance cross-reference file has been specified, the path to the extracted netlist file must also be provided for the binder to improve the existing bindings. Environment variable: bindExtractedNetlistFile

© 2023 Cadence Design Systems, Inc.

Field	Description
Input Scale	Specifies the floating point input scale for running PVS. By default, input_scale is 0, which means that dbuPerUu for the layout is used. But, if the PVS input scale is different from that of the layout, and the PVS rule file is specified, the PVS rule deck is parsed to determine the appropriate value. Alternatively, you can specify the PVS input scale to be used. Environment variables: bindPVSScale, bindPVSRulesFile
Report	This section lets you specify options for reporting.
Permute rules	Reports cells that have no permute rules defined. Environment variable: checkPermute
Component types	Reports cells that have a missing component type. Environment variable: checkCompType
Remove devices	Reports parasitic devices that could be ignored. Environment variable: checkRemoveDevice
Dummies	Reports unbound dummy layout instances. Environment variable: bindReportDummies
Route cells	Reports layout instances that have an ignore property set. Environment variable: bindReportRouteCells
Hierarchy binding	Reports the schematic and layout cellviews that are being descended into to create the bindings. Environment variable: bindReportHierarchy
Unconnected instance	Reports instances that have an unconnected instance terminal. Environment variable: bindReportUnconnInstTerms
Unmatched terminals	Reports cellviews with instance terminals that do not match any instances in the corresponding schematic view. Environment variable: bindReportTerminalMismatch
Opens and shorts	Reports cellviews with shorts between instance terminals and opens between strongly- or weekly-connected pins. Environment variable: checkShortsAndOpens

© 2023 Cadence Design Systems, Inc.

Field	Description
Cellviews with no terminals	Reports cellviews with no terminals. Environment variable: bindReportCellsNoTerms
Terminals with no pins	Reports instance terminals with no pins. Environment variable: checkMissingTermPins
Duplicate objects	Checks and reports duplicate instances, shapes, and vias in the layout cellview. Environment variable: checkDuplicateObjects

Related Topics

Instance Cross Reference

Configure Physical Hierarchy Window

Flatten layout instances

validLayers – Stop Layers Specification

Binding Updates

lxUpdateBinding

Transparent Layout Hierarchy Generation

Device Correspondence Updates

Define Device Correspondence Form

Update Binding Form

Issues Faced when Setting up the PDK to Improve Layout XL Compliance

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Update Clone Families Form

Use the Update Clone Families form to manage the families of synchronized clones in your design. You can update the status of a clone family, rename a clone family, remove a clone from a family, create and remove clone families, and add a group to an existing family.

Caution

You cannot undo the changes you make using this form.



© 2023 Cadence Design Systems, Inc.

Field	Description
Automatically zoom to selected	Zooms in on the clone or clone family that is currently selected in the *Clone Families* pane, or on the layout group selected in the *Layout groups* pane. Clicking on the top level fits the whole cellview in the layout window. If you select more than one layout group, the software zooms in on the bounding box that encloses the selected groups. This option is considered only when you make the selection in the Update Clone Families form; selecting a synchronous clone in the canvas does not zoom the display to the selected clone or group.
Clone Families	Lists the clone families in the current design. You can expand each entry to display the clones it contains.
Buttons	This section lets you select buttons on the right-hand side of the form that operates on the clone families displayed in the left pane.
Verify Synchronization	Checks the consistency of the clones in the selected family or in all the families if no family is selected, and makes any changes required to the existing clone families depending on the design situation. If a clone is out of synch with the other clones in the family, it is removed from the family. Although the clone group is still retained, the group type is changed from syncClone to none. If two or more clones are out of synch with the family but in synch with each other, they are moved into a new clone family. If all the clones in the family are different, the family is removed altogether. The results are printed in the CIW. You cannot undo *Verify Synchronization*.
Rename Family	Renames the selected clone family and the member clones, if set for update. This button is enabled only when you select a clone family. If the clone family contains virtual clones that cannot be renamed, the *Rename Clones* check box is disabled.

© 2023 Cadence Design Systems, Inc.

Field	Description
Remove Family	Removes the selected clone family from the design. This button is enabled only when you select a clone family. You cannot remove a family if one of its members is currently being edited in place. You need to exit the *Edit In Place* command first.
Remove Clone	Removes the selected clone from a family. This button is enabled only when you select a clone. You cannot remove a clone if one of its members is currently being edited in place. You need to exit the *Edit In Place* command first.
Create/Add To Family	Expands the form to let you create a new family of synchronous clones or add clones to an existing family.
Layout groups	Lists all the regular layout groups not currently assigned to a clone family. It does not list any other type of group. When you remove a clone from its family, its type is changed to `none` and it is added to the list of layout groups. However, its name is left unchanged, which means that the list might contain names like `cloneFamily1_1`. Such groups are regular groups and not synchronous clones.
Synchronization reference	Lets you choose how the members of the new or updated family are synchronized. All routine geometry is synchronized, including shapes and vias. However, labels are not synchronized. If you are creating a new family, the drop-down list contains all the selected layout groups to be included in the family. From the *Synchronization reference* drop-down, choose the reference group with which the other groups are to be synchronized. If you are adding groups to an existing family, the drop-down list contains the family itself and all the groups you want to add to it. If you choose the family, the new members are synchronized against the existing members of the family. If you choose one of the groups, the new and existing members are synchronized against the group you selected.
Create Family	Creates a new family containing the members highlighted in the *Layout groups* list. This button is enabled only when you select two or more layout groups and there is no existing family selected in the *Clone Families* pane.

© 2023 Cadence Design Systems, Inc.

Field	Description
Add To Family	Adds one or more selected groups to an existing family. This button is enabled only when you select an existing family in the *Clone Families* pane and at least one group from the *Layout groups* pane.

Related Topics

Updating Clone Families

Verifying a Clone Family

Removing a Clone from a Family

Removing a Clone Family from Design

Creating a New Clone Family

Adding a Group to an Existing Family

Editing a Clone in a Synchronized Family

Generating a Family of Synchronous Clones

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Update Components and Nets Form

Use the Update Components and Nets form to automatically update your layout to take account of the instances, pins, and connectivity you have changed in the schematic.

The Update Components and Nets form consists of the following tabs.

Tab	Description
Update	Specifies how much of the design is updated and what happens to existing layout components that need to be changed or removed from the design. The tab also specifies which missing design objects are generated in the layout view.
I/O Pins	Specifies pin attributes to be applied when pins are generated in the layout.
PR Boundary	Specifies the shape and size of the PR boundary to be generated in the layout.
Hierarchy	Specifies the scope for generating virtual hierarchies in the layout.

Update

The following table describes the fields available on the *Update* tab of the Update Components and Net form.

Field	Description
Update	This section lets you specify how much of the design is updated and what happens to existing layout components that need to be changed or removed from the design
Update Selected Layout Components Only	Updates only the instances and pins currently selected in the layout window. This option automatically disables the generation of new components. However, if the *Update Instance Masters* option is enabled and set to *Creating a New*, Layout XL may generate new instances to replace existing instances with incorrect masters; you have the option to switch on *Chain, Fold, and Chain Folds* for these new instances.

© 2023 Cadence Design Systems, Inc.

Field	Description
Update Nets and Instance Name Mismatches Only	Specifies that only net assignments and instance, terminal, and net names are updated. Enabling this option also automatically preserves user-defined mappings. All other options on the form are disabled except *Update Selected Layout Components Only*. If any pins have got swapped due to a schematic ECO, running the *Update* with *Update Nets and Instance Name Mismatches Only* selected ensures that the original pin positions are retained in the layout. To reflect the pin swaps in the layout, run the *Update* with *Update Nets and Instance Name Mismatches Only* deselected. Environment variable: updateNetsOnly
Update Net Signal Types	Specifies that the signal types assigned to nets in the schematic are updated in the layout view. The signal types can be set in the schematic and are automatically transferred to the layout during *Generate All From Source*. However, once set, the signal type of a net can be manually updated using the Property Editor assistant. If the signal type of a net is changed in the schematic, the *Update Net Signal Types* option can be used to update the layout to keep the two views in sync. Environment variable: updateNetSigType
Update Net MinV/MaxV	Specifies that the minimum and maximum voltages assigned to nets in the schematic are updated in the layout view. Environment variable: updateNetMinMaxVoltage Only those voltages that are manually set on the schematic net are updated in the layout view. Layout versus schematic mismatches that may arise due to this selective transfer of voltages are not reported by the *Check Against Source* command.

© 2023 Cadence Design Systems, Inc.

Field	Description
Update Instance Masters by	Updates an existing instance that uses an incorrect master to use the correct master. ■ *Replacing the Old or Creating a New Instance*: The default is *Replacing the Old*, which means the command removes the incorrect instance and replaces it with an instance of the correct master in the same location. When set to *Creating a New*, the command puts a marker on the instance with the incorrect master and renames it *name*_old. It then creates a new instance with the correct master and places it below the PR boundary. Because you are creating a new instance, you can optionally enable *Chain, Fold, and Chain Folds* in this mode. Environment variable: updateWithMarkers
Delete Unmatched Pins	Deletes layout pins that are no longer present in the schematic. Redundant nets and terminals are deleted from the layout view at the same time. When switched off, unmatched pins are not deleted but are instead indicated with a marker in the layout view. Environment variable: deleteUnmatchedPins
Delete Unmatched Instances	Deletes layout instances that are no longer present in the schematic. When switched off, unmatched instances are not deleted but are instead indicated with a marker in the layout view. Environment variable: deleteUnmatchedInsts
Update Layout Parameters	Updates the parameters and parameter values automatically on layout instances to match those on their schematic counterparts. Parameters that are set in layout instances but are not present on their schematic counterparts are not removed. Environment variable: updateLayoutParameters

© 2023 Cadence Design Systems, Inc.

Field	Description
Move Changed Overlapping Instances Below PR Boundary	Places instances automatically below the PR boundary if they overlap other instances during the update. The overlapping instances are the ones that have changed in size, and if placed at their original location, may overlap other unchanged instances.This data is then used by the Virtuoso Custom Digital Placer (VCP) for Eco Mode placement. Environment variable: updatePlaceBelowBoundary
Generate	This section lets you specify which missing design objects are generated in the layout view.
Instances	Generates instances that are in the schematic but are not currently in the layout.
Chain	Automatically abuts transistors into chains during layout generation.
Fold	Automatically divides devices into folds to prevent the gate width from exceeding a specified size. Environment variable: initDoFolding If the disableFolding environment variable is set and the folding threshold for the component type is set to `0`, folding is disabled.
Chain Folds	Chains the individual folds of a transistor automatically. Environment variable: initDoStacking For the *Chain Folds* option to be available, only the *Fold* check box must be selected. If you select the *Chain* check box as well, *Chain Folds* is deactivated.
I/O Pins	Generates all the pins listed on the *I/O Pins* tab. The generated pins are automatically snapped to the placement grid. Environment variable: initCreatePins
Except Global Pins	Stops Layout XL generating layout pins for the global nets in the schematic.

© 2023 Cadence Design Systems, Inc.

Field	Description
Except Pad Pins	Stops Layout XL from generating layout pins for schematic pins that are connected to I/O pads (cells of type `pad`, `padSpacer`, or `padAreaIO`). When unchecked, the software generates both pads and pins. Environment variable: initCreatePadPins
PR Boundary	Regenerates the new PR boundary based on the settings on the *PR Boundary* tab. All placements and estimations are based on the generated PR boundary. When switched off, the existing boundary is retained.
Snap Boundary	Generates a rectangular snap boundary that encloses the generated PR boundary. You can generate a snap boundary only if the *PR Boundary* option is switched on. Environment variable: initCreateSnapBoundary
Device Correspondence	■ *Preserve User Defined Bindings*: Preserves user-defined bindings of devices between the schematic and the layout. This option preserves only user-defined one-to-one, many-to-many, many-to-one, and one-to-many device bindings defined in the Define Device Correspondence form. It does not report missing devices or shapes within a bound group. Environment variable: initCreateMTM

I/O Pins

The following table describes the fields available on the *I/O Pins* tab of the Update Components and Net form**.**

Field	Description
Default Values For Electrical Pins	This section lets you specify attribute values and *Apply* them to all the pins shown in the list box. Click *Apply* to apply the *Layer*, *Width*, *Height*, *Num*, and *Create* settings for all the listed pins.

© 2023 Cadence Design Systems, Inc.

Field	Description
Layer	Specifies the layer on which the pin is generated. The cyclic field lists only the conducting layers. You can use the initIOPinLayer environment variable to specify the layer-purpose pair to use for generating the pins. The default is the current layer selected in the Layer Assistant, provided it is a valid layer. See validLayers. Else, it is the first extractable layer in the technology file that has a `pin` purpose. If an extractable layer with a `pin` purpose does not exist, the first extractable layer with a `drawing` purpose is selected as the default. If there are no extractable layers, the cyclic field lists all the valid layout layers. Layers are extracted from the `validLayers` constraint only if they have a purpose defined. However, the purposes are not used to restrict the displayed layer-purpose pairs. To filter the purposes, use the initIOPinPurposeNames environment variable.
Width	Specifies the width for each pin. The default is the `minWidth` value set for the current layer in the technology file. Any change to the value is applied only if the new value is greater than the default value.
Height	Specifies the height for each pin. The default is the `minWidth` value set for the current layer in the technology file. Any change to the value is applied only if the new value is greater than the default value.
Num	Specifies how many instances of this pin to generate. If you type 0, the pin is not generated.
Create	Specifies that pins are to be generated in the layout.
Specify Pins to be Generated	This section lets you select pins from the list box and update the attribute values used when those pins are generated in the layout. Click *Update* to update the *Layer*, *Width*, *Height*, *Num*, and *Create* settings for the currently selected pins.
Select	Lets you type in a complete or partial pin name to select one or more pins in the list box.

© 2023 Cadence Design Systems, Inc.

Field	Description
Number Of Matches	Displays the number of pin with names that match the string you typed into the *Select* field.
Add New Pin	Opens the Add A New Pin Form dialog, where you can specify the name of a terminal for which to generate a new pin. This option is available when you right-click in the *Specify Pins to be Generated* section.
Term Name	Specifies the schematic terminal name. You cannot change this value.
Net Name	Specifies the net associated with the pin in the layout. You cannot change this value. If terminal and net names differ in the schematic, Layout XL generates a pin with the same name as the schematic terminal and a net with the same name as the net attached to the terminal in the schematic. If there is no explicit net label in the schematic, both the pin and the net name in the layout are the same as the schematic terminal. This is the default behavior.
Layer	Specifies the layer on which the selected pins are generated. The cyclic field offers only the conducting layers. The default is the current drawing layer if it is defined as an extractable layer in the technology file. Otherwise, it is the first extractable layer found. If there are no extractable layers, the cyclic field lists all valid layout layers.
Width	Specifies the width for each selected pin. The default is the `minWidth` value set for the current layer in the technology file. Any change to the value is applied only if the new value is greater than the default value.
Height	Specifies the height for each selected pin. The default is the `minWidth` value set for the current layer in the technology file. Any change to the value is applied only if the new value is greater than the default value.
Num	Specifies how many instances of each selected pin are generated. If you type 0, the selected pin is not generated.

© 2023 Cadence Design Systems, Inc.

Field	Description
Create	Specifies whether or not the selected pins are to be generated in the layout. To limit the number of pins that are displayed in the pin table, you can set generatePinsDisplayLimit to the required display limit. If the design includes more pins than the number allowed to display in the form, the additional pins are suppressed and the form is updated to mention the number of suppressed pins. You can right-click the suppressed pin notification, and choose *Un-suppress* to display all the pins.
Pin Label	This section lets you specify the type of label generated when you create a pin. This setting is honored by the *Generate All From Source* and *Generate Selected From Source* commands.
Create Label As	Specifies whether a pin label is created when the pin is generated. You can choose to create either a *Label* object or a *Text Display*. Environment variable: initIOLabelType Click the *Options* button to access the Set Pin Label Text Style Form, where you can set the size, font, style, justification and orientation of the label lettering, and the drawing or pin layer on which the labels are displayed. If your pin labels are not visible in the canvas, turn on the *Pin Names* option in the Display Options form. For this setting to take effect, you must set the createPinLabel environment variable to `t`.

PR Boundary

The following table describes the fields available on the *PR Boundary* tab of the Update Components and Net form.

Field	Description
Shape group box	This section lets you specify whether the place and route boundary is a rectangle or a polygon

© 2023 Cadence Design Systems, Inc.

Field	Description
Rectangle	Specifies a rectangular place and route boundary. Use the *Area Estimation* group box to specify how the size of the boundary is calculated.
Origin	Specifies the coordinates of the boundary's origin. The default is (0.0 0.0).
Polygon	Specifies a polygonal boundary. Use the *Points List* to specify the coordinates of each of the vertices of the polygon.
Area Estimation group box	This section lets you specify how the system calculates the shape and size of a rectangular boundary. Area estimation comprises two parts: the first specifies the aspect ratio and utilization of the boundary; the second estimates the size of boundary required to accommodate the components to be generated. To specify the aspect ratio and utilization, set two of the following four parameters. Choose one of the parameters from the first cyclic list and any one of the remaining valid choices from the second cyclic list.
Utilization (%)	Specifies the percentage of area within the cell boundary that you want to fill. The default is `25`. Environment variable: initUtilization When the *Chain, Fold, or Chain Folds* options are switched on in the *Generate* tab, the *Utilization* value is applied only after chaining and folding is complete so that the size of the boundary is calculated accurately.
Aspect Ratio (W/H)	Specifies the width-to-height ratio of the design boundary. A value of 1 specifies a square boundary; 0.5 specifies a boundary twice as high as it is wide; and 2 specifies a boundary twice as wide as it is high. The default is `1`. Environment variable: initAspectRatio and initAspectRatioOption
Width	Specifies the width of the design boundary. The default is the size of the last boundary or `10`. Environment variable: initPrBoundaryW

© 2023 Cadence Design Systems, Inc.

Field	Description
Height	Specifies the height of the design boundary. The default is the size of the last boundary or `10`. Environment variable: initPrBoundaryH
Estimator	Lets you choose the type of area estimator to be used.
PR Boundary based	Calculates the area based on the sum of the areas enclosed by all the place and route boundaries of the components to be updated.
BBox based	Calculates the area based on the sum of the areas enclosed by all the bounding boxes of the components to be updated.
Register	Opens the Add Area Estimators Form, which lets you register a user-defined SKILL function to estimate the area enclosed by a PR boundary or by the area boundary of a virtual hierarchy block. The SKILL function registered for estimating the PR boundary appears in the *Estimator* drop-down list and can be used to estimate the area of the top-level boundary when updating a layout. The SKILL function registered for estimating the area boundary appears in the *Soft Block – Area* drop-down list and can be used to estimate the area of the virtual hierarchies when updating a layout. If you specified `deferred` when you registered the PR boundary area estimator and are now choosing to generate virtual hierarchies with area boundaries, the PR boundary estimation is deferred until the area boundary of the virtual hierarchy is created. This allows the area estimation function to consider the virtual hierarchy area boundary sizes when calculating the PR boundary area.
Virtual Hierarchy Area Boundary	This section lets you specify how the area boundary of a virtual hierarchy is estimated. This group box is available only when you have the Layout EXL or higher license checked out.
Enclose by	Specifies the distance from the objects inside the virtual hierarchy at which the area boundary is created.
Utilization (%)	Specifies the acceptable area utilization percentage for deriving the size of the area boundary for the virtual hierarchy. Environment variable: areaBoundaryUtilization

© 2023 Cadence Design Systems, Inc.

Field	Description
Top Level	Specifies that the selected area boundary settings be used for creating virtual hierarchy area boundaries only at the top level.
All Levels	Specifies that the selected area boundary settings be used for creating virtual hierarchy area boundaries at all levels within the virtual hierarchy of the top-level design.
None	Specifies that the virtual hierarchy area boundaries are not created, instead existing virtual hierarchy boundaries are used.
Soft Block	This section lets you specify the area of a soft block to be created. This group box is available only when you have the Layout EXL or higher license checked out.
Area	Specifies the area of the soft blocks to be created.LLIf a SKILL function has been registered to estimate the area of a virtual hierarchy, the function appears in the drop-down list associated with the *Area* field. The same area estimator function can be accessed using the Adjust Area Boundary form in the Virtuoso Design Planner. Environment variable: softBlockArea

Hierarchy

The following table describes the fields available on the *Hierarchy* tab of the Update Components and Net form.

Field	**Description**
Scope	This section lets you specify the extent in the hierarchy to which the design is updated
All cells in design	Specifies that the selected updates are carried out to all the nontransparent cellviews in the design hierarchy. When this option is enabled, the *Generate – PR Boundary* option on the *Update* tab is disabled to prevent recreating the PR boundaries across the hierarchy.
Design Planning	This section lets you specify if the virtual hierarchy is updated to match the schematic hierarchy and if soft blocks are automatically generated for missing layouts. The group box is available only when you have the Layout EXL higher license checked out.

© 2023 Cadence Design Systems, Inc.

Field	Description
Virtual Hierarchy	Lets you generate a virtual hierarchy using the area boundary options selected in the *PR Boundary* tab, or updates an existing virtual hierarchy to match the schematic hierarchy. When updating an existing virtual hierarchy, the area boundary of the virtual hierarchy is not updated. Environment variable: generateVirtualHierarchy
Auto Generate Soft Blocks	Lets you generate soft blocks for top-level virtual hierarchy blocks that have no schematics available or have schematics with only pins but no instances or physical binding. For the virtual hierarchy blocks that have no schematic available, soft block generation uses the bound symbol view to generate pins. Soft block boundary is generated using the options defined on the *PR Boundary* tab. Environment variable: generateSoftBlocks
Update soft blocks when symbol is modified	Updates the soft block in the layout to match the schematic symbol. Environment variable: updateSoftBlocksFromSymbol
Auto place generated instances inside existing virtual hierarchy	Places instances missing from a virtual hierarchy inside the virtual hierarchy automatically, retaining the placement of the existing instances. After placement, if the virtual hierarchy contents extend beyond the area boundary and the area boundary is rectangular, the boundary automatically adjusts to enclose the contents. Rectilinear area boundaries that have their contents extending beyond the boundary need to be manually adjusted and placed. For the generated instances to be automatically placed inside an existing virtual hierarchy, the placement status of the virtual hierarchy should be set to *None*.

Related Topics

Eco Mode

Add A New Pin Form

Adjust Area Boundary

© 2023 Cadence Design Systems, Inc.

Set Pin Label Text Style Form

Define Device Correspondence Form

Creating and Registering a User-Defined Area Estimation Function

Components and Nets Updates

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Update Connectivity Reference Form

Use the Update Connectivity Reference form to update the source for your layout view and to change the physical configuration which specifies how the layout is generated from the source.

Field	Description
Source View Definition	This section lets you specify the view to use as the connectivity source
Use schematic view	Specifies Layout XL to use a schematic view as the connectivity reference. If you uncheck the box, it means there is no connectivity reference.To specify the schematic to use, do one of the following. ■ Type the *Library*, *Cell*, and *View* names of the schematic in the fields provided. You can type in the name of a different library. The default view name is `schematic`. ■ Use *Sel by Cursor* and click an open schematic window. ■ Use the *Browse* button to browse your file system for the schematic you want.
Update layout cellviews hierarchically	Descends into a hierarchical layout cellview and updates the library setting for any lower-level cellviews in accordance with the library name specified for the top-level cell, except: ■ Where the lower-level cellview is a leaf (i.e., there is no instance in the cellview). ■ The current setting points to a non-existent cellview.Lower-level cellviews updated in this way are not saved until you confirm that you want to do so at the end of the session.

© 2023 Cadence Design Systems, Inc.

Field	Description
Physical Configuration View Definition	This section lets you specify definitions for physical configuration view.
Use configuration view	Lets you specify the physical configuration view to use for the schematic-layout pair. If you deselect this check box, Layout XL runs without a physical configuration.To specify a physical configuration to use, do one of the following: ■ Type the *Library*, *Cell*, and *View* names of the view in the fields provided. You can type in the name of a different library. The default view name is `physConfig`. ■ Use the *Browse* button to browse your file system for the view you want. Environment variables: cphUseLayoutLibrary, cphUseLayoutViewName
Stop Level	Specifies the stop level for elaboration of the logical design hierarchy. Once saved in the layout, the specified stop level value remains unchanged until the value is manually updated. The value set here takes precedence over the one set in CPH or using the environment variable. Environment variable: cphStopLevel

Related Topics

Updating the Schematic View

Updating the Physical Configuration View

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

Update Layout Device List Form

The Update Layout Device List form displays all the devices in the design. Use it to select specific devices when using the *Update Layout Parameters* and *Update Schematic Parameters* commands.

Field	Description
Search	This section lets you find and select the names of devices you specify. Specify a device name, part of a device name, or a regular SKILL expression using part of a device name in the field to the right of the *Search* button.
A device name	Highlights the device and scrolls the list to make the device name visible.
Part of a device name (for example, `IN`)	Highlights all the devices with `IN` as an element in the name (`IN3`, `IN6`, and so on) and scrolls the list to make the first example visible.
A regular SKILL expression using part of a device name (for example, `*mos`)	Highlights all devices with the string `*mos` (for example, `nmos`, `pmos`) and scrolls the list to make the first example visible.

Related Topics

Layout XL Design Data Updates

Updating Layout Parameter Values in Postselection Mode

Updating Schematic Parameter Values in Postselection Mode

Layout XL Forms

© 2023 Cadence Design Systems, Inc.

View By Form

Use the **View By** form to specify how the annotations are organized in categories and how the categories are sorted in the browser pane.

To open the form, right-click in the browser pane and choose *View By*.

Alternatively, you can open the **View By** form by right-clicking in the Annotation Browser column header and choosing *View By – Advanced*. For more information, see Annotation Browser Shortcut Menus.

Field	Description
View items by	Specifies a category by which to view items. ■ **Ascending**: Sorts the items by the lowest number or in alphabetical order ■ **Descending**: Sorts the items by the highest number or reverse alphabetical order.
Then by	Specify category by which to view items for second, third, and fourth-level groupings.
Defaults	Restores the default grouping for the tab (for example, the default grouping for the *Connectivity* tab is *Type/Net*)
Apply	Applies changes to the markers currently displayed in the Annotation Browser.
OK	Applies the changes and to closes the *View By* form.

Related Topics

Annotation Browser Assistant

Annotation Browser Options Form

Custom Show Filter Form

Incomplete Net Filter Form

© 2023 Cadence Design Systems, Inc.

XL Probe Form

Use the XL Probe Form form to filter the objects that can be selected by clicking in a window and to locate a specific object to probe.

Field	Description
Object Filter	Specifies the types of objects that can be probed when you click in the layout or schematic window. For example, if you check only *Pins*, you can create probes only for pin objects by clicking in either window. Clicking on a net or instance has no effect. The object filters are applied in the order in which they appear in the form. For example, if you check all three then click in the layout window, Layout XL first looks for a pin shape and creates a pin probe if it finds one. If not, it looks for a routing shape and creates a net probe if it finds one. If not, it looks for an instance and creates an instance probe if it finds one. When you create a probe on a pin, Layout XL probes only the pinFigs associated with the pin. When you create a probe on a net, Layout XL probes all the shapes on the specified net, including routing shapes and pinFigs. The *Object Filter* option has no effect on probe creation using the object list in the XL Probe form, only on probe creation by clicking directly in the layout or schematic window. Environment variables: probeNet and probePin
Send Messages to CIW	Sends information about the objects probed to the CIW and displays it in the box at the bottom of the form. Environment variable: probeInfoInCIW

© 2023 Cadence Design Systems, Inc.

Field	Description
Show	Lists the names of the nets, pins, net classes, or instances in the layout depending on which category you select in the cyclic field. When you click one or more items in the list box, the items are highlighted in the schematic and the layout, and the name, window, and equivalency information about each item is shown in the display box. When you remove a probe, that information disappears from the display box. CMX legacy net classes are listed in the XL Probe form but are not shown under the *Net Class* category in the Constraint Manager assistant. This does not impact the individual constraints migrated.
Probing Options	Opens the Highlight Options form, where you can specify how nets, instances, and terminals are probed and which layer and colors are used to draw the probes.
Zoom	Zooms to the bounding box of the currently selected objects.

Related Topics

Highlight Options

Design Probing

Probing Objects by Clicking in a Window

Probing Components in OTM, MTM, and MTO Relationships

Probing with Multiple Layout Cellviews

Probing a Hierarchical Design

© 2023 Cadence Design Systems, Inc.

© 2023 Cadence Design Systems, Inc.

E

Configure Physical Hierarchy Window

The Configure Physical Hierarchy window lets you specify how a layout implementation is generated from a schematic design.

Field	Description
CPH Menu Bar	Lets you access menu commands to perform different types of tasks in the Configure Physical Hierarchy window.
CPH Toolbar	Lets you access Configure Physical Hierarchy commands directly without opening any menus.
Hierarchy Configuration Mode in the CPH Window	Controls how the physical hierarchy is generated from the logical design, including which logical components are to be ignored in the physical implementation and which physical views are used to implement the logical components
Component Types Mode in the CPH Window	Lets you create, edit, and remove component types, set the parameters for device chaining and folding, and control how the components are placed in rows.
Soft Block Mode in the CPH Window	Lets you partition the logical design by defining and editing parameters for the soft blocks that will be created when you generate the physical hierarchy.
Top Cell Pane	Displays the top-level logical cellview and the corresponding physical cellview to be generated in the context of the current physical configuration.
	It is a dockable pane, which you can dock on either side of the main Configure Physical Hierarchy window, or leave undocked as a floating window.
	This pane is available in Hierarchy Configuration and Soft Block modes.

© 2023 Cadence Design Systems, Inc.

Field	Description
Logical	Shows the library, cell, and view names of the logical view associated with the current physical configuration view. Use the *Open* button to open (or raise, if already open) the specified logical view in the context of the current physical configuration.
Physical	Shows the library, cell, and view names of the physical view associated with the current physical configuration view. Use the *Open* button to open (or raise, if already open) the specified physical view in the context of the current physical configuration.
Open	Opens (or raises, if already open) the specified views in the context of the current physical configuration view, which might be different from the configuration the schematic was using previously. If either cellview is not fully specified, this is indicated by the text `<unknown>` in the relevant field and the *Open* button is grayed out. If there are multiple open layout views, all using the same physical configuration view, the Top Cell pane shows the layout view in the last layout window you clicked.
Global Bindings Pane	Specifies how the logical design is traversed and how the logical to physical correspondence for leaf cells (stop points) is made.
	It is a dockable pane, which you can dock on either side of the main Configure Physical Hierarchy window, or leave undocked as a floating window. This pane is available in Hierarchy Configuration and Soft Block modes.
Physical library list	Specifies the list of libraries that are searched to find the corresponding physical cell for a given logical cell. By default, the library containing the logical cell is always searched first. Environment variable: lxUseLibList
Logical switch view list	Specifies the view names that are used to descend into a hierarchical design to find layout views. Environment variable: viewList

© 2023 Cadence Design Systems, Inc.

Field	Description
Physical stop view list	Specifies the view names that are used to determine the corresponding physical view for a given logical view. When traversing a hierarchy, Configure Physical Hierarchy stops when it encounters a view with one of the specified names. Environment variable: stopList
Constraint view list	Lists the names of the views containing constraint data. This list is passed to the Virtuoso Schematic Editor when it is opened in the context of a physical configuration, allowing it to determine whether the contents of the Constraint Manager must be updated.

Related Topics

Physical Hierarchy Configuration

© 2023 Cadence Design Systems, Inc.

CPH Menu Bar

Menu Command	Description
File	
New	Creates a new physical configuration view in the current window.
Open	Opens an existing physical configuration view.
Open Schematic	Opens the schematic window.
Save	Saves the current physical configuration view.
Save As	Saves the current physical configuration view under a different name.
Discard Edits	Discards all the edits made since the last time you saved the view.
Make Editable Make Read Only	*Make Editable* makes a read-only physical configuration view editable. *Make Read Only* makes an editable physical configuration view read-only.
Load Floorplan File	Loads soft block parameters from a floorplan file.
Save Floorplan File	Saves the current soft block parameters to a floorplan file, which you can then re-use in a different Virtuoso session.
Exit	Closes the window. If the view contains any unsaved changes, you are prompted to save them before the window closes.
Window	
Panes – Top Cell	Toggles the display of the Top Cell pane.
Panes – Global Bindings	Toggles the display of the Global Bindings pane.
Panes – Attributes	Toggles the display of the Attributes pane.
Toolbars – File	Toggles the display of the File toolbar.
Options	
Soft Block Options	Opens the Soft Block Global Options Form where you specify global settings for the pin labels in the soft blocks of your design.

© 2023 Cadence Design Systems, Inc.

Related Topics

Physical Hierarchy Configuration

CPH Toolbar

Lets you access certain Configure Physical Hierarchy commands directly without opening any menus. As with all toolbars, you can use the handle to reposition it anywhere within the Configure Physical Hierarchy window. r

Icon	Description
	*File – New* Opens the New Physical Configuration form.
	*File – Open* Opens the Open Physical Configuration form.
	*File – Open Schematic* Opens the schematic window.
	*File – Save* Saves changes to the physical configuration of the design.
	*Choose Mode* Opens the Configure Physical Hierarchy window in the selected mode: Hierarchy Configuration, Component Types, or Soft Block.
	*Toggle Occurrence Mode* Switches between Instance and Occurrence modes in CPH.
	*Set Stop Level* Sets a global stop level at which to stop the elaboration of the logical design hierarchy. The stop level specified here overrides the value specified using the `cphStopLevel` environment variable.

Related Topics

Instance Mode and Occurrence Mode

cphStopLevel

© 2023 Cadence Design Systems, Inc.

Hierarchy Configuration Mode in the CPH Window

Hierarchy Configuration mode controls how the physical hierarchy is generated from your logical design, including which logical components are to be ignored in the physical implementation and which physical views are used to implement the logical components. For information on colors used in Hierarchy Configuration Instance and Cells tables; see Color Coding in the Hierarchy Configuration Instances and Cells Tables. The table also has a shortcut menu that lets you easily access the required commands; see Shortcut menu for the Hierarchy Configuration Instances and Cells Tables.

Field	Description
Instances	Shows a hierarchical representation of the schematic top-level cell and how each instance in the logical design is mapped to a layout view.
	For each instance in the design, the table shows the view list used to reach the schematic instance and the corresponding layout cellview found.
Name	Specifies the name of the top cell, hierarchical node, or schematic leaf instance derived using either the *Inherited View List* or the *View To Use* field.
Inherited View List	Specifies the list of view names to switch into when traversing the hierarchy from this node onward. This setting is inherited by all the subnodes and leaf cells under the current node, unless it is overridden by an *Inherited View List* or a specific *View To Use* setting on a subnode or leaf cell.
View To Use	Specifies the name of the view to use for a node or leaf cell in the hierarchy. If there is no *View To Use* specified, the leaf or node in question inherits the view list from its parent. The *View To Use* setting is not inherited by the children of any node on which it is set. If the design contains two instances of the same cell and you specify different views to be used for each instance, then the *Cells* table lists two entries for that cell, one for each view to use.
Physical Library	The name of the physical library containing the corresponding layout cellview.
Physical Cell	The name of the physical cell to use.
Physical View	The name of the physical view to use.

© 2023 Cadence Design Systems, Inc.

Field	Description
Cells	Shows a flat view of the cellviews used in the schematic design and lists the corresponding layout cellview.
	The table lists each cellview explicitly, so if your design contains two instances of the same cell and you specify different views to be used for each instance, then the *Cells* table lists two entries for that cell, one for each view to use.
Logical Library	The name of the logical library.
Logical Cell	The name of the logical cell.
Logical View	The name of the logical view.
Inherited View List	Specifies the list of view names to switch into when traversing the hierarchy from this node onward. This setting is inherited by all the subnodes and leaf cells under the current node, unless it is overridden by an *Inherited View List* or a specific *View To Use* setting on a subnode or leaf cell.
View To Use	Specifies the name of the view to use for a node or leaf cell in the hierarchy. If there is no *View To Use* specified, the leaf or node in question inherits the view list from its parent. The *View To Use* setting is not inherited by the children of any node on which it is set. If your design contains two instances of the same cell and you specify different views to be used for each instance, then the *Cells* table lists two entries for that cell, one for each view to use.
Physical Library	The name of the physical library containing the corresponding layout cellview.
Physical Cell	The name of the physical cell to use.
Physical View	The name of the physical view to use.

© 2023 Cadence Design Systems, Inc.

Field	Description
Inherited Stop List	Specifies the view names that cause hierarchy traversal to stop its search for a layout cellview. If a view with one of the specified names if found, it is used as the physical counterpart of the schematic instance in question. This setting is inherited by all the nodes and leaf cells under the current node, unless it is overridden by another *Inherited Stop List* or by a specific *Physical View* setting on a subnode or leaf cell.
Attributes	Lets you specify a number of cell and instance attributes that impact instance generation.
	The attribute label is drawn in *italic* text when it is unchanged from its default value. Hovering over a label shows more information about the current value.
Revert to default	 Restores the selected value to its default.
Generation	Lets you override a number of properties that affect layout generation, previously stored with the schematic.
Ignore for generation	Prevents the selected instance or cell from being generated in the layout by the *Generate All From Source* or *Generate Selected From Source* commands. This includes any instance or cell with the *ignore* property set and any instance or cell with a property listed in the propsUsedToIgnoreObjs environment variable.
Ignore for check	Prevents the selected instance or cell from being checked by the *Check Against Source* command. Any instance or cell that is ignored for generation is automatically ignored for check as well, along with any instance or cell which has a property listed in the propsUsedToIgnoreObjsForCheck environment variable.

© 2023 Cadence Design Systems, Inc.

Field	Description
Split mfactored devices	Controls whether Virtuoso Layout XL generates a schematic device with the mfactor Property property as a single device (*off*) or as multiple parallel-connected devices (*on*) in the layout. This option overrides the global option set on the *Generation* tab on the Connectivity form. Environment variable: mfactorSplit *Split mfactored devices* controls only whether Layout XL generates one device or several devices in the layout view. It does not change the widths of the generated devices. It is grayed out if the selected schematic instances do not have the mfactor property set.
Split sfactored devices	Controls whether Layout XL generates a schematic device with the sfactor Property property as a single device (*off*) or as multiple series-connected devices (*on*) in the layout. This option overrides the global option set on the *Generation* tab on the Connectivity form. Environment variable: sfactorSplit *Split sfactored devices* controls *only* whether Layout XL generates one device or several devices in the layout view. It does not change the widths of the generated devices. It is grayed out if the selected schematic instances do not have the sfactor property set.
Split fingered devices	Controls whether each finger of a schematic device will be placed in the layout as a separate device. Environment variable: fingerSplit
Remove device	Causes parasitic devices to be ignored by merging nets connected to the terminals of a single instance. See Ignoring Parasitic Devices.
Stop level	Specifies the hierarchy level to which the selected instance will elaborate in relation to the top level. See Setting the Instance Stop Level.

© 2023 Cadence Design Systems, Inc.

Field	Description
Parameter	Lets you specify the names of parameters to be ignored during layout generation or by *Check Against Source*, how parameter names are mapped between schematic and layout, and how parameter values are rounded.
To check	Lists the parameters that are checked by the *Check Against Source* command.
Ignore for generation	Lists the parameters that are ignored by the following commands: *Generate All From Source*, *Generate Selected From Source*, *Generate Clones*, *Check Against Source*, *Update Components And Nets*, *Update Layout Parameters*, *Update Schematic Parameters*. The following parameters are always ignored: ■ `instancesLastChanged` ■ `instNamePrefix` ■ `lxIgnoreParamForCAS` ■ `lxIgnoredParams` ■ `lxMFactorNum` ■ `lxParamsToIgnore` ■ `lxParamsToIgnoreForCheck` ■ `lxPlacementStatus` ■ `lxRounding` ■ `lxStopList` ■ `lxTimeStamp` ■ `lxUseCell` ■ `pin#` ■ `posi`
Ignore for check	Lists the parameters that are ignored by the *Check Against Source* command. Mismatches for any of the listed parameters are not reported by the check. The list inherits all the parameter names from the *Ignore for generation* list.

© 2023 Cadence Design Systems, Inc.

Field	Description
Name mapping	Defines the mapping between parameter names in the schematic and layout cellviews; for example: `l L ; w W ;` Maps schematic parameters `l` and `w` to layout parameters `L` and `W` respectively. This field is not available as an instance attribute because you cannot override this setting for an individual instance of a cell.
Rounding	Defines how the value of a specified parameter on a schematic cell master or instance is rounded when it is evaluated by Layout XL. You can use this property in conjunction with the *Width parameter* to prevent folded devices from becoming off-grid; for example, the following setting rounds the value of `w` to the closest multiple of 0.05 microns. `(w 0.05 round)` See Rounding Parameter Values.
Terminals	Lets you specify the names of terminals to be ignored during layout generation or by *Check Against Source*, and how terminal names are mapped between schematic and layout.
Ignore for generation	Lists the terminals that are ignored by the following commands; *Generate All From Source*, *Generate Selected From Source*, *Generate Clones*, *Check Against Source*, *Update Components And Nets*, *Update Layout Parameters*, *Update Schematic Parameters*.
Ignore for check	Lists the terminals that are ignored by the *Check Against Source* command. Mismatches for any of the listed terminals are not reported by the check.
Name mapping	Defines the mapping between terminal names in the schematic and layout cellviews; for example: `B G ; X D ;` Maps schematic terminals `B` and `X` to layout terminals `G` and `D` respectively. This field is not available as an instance attribute because you cannot override this setting for an individual instance of a cell.

© 2023 Cadence Design Systems, Inc.

Color Coding in the Hierarchy Configuration Instances and Cells Tables

Text Color	Description
Instances	Displays the color coding used for different entries in the *Instances* table.
Gray	If the text label for a schematic instance has a strikethrough and the entries against it in the physical library, cell, and view fields are in gray text it means that the schematic instance is set to be ignored. Instance bindings under a valid physical stop point are also shown in gray text, indicating that although the binding has been set up, it is never reached because of the physical stop point at a higher level.
Black on pink background	For a: ■ Layout field, this means that the schematic instance has no corresponding layout. ■ Leaf cell, this means that it has no corresponding layout. ■ For a block, it means that at least one of its children does not have a corresponding layout.
Red	The physical binding points to a cellview that does not exist.
Blue	The value was typed in manually by the user, or a property was created while converting data to use the Layout XL schema.
Cells	Displays color coding used for different entries in the *Cells* table.
Gray	If the text label for a schematic instance has a strikethrough and the entries against it in the physical library, cell, and view fields are in gray text it means that the schematic instance is set to be ignored. **Instance bindings under a valid physical stop point are also shown in gray text, indicating that although the binding has been set up, it is never reached because of the physical stop point at a higher level.**
Red	The physical binding points to a cellview that does not exist.
Blue	The value was typed in manually by the user, or a property was created while converting data to use the Layout XLschema.

© 2023 Cadence Design Systems, Inc.

Shortcut menu for the Hierarchy Configuration Instances and Cells Tables

The shortcut commands, also called the shortcut menu commands, are available on right-clicking an instance within the Hierarchy Configuration Instances and Cells tables. These commands are enabled and disabled dynamically depending on the instance that is selected in the table view.

Command Name	Description
Instances	Lets you access the shortcut menu commands that you can use to perform specific tasks in the *Instances* table.
Set/clear stop point	Makes or clears the selected instance stop point in the hierarchy traversal. A 'STOP' icon against the schematic instance indicates that traversal stops at this instance.
Set/remove force descend	Forces hierarchy traversal to proceed beyond a leaf instance or removes the setting from the selected instance. A 'go' icon indicates that the selected instance will now be traversed even though it was originally being considered as a leaf instance. You can set the force descend only on instances that have a physical binding.
Create Physical Binding	Lets you specify the physical binding information, such as physical library and cellview, for the selected instance.
Clear Physical Binding	Lets you clear the existing physical binding information for the selected instance.
Edit Multiple <fieldName>	Lets you change the value of the specified field for multiple instances to a common value either by typing in the new setting or by selecting a value from the drop-down list. This command appears in the shortcut menu only when multiple instances are selected.
Cells	Lets you access the shortcut menu commands that you can use to perform specific tasks in the *Cells* table.
Create Physical Binding	Lets you specify the physical binding information, such as physical library and cellview, for the selected cell.
Clear Physical Binding	Lets you clear the existing physical binding information for the selected cell.

© 2023 Cadence Design Systems, Inc.

Command Name	Description
Edit Multiple <fieldName>	Lets you change the value of the specified field for multiple cells to a common value either by typing in the new setting or by selecting a value from the drop-down list. ■ This command appears in the shortcut menu only when multi*ple cells are selected.* ■ The command does not support updating the logical library, cell, and view values.
Create/remove VPLGen	Lets you create (or remove) predefined layouts for schematic instances that differ only in terms of their inherited parameter values.
Create/remove VLGen	Lets you create (or remove) the layout for schematic symbols that do not have an existing layout available to reuse and have no CDF parameters defined.
Create/remove Virtual Hierarchy	Lets you create (or remove) virtual hierarchies for the selected cells when selective virtual hierarchy generation mode is enabled. To disable the selective virtual hierarchy generation and to allow generation of all virtual hierarchies, use the vhSelectiveMode environment variable.

Related Topics

Hierarchy Configuration Mode in the CPH Window

Component Types Mode in the CPH Window

Soft Block Mode in the CPH Window

© 2023 Cadence Design Systems, Inc.

Component Types Mode in the CPH Window

Component Types Mode lets you view the library-level component types. In addition, this mode lets you create, edit, and remove the design-level component types that identify the *NMOS* and *PMOS* transistor cells in the design and set the parameters for device chaining and folding. The table also has a shortcut menu that lets you easily access the required commands; see Shortcut Menu for the Component Types Table

Field	Description
Cells	Shows the cells in the design organized by component type. If a cell is not assigned to a component type, it is put in the *No component type* folder.
Physical Library	Shows the name of the library containing the physical representation of the logical leaf cell.
Physical Cell	Shows the physical equivalent of the logical leaf cell.
Attributes	Lets you edit the attributes for the selected component type. These controls are enabled only when you have a single component type selected in the *Cells* table
Component class	Specifies the type of devices in the component type. Valid values are: Leave the field blank to set component class as undefined. ■ Choose *PMOS* or *NMOS* if you are using the MOS component-assisted row generation. The *Views* field is disabled in this case. ■ Choose *PFIN* or *NFIN* if you are using FinFET devices. ■ Choose *RESISTOR*, *CAPACITOR*, or *INDUCTOR* to recognize the split parameter on these devices when manually folding them. The information is used by the Layout XL *Check* and *Update* commands to verify if the parameter has been split and if the split has been accurate. ■ Choose *STDCELL* if you are using the Standard Cell component-assisted row generation. All other settings are disabled in this case.

© 2023 Cadence Design Systems, Inc.

Field	Description
	■ Choose *STDSUBCONT* or *FILLER* for substrate contacts and filler cells respectively. Only the *Views* field is enabled; everything else is disabled in this case. ■ Choose *BOUNDARYCELL* for boundary cells. Only the *Views* and *Edge type* fields are enabled; everything else is disabled in this case. ■ Choose *ROUTING* for routing only cells, which are cells that contain no devices ■ Choose *GUARDRING* for guard-ring cells, which are cells that prevent unintended signal leakage to or from the components they surround. ■ (Virtuoso Advanced Node Option for GAA) Choose *NGAA* or *PGAA* if you are using Gate All Around transistor devices. These component classes are available in Layout EXL or higher tiers.
Active layer	Specifies the diffusion layer-purpose pair for *NMOS* and *PMOS* devices. The drop-down lists all the layers from the technology file for which the function is set to one of the following: `pdiff`, `ndiff`, `pwell`, `nwell`, `pplus`, `nplus`, or `diffusion`. This parameter is required for *NMOS and PMOS* component types to control device chaining and folding during placement. The Custom Digital Placer also uses this layer to determine the spacing rules to be used when seeding the defined spacing type in the fields on the Placement Planning (Assisted CMOS) form. The *Calculate Estimates* command for Assisted CMOS placement uses the spacing rule between adjacent MOS device chains to achieve more accurate estimates.
Width parameter	Specifies the name of the transistor width parameter on the device master cell. The default name is `w`. If you use a different name for the transistor width parameter, type that name into the text field. You must set this parameter for *NMOS* and *PMOS* component types even if the devices in the component type will not be folded.
Length parameter	Specifies the name of the transistor length parameter on the device master cell. The default name is `l`. If you use a different name for the transistor width parameter, type that name here.

© 2023 Cadence Design Systems, Inc.

Field	Description
Split parameter	Specifies the name of the parameter to be split.
Folding minimum	Specifies the minimum width that can be used when folding transistors with width greater than the folding threshold.
Folding threshold	Specifies the maximum width beyond which transistors are automatically folded. You must set this parameter for *NMOS* and *PMOS* component types even if the devices in the component type will not be folded. The value applies globally to all devices within the scope of the component type definition (that is, the library containing the current cell or just the current cell).
Folding step	Specifies the available width values that lie between the *Folding minimum* and the *Folding threshold v*alues and can be used for folding. If the *Folding min = 0.3u, Folding threshold = 0.9u, and the Folding step = 0.3u*, the following commands will create or update devices to be *0.3u, 0.6u, or 0.9u* wide. ■ *Generate All From Source* ■ *Generate Folded Devices* ■ *Update Components And Nets* ■ *Update Layout Parameters*
Edge type	Specifies the edge along which the boundary cells can be placed around core cells.It is supported only for the cells that have their *Component class* set to BOUNDARYCELL. Choose the edge type from: *Left*, *Right*, *LeftTop*, *LeftBottom*, *RightTop*, and *RightBottom*.
Views	Specifies the view name to be used when defining a standard cell substrate contact or filler cell. This parameter is valid only when the *Component class* is set to *STDSUBCONT* or *FILLER*.

© 2023 Cadence Design Systems, Inc.

Field	Description
Orientation parameter	Reads the orientation value, if defined in the LAM file. Else, specifies the orientation value for the selected component type to be used during generation. The orientation value defined using the environment variable, lxGenerationOrientation, is considered when the value is not already defined in the LAM file or using the *Orientation parameter* field.
Components parameters	Specifies multiple component parameters for a single cell to allow the same cell to be assigned different component types. To determine the supported component types for a cell, the component parameters are compared to the CDF parameters on the device master cell. The component parameters for a design master cell can only be specified or edited using the LAM file. The value is not editable using the Configure Physical Hierarchy window.
Terminal Names	Lets you edit the default names used to identify the *Drain*, *Gate*, *Source*, and *Bulk* terminals in the cells assigned to the component type. The names are not case-sensitive. The defaults are: `drain` and `D`; `gate` and `G`; `source` and `S`; and `bulk` and `B`, respectively.
Number of fins	Specifies the name of the number of fins parameter on the FinFET master cell. You can choose *nFins*, *nfins*, *NFINS*, or an empty string from the drop-down list. You can also specify a default value for this field by defining it in the LAM File Component Type Section. The value specified in this field is used to determine the whole number of fins that a FinFET device should be folded into during layout generation using commands such as *Generate All From Source*. To fold or chain FinFET devices based on number of fins, the *Number of fins* parameter and the *Folding threshold fins* values are required. *Width parameter*, *Folding threshold*, *Views*, and *Width per fin* are disabled in this case. This attribute supports only the *NFIN* and *PFIN* component types. The parameter is required for driving FinFET chaining and folding.

© 2023 Cadence Design Systems, Inc.

Field	Description
Number of fins	Specifies the name of the number of fins parameter on the FinFET master cell. You can choose *nFins*, *nfins*, *NFINS*, or an empty string from the drop-down list. You can also specify a default value for this field by defining it in the LAM File Component Type Section. The value specified in this field is used to determine the whole number of fins that a FinFET device should be folded into during layout generation using commands such as *Generate All From Source*. To fold or chain FinFET devices based on number of fins, the *Number of fins* parameter and the *Folding threshold fins* values are required. *Width parameter*, *Folding threshold*, *Views*, and *Width per fin* are disabled in this case. This attribute supports only the *NFIN* and *PFIN* component types. The parameter is required for driving FinFET chaining and folding.
Width per fin	Specifies the width of each fin. To fold or chain FinFET devices based on the *Width parameter*, the *Width per fin*, and the *Folding threshold fins* values are required. *Folding threshold, Views,* and *Number of fins* are disabled in this case, as displayed in the figure below. If automatic folding is driven by the *Width parameter,* it uses the product of the *Width per fin* value and the *Folding threshold fins* value to achieve a whole number of fins per fold. This attribute supports only the *NFIN* and *PFIN* component types. The parameter is required for driving FinFET chaining and fold
Folding threshold fins	Specifies the maximum number of fins beyond which FinFETs are automatically folded. This attribute supports only the *NFIN* and *PFIN* component types. The parameter is required for driving FinFET chaining and folding.

© 2023 Cadence Design Systems, Inc.

Shortcut Menu for the Component Types Table

The shortcut commands, also called the shortcut menu commands, are available on right-clicking a cell within the Components Type table. These commands are enabled and disabled dynamically depending on the cell that is selected in the table view.

Command Name	Description
Cells	**Lets you access the shortcut menu commands that you can use to perform specific tasks in the *Cells* table.**
Add component type	Opens the Create Component Type dialog, which lets you create a new component type to be added to the table view. See Defining a Design-Level Component Type.
Remove component type	Removes the selected component type from the table view. You can remove only design-level component types. Evicted cellviews are put into the *No component type* folder. See Removing a Component Type.
Move cells	Opens the Move Cells dialog, which lets you move selected cells between component types. See Defining a Design-Level Component Type.
Show cells	Choose *In design* (the default) only to show the physical cells that will be generated in the layout based on the current physical configuration view. Choose one or more of the library names to toggle the display of cells from that library. The submenu lists all the libraries from the library definitions file, `cds.lib`. See Component Types Cells Table.f

Related Topics

Component Types Mode in the CPH Window

Hierarchy Configuration Mode in the CPH Window

Soft Block Mode in the CPH Window

Folding a Transistor or FinFET Device

© 2023 Cadence Design Systems, Inc.

Soft Block Mode in the CPH Window

In Soft Block mode you can partition your logical design by defining and editing parameters for the soft blocks that will be created when you generate the physical hierarchy. The table also has a shortcut menu that lets you easily access the required commands; see Shortcut Menu for the Soft Block Instances Table

Field	Description
Instances	Lets you specify instance parameters for soft blocks.
Names	Lists the instances in the design organized by cell type into *Core*, *Custom*, *Hard Blocks*, *I/Os*, and *Soft Blocks*. Instances with no physical view are also grouped together.
Inherited View List	Shows the logical switch view list used to derive the leaf cells in the logical hierarchy.
Physical Library	Shows the name of the library containing the physical representation of the logical leaf cell.
Physical Cell	Shows the physical equivalent of the logical leaf cell.
Physical View	Shows the name of the physical view.
Block Type	Specifies the type of block to be generated, either CUSTOM or DIGITAL. This influences how the place and route boundary and pins in the design are snapped during level-1 editing. ■ *CUSTOM* specifies that the pin edges and the place and route boundary are snapped to the manufacturing grid. ■ *DIGITAL* specifies that the pin centers are snapped to the routing grid and the place and route boundary is snapped to the placement grid.

© 2023 Cadence Design Systems, Inc.

Field	Description
Type	Specifies whether the view that will be generated for a soft block is a *LAYOUT* (the default) or an *ABSTRACT*. ■ *LAYOUT* specifies a layout view that contains instances, connectivity, pins, place and route boundary and other detailed information, generated from the schematic view by the *Generate Physical Hierarchy* command. Use this setting if there are physical representations for the logical cells at lower levels of the hierarchy. ■ *ABSTRACT* specifies a special type of layout view containing only pins and place and route boundary based on symbol information, generated using the *Generate Abstract From Symbol* command. Use this setting if you do not have detailed physical views for the lower levels of the hierarchy but you want to start the floorplan by generating abstract views. This field is blank for all instances other than soft blocks that exist already or instances for which soft block parameters have been defined.
Inherited Stop List	Shows the physical view list used to determine the corresponding physical view for a given logical view.
Attributes	Lets you edit the attributes for the selected soft block.
Save Soft Block	Saves all the attribute settings for the block.
Discard Changes to Soft Block	Discards any changes you have made since the last time the block was saved.

© 2023 Cadence Design Systems, Inc.

Field	Description
Boundary	Specifies the shape and size of the boundary that encloses the soft block
Rectangle	Specifies a rectangular boundary. Choose two of the available parameters to determine the dimensions of the boundary for a given value of *Core Area*. ■ *Aspect Ratio (W/H)* ■ *Boundary Width (W)* ■ *Boundary Height (H)* ■ *Utilization* Choose one of the above from the first cyclic list and any one of the remaining valid choices from the second cyclic list. If one of the parameters you choose is *Utilization*, you need to specify the core area to which the utilization percentage can be applied.
Area Calculation	This group box offers you three ways to specify the area value.
Manual	Lets you type the value you want into the *Core Area* field at the bottom of the form.
Avg. Area Per Gate	Lets you specify the gate count and the average area per gate. The system calculates the area by multiplying the two values.
Use Estimator	Lets you choose between PR Boundary Based and BBox Based area estimator. The PR Boundary Based estimation sums up the polygonal area of the individual instances to derive the overall area estimation. The BBox Based estimation, on the other hand, sums up the minimum bounding box area of the individual instances to derive the overall area estimation. You can also register your own area estimation functions and use them, as appropriate. See Creating and Registering a User-Defined Area Estimation Function.
Estimate	Modifies any user-defined parameters for the area estimation function before running the function.
Rail Height	Specifies an optional parameter. The rail height value is used to extend the total area to accommodate the rails at the top and bottom of the core area.

© 2023 Cadence Design Systems, Inc.

Field	Description
Polygon	Specifies a polygonal boundary with no area estimation. Specify the coordinates required to define the boundary in the fields provided.
I/O Pins	Lists the I/O pins in the design.
	All fields with a white background are editable. By default the table lists all the top-level pins detected in the schematic design, including schematic pins that are connected to I/O pads (cells of type `pad`, `padSpacer`, or `padAreaIO`). To generate only pads but no pins, set the initCreatePadPins environment variable to `nil`. If the soft block has already been generated in the layout, the *Layer*, *Width*, and *Height* fields show the dimensions and layer purpose of the pin shape in the layout. If the soft block has not been generated, the default minimum width and height specified in the technology file are shown, while the default layer is specified by the Layout XL initIOPinLayer environment variable.
Add Pin	Adds a new pin to the table.
Delete Pin	Deletes a selected pin.
Create Label	Generates a label for each of the pins automatically in the current soft block. To set the style for the labels, choose *Options – Soft Block Options* from the Configure Physical Hierarchy menu bar and choose the *Create Label* check box to activate the options under the *Pin Label Text Style* group box.
LayerHalo Obstruction	Lets you create routing and placement obstructions. You can create only one placement obstruction per soft block.
Blockage Type	Specifies the type of obstruction to be created. Choose from *placement*, *routing*, *fill*, *slot*, *pin*, *feedthru*, or *screen*.
Material	Filters the list of available layers based on the material defined for that layer in the technology file. This field is not available when defining a placement obstruction.
Layer	Specifies the layer on which the obstruction is created. This field is not available when defining a placement obstruction.
Offsets	Define the position of the obstruction.

© 2023 Cadence Design Systems, Inc.

Field	Description
Add	Adds the obstruction to the soft block definition, or
Delete	Deletes a selected obstruction.
Cover Obstruction	Lets you create a cover obstruction on the selected layer and lower layers.
Select a Top Layer	Specifies the top layer on which the obstruction is created.
Obstruction Applied on Layers	Lists the layers on which obstructions are created.
Create	Creates a new cover obstruction.
Delete	Deletes a cover obstruction.
Allow PGNet	Allows power and ground nets.

Shortcut Menu for the Soft Block Instances Table

Click the right mouse button to reveal a shortcut menu with a number of other options. These are enabled and disabled dynamically depending on what is selected in the table view.

Command Name	Description
Define Soft Block Parameters *Define Top Level Parameters* (if the top-level block is selected)	Marks the selected nodes of a schematic or netlist view so that the *Floorplan – Generate Physical Hierarchy* command can create soft blocks for them in the physical hierarchy. See Defining Soft Block Parameters. If the selected component already has soft block parameters defined, this menu item changes to *Edit Soft Block Parameters* (or *Edit Top Level Parameters* if the top-level block is selected).
Initialize Soft Block Parameters Using Physical View	Imports soft block parameters from another layout cellview. This command uses the same functionality as the *Load Physical View* command. See Saving and Loading Soft Block Parameters to a Floorplan File Defining Soft Block Parameters.

© 2023 Cadence Design Systems, Inc.

Command Name	Description
Remove Soft Block Parameters	Removes the soft block parameters from the selected nodes in the tree. See Removing Soft Block Parameters.
Set Master Cell Type	Sets the cell type for the selected instances. Cell types are used to categorize the blocks in your design. See Setting the Cell Type.
Create Physical Binding *Clear Physical Binding*	Sets or clears a physical binding for the selected instance. See Creating a Physical Binding

Soft Block Table View State Indicators

The following icons are used to indicate different states in the soft block table view.

Directory Name	Icon	Description
Core *IO* *Hard block*		Indicates a physical leaf layout.
Soft block		With the hierarchy symbol (+), indicates a schematic logical node that has a corresponding physical view which is a soft block. Without the hierarchy symbol, indicates a symbol logical node that has a corresponding physical view which is a soft block. **Note:** You cannot see the hierarchy for the logical node.
		Indicates that the soft block parameters were redefined.

© 2023 Cadence Design Systems, Inc.

No Physical	*Hierarchy symbol (+)* AND *File Directory*	Indicates that there is hierarchy underneath the node; for example, a schematic logical node.
	Octagon symbol	Indicates a logical node.
		Indicates that the soft block parameters are defined.

Related Topics

Hierarchy Configuration Mode in the CPH Window

Component Types Mode in the CPH Window

© 2023 Cadence Design Systems, Inc.

© 2023 Cadence Design Systems, Inc.

F

Library and Attributes Mapping File Syntax

This topic describes the Virtuoso® Layout Suite XL library and attributes mapping (LAM) file, that supports the physical hierarchy configuration functionality.

The LAM file stores the symbol properties that drive layout generation, component type information for a given library, and library and cell-level mapping information. The file is called `cph.lam` and is created automatically when you convert your library data to use the Layout XL schema using the `cphUprevLibrary` SKILL function. For more information, see cphUprevLibrary in the *Virtuoso Layout Suite SKILL Reference*.

You can override the settings in the library LAM file by creating your own version called *`library`*`.lam` and placing it in one of the locations defined in your `setup.loc` file. For more information, see LAM File Locations.

Important

You cannot export a LAM file from the Configure Physical Hierarchy window and Virtuoso does not provide any graphical user interface to help you create or edit a LAM file. Cadence recommends that your CAD group generate the required LAM files for a given GPDK from your golden library sources, or that you use a text or XML editor to create or edit your own LAM files.

Related Topics

Physical Hierarchy Configuration

LAM File Locations

Basic LAM File Structure

LAM File Logical Section

LAM File Physical Section

© 2023 Cadence Design Systems, Inc.

LAM File Component Type Section

LAM File Mapping Section

Sample LAM File

LAM File Locations

The LAM file generated as part of the library conversion process is stored in the following location.

- *library*/.cadence/dfII/cph/lam/cph.lam

During LAM initialization, this file is imported for each library defined in the library definitions file, cds.lib.

The system then searches the locations defined in the setup.loc file for any other *library*.lam file and loads the first one it finds. The settings in the *library*.lam file override the settings in the *library*/.cadence/dfII/cph/lam/cph.lam file.

Example

The default setup.loc file is

```
.                          current working directory always searched first
@LIBRARY                   look in the design libraries for the file
$CDS_WORKAREA              user workarea if defined
$CDS_SEARCHDIR             this is set by various tools during tool startup
$HOME
$CDS_PROJECT               project storage area, ignored if not defined
$CDS_SITE                  Site Setup Info - default is $CDS_INST_DIR/share/local
$(compute:THIS_TOOL_INST_ROOT)/share    Cadence Default Setup Info
```

For example, if a library called acpd is part of the library list, then during LAM initialization the system imports

- acpd/.cadence/dfII/cph/lam/cph.lam

And then checks for the following files

- ./.cadence/dfII/cph/acpd/lam/acpd.lam
- <alllibs- inc acpd>/.cadence/dfII/cph/acpd/lam/acpd.lam
- $CDS_WORKAREA/.cadence/dfII/cph/acpd/lam/acpd.lam
- ...

© 2023 Cadence Design Systems, Inc.

The system imports the first `acpd.lam` it finds and stops searching.

You can use the following syntax to insert a comment in a LAM file.

```
<!-- This is a comment -->
```

Related Topics

Library and Attributes Mapping File Syntax

© 2023 Cadence Design Systems, Inc.

Basic LAM File Structure

```
<!DOCTYPE LAM_FILE>
<LIBRARY_DATA>
  <VERSION NUMBER="versionNumber" />
  <LOGICAL>
    <!-- logical section goes here -->
  </LOGICAL>
  <PHYSICAL>
    <COMPONENT_TYPE_GROUPS>
      <!-- component type definitions -->
    </COMPONENT_TYPE_GROUPS>
    <>
      <!-- logical to physical  -->
    </>
  </PHYSICAL>
</LIBRARY_DATA>
```

Description

Identifies the file as a library and attributes mapping (LAM) file, indicates the version of the data it contains, and differentiates between logical and physical information contained in the file. The physical section is further divided into a section that defines component type groups and a section that specifies logical-to-physical mapping for the design.

Statements

Statement	Description
`<!DOCTYPE LAM_FILE>`	Identifies the file as a library and attributes mapping (LAM) file.
`<LIBRARY_DATA>`	Encloses the data in the current LAM file.
`<VERSION NUMBER="versionNumber" />`	Indicates the version of the attributes and mappings in the current file. For example, `<VERSION NUMBER="0.2" />`
`<LOGICAL>`	Encloses the section of the LAM file that contains logical information. For more information, see LAM File Logical Section on page 2054.
`<PHYSICAL>`	

© 2023 Cadence Design Systems, Inc.

Statement	Description
	Encloses the section of the LAM file that contains physical information. For more information, see LAM File Physical Section on page 2063.
`<COMPONENT_TYPE_GROUPS>`	
	Encloses the library-level component type definitions in the LAM file. For more information, see LAM File Component Type Section on page 2065.
`<>`	
	Encloses all the statements used to map physical cellviews, parameters, and terminals to their counterparts in the logical design. For more information, see LAM File Mapping Section on page 2080.

Related Topics

Library and Attributes Mapping File Syntax

LAM File Logical Section

LAM File Physical Section

LAM File Component Type Section

LAM File Mapping Section

Sample LAM File

© 2023 Cadence Design Systems, Inc.

LAM File Logical Section

```
<LOGICAL>
  <LIB NAME="libName" >
    <CELL NAME="cellName" >
      <VIEW NAME="viewName" >
        <MFACTOR_SPLIT>t | nil</MFACTOR_SPLIT>
        <SFACTOR_SPLIT>t | nil</SFACTOR_SPLIT>
        <REMOVE_DEVICE>expression</REMOVE_DEVICE>
        <STOP_LIST>list_of_viewNames</STOP_LIST>
        <VIEW_LIST>list_of_viewNames</VIEW_LIST>
        <PARAMETERS>
          <INCLUDE>
            <CHECK>list_of_parameterNames</CHECK>
          </INCLUDE>
          <IGNORE>
            <GEN>list_of_paramNames</GEN>
            <CHECK>list_of_paramNames</CHECK>
          </IGNORE>
        </PARAMETERS>
        <TERMINALS>
          <IGNORE>
            <GEN>list_of_termNames</GEN>
            <CHECK>list_of_termNames</CHECK>
          </IGNORE>
        </TERMINALS>
      </VIEW>
    </CELL>
  </LIB>
</LOGICAL>
```

Description

Specifies for each cellview in the logical design the view list that was used to find the cellview in the hierarchy, the stop list that will be used to find the corresponding physical view, and options that specify whether the device in question is to be generated or ignored in the physical view.

Statements

Statement	Description
<LIB NAME="*libName*" >	Specifies the logical library name and encloses all the statements relating to that library. The name must be enclosed in quotation marks.
<CELL NAME="*cellName*" >	

© 2023 Cadence Design Systems, Inc.

Statement	Description
	Specifies the logical cell name and encloses all the settings relating to that cell. The name must be enclosed in quotation marks.
`<VIEW NAME="viewName" >`	
	Specifies the logical view name and encloses all the settings relating to that cellview. The name must be enclosed in quotation marks.
`<MFACTOR_SPLIT>t | nil</MFACTOR_SPLIT>`	
	Controls whether Layout XL generates logical devices with the mfactor Property property as multiple devices in the physical view. **GUI:** *Split mfactored devices*
`<SFACTOR_SPLIT>t | nil</SFACTOR_SPLIT>`	
	Controls whether Layout XL generates logical devices with the sfactor Property property as multiple devices in the physical view. **GUI:** *Split sfactored devices*
`<REMOVE_DEVICE>expression</REMOVE_DEVICE>`	
	Causes parasitic devices to be ignored by merging nets connected to the terminals of a single instance. For example, to short terminals `PLUS` and `MINUS` on a resistor, type the following. `<REMOVE_DEVICE>(short(PLUS MINUS))</REMOVE_DEVICE>` Do not enclose the expression in quotation marks. **GUI:** *Remove device*
`<STOP_LIST>list_of_viewNames</STOP_LIST>`	
	Lists the view names to be used to determine the corresponding physical view for a given logical view. When traversing a hierarchy, Configure Physical Hierarchy stops when it encounters a view with one of the specified names. Separate each name with a single whitespace. Do not enclose the list in quotation marks. **GUI:** *Physical stop view list*

© 2023 Cadence Design Systems, Inc.

Statement	Description
`<VIEW_LIST>`*`list_of_viewNames`*`</VIEW_LIST>`	
	Specifies the view names used to descend into a hierarchical design to find logical views. Separate each name with a single whitespace. Do not enclose the list in quotation marks. **GUI:** *Logical switch view list*
`<PARAMETERS>`	
	Encloses the `<INCLUDE>` and `<IGNORE>` statements that specify which logical instance parameters are to be checked or which are the ones to be ignored for generation and check. If the `<INCLUDE>` statements are specified, these are given precedence over the `<IGNORE>` statements. **GUI:** Options on the *Parameters* tab
`<TERMINALS>`	
	Encloses the `<IGNORE>` statements that specify which logical instance terminals are to be ignored for generation and check. **GUI:** Options on the *Terminals* tab
`<IGNORE>`	
	Encloses the `<GEN>` and `<CHECK>` statements that specify which logical instance parameters (or terminals) are to be ignored for generation or check. The `ignore` parameter is also used by the dummy instance backannotation commands to determine which parameters are to be ignored for backannotation. For more information, see Dummy Instances Backannotation.
`<GEN>`*`list_of_names`*`</GEN>`	

© 2023 Cadence Design Systems, Inc.

Statement	Description
	Lists the logical instance parameters (or terminals) that are ignored by the following commands; *Generate All From Source*, *Generate Selected From Source*, *Generate Clones*, *Check Against Source*, *Update Components And Nets*, *Update Layout Parameters*, *Update Schematic Parameters*. Separate each name with a single whitespace. Do not enclose the list in quotation marks. **GUI:** *Ignore for generation*
`<CHECK>list_of_names</CHECK>`	Lists the logical instance parameters (or terminals) that are ignored by the *Check Against Source* command. Mismatches for any of the listed parameters are not reported by the check. The list inherits all the parameter names from the *Ignore for generation* list. Separate each name with a single whitespace. Do not enclose the list in quotation marks. **GUI:** *Ignore for check*

Example 1

```
<LOGICAL>
  <LIB NAME="basic" >
    <CELL NAME="cellA" >
      <VIEW NAME="symbol" >
        <REMOVE_DEVICE>(short(src dst))</REMOVE_DEVICE>
      </VIEW>
    </CELL>
  </LIB>
  <LIB NAME="lib1" >
    <CELL NAME="cellB" >
      <VIEW NAME="symbol" >
        <MFACTOR_SPLIT>t</MFACTOR_SPLIT>
        <REMOVE_DEVICE>(short(PLUS MINUS) funcR(r))</REMOVE_DEVICE>
        <STOP_LIST>abstract layout layoutS</STOP_LIST>
        <VIEW_LIST>schematic symbol layout compacted</VIEW_LIST>
        <PARAMETERS>
          <IGNORE>
            <GEN>w l</GEN>
            <CHECK>a b</CHECK>
          </IGNORE>
        </PARAMETERS>
      </VIEW>
    </CELL>
  </LIB>
</LOGICAL>
```


© 2023 Cadence Design Systems, Inc.

In the example above,

For cellview	LAM specifies that
`basic/cellA/symbol`	The cellview is to be ignored by merging the nets connected to its terminals `src` and `dst`.
`lib1/cellB/symbol`	Multiple physical devices are to be generated for mfactored instances of this cellview. The cellview is to be ignored by merging the nets connected to terminals `PLUS` and `MINUS` but only if the function funcR(r) returns a value that is non-nil. The view to be used to generate the physical equivalent of this device is either `abstract`, `layout`, or `layoutS`. The `w` and `l` parameters are to be ignored during generation and that the `a` and `b` parameters are to be ignored during check.

Example 2

```
<PHYSICAL>
    <MAPPINGS>
      <LIB NAME="hierarchy" >
        <CELL NAME="inv" >
          <VIEW NAME="layout" >
            <LOGICAL_CELL>
              <LIB NAME="hierarchy" >
                <CELL NAME="inv" >
                  <VIEW NAME="symbol" >
                    <VLGEN>t</VLGEN>
                    <VPLGEN>t</VPLGEN>
                    <PARAMETERS>
                     <VPL>pl 1;pw 1;nl 1;nw 1</VPL>   #VPLGEN parameters and
their default values
                    </PARAMETERS>
                  </VIEW>
                </CELL>
              </LIB>
            </LOGICAL_CELL>
          </VIEW>
```


© 2023 Cadence Design Systems, Inc.

```
        </CELL>
        <CELL NAME="nand2" >
          <VIEW NAME="layout" >
            <LOGICAL_CELL>
              <LIB NAME="hierarchy" >
                <CELL NAME="nand2" >
                  <VIEW NAME="symbol" >
                    <VLGEN>t</VLGEN>
                  </VIEW>
                </CELL>
              </LIB>
            </LOGICAL_CELL>
          </VIEW>
        </CELL>
      </LIB>
    </MAPPINGS>
</PHYSICAL>
```

The LAM file above specifies the schematic cells that need to be generated as VPLGens or VLGens.

When generating VPLGens, the following additional parameters can be specified:

```
<VPL>pl;pw;nl;nw</VPL>
```

If the default value is a CDF parameter for the symbol, you can choose not to mention the default value. The specified value for a VPLGen or VLGen parameter can be of type: float, boolean, or a string.

Example 3

```
<PHYSICAL>
    <MAPPINGS>
      <LIB NAME="hierarchy" >
        <CELL NAME="inv" >
          <VIEW NAME="layout" >
            <LOGICAL_CELL>
              <LIB NAME="hierarchy" >
                <CELL NAME="inv" >
                  <VIEW NAME="symbol" >
                    <VLGEN>t</VLGEN>
                    <VPLGEN>t</VPLGEN>
                    <PARAMETERS>
```


© 2023 Cadence Design Systems, Inc.

```
                    <VPL>pl 1;pw 1;nl 1;nw 1</VPL>   #VPLGEN parameters and
their default values
                  </PARAMETERS>
                </VIEW>
              </CELL>
            </LIB>
          </LOGICAL_CELL>
        </VIEW>
      </CELL>
      <CELL NAME="nand2" >
        <VIEW NAME="layout" >
          <LOGICAL_CELL>
            <LIB NAME="hierarchy" >
              <CELL NAME="nand2" >
                <VIEW NAME="symbol" >
                  <VLGEN>t</VLGEN>
                </VIEW>
              </CELL>
            </LIB>
          </LOGICAL_CELL>
        </VIEW>
      </CELL>
    </LIB>
  </MAPPINGS>
</PHYSICAL>
```

The LAM file above specifies the schematic cells that need to be generated as VPLGens or VLGens.

When generating VPLGens, the following additional parameters can be specified:

```
<VPL>pl;pw;nl;nw</VPL>
```

If the default value is a CDF parameter for the symbol, you can choose not to mention the default value. The specified value for a VPLGen or VLGen parameter can be of type: float, boolean, or a string.

Related Topics

Library and Attributes Mapping File Syntax

LAM File Logical Section

© 2023 Cadence Design Systems, Inc.

LAM File Physical Section

LAM File Component Type Section

LAM File Mapping Section

Sample LAM File

LAM File Support for SDL Pcell

The LAM file supports CAD teams to define a common library that can contain SDL Pcells. In addition, the CAD teams can specify the location of the SDL Pcells that are created in a different library and allow layout designers to override the physical bindings in the design library to select the SDL Pcells defined outside.

Here is a sample representing the LAM file definition for the main library in the scenario above.

```
<!DOCTYPE LAM_FILE>
<LIBRARY_DATA>
    <VERSION NUMBER="0.2" />
    <LOGICAL>
    <LIB NAME="lib">
        <CELL NAME="inv">
        <VIEW NAME="symbol">
            <SDL_PCELL>t</SDL_PCELL>
            <PARAMETERS>
            <SDL>a 1;b 2</SDL>
            </PARAMETERS>
        </VIEW>
        </CELL>
...
    <MAPPINGS>
        <LIB NAME="lib_phys">
        <CELL NAME="inv">
            <VIEW NAME="layout">
              <LOGICAL_CELL>
            <LIB NAME="lib">
              <CELL NAME="inv">
                <VIEW NAME="symbol">
               </VIEW>
              </CELL>
```


© 2023 Cadence Design Systems, Inc.

```
            </LIB>
              </LOGICAL_CELL>
            </VIEW>
          </CELL>
```

The Mapping section of the other LAM file for the above scenario could be represented as below.

```
<MAPPINGS>
        <LIB NAME="mylib">
        <CELL NAME="inv">
            <VIEW NAME="layout">
              <LOGICAL_CELL>
            <LIB NAME="lib">
              <CELL NAME="inv">
                <VIEW NAME="symbol">
               </VIEW>
              </CELL>
            </LIB>
              </LOGICAL_CELL>
            </VIEW>
          </CELL>
```

You can override the SDL_PCELL and its parameters at any level.

Related Topics

Library and Attributes Mapping File Syntax

Basic LAM File Structure

LAM File Physical Section

LAM File Component Type Section

LAM File Mapping Section

Sample LAM File

© 2023 Cadence Design Systems, Inc.

LAM File Physical Section

```
<PHYSICAL>
  <COMPONENT_TYPE_GROUPS>
    <!-- component type definitions -->
  </COMPONENT_TYPE_GROUPS>
  <>
    <!-- logical to physical  -->
  </>
</PHYSICAL>
```

Description

The physical section of the LAM file is divided into two subsections. One of these defines the library-level component types, the other specifies the logical-to-physical mapping for the design.

Statements

Statement	Description
`<COMPONENT_TYPE_GROUPS>`	Encloses the library-level component type definitions.
`<>`	Encloses all the statements used to map physical cellviews, parameters, and terminals to their counterparts in the logical design.

The logical-to-physical mapping of the design is dependent on:

- The order in which the libraries are defined in `cds.lib` because that determines the order in which the corresponding LAM files will be read for appropriate physical binding.
- The availability of the exact matching physical binding in the logical library.
  - If the logical library has a matching physical binding, the physical binding in the logical library gets used to create a logical-to-physical map.
  - If a matching physical binding is not available in the logical library, the libraries defined in `cds.lib` are read in the reverse order until a matching physical binding is found. If the library definition order in `cds.lib` is modified, the order in which the libraries are read will change as well.

© 2023 Cadence Design Systems, Inc.

Related Topics

Library and Attributes Mapping File Syntax

LAM File Component Type Section

LAM File Mapping Section

Basic LAM File Structure

LAM File Logical Section

Sample LAM File

© 2023 Cadence Design Systems, Inc.

LAM File Component Type Section

```
<COMPONENT_TYPE_GROUPS>
  <LIB NAME="libName" >
    <COMPONENT_TYPE_GROUP NAME="compTypeName" >
      <COMP_CLASS>PMOS | NMOS | NFIN | PFIN | STDCELL | STDSUBCONT | BOUNDARY
CELL | FILLER</COMP_CLASS>
      <WIDTH>WIDTH</WIDTH>
      <LENGTH>LENGTH</LENGTH>
      <SPLIT>SPLIT</SPLIT>
      <ORIENTATION>device_orientation</ORIENTATION>
      <DRAIN>list_of_drainTermNames</DRAIN>
      <GATE>list_of_gateTermNames</GATE>
      <SOURCE>list_of_sourceTermNames</SOURCE>
      <BULK>list_of_bulkTermNames</BULK>
      <ACTIVE_LP>layerPurposePairName</ACTIVE_LP>
      <FOLD_MIN>float</FOLD_MIN>
      <FOLD_THRESH>float</FOLD_THRESH>
      <COMPTYPE_PARAMS>list_of_parameters_and_value_pairs</COMPTYPE_PARAMS>
      <FOLD_STEP>float</FOLD_STEP>
      <VIEWS>list_of_viewNames</VIEWS>
      <COMPTYPE_PARAMS>list_of_parameters</COMPTYPE_PARAMS>
      <NUM_FINS>number_of_fins_ParameterName</NUM_FINS>
      <FOLD_THRESH_FINS>int</FOLD_THRESH_FINS>
      <WIDTH_FIN>float</WIDTH_FIN>
    </COMPONENT_TYPE_GROUP>
  </LIB>
</COMPONENT_TYPE_GROUPS>
```

The following attributes in the LAM file above are supported for FinFET devices:

- `<COMP_CLASS>NFIN | PFIN</COMP_CLASS>`
- `<NUM_FINS>`*`number_of_fins_ParameterName`*`</NUM_FINS>`
- `<FOLD_THRESH_FINS>`*`int`*`</FOLD_THRESH_FINS>`
- `<WIDTH_FIN>`*`float`*`</WIDTH_FIN>`

Description

Specifies the library-level component types, which identify *NMOS* and *PMOS* transistor cells and sets the parameters for device chaining and folding.

For FinFET devices, the Component Type section of the LAM file also specifies the library-level component types that identify *NFIN* and *PFIN* FinFET devices and sets the parameters for chaining and folding for these FinFET devices.

The LAM file Component Type section also controls how components are assigned to rows when using the Virtuoso Custom Digital placer. The LAM file component type settings are inherited by all the cells in the specified library. They can be overridden, but not overwritten in the Configure Physical Hierarchy window.

© 2023 Cadence Design Systems, Inc.

By default, CPH only displays used cells for a design. If you want to display in CPH the cell assignments that are made in LAM (although the cells are not being used), do this:

➡ In CPH *Cells* tab, right-click the library for which you made the cell assignments in LAM and choose *Show Cells*.

CPH displays the LAM cell assignments for the library.

The component type settings defined in the LAM file are created automatically during the logical library conversion. This takes the component types stored in CDF files or (if none exist in CDF) in the `data.dm` file in the library directory and converts them to LAM file format.

If the same component type is defined with identical attributes in two different libraries, the definitions are merged into a single definition in the LAM file. All required cells are then assigned to that single component type.

You can also add new component type groups using the syntax described in this section. for more information, see Data Conversion Flow for the Layout XL Schema on page 79.

When using multiple SKILL APIs for working with component types, it is recommended that you checkout a Layout XL license beforehand to enhance the system performance. But if you have a CPH window or an XL session open already, the XL license will already be checked out.

After you have defined the component type group, you must specify the cells that are assigned to this group using the `<COMPONENT_TYPE>` statement. See LAM File Mapping Section for more information.

Statements

Statement	Description
`<COMPONENT_TYPE_GROUPS>`	Encloses the library-level component type definitions in the LAM file.
`<LIB NAME="libName" >`	Specifies the name of the library to which the component type definitions apply.
`<COMP_CLASS>PMOS | NMOS |PFIN | NFIN | RESISTOR | CAPACITOR | INDUCTOR | STDCELL | STDSUBCONT | BOUNDARYCELL | FILLER</COMP_CLASS>`	

© 2023 Cadence Design Systems, Inc.

Statement	Description
	Specifies the type of devices in the component type. Valid values are `PMOS`, `NMOS`, `PFIN`, `NFIN`, `RESISTOR`, `CAPACITOR`, `INDUCTOR`, `STDCELL`, `STDSUBCONT`, `BOUNDARYCELL`, `FILLER`. Leave blank to set to undefined. `PFIN` and `NFIN` are valid component types only for designs using FinFET devices. ■ Choose `PMOS` or `NMOS` if you are using the placer's MOS component-assisted row generation. The `<VIEWS>` statement is ignored. ■ Choose `RESISTOR`, `CAPACITOR`, or `INDUCTOR` to recognize the split parameter on these devices when manually folding them. ■ The information is used by the Layout XL *Check* and *Update* commands to verify if the parameter has been split and if the split has been accurate. ■ Choose `PFIN` or `NFIN` if you are using FinFET devices for generation. ■ Choose `STDCELL` if you are using the placer's Standard Cell component-assisted row generation. All other settings are ignored. ■ Choose `STDSUBCONT` or `FILLER` for substrate contacts and filler cells, respectively. Only the `<VIEWS>` statement is considered; everything else is ignored. ■ Choose `BOUNDARYCELL` for boundary cells. Only the `<VIEWS>` statement is considered; everything else is ignored. **GUI:** *Component class*

© 2023 Cadence Design Systems, Inc.

Statement	Description
`<WIDTH>WIDTH</WIDTH>`	Specifies the name of the transistor width parameter on the device master cell. The default name is `w`. If you use a different name for the transistor width parameter, type that name here. You must set this parameter for `NMOS` and `PMOS` component types even if the devices in the component type will not be folded. **GUI:** *Width parameter*
`<LENGTH>LENGTH</LENGTH>`	Specifies the name of the transistor length parameter on the device master cell. The default name is `l`. If you use a different name for the transistor length parameter, type that name here. **GUI:** *Length parameter*
`<SPLIT>SPLIT</SPLIT>`	Specifies the name of the parameter to split. **GUI:** *Split parameter*
`<ORIENTATION>`*`device_orientation`*`</ORIENTATION>`	For the selected component class, specifies the orientation to be used during layout generation.
`<DRAIN>`*`list_of_drainTermNames`*`</DRAIN>`	Specifies the default names used to identify the *Drain* terminal in the cells assigned to the component type. The names are not case-sensitive. The default names are `drain` and `D`. Separate each name with a single space. Do not enclose the list in quotation marks. **GUI:** *Drain*
`<GATE>`*`list_of_gateTermNames`*`</GATE>`	

© 2023 Cadence Design Systems, Inc.

Statement	Description
	Specifies the default names used to identify the *Gate* terminal in the cells assigned to the component type. The names are not case-sensitive. The default names are `gate` and `G`. Separate each name with a single space. Do not enclose the list in quotation marks. **GUI:** *Gate*
`<SOURCE>`*`list_of_sourceTermNames`*`</SOURCE>`	Specifies the default names used to identify the *Source* terminal in the cells assigned to the component type. The names are not case-sensitive. The default names are `source` and `S`. Separate each name with a single space. Do not enclose the list in quotation marks. **GUI:** *Source*
`<BULK>`*`list_of_bulkTermNames`*`</BULK>`	Specifies the default names used to identify the *Bulk* terminal in the cells assigned to the component type. The names are not case-sensitive. The default is `bulk` and `B`. Separate each name with a single space. Do not enclose the list in quotation marks. **GUI:** *Bulk*
`<ACTIVE_LP>`*`layerPurposePairName`*`</ACTIVE_LP>`	Specifies the diffusion layer-purpose pair for NMOS and PMOS devices. This parameter is required for component types of the component class `NMOS` and `PMOS` to control device chaining and folding during placement. For example, to specify the ndiff drawing layer as the diffusion layer, type the following. `<ACTIVE_LP>ndiff drawing</ACTIVE_LP>` Separate the layer and purpose names with a single space. Do not enclose the names in quotation marks. **GUI:** *Active layer*
`<FOLD_MIN>`*`float`*`</FOLD_MIN>`	

© 2023 Cadence Design Systems, Inc.

Statement	Description
	Specifies the minimum width that can be used when folding transistors with width greater than the folding threshold. **GUI:** *Folding minimum*
`<FOLD_THRESH>`*`float`*`</FOLD_THRESH>`	
	Specifies the maximum width beyond which MOS devices are automatically folded. **Note:** You must set this parameter for `NMOS` and `PMOS` component types even if the devices in the component type will not be folded. **GUI:** *Folding threshold*
`<COMPTYPE_PARAMS>`*`list_of_parameters_and_value_pairs`*`</COMPTYPE_PARAMS>`	
	Specifies the parameters names and values required for the cell to belong to this component type. **GUI:** *Component parameters*
`<FOLD_STEP>`*`float`*`</FOLD_STEP>`	
	Specifies the available width values that lie between the *Folding minimum* and the *Folding threshold v*alues and can be used for folding. **GUI:** *Folding step*
`<VIEWS>`*`list_of_viewNames`*`</VIEWS>`	
	Specifies the view name to be used when defining a standard cell substrate contact or filler cell. This parameter is valid only when the component class is either `STDSUBCONT` or `FILLER`. Separate each name with a single space. Do not enclose the list in quotation marks. **GUI:** *Views*

© 2023 Cadence Design Systems, Inc.

Statement	Description
`<NUM_FINS>`*`number_of_fins_parameterName`*`</NUM_FINS>`	Specifies the name of the *Number of fins* parameter on the FinFET device master cell. `NUM_FINS` must be set for FinFET devices that are driven for folding by number of fins. **GUI:** *Number of fins*
`<WIDTH_FIN>`*`float`*`</WIDTH_FIN>`	Specifies the width per fin value such that the folding threshold for automatic folding is set using the product of the width per fin value and the folding threshold as a number of fins. Set this parameter to fold FinFET devices driven by setting the *Width parameter*. **GUI:** *Width per fin*
`<FOLD_THRESH_FINS>`*`int`*`</FOLD_THRESH_FINS>`	Specifies the folding threshold in terms of number of fins. `FOLD_THRESH_FINS` must be set to drive automatic folding of FinFET devices. **GUI:** *Folding threshold fins*
`<COMPTYPE_PARAMS>list_of_parameters</COMPTYPE_PARAMS>`	Specifies multiple component parameters for a single cell to allow the same cell to be assigned different component types. To determine the supported component types for a cell, the component parameters are compared to the CDF parameters on the device master cell. The component parameters for a design master cell can only be specified or edited using the LAM file. The value is not editable using the Configure Physical Hierarchy window.

Let us consider some examples depicting the use of the various attributes.

Example 1 *LAM file depicting the use of various component types*

```
<COMPONENT_TYPE_GROUPS>
  <LIB NAME="lib1" >
```


© 2023 Cadence Design Systems, Inc.

```
      <COMPONENT_TYPE_GROUP NAME="nmos" >
        <COMP_CLASS>NMOS</COMP_CLASS>
        <WIDTH>w</WIDTH>
        <ORIENTATION>R90</ORIENTATION>
        <DRAIN>d</DRAIN>
        <GATE>g</GATE>
        <SOURCE>s</SOURCE>
        <BULK>b</BULK>
        <ACTIVE_LP>pdiff drawing</ACTIVE_LP>
        <FOLD_MIN>0.3</FOLD_MIN>
        <FOLD_THRESH>0.9</FOLD_THRESH>
        <FOLD_STEP>0.3</FOLD_STEP>
        <VIEWS>layout filler</VIEWS>
      </COMPONENT_TYPE_GROUP>
      <COMPONENT_TYPE_GROUP NAME="pmos" >
        <COMP_CLASS>PMOS</COMP_CLASS>
        <WIDTH>w</WIDTH>
        <DRAIN>d</DRAIN>
        <GATE>g</GATE>
        <SOURCE>s</SOURCE>
        <BULK>b</BULK>
        <ACTIVE_LP>ndiff drawing</ACTIVE_LP>
        <FOLD_THRESH>0.9</FOLD_THRESH>
        <VIEWS>layout</VIEWS>
      </COMPONENT_TYPE_GROUP>
    </LIB>
    <LIB NAME="lib2" >
      <COMPONENT_TYPE_GROUP NAME="lib2" />
    </LIB>
  </COMPONENT_TYPE_GROUPS>
```

The example above defines three library-level component types.

- One called `nmos` and one called `pmos` for a library called `lib1`.
- One called `lib2` for a library called `lib2`.

The *Folding minimum*, *Folding threshold*, and *Folding step* values set in the LAM file result in the following commands to create or update devices to be *0.3u, 0.6u, or 0.9u wide*.

- *Generate All From Source*
- *Generate Folded Devices*
- *Update Components And Nets*
- *Update Layout Parameters*

Let us now consider some examples depicting the component types section of LAM files for FinFET devices.

Example 1 *LAM File for FinFET devices that have the Number of fins parameter set*

```
<PHYSICAL>
 <COMPONENT_TYPE_GROUPS>
```


© 2023 Cadence Design Systems, Inc.

```
 <LIB NAME="cdn20">
  <COMPONENT_TYPE_GROUP_NAME="nfin">
   <COMP_CLASS>NFIN</COMP_CLASS>
   <NUM_FINS>nFins</NUM_FINS>
  </COMPONENT_TYPE_GROUP>
  <COMPONENT_TYPE_GROUP_NAME="pfin">
   <COMP_CLASS>PFIN</COMP_CLASS>
   <NUM_FINS>pFins</NUM_FINS>
  </COMPONENT_TYPE_GROUP>
 </LIB>
</COMPONENT_TYPE_GROUPS>
<MAPPINGS>
 <LIB NAME="cdn20">
  <CELL NAME="nfinFET">
   <COMPONENT_TYPE>
    <NAME>nfin</NAME>
    <LIB>cdn20</LIB>
   </COMPONENT_TYPE>
  </CELL>
  <CELL NAME="pfinFET">
   <COMPONENT_TYPE>
    <NAME>pfin</NAME>
    <LIB>cdn20</LIB>
   </COMPONENT_TYPE>
  </CELL>
 </LIB>
</MAPPINGS>
</PHYSICAL>
```

The example above defines the components of the class *NFIN* and *PFIN* that have the *Number of fins* parameter set as `nFins` and `pFins`, respectively.

Example 2 *LAM File for FinFET devices that have the Width parameter set*

```
<PHYSICAL>
 <COMPONENT_TYPE_GROUPS>
  <LIB NAME="cdn20">
   <COMPONENT_TYPE_GROUP_NAME="nfin">
    <COMP_CLASS>NFIN</COMP_CLASS>
    <WIDTH>w</WIDTH>
    <WIDTH_FIN>0.1</WIDTH_FIN>
```


© 2023 Cadence Design Systems, Inc.

```
    </COMPONENT_TYPE_GROUP>
    <COMPONENT_TYPE_GROUP_NAME="pfin">
     <COMP_CLASS>PFIN</COMP_CLASS>
     <WIDTH>w</WIDTH>
     <WIDTH_FIN>0.3</WIDTH_FIN>
    </COMPONENT_TYPE_GROUP>
   </LIB>
  </COMPONENT_TYPE_GROUPS>
  <MAPPINGS>
   <LIB NAME="cdn20">
    <CELL NAME="nfinFET">
     <COMPONENT_TYPE>
      <NAME>nfin</NAME>
      <LIB>cdn20</LIB>
     </COMPONENT_TYPE>
    </CELL>
    <CELL NAME="pfinFET">
     <COMPONENT_TYPE>
      <NAME>pfin</NAME>
      <LIB>cdn20</LIB>
     </COMPONENT_TYPE>
    </CELL>
   </LIB>
  </MAPPINGS>
</PHYSICAL>
```

The example above defines the components of the class *NFIN* and *PFIN* that have the *Width* parameter set as `w`.

Example 3 *LAM File for Automatic folding of FinFET devices based on Number of fins*

```
<PHYSICAL>
 <COMPONENT_TYPE_GROUPS>
  <LIB NAME="cdn20">
   <COMPONENT_TYPE_GROUP_NAME="nfin">
    <COMP_CLASS>NFIN</COMP_CLASS>
    <NUM_FINS>nFins</NUM_FINS>
    <FOLD_THRESH_FINS>2</FOLD_THRESH_FINS>
   </COMPONENT_TYPE_GROUP>
   <COMPONENT_TYPE_GROUP_NAME="pfin">
```


© 2023 Cadence Design Systems, Inc.

```
    <COMP_CLASS>PFIN</COMP_CLASS>
    <NUM_FINS>pFins</NUM_FINS>
    <FOLD_THRESH_FINS>3</FOLD_THRESH_FINS>
   </COMPONENT_TYPE_GROUP>
  </LIB>
 </COMPONENT_TYPE_GROUPS>
 <MAPPINGS>
  <LIB NAME="cdn20">
   <CELL NAME="nfinFET">
    <COMPONENT_TYPE>
     <NAME>nfin</NAME>
     <LIB>cdn20</LIB>
    </COMPONENT_TYPE>
   </CELL>
   <CELL NAME="pfinFET">
    <COMPONENT_TYPE>
     <NAME>pfin</NAME>
     <LIB>cdn20</LIB>
    </COMPONENT_TYPE>
   </CELL>
  </LIB>
 </MAPPINGS>
</PHYSICAL>
```

The example above indicates that the FinFET devices have the number of fins parameter set. As a result, the devices will automatically fold based on the fins per fold when the folding threshold defined by the `FOLD_THRESH_FINS` is reached.

Example 4 *LAM File for Automatic folding of FinFET devices based on fin width*

```
<PHYSICAL>
 <COMPONENT_TYPE_GROUPS>
  <LIB NAME="cdn20">
   <COMPONENT_TYPE_GROUP_NAME="nfin">
    <COMP_CLASS>NFIN</COMP_CLASS>
    <WIDTH>w</WIDTH>
    <WIDTH_FIN>0.1</WIDTH_FIN>
    <FOLD_THRESH_FINS>2</FOLD_THRESH_FINS>
   </COMPONENT_TYPE_GROUP>
   <COMPONENT_TYPE_GROUP_NAME="pfin">
    <COMP_CLASS>PFIN</COMP_CLASS>
```


© 2023 Cadence Design Systems, Inc.

```
    <WIDTH>w</WIDTH>
    <WIDTH_FIN>0.3</WIDTH_FIN>
    <FOLD_THRESH_FINS>3</FOLD_THRESH_FINS>
   </COMPONENT_TYPE_GROUP>
  </LIB>
 </COMPONENT_TYPE_GROUPS>
 <MAPPINGS>
  <LIB NAME="cdn20">
   <CELL NAME="nfinFET">
    <COMPONENT_TYPE>
     <NAME>nfin</NAME>
     <LIB>cdn20</LIB>
    </COMPONENT_TYPE>
   </CELL>
   <CELL NAME="pfinFET">
    <COMPONENT_TYPE>
     <NAME>pfin</NAME>
     <LIB>cdn20</LIB>
    </COMPONENT_TYPE>
   </CELL>
  </LIB>
 </MAPPINGS>
</PHYSICAL>
```

The example above indicates that the FinFET devices have the width parameter set. As a result, the devices will automatically fold when the folding threshold defined by the `FOLD_THRESH_FINS` is reached such that each fold gets a whole number of fins.

Example 5 *LAM File depicting the use of the* `COMPTYPE_PARAMS` *attribute*

```
<!DOCTYPE LAM_FILE>
<LIBRARY_DATA>
<VERSION NUMBER="0.3" />
<PHYSICAL>
<COMPONENT_TYPE_GROUPS>
<LIB NAME="cdn20FFBase" >
<COMPONENT_TYPE_GROUP NAME="NFIN" >
<COMP_CLASS>NFIN</COMP_CLASS>
<NUM_FINS>nFins</NUM_FINS>
<FOLD_THRESH_FINS>2</FOLD_THRESH_FINS>
<COMPTYPE_PARAMS>list(list("rSADPShoulder" "auto") list("lSADPShoulder"
"auto"))</COMPTYPE_PARAMS>
```


© 2023 Cadence Design Systems, Inc.

```
      </COMPONENT_TYPE_GROUP>
      <COMPONENT_TYPE_GROUP NAME="NFIN2" >
        <COMP_CLASS>NFIN</COMP_CLASS>
        <NUM_FINS>nFins</NUM_FINS>
       <FOLD_THRESH_FINS>2</FOLD_THRESH_FINS>
         <COMPTYPE_PARAMS>list(list("rSADPShoulder" "au*") list("lSADPShoulder"
"*to"))</COMPTYPE_PARAMS>
      </COMPONENT_TYPE_GROUP>
      <COMPONENT_TYPE_GROUP NAME="NFIN3" >
        <COMP_CLASS>NFIN</COMP_CLASS>
        <NUM_FINS>nFins</NUM_FINS>
       <FOLD_THRESH_FINS>2</FOLD_THRESH_FINS>
           <COMPTYPE_PARAMS>list(list("rSADPShoulder" "*au*")
list("lSADPShoulder" "*ut*"))</COMPTYPE_PARAMS>
      </COMPONENT_TYPE_GROUP>
      <COMPONENT_TYPE_GROUP NAME="PFIN" >
           <COMP_CLASS>PFIN</COMP_CLASS>
           <WIDTH>nFins</WIDTH>
           <WIDTH_FIN>3e-06</WIDTH_FIN>
           <FOLD_THRESH_FINS>3</FOLD_THRESH_FINS>
         </COMPONENT_TYPE_GROUP>
       </LIB>
     </COMPONENT_TYPE_GROUPS>
     <MAPPINGS>
       <LIB NAME="cdn20FFBase" >
         <CELL NAME="nfinFET" >
           <COMPONENT_TYPE>
             <NAME>NFIN</NAME>
             <LIB>cdn20FFBase</LIB>
           </COMPONENT_TYPE>
         </CELL>
         <CELL NAME="nfinFET" >
           <COMPONENT_TYPE>
             <NAME>NFIN2</NAME>
             <LIB>cdn20FFBase</LIB>
           </COMPONENT_TYPE>
         </CELL>
         <CELL NAME="nfinFET" >
           <COMPONENT_TYPE>
             <NAME>NFIN3</NAME>
             <LIB>cdn20FFBase</LIB>
```


© 2023 Cadence Design Systems, Inc.

```
          </COMPONENT_TYPE>
        </CELL>
        <CELL NAME="pfinFET" >
          <COMPONENT_TYPE>
            <NAME>PFIN</NAME>
            <LIB>cdn20FFBase</LIB>
          </COMPONENT_TYPE>
        </CELL>
      </LIB>
    </MAPPINGS>
  </PHYSICAL>
</LIBRARY_DATA>
```

The LAM file defines multiple component types with different parameters and assigns the same cell with multiple component types. This allows creating different component types based on different cell properties for the same cell.

Example 6 *LAM File depicting the use of the* `COMPTYPE_PARAMS` *attribute, defining some component types and assigning cells.*

```
<!DOCTYPE LAM_FILE>
<LIBRARY_DATA>
  <VERSION NUMBER="0.3" />
  <PHYSICAL>
    <COMPONENT_TYPE_GROUPS>
      <LIB NAME="testLib" >
<COMPONENT_TYPE_GROUP NAME="NDevice1" >
<COMP_CLASS>NFIN</COMP_CLASS>
<NUM_FINS>nFins</NUM_FINS>
<FOLD_THRESH_FINS>2</FOLD_THRESH_FINS>
<COMPTYPE_PARAMS>list(list("param1" value1) list("param2" value2))</
COMPTYPE_PARAMS>
</COMPONENT_TYPE_GROUP>
<COMPONENT_TYPE_GROUP NAME="PFIN" >
<COMP_CLASS>PFIN</COMP_CLASS>
<NUM_FINS>nFins</NUM_FINS>
<FOLD_THRESH_FINS>3</FOLD_THRESH_FINS>
<COMPTYPE_PARAMS>list(list("param1" value1) list("param2" value2)
list("param3" value3)</COMPTYPE_PARAMS>
</COMPONENT_TYPE_GROUP>
      </LIB>
    </COMPONENT_TYPE_GROUPS>
    <MAPPINGS>
```


© 2023 Cadence Design Systems, Inc.

```
      <LIB NAME="testLib" >
<CELL NAME="nfinCell1" >
<COMPONENT_TYPE>
<NAME>NDevice1</NAME>
<LIB>testLib</LIB>
</COMPONENT_TYPE>
</CELL>
<CELL NAME="pfinCell1" >
<COMPONENT_TYPE>
<NAME>PDevice1</NAME>
<LIB>testLib</LIB>
</COMPONENT_TYPE>
</CELL>
</LIB>
    </MAPPINGS>
  </PHYSICAL>
</LIBRARY_DATA>
```

Related Topics

Library and Attributes Mapping File Syntax

LAM File Physical Section

LAM File Mapping Section

Basic LAM File Structure

LAM File Logical Section

Sample LAM File

© 2023 Cadence Design Systems, Inc.

LAM File Mapping Section

```
<>
  <LIB NAME="libName" >
    <CELL NAME="cellName" >
      <COMPONENT_TYPE>
        <NAME>compTypName</NAME>
        <LIB>libName</LIB>
      </COMPONENT_TYPE>
      <VIEW NAME="viewName" >
        <LOGICAL_CELL>
          <LIB NAME="libName" >
            <CELL NAME="cellName" >
              <VIEW NAME="viewName" >
                <ROUNDING>roundingExpr</ROUNDING>
                <PARAMETERS>
                  <MAP>p1 p2 ; p3 p4 ; </MAP>
                  <INCLUDE>
                    <CHECK>l w fw</CHECK>
                  </INCLUDE>
                  <IGNORE>
                    <GEN>list_of_paramNames</GEN>
                    <CHECK>list_of_paramNames</CHECK>
                  </IGNORE>
                </PARAMETERS>
                <TERMINALS>
                  <MAP>t1 t2 ; t3 t4 ; </MAP>
                  <IGNORE>
                    <GEN>list_of_termNames</GEN>
                    <CHECK>list_of_termNames</CHECK>
                  </IGNORE>
                </TERMINALS>
              </VIEW>
            </CELL>
          </LIB>
        </LOGICAL_CELL>
      </VIEW>
    </CELL>
  </LIB>
</>
```

Description

Maps physical cellviews and parameter and terminal names to their counterparts in the logical design, specifies which parameters and terminal names are to be ignored for generation and for check, and overrides the component type setting for a specific physical cellview.

© 2023 Cadence Design Systems, Inc.

Statements

Statement	Description
`<>`	Encloses all the statements used to map physical cellviews, parameters, and terminals to their counterparts in the logical design.
`<LIB NAME="libName" >`	Specifies the physical library name and encloses all the statements relating to that library. The name must be enclosed in quotation marks.
`<CELL NAME="cellName" >`	Specifies the physical cell name and encloses all the statements relating to that cell. The name must be enclosed in quotation marks.

`<COMPONENT_TYPE>`	Encloses the component type group to which the current cell is assigned.
	For example, the syntax below assigns the current cell to component type `myPmos` defined for `lib1`.
	`<COMPONENT_TYPE>`
	`  <NAME>myPmos</NAME>`
	`  <LIB>lib1</LIB>`
	`</COMPONENT_TYPE>`

`<VIEW NAME="viewName" >`	Specifies the physical view name and encloses all the statements relating to that cellview. The name must be enclosed in quotation marks.

© 2023 Cadence Design Systems, Inc.

Statement	Description

<LOGICAL_CELL>	Specifies the logical cellview that corresponds to the current physical cellview. For example,
	`<LOGICAL_CELL>`
	`  <LIB NAME="lib1">`
	`  <CELL NAME="cell1">`
	`  <VIEW NAME="schematic">`
	`  .`
	`  .`
	`  .`
	`  </VIEW>`
	`  </CELL>`
	`  </LIB>`
	`</LOGICAL_CELL>`

<ROUNDING>*expression*</ROUNDING>	Defines how parameter values are rounded when evaluated by Layout XL; for example, `<SROUNDING>(w 0.05 round)</ROUNDING>` Rounds the value of `w` to the nearest multiple of 0.05 nanometers.
<PARAMETERS>	Encloses the <IGNORE>> statements that specify which logical instance parameters are to be ignored for generation and check. **GUI:** Options on the *Parameters* tab
<TERMINALS>	Encloses the <IGNORE> statements that specify which logical instance terminals are to be ignored for generation and check. **GUI:** Options on the *Terminals* tab

© 2023 Cadence Design Systems, Inc.

Statement	Description
`<MAP>`*`p1 p2 ; p3 p4 ;`* `</MAP>`	Maps a logical parameter (or terminal) name to the corresponding parameter (or terminal) in the physical view. For example, to map logical parameters `D` and `G` to physical parameters `d` and `g`, type the following. `<MAP>D d ; G g ; </MAP>` Do not enclose the mapping in quotation marks. **GUI:** *Name mapping* (cell attributes only)
`<INCLUDE>`	Encloses the `<CHECK>` statements that specify which logical instance parameters are to be checked.
`<CHECK>`*`list_of_parameter_names`*`</CHECK>`	Lists the names of logical instance parameters that are checked by *Check Against Source*. Separate each name with a single whitespace. Do not enclose the list in quotation marks. **GUI:** *To check*
`<IGNORE>`	Encloses the `<GEN>` and `<CHECK>` statements that specify which logical instance parameters (or terminals) are to be ignored for generation or check.
`<GEN>`*`list_of_names`*`</GEN>`	Lists the names of logical instance parameters (or terminals) that are ignored by the following commands; *Generate All From Source*, *Generate Selected From Source*, *Generate Clones*, *Check Against Source*, *Update Components And Nets*, *Update Layout Parameters*, *Update Schematic Parameters*. Separate each name with a single whitespace. Do not enclose the list in quotation marks. **GUI:** *Ignore for generation*

© 2023 Cadence Design Systems, Inc.

Statement	Description
<CHECK>*list_of_names*</CHECK>	
	Lists the names of logical instance parameters (or terminals) that are ignored by the *Check Against Source* command. Mismatches for any of the listed parameters are not reported by the check. Separate each name with a single whitespace. Do not enclose the list in quotation marks. The list inherits all the parameter names from the *Ignore for generation* list. **GUI:** *Ignore for check*

Example

```
<>
  <LIB NAME="lib1" >
    <CELL NAME="cellB" >
      <VIEW NAME="layout" >
        <LOGICAL_CELL>
          <LIB NAME="lib2" >
            <CELL NAME="cellF" >
              <VIEW NAME="symbol" />
            </CELL>
          </LIB>
        </LOGICAL_CELL>
      </VIEW>
    </CELL>
  </LIB>
  <LIB NAME="lib2" >
    <CELL NAME="cellE" >
      <COMPONENT_TYPE>
        <NAME>myNmos</NAME>
        <LIB>lib2</LIB>
      </COMPONENT_TYPE>
      <VIEW NAME="layout" >
        <LOGICAL_CELL>
          <LIB NAME="lib1" >
            <CELL NAME="cellA" >
              <VIEW NAME="schematicB" >
                <ROUNDING>(w 0.01 round)</ROUNDING>
                <PARAMETERS>
                  <MAP>l L ; w W ; x X2 ; </MAP>
                  <IGNORE>
                    <GEN>w l</GEN>
                    <CHECK>a b c</CHECK>
                  </IGNORE>
                </PARAMETERS>
                <TERMINALS>
                  <MAP>B G ; X D ; Y S ; </MAP>
                </TERMINALS>
              </VIEW>
              <VIEW NAME="symbol" >
                <ROUNDING>(w 0.05 round)</ROUNDING>
```


© 2023 Cadence Design Systems, Inc.

```
                <PARAMETERS>
                  <MAP>l L ; w W ; </MAP>
                  <IGNORE>
                    <GEN>x y z</GEN>
                    <CHECK>d e f</CHECK>
                  </IGNORE>
                </PARAMETERS>
                <TERMINALS>
                  <MAP>D d ; G g ; S s ; </MAP>
                  <IGNORE>
                    <GEN>X</GEN>
                  </IGNORE>
                </TERMINALS>
              </VIEW>
            </CELL>
          </LIB>
        </LOGICAL_CELL>
      </VIEW>
    </CELL>
    <CELL NAME="cellG" >
      <VIEW NAME="layout" >
        <LOGICAL_CELL>
          <LIB NAME="lib1" >
            <CELL NAME="cellD" >
              <VIEW NAME="symbol" />
            </CELL>
          </LIB>
        </LOGICAL_CELL>
      </VIEW>
      <VIEW NAME="layoutS" >
        <LOGICAL_CELL>
          <LIB NAME="lib1" >
            <CELL NAME="cellD">
              <VIEW NAME="symbolS" />
            </CELL>
          </LIB>
        </LOGICAL_CELL>
      </VIEW>
    </CELL>
  </LIB>
</>
```

The example above

For cellview	LAM specifies that
`lib1/cellB/layout`	The cellview maps to logical cellview `lib2/cellF/symbol`.
`lib2/cellE`	The cell is assigned to component type `myNmos` defined in library `lib2`.

© 2023 Cadence Design Systems, Inc.

For cellview	LAM specifies that
`lib2/cellE/layout`	The cellview maps to logical cellviews `lib1/cellA/schematicB` and `lib1/cellA/symbol`. When `lib1/cellA/schematicB` is used: ■ The value of the `w` parameter is rounded to the nearest multiple of 0.01 nanometers. ■ Logical instance parameters `l`, `w`, and `x` are mapped to physical parameters `L`, `W`, and `X2` respectively. ■ Logical instance parameters `w` and `l` are not copied during layout generation; parameters `a`, `b`, and `c` are not checked by the *Check Against Source* command ■ Logical terminals `B`, `X`, and `D` are mapped to physical terminals `G`, `D`, and `S` respectively. When `lib1/cellA/symbol` is used: ■ The value of the `w` parameter is rounded to the nearest multiple of 0.05 nanometers. ■ Logical instance parameters `l` and `w` are mapped to physical parameters `L` and `W` respectively. ■ Logical instance parameters `x`, `y`, and `z` are not copied during layout generation; parameters `d`, `e`, and `f` are not checked by the *Check Against Source* command ■ Logical terminals `D`, `G`, and `S` are mapped to physical terminals `d`, `g`, and `s` respectively. ■ Logical terminal `X` is ignored during layout generation.
`lib2/cellG/layout`	The cellview maps to logical cellview `lib1/cellD/symbol`.
`lib2/cellG/layoutS`	The cellview maps to logical cellview `lib1/cellD/symbolS`.

Wildcard Support for Cell-Level Mapping

For cell-level mapping, wildcards, such as "`\*`" can be used for cell names either as a prefix or as a postfix.

© 2023 Cadence Design Systems, Inc.

The wildcards can be used for both physical and logical cell names in the Mapping section of the LAM file. The use of wildcards in cell names makes it simple to update cell names, if required, because multiple cell names can be updated at a time.

Let us consider some examples that demonstrate the use of wildcards in the Mapping section of the LAM file.

Example 1: The example below demonstrates how the wildcards can be used as a postfix when mapping the physical and logical cells.

```
<!DOCTYPE LAM_FILE>
<LIBRARY_DATA>
  <VERSION NUMBER="0.3" />
  <PHYSICAL>
    <MAPPINGS>
      <LIB NAME="acpd" >
        <CELL NAME="spc\*" >
          <VIEW NAME="layout" >
            <LOGICAL_CELL>
              <LIB NAME="acpd" >
                <CELL NAME="spc\*" >
                  <VIEW NAME="symbol" />
                </CELL>
              </LIB>
            </LOGICAL_CELL>
          </VIEW>
        </CELL>
      </LIB>
    </MAPPINGS>
  </PHYSICAL>
</LIBRARY_DATA>
```

You can also use the wildcard as a prefix, such as when specifying the cell name as `<CELL NAME="\*spc">`.

Example 2: The example below demonstrates how filler cells can be defined in the Mapping section of the LAM file so that all the filler cells that have their cell name defined using the wildcard, such as `"FILLER\*"`, are ignored by the binder.

```
<!DOCTYPE LAM_FILE>
<LIBRARY_DATA>
  <VERSION NUMBER="0.3" />
  <PHYSICAL>
    <COMPONENT_TYPE_GROUPS>
```


© 2023 Cadence Design Systems, Inc.

```
      <LIB NAME="testLib090" >
    <COMPONENT_TYPE_GROUP NAME="filler" >
      <COMP_CLASS>FILLER</COMP_CLASS>
      <VIEWS>layout filler</VIEWS>
    </COMPONENT_TYPE_GROUP>
      </LIB>
    </COMPONENT_TYPE_GROUPS>
    <MAPPINGS>
      <LIB NAME="testLib090" >
    <CELL NAME="FILLER\*" >
      <COMPONENT_TYPE>
        <NAME>filler</NAME>
        <LIB>testLib090</LIB>
      </COMPONENT_TYPE>
    </CELL>
      </LIB>
    </MAPPINGS>
  </PHYSICAL>
</LIBRARY_DATA>
```

Example 3: To view the sample LAM file for hierarchically mapping library names and cell names by providing only library names, click here. Viewing the example requires Cadence Online Support access.

Related Topics

Library and Attributes Mapping File Syntax

LAM File Physical Section

LAM File Component Type Section

Basic LAM File Structure

LAM File Logical Section

Sample LAM File

© 2023 Cadence Design Systems, Inc.

Sample LAM File

```
<!DOCTYPE LAM_FILE>
<LIBRARY_DATA>
  <VERSION NUMBER="0.2" />
  <LOGICAL>
    <LIB NAME="basic" >
      <CELL NAME="cellA" >
        <VIEW NAME="symbol" >
          <REMOVE_DEVICE>(short(src dst))</REMOVE_DEVICE>
        </VIEW>
      </CELL>
    </LIB>
    <LIB NAME="lib1" >
      <CELL NAME="cellB" >
        <VIEW NAME="symbol" >
          <MFACTOR_SPLIT>t</MFACTOR_SPLIT>
          <REMOVE_DEVICE>(short(PLUS MINUS) funcR(r))</REMOVE_DEVICE>
          <STOP_LIST>abstract layout layoutS</STOP_LIST>
          <VIEW_LIST>schematic symbol layout compacted</VIEW_LIST>
          <PARAMETERS>
            <IGNORE>
              <GEN>w l</GEN>
              <CHECK>a b</CHECK>
            </IGNORE>
          </PARAMETERS>
        </VIEW>
      </CELL>
      <CELL NAME="cellC" >
        <VIEW NAME="symbol" >
          <MFACTOR_SPLIT>t</MFACTOR_SPLIT>
          <STOP_LIST>abstractX layoutX layoutSX</STOP_LIST>
          <VIEW_LIST>schematicX symbolX layoutX</VIEW_LIST>
          <TERMINALS>
            <IGNORE>
              <GEN>x y z</GEN>
              <CHECK>a b c</CHECK>
            </IGNORE>
          </TERMINALS>
        </VIEW>
      </CELL>
```


© 2023 Cadence Design Systems, Inc.

```
  </LIB>
</LOGICAL>
<PHYSICAL>
  <COMPONENT_TYPE_GROUPS>
    <LIB NAME="lib1" >
      <COMPONENT_TYPE_GROUP NAME="nmos" >
        <COMP_CLASS>NMOS</COMP_CLASS>
        <WIDTH>w</WIDTH>
        <DRAIN>d</DRAIN>
        <GATE>g</GATE>
        <SOURCE>s</SOURCE>
        <BULK>b</BULK>
        <ACTIVE_LP>ndiff drawing</ACTIVE_LP>
        <FOLD_THRESH>0.92</FOLD_THRESH>
        <VIEWS>layout filler</VIEWS>
      </COMPONENT_TYPE_GROUP>
      <COMPONENT_TYPE_GROUP NAME="pmos" >
        <COMP_CLASS>PMOS</COMP_CLASS>
        <WIDTH>w</WIDTH>
        <DRAIN>d</DRAIN>
        <GATE>g</GATE>
        <SOURCE>s</SOURCE>
        <BULK>b</BULK>
        <ACTIVE_LP>pdiff drawing</ACTIVE_LP>
        <FOLD_THRESH>0.9</FOLD_THRESH>
        <VIEWS>layout</VIEWS>
      </COMPONENT_TYPE_GROUP>
    </LIB>
    <LIB NAME="lib2" >
      <COMPONENT_TYPE_GROUP NAME="myNmos" />
    <COMP_CLASS>NMOS</COMP_CLASS>
        <WIDTH>w</WIDTH>
        <DRAIN>d</DRAIN>
        <GATE>g</GATE>
        <SOURCE>s</SOURCE>
        <BULK>b</BULK>
        <ACTIVE_LP>ndiff drawing</ACTIVE_LP>
        <FOLD_THRESH>0.6</FOLD_THRESH>
        <VIEWS>layout filler</VIEWS>
      </COMPONENT_TYPE_GROUP>
    </LIB>
```


© 2023 Cadence Design Systems, Inc.

```
</COMPONENT_TYPE_GROUPS>
<MAPPINGS>
  <LIB NAME="lib1" >
    <CELL NAME="cellB" >
      <VIEW NAME="layout" >
        <LOGICAL_CELL>
          <LIB NAME="lib2" >
            <CELL NAME="cellF" >
              <VIEW NAME="symbol" />
            </CELL>
          </LIB>
        </LOGICAL_CELL>
      </VIEW>
    </CELL>
  </LIB>
  <LIB NAME="lib2" >
    <CELL NAME="cellE" >
      <COMPONENT_TYPE>
        <NAME>myNmos</NAME>
        <LIB>lib2</LIB>
      </COMPONENT_TYPE>
      <VIEW NAME="layout" >
        <LOGICAL_CELL>
          <LIB NAME="lib1" >
            <CELL NAME="cellA" >
              <VIEW NAME="schematicB" >
                <ROUNDING>(w 0.01 round)</ROUNDING>
                <PARAMETERS>
                  <MAP>l L ; w W ; x X2 ; </MAP>
                  <IGNORE>
                    <GEN>w l</GEN>
                    <CHECK>a b c</CHECK>
                  </IGNORE>
                </PARAMETERS>
                <TERMINALS>
                  <MAP>B G ; X D ; Y S ; </MAP>
                </TERMINALS>
              </VIEW>
              <VIEW NAME="symbol" >
                <ROUNDING>(w 0.05 round)</ROUNDING>
                <PARAMETERS>
```


© 2023 Cadence Design Systems, Inc.

```
                    <MAP>l L ; w W ; </MAP>
                    <IGNORE>
                      <GEN>x y z</GEN>
                      <CHECK>d e f</CHECK>
                    </IGNORE>
                  </PARAMETERS>
                  <TERMINALS>
                    <MAP>D d ; G g ; S s ; </MAP>
                    <IGNORE>
                      <GEN>X</GEN>
                    </IGNORE>
                  </TERMINALS>
                </VIEW>
              </CELL>
            </LIB>
          </LOGICAL_CELL>
        </VIEW>
      </CELL>
      <CELL NAME="cellG" >
        <VIEW NAME="layout" >
          <LOGICAL_CELL>
            <LIB NAME="lib1" >
              <CELL NAME="cellD" >
                <VIEW NAME="symbol" />
              </CELL>
            </LIB>
          </LOGICAL_CELL>
        </VIEW>
        <VIEW NAME="layoutS" >
          <LOGICAL_CELL>
            <LIB NAME="lib1" >
              <CELL NAME="cellD">
                <VIEW NAME="symbolS" />
              </CELL>
            </LIB>
          </LOGICAL_CELL>
        </VIEW>
      </CELL>
    </LIB>
  </MAPPINGS>
</PHYSICAL>
```


© 2023 Cadence Design Systems, Inc.

```
</LIBRARY_DATA>
```

Related Topics

Library and Attributes Mapping File Syntax

Basic LAM File Structure

LAM File Logical Section

LAM File Physical Section

LAM File Component Type Section

LAM File Mapping Section

© 2023 Cadence Design Systems, Inc.

© 2023 Cadence Design Systems, Inc.

G

Layout XL Assistants

This section describes the dockable assistants available in the Virtuoso® Layout Suite XL layout editor (Layout XL).

Some of the assistants described in this section are inherited from the Virtuoso Studio Design Environment and are not described in full in this section. Instead, links are provided to the relevant documentation.

- Annotation Browser Assistant - Lets you view and manage the violation markers generated for the current design by different Virtuoso applications.
- Navigator - Lets you view objects across the design hierarchy using a tree representation.
- Schematic Assistant - Shows you the current connectivity reference as displayed in the Virtuoso Schematic Editor.

Related Topics

Controlling Assistant Panes

© 2023 Cadence Design Systems, Inc.

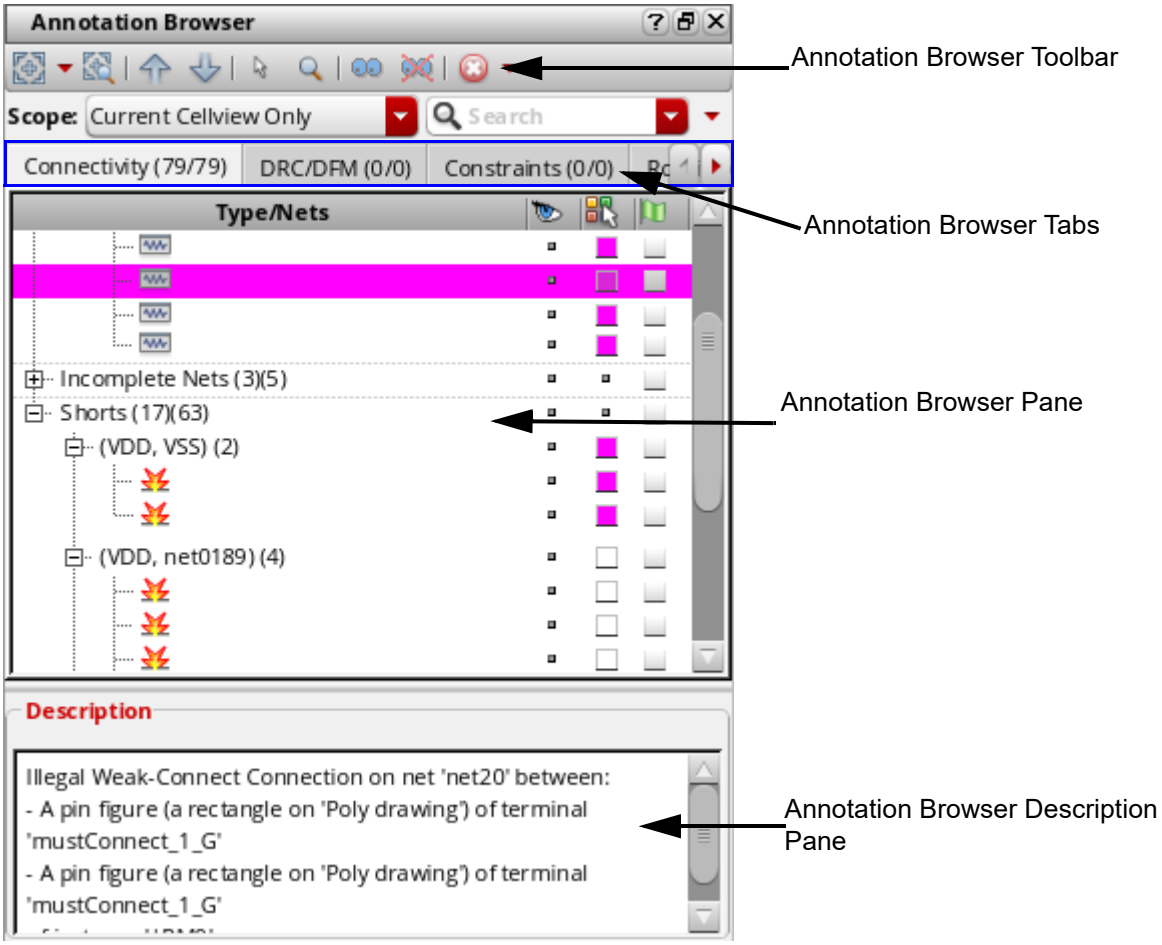
Annotation Browser Assistant

Use the **Annotation Browser** to view and manage the violation markers generated for the current design by different Virtuoso applications.

The Annotation Browser displays the violation markers due to the current environment of the design window. If you modify the environment by editing a cellview in place or descending into a hierarchical cellview, the Annotation Browser gets updated accordingly to reflect a different set of violation markers.

In addition to the generic methods for showing or hiding an assistant, you can use the Layout XL toolbar button,  , to show and hide the Annotation Browser.

The Annotation Browser graphical user interface consists of the following elements.



© 2023 Cadence Design Systems, Inc.

Annotation Browser Toolbar	Provides functions to filter and manage the violation markers displayed in the browser pane, and to define how the markers are displayed in the design window.
Annotation Browser Tabs	Contains multiple tabs each displaying a unique set of violation markers.
Annotation Browser Pane	Displays the markers corresponding to each application category supported by a tab.
Annotation Browser Description Pane	Displays the message stored on the current marker.

Related Topics

Annotation Browser Shortcut Menus

Annotation Browser Options Form

Markers Generated by Connectivity Extractor

© 2023 Cadence Design Systems, Inc.

Annotation Browser Toolbar

The toolbar provides functions to filter and manage the violation markers displayed in the browser pane, and to define how the markers are displayed in the design window.

Icon	Name	Description

	*Auto Zoom*	Zooms and pans to the currently selected markers when using the Up and Down arrows in the toolbar, or to the merged bounding box of all the selected nodes when a selection is changed in the browser.
		
		For example, if you select a node in the tree; the tool zooms to the merged bounding box of all the markers under that node. So, in the example below
		
		`SRC`
		`  Incomplete Nets`
		`    m1`
		`    m2`
		`  Shorts`
		`    m3`
		`    m4`
		
		If you select `Incomplete Nets`, the bounding boxes of `m1` and `m2` are merged and the tool zooms in on the resultant box. If you select `SRC`, the bounding boxes of `m1`, `m2`, `m3`, and `m4` are merged and the tool zooms on this box.
		
		Environment variable: autoZoomEnabled

	Auto zoom mode and scale	Sets *Fixed* or *Minimal* scale for auto zoom. ■ In Fixed mode, the ratio is applied to the bounding box of all the selected markers. So, if set to 4, the merged bounding box of all the selected markers occupies 1/4 of the canvas area after the *Auto Zoom* is performed. ■ In Minimal mode, the Annotation Scale defines a minimum zoom level that will ensure all the selected nodes are visible in the canvas. For example, assume the *Annotation Scale* is set to 4.

© 2023 Cadence Design Systems, Inc.

Icon	Name	Description
		❑ If the merged bounding box of the selected markers is greater than the canvas size, the display zooms out so that the bounding box fits in the canvas. ❑ If the canvas size is greater than the merged bounding box of the selected nodes, which in turn is greater than 1/4 of the canvas size, the display pans to the bounding box of the selected markers. ❑ If the merged bounding box of the selected markers is less than1/4 of the canvas size, the display zooms in on the bounding box so that it occupies 1/4 of the canvas size.
	Annotation Scale slider	Defines a ratio specifying how much of the available canvas area is occupied by the merged bounding box of the selected markers after the zoom is performed. See autoZoomIsFixed. For example, if you set it to 4, the ratio is 1/4; if you set it to 10, the ratio is 1/10, and so on.
	*Zoom To Selected*	Zooms to the selected set of markers in the design window. For markers in the hierarchy, the *Zoom To Selected* functionality highlights the markers of the first instance that satisfies the specified Annotation Browser depth and the display start- and stop-level criteria. If a marker is present in the Annotation Browser tree but cannot be visualized because no instance containing those markers can be highlighted on canvas, then a "not allowed" icon () is displayed against its *Highlight State* column.

© 2023 Cadence Design Systems, Inc.

Icon	Name	Description
	*Go To The Previous Selected Marker* *Go To The Previous Row* *Go To The Last Row*	The behavior (and label) of this button changes depending on what is selected in the browser. ■ For multiple selected markers, it makes the previous marker in the selected set the current marker. This works even if the markers were selected in the canvas and their parent category is not expandable in the browser pane. ■ For an individual selected marker or marker category, it deselects the currently selected entry and selects the entry immediately above it in the list. ■ For multiple selected marker categories, it deselects all the currently selected entries and selects the entry immediately above the original selected set. ■ If there is nothing selected in the browser pane, it selects the last entry in the list. You can use the Up arrow on your keyboard to perform the last three functions listed above. Only the toolbar button performs the first function.

© 2023 Cadence Design Systems, Inc.

Icon	Name	Description
	*Go To The Next Selected Marker* *Go To The Next Row* *Go To The First Row*	The behavior (and label) of this button changes depending on what is selected in the browser. ■ For multiple selected markers, it makes the next marker in the selected set the current marker. This works even if the markers were selected in the canvas and their parent category is not expandable in the browser pane. ■ For an individual selected marker or marker category, it deselects the currently selected entry and selects the entry immediately below it in the list. ■ For multiple selected marker categories, it deselects all the currently selected entries and selects the entry immediately below the original selected set. ■ If there is nothing selected in the browser pane, it selects the first entry in the list. You also can use the Down arrow on your keyboard to perform the last three functions listed above. Only the toolbar button performs the first function.
	*Make Highlighted Markers Selectable*	Specifies that markers that are highlighted in the canvas can be selected even if marker selection has been switched off in the Palette assistant. This option is off by default, meaning that you cannot select highlighted markers in the canvas. Environment variable: highlightedIsSelectable

© 2023 Cadence Design Systems, Inc.

Icon	Name	Description
	*Find Markers From Canvas*	Selects a marker in the Annotation Browser, if the corresponding marker object in the layout canvas is clicked. Even when the marker object is clicked in the layout canvas, the selection in the canvas does not change. See the `abHiFindMarker` SKILL function documentation for more information. For markers that do not display in the Annotation Browser due to the applied filtering, nothing is selected in the Annotation Browser.
	*Show Checked Markers*	Shows markers that are marked as *Checked* in the Annotation Browser. This command has no effect on the marker display in the canvas, it only impacts the display in the Annotation Browser.
	*Hide Checked Markers*	Hides markers that are marked as *Checked* in the Annotation Browser. ■ This command has no effect on the marker display in the canvas, it only impacts the display in the Annotation Browser. ■ During interactive commands, such as *Move*, existing markers, if any, are deleted and new markers created. Therefore, if the markers were previously set as “checked” and “hidden”, new markers may appear after the *Move*.
	*Delete Selected Markers*	Deletes the selected markers from the Annotation Browser and design window.
	*Delete All Unfiltered Markers*	Deletes the markers from the currently selected tab in the Annotation Browser that are not impacted by any of the filter settings currently applied.
	*Delete All Markers*	Deletes all the markers from the currently selected tab in the Annotation Browser.

© 2023 Cadence Design Systems, Inc.

Icon	Name	Description
	Scope	Specifies the scope of the violations displayed. Choose from ■ *Top cellview only* ■ *Current cellview only* ■ *Current Cellview To Depth(3)* Hierarchy from the current cellview to a depth set in the Navigator. The number in the bracket displays the depth specified in the Annotation Browser Options form. The default is `3`. ■ *Top Cellview To Depth(3)* Hierarchy from the top cellview to a depth set in the Navigator. The number in the bracket displays the depth specified in the Annotation Browser Options form. The default is `3`. All markers that do not belong to the current cellview are dimmed. For example, when you descend into a hierarchical cellview, the markers not at the current level are dimmed. You can still delete or mark these as checked, but you cannot highlight them in the canvas. When you return to the higher level, the browser restores the highlighting that was set for that level prior to the descend. a.) With the scope set to “current cellview to depth” or “top to depth”, if a marker reported in the Annotation Browser does not belong to the visible hierarchy—window cellview to edit cellview hierarchy—the marker displays an invalid icon ( ) in the Visibility column to indicate that the visibility of the marker cannot be controlled.

© 2023 Cadence Design Systems, Inc.

Icon	Name	Description
		b.) If the edit cellview in the window is the top cellview and the scope in the Annotation Browser is set to “Top cellview only”, the update commands corresponding to the markers displayed in the CAS tab, such as *Bind*, *Generate*, *Update Pin Direction*, and *Update All Bus Terminals* are available in the shortcut menu. You can choose an appropriate command to interactively fix the marker.
	Search	Filters the results in the browser pane based on a user-specified string. When you type a string in the field, the browser updates automatically to show only those markers that feature the specified string in all the displayed columns and in the *Description* column, even if the *Description* column is not displayed. You can use the alwaysGenerateProps environment variable to control whether the *Description* column is always searched or not. The browser pane displays the markers based on the Search string. For example, if you type in an instance or net name, the Annotation Browser searches the displayed columns (and the *Description* column, if alwaysGenerateProps is set to `t)` to show only the markers associated with that particular instance or net. Previous filters used in the current session are accessible from the drop-down list to the right of the entry field.

© 2023 Cadence Design Systems, Inc.

Icon	Name	Description
		There is also a further drop-down menu which you can use to specify the criteria to be used for the search. ■ Find With: *All Of The Words*, *Any Of The Words*, *The Exact Phrase*, or *None Of The Words* ■ Match Word: *Substring*, *Prefix*, *Exactly*, or *Suffix* ■ Using Case: *Insensitive Match* or *Sensitive Match* The Virtuoso Search auto complete functionality is not supported in the Annotation Browser.

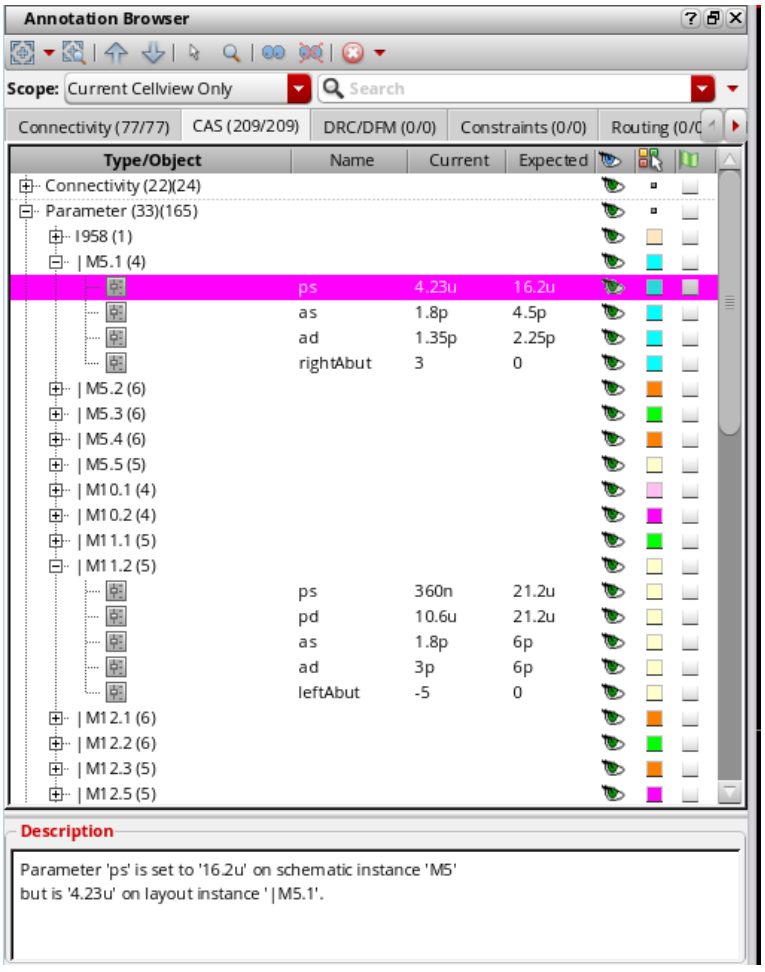
© 2023 Cadence Design Systems, Inc.

Annotation Browser Tabs

The Annotation Browser contains the following tabs, each of which, displays a unique set of violation markers. The *Connectivity* tab, for example, displays markers that indicate connectivity violations in the design.

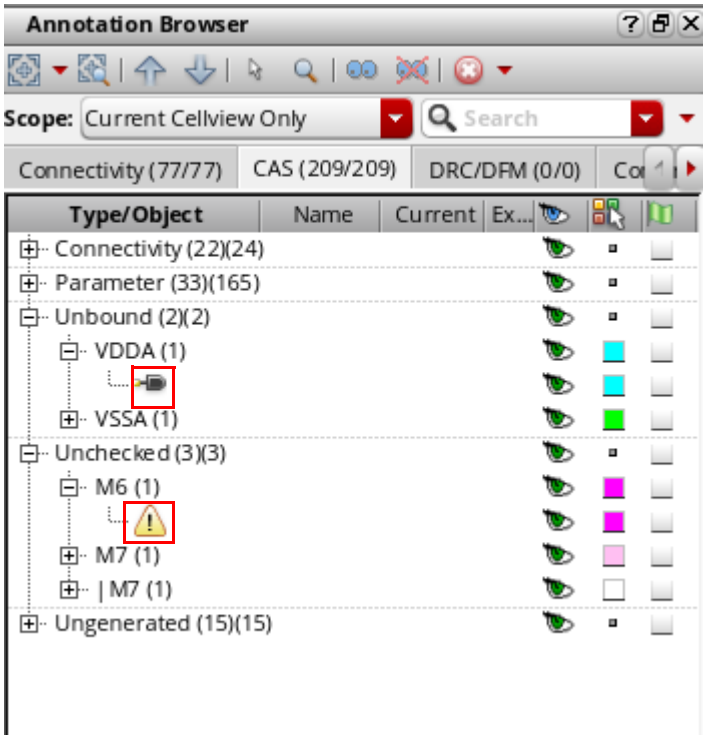
Tab	Description
Connectivity	The **Connectivity tab** carries the connectivity extractor markers that indicate connectivity violations in the design. The markers supported by this tab are: ■ *Opens* ■ *Shorts* ■ *Weak Connect Violations* ■ *Must Connect Violations* ■ *Illegal Overlaps* ■ *Invalid Overlaps* ■ *Unnecessary Pseudo-Parallel Violations* The *Open* markers created by the connectivity extractor between unassigned must-join instance terminals are displayed using a name that reflects the internal net name and the instance name concatenated with "|". You can choose to display the listed violations in the tab based on signal type or the net for which these violations are reported.
DRC/DFM	The **DRC/DFM tab** carries violation markers generated during the Match and Fix Flow. In addition, the tab carries integrated PVS, Width Spacing Pattern, and DRD (`drdEdit`) markers.

© 2023 Cadence Design Systems, Inc.

Tab	Description
CAS	The **CAS tab** is automatically displayed to report the markers generated by the *Connectivity – Check – Against Source (CAS)* command. The markers in the CAS tab report the differences between the schematic and the layout in terms of instances, masters, connectivity, parameters, and instance or net names, depending on the checks that are selected on the Check Against Source Form. The figure below displays the Annotation Browser CAS tab reporting various types of markers. Where applicable, the CAS tab also displays the *Current* and *Expected* values against each marker. The *Description* field at the bottom of the CAS tab, similar to the other tabs, provides more information about the selected marker and hints at how the marker can be resolved. 

© 2023 Cadence Design Systems, Inc.

Tab	Description
	The marker of each type is identified using a unique icon, as displayed in the figure below.



Having the CAS markers display in the Annotation Browser assistant makes the CAS makers available in the format that is already being used to view the other marker types in the design, such as those generated by the connectivity extractor. This also makes it simple to visualize, select, sort, filter, and group the schematic and layout differences.

© 2023 Cadence Design Systems, Inc.

Tab	Description
Constraints	Carries the markers generated by various Virtuoso tools that verify the design for Placement, Specialty Routing, and Design and Process Rules. The supported tools conducting these checks are: ■ DiffpairCheck ■ Symmetry Check ■ checkRoutabilityTool ■ VLS Constraint Aware Editing For a marker to be supported by the *Constraints* tab, it must be associated with a tool listed above. Else, the marker is automatically placed in the *Misc* tab, unless it corresponds to an application supported by the *Connectivity* or *DRC/DFM* tabs.
Misc	The Misc tab of the Annotation Browser displays all those markers that are not associated with any of the applications supported by other tabs. The Misc tab also lists violation markers for any process rule violations in the design and any room violations generated during placement.
IDF	The **IDF tab** carries violation markers generated using the DRC/DFM fixing flow.
Routing	The **Routing tab** carries violation markers generated during design verification. For more information, see Routability Check in the *Virtuoso Space-based Router User Guide*.

The markers supported by each tab are displayed in the Annotation Browser Pane. Each tab name in the Annotation Browser includes a value within parentheses. This value indicates the number of markers currently shown in the browser pane as opposed to the total number of markers in the tab.

For example, *Connectivity (23/120)* means that the design contains 120 markers generated by the connectivity extractor, of which only 23 are being displayed in the browser pane.

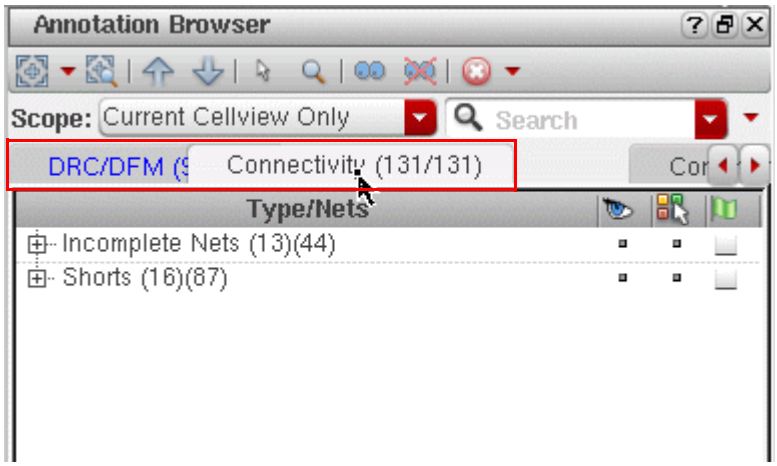
Depending on your preferences, you can change the position of the tabs in the Annotation Browser. Although, by default, the tabs are displayed horizontally at the top of the Annotation

© 2023 Cadence Design Systems, Inc.

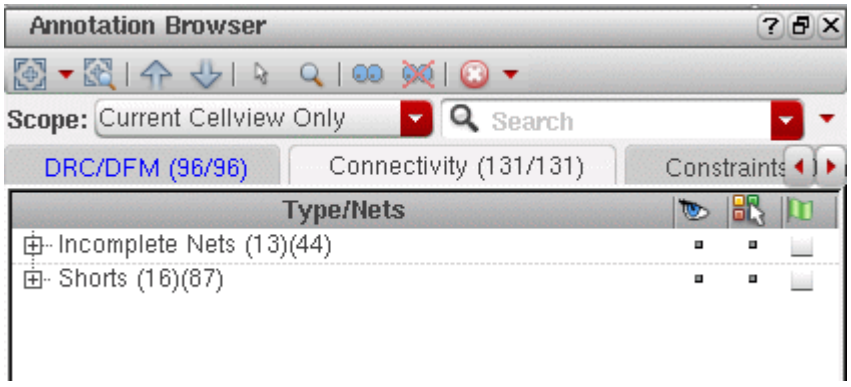
Browser assistant, you can also choose to display the tabs vertically on the side. To control the position of the tabs, use the tabPosition environment variable.

In addition to choosing the position of the tabs, you can choose the order in which the tabs are displayed by manually dragging and dropping them to form the desired order. After you have decided upon the tab display order, you can save your settings so that the tabs display in the same order the next time you open the browser.

For example, the figure below shows the *Connectivity* tab being dragged to the right so that it is the second tab to be displayed the next time the Annotation Browser assistant is opened.



Notice in the figure below that the *Connectivity* tab is now the second tab to be displayed. *DRC/DFM* is the first.

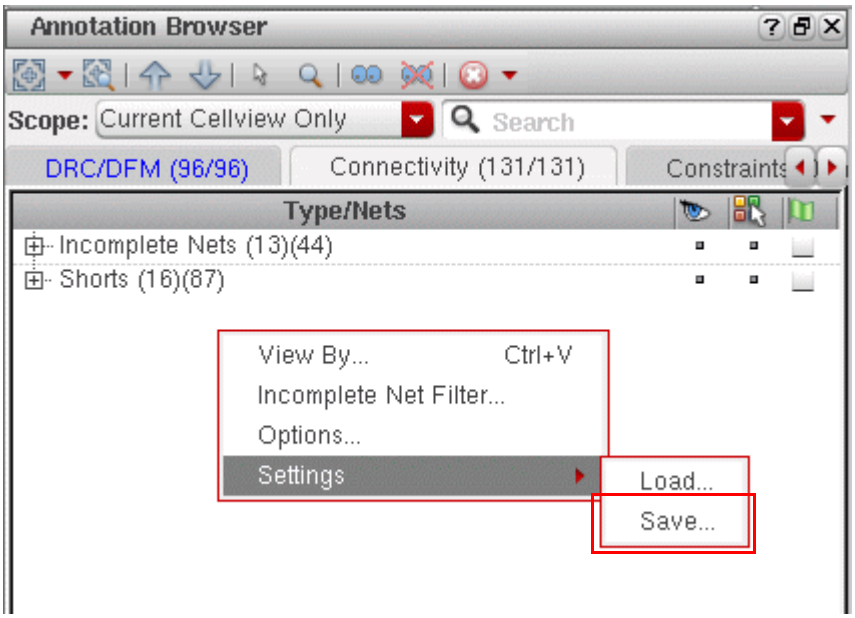


Saving the Annotation Browser Settings

To make sure the order that you choose for the Annotation Browser tabs is retained even after you close the current Layout XL session, you must save your tab order preferences to a `*.ini` file. To do this:

© 2023 Cadence Design Systems, Inc.

➡ In the browser pane, right-click and select *Settings – Save* from the shortcut menu, as shown in the figure below.



Your current tab order settings will get saved to a `.ini` file. You can then load the `.ini` file using the *Settings – Load* command to restore the saved settings the next time you launch Annotation Browser.

Alternatively, you can save your settings to the `ab.ini` file, which gets loaded automatically when you launch the Annotation Browser. The `ab.ini` file is, by default, saved under the `.cadence/dfII/ab` directory, unless referenced at some other location as defined in the `setup.loc` file.

Related Topics

Annotation Browser Assistant

Incomplete Nets

Incomplete Nets Commands

Showing Newly Created Incomplete Nets Automatically

Showing and Hiding Current Incomplete Nets

Checking Shorts, Incomplete Nets, and Illegal Overlaps

© 2023 Cadence Design Systems, Inc.

Annotation Browser Pane

The Browser Pane displays the markers corresponding to each application category supported by a tab. For example, in the *Constraints* tab, the browser pane displays markers for all the supported categories: Specialty Routing, Placement, and Design and Process Rules.

When you select a marker in the browser pane, it is automatically highlighted in the canvas and vice versa. The current marker is colored magenta in the browser and is haloed in the canvas. If you selected a marker in the canvas and that marker is listed in a tab that is currently hidden, that tab is automatically brought to the front and the tree is scrolled to show the selected marker. Where there are multiple selected markers on different tabs, the focus is given to the current (magenta) marker of the selected set.

The columns in the table show the following information for each marker.

Depending on the tab you are using to view the markers, one or more of the columns listed below may not be available. For example, the *Tool* column is not available in the *Connectivity* and *IDF* tabs. Likewise, the *Signal Type* column is supported by the *Connectivity* tab, but not supported by the other tabs.

Icon	Name	Description
	Tree	Shows the various categories of markers generated by an application. The marker categories are displayed in a Tree structure. The top category node indicates the marker type and the nodes below list marker sub-categories, if any, or the impacted instances or nets, and their location. Every category in the tree is followed by two numbers in parentheses. The first of these is the number of categories one level below the current node; the second is the total number of markers under the current node.

© 2023 Cadence Design Systems, Inc.

Icon	Name	Description
	*Set Highlight State*	Shows and sets the highlight state of the corresponding marker in the design window. Clicking in this column for a particular node sets the highlight state for all the markers under that node. A grayed-out version of the icon displayed against a node indicates that only some of the markers under that node are highlighted. To set or unset the highlight state for all the markers, click the column header.
	*Set Highlight Color*	Shows and sets the color used to draw the corresponding marker in the design window. Clicking in this column for a node sets the highlight color for all the markers under that node. To set the same color for all the markers, click the column header and choose a color from the color menu. Choose cycle to let Layout XL select the colors automatically by cycling through a predefined list. If the node has the dot ( ) icon in this column, it implies that no single color is assigned for the markers in the node. The markers either have no color assigned or at least one marker has a different color from the rest. If the node has a color in this column, it implies that all the markers in the node have the same highlight color assigned. This command does not change the color assigned to the marker in the database, it only changes the color used to draw the marker in the canvas.

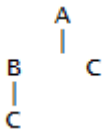
© 2023 Cadence Design Systems, Inc.

Icon	Name	Description
	*Set Marker Check State*	Shows and sets the *Checked* state of the corresponding marker. When a marker is set to *Checked*, you can use the *Hide Checked Markers* button to hide it in the Annotation Browser. Clicking in this column for a particular node sets the *Checked* state for all the markers under that node. To set the Checked state for all the markers, click the column header. ■ If the node displays a tick () in the check box, it implies that all the markers in the node have the *Checked* state set. ■ If the node displays a dot () in the check box, it implies that at least one marker under the node is unchecked. ■ If the node displays an empty check box (), it implies that the *Checked* state is not applied. This command has no effect on the marker display in the canvas, it only controls the marker display in the Annotation Browser.
	Description	Shows a message describing the marker in more detail. Similar information is presented in the *Description* pane at the bottom of the assistant. This column is hidden by default.
	Lib Name	Shows the master library of the cell associated with the marker. This column is hidden by default.
	Cell Name	Shows the master cell name of the cell associated with the marker. This column is hidden by default.

© 2023 Cadence Design Systems, Inc.

Icon	Name	Description
	View Name	Shows the master view name of the cell associated with the marker. This column is hidden by default.
	Layers	Shows the name of the layer associated with the marker. This column is hidden by default in the layout window Annotation Browser. It is not available in the schematic window Annotation Browser.
	Purposes	Shows the purpose of the layer associated with the marker. This column is hidden by default in the layout window Annotation Browser. It is not available in the schematic window Annotation Browser.
	Nets	Shows the net associated with the marker. This column is hidden by default.
	Tools	Shows the Virtuoso tool, such as *VLS Constraint Aware Editing*, that generated the marker. This column is hidden by default.
	Signal Type	Shows the signal type of the nets associated with the marker.
	Severity	Shows the criticality of the marker. This column is hidden by default.
	Short Message	Shows a brief message that gives a quick overview to the marker. The message provided in this column is much shorter compared to the one displayed in the *Description* pane. The *Short Message* column is hidden by default.

© 2023 Cadence Design Systems, Inc.

Icon	Name	Description
	Bbox	Shows the coordinates of the marker bounding box. This column is hidden by default.
	Depth	Shows the lowest depth at which the cellview that contains the marker was found. Let us consider the simple hierarchy represented by the image below:  ■ If the Annotation Browser scope is set to *Top Cellview To Depth* with a depth of `3`, then the markers associated with `C` will display `2` in the *Depth* column. ■ If the Annotation Browser scope is set to *Top Cellview To Depth* with a depth of `1`, then the markers associated with `C` will display `1` in the *Depth* column.
	Type	Shows the marker type. This column is hidden by default.
	State	Shows the current state of a marker. The various states supported are: *Fixed*, *Signoff*, *Waiver*, or a blank *State* column.
	Expected Value	Shows the expected value of the marker.
	Same Mask	Shows the same mask spacing violation markers, which indicate that the distance between two shapes locked to the same color is less than the required same mask spacing as defined in the technology file.

© 2023 Cadence Design Systems, Inc.

Icon	Name	Description
	Rule	Shows the rule ID of a design rule from which the “rule” constraint originates. The rule ID is derived from the Virtuoso Layout Migrate technology database. If the constraint is not originating from a “rule”, the *Rule* column displays a string to indicate the origin of the constraint. For example, *connectivity*. The *Rule* column is only supported by the *IDF* tab.
	Dist	Shows the current distance of the constraint with an optional bracket to show the original distance if the constraint was relaxed, (for example,. “.13 [.26]”). The *Dist* column is only supported by the *IDF* tab.
	Cost	Shows the current cost of the constraint. The *Cost* column is only supported by the *IDF* tab.

Related Topics

Annotation Browser Assistant

Annotation Browser Tabs

Annotation Browser Shortcut Menus

Annotation Browser Options Form

Markers Generated by Connectivity Extractor

Layout XL Assistants

Annotation Browser Description Pane

The *Description* pane displays the message stored on the current marker. The current marker is colored magenta in the browser and haloed in the canvas. If there are multiple markers selected, the current marker and its description changes as you use the *Go To The Next Selected Marker* and *Go To The Previous Selected Marker* buttons to browse the

© 2023 Cadence Design Systems, Inc.

selected set. (This works even if you select the markers in the canvas and their parent category is not expandable in the browser pane.)

Depending on the selection in the Annotation Browser, the *Description* pane shows:

- Nothing, if multiple category nodes are selected.
- The marker node description, if a selected category node has one marker child.
- Description of the first marker, if a selected category node has multiple marker children and the first and the last marker nodes (in sorted order) have the same description.
- HTML links and links to SKILL functions. If markers have a description field created with a reference tags. For example:

  SKILL function

  ```
  dbCreateMarker(cv "<a href=\"skill:myLinkWrapper()\">
  ```

  The SKILL link will be:

  ```
  </a>" "link test" list(0:0 0:2 2:2 2:0) nil t nil "warning"
  "marker with link")
  ```

  HTML

  ```
  dbCreateMarker(cv "<a href=\"www.cadence.com\">
  ```

  The HTML link will be:

  ```
  </a>" "html link test" list(0:0 0:2 2:2 2:0) nil t nil "warning" "marker with
  html link")
  ```

Related Topics

Annotation Browser Assistant

Annotation Browser Tabs

Annotation Browser Shortcut Menus

Annotation Browser Options Form

Markers Generated by Connectivity Extractor

Layout XL Assistants

© 2023 Cadence Design Systems, Inc.

Annotation Browser Shortcut Menus

The Annotation Browser has three shortcut menus.

Right-click a **tab** to:

- Show or hide the tabs selectively.

  Alternatively, you can use the autoHideEmptyTabs environment variable to specify whether to show or hide the empty tabs.

- Hide all tabs.

  Even when you choose to hide all the tabs, the Annotation Browser will display at least one tab. This could either be the currently active tab or the tab that is assigned the first position.

- Automatically hide any tabs that are empty.

  Alternatively, you can use the autoHideEmptyTabs environment variable to specify whether to show or hide the empty tabs.

Right-click a **table header** to:

- Show or hide different columns in the table.
- View the markers in the current tab grouped by the selected column.
- Sort the markers in the current tab grouped by the selected column.

Right-click the **table body** to access the following commands. (These are enabled and disabled dynamically depending on what is selected in the browser window.)

Command Name	Description
Open (<mode>/<tab>)	Opens the cellview containing the selected marker with the behavior specified in the command name. For example, *Open (edit/new tab)* opens the cellview in edit mode in a new tab; *Open (read/current tab)* opens the cellview in read mode in the current tab. You specify the default behavior for this command in the Navigator. To open a different view of the current cell, use the *Open* command instead.

© 2023 Cadence Design Systems, Inc.

Open...	Brings up the Open form where you can specify ■ Which view of the current cell you want to open ■ Whether that view is opened in edit or read mode ■ Whether it is opened in a new tab, the current tab, or in a new window.
Edit In Place	Edits in place by choosing the best hierarchical path according to the current context, when one or more rows are selected in the Annotation Browser that involve markers on the same cellview. This option is only available in the `maskLayout` views.
Expand All Below	Expands the hierarchy to display everything below the currently selected node. You can expand a single level of hierarchy for a selected entry using the Right arrow on your keyboard.
Collapse All Below	Collapses the hierarchy to hide everything below the currently selected node. You can collapse a single level of hierarchy for a selected entry using the Left arrow on your keyboard.
Select All Below	Expands the hierarchy and selects all the marker entries below the currently selected node.
Zoom To Selected	Zooms to the selected set of markers in the design window. For markers in the hierarchy, the *Zoom To Selected* functionality highlights the markers of the first instance that satisfies the specified Annotation Browser depth and the display start- and stop-level criteria. If a marker is present in the Annotation Browser tree but cannot be visualized because no instance containing those markers can be highlighted on canvas, then a "not allowed" icon () is displayed against its *Highlight State* column.

© 2023 Cadence Design Systems, Inc.

Check/Uncheck	Sets the *Checked* state for the selected markers. When a marker is set to *Checked*, you can use the *Hide Checked Markers* button to hide it in the Annotation Browser. This command has no effect on the marker display in the canvas, only in the Annotation Browser. See Set Marker Check State for more information.
Set/Unset Highlight State	Sets the highlight state of the selected markers in the design window. Using this command on a node sets the visibility for all the markers under that node. A grayed-out version of the eye icon displayed against a node indicates that only some of the markers under that node are highlighted. See Set Highlight State for more information.
Suppress/Unsuppress Expansion	Suppresses (or unsuppresses) the expansion of the node under the cursor. Suppress expansion globally by setting a limit on the number of violations using the *Suppress expansion* option in the Navigator. You then use these commands to dynamically unsuppress (and suppress) the expansion for a selected node in the browser. These commands appear only when expansion has been suppressed as a result of the limit set using the *Suppress expansion* option.
Highlight Color	Sets the color used to draw the corresponding marker in the canvas. Using this command for a node sets the color for all the markers under that node. Choose *cycle* to let Layout XL select the color automatically by cycling through a predefined list. This command does not change the color assigned to the marker in the database, only the color used to draw the marker in the canvas. See Set Highlight Color for more information.

© 2023 Cadence Design Systems, Inc.

Delete	Deletes the selected marker or marker category from the Annotation Browser and the design window.
Auto Net Probing	Probes the nets associated with the highlighted and selected markers.
Select Nets	Selects the nets associated with the currently highlighted markers in the layout canvas and Navigator assistant.
Trace Net	Traces the physical and logical connectivity of the selected nets. This command is supported only for the *Connectivity* tab. For more information, see Tracing Nets.
Incomplete Net Filter	Opens the Incomplete Net Filter Form where you filter the incomplete net markers in the browser based on the set of currently selected nets in the layout canvas or Navigator assistant. For more information, see Setting an Incomplete Net Filter.
Locate Short	Opens the Short Locator Form that can be used to locate the short that is causing an error. For more information, see Locating Shorts Using Short Locator.
Extract Nets	Extracts all the nets associated with the selected markers. If the selected marker does not have a net associated, the *Extract Nets* menu item is disabled.
Extract Unverified Areas	Extracts connectivity from the areas that were not interactively verified by the connectivity extractor.
View By	Opens the Navigator where you specify how the list of annotations is grouped and sorted in the browser. For more information, see Grouping Markers in the Annotation Browser.

© 2023 Cadence Design Systems, Inc.

Options	Opens the Navigator, where you specify: ■ The number of violations, which prevent a node from being expanded ■ The levels of hierarchy searched for violation markers ■ The default open behavior ■ How new markers are highlighted
Settings	Loads or saves the following user-defined settings for the selected tab; ■ the “View By” defined for the tab ■ the columns to be displayed and the order in which they appear ■ the size and position of each column

You can delete and set the checked state of markers only in the cellviews that are open in edit mode. If you try to use the *Delete* or *Check/Uncheck* commands (or their Annotation Browser Pane equivalents) on a read-only cellview, the application prompts you to confirm if you want to re-open the cellview in edit mode before performing the requested operation.

This prompt represents a behavior change from the previous releases, where the software automatically re-opened the cellview in edit mode, if required, to delete a marker. To restore this behavior, set one or both of the following environment variables to `nil`.

- deleteConfirmModeChange
- checkStateConfirmModeChange

Related Topics

Annotation Browser Assistant

Annotation Browser Tabs

Annotation Browser Options Form

Grouping Markers in the Annotation Browser

Sorting Markers in the Annotation Browser

Filtering Markers in the Annotation Browser

© 2023 Cadence Design Systems, Inc.

Layout XL Assistants

© 2023 Cadence Design Systems, Inc.

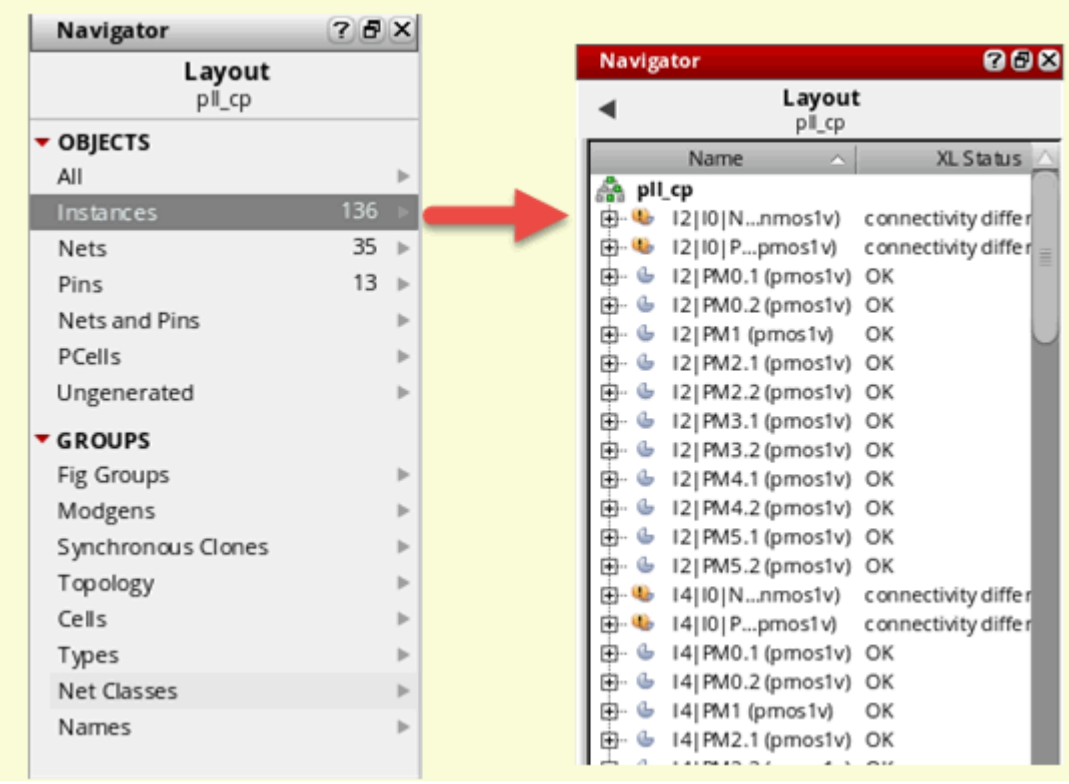
Navigator

This section describes only the extended Navigator functionality enabled in the Layout suite.

The Navigator assistant is divided into two main panes—*Summary* and *Details*. The *Summary* pane lists the datasets in various default and user-defined categories and displays the number of objects that each dataset contains. The *Details* pane, as displayed in the figure below, hierarchically lists the objects that exist within the dataset selected in the *Summary* pane.

Summary pane

Details pane



The *Summary* pane, as mentioned above, lists certain predefined data categories, such as *Instances*, *Pins*, *Ungenerated*, and so on. You can click any of these categories to display the associated objects in the *Details* pane. The *Ungenerated* category, for example, displays a list of all the ungenerated objects in the design—objects that are available in the schematic view but are not yet generated in the layout. For more information on how to use the *Navigator* assistant to view these instances and how the ungenerated instances can be generated in the layout, see Viewing Ungenerated Instances In Navigator.

© 2023 Cadence Design Systems, Inc.

The hierarchical display of the objects in the *Details* pane of the *Navigator* assistant simplifies the task of navigating through a hierarchical design to find the components you are looking for. The hierarchical display can represent devices, nets, pins, and parameterized cells (Pcells).

An unelaborated instance in the *Details* pane displays a [+] sign before the icon; indicating the instance holds some devices, nets, or pins within. To elaborate an instance, you must click the instance once. If the sub instance is a leaf, it does not carry the [+] sign. Else, the sub instances carry the [+] sign till you explore them and browse to the leaf level.

When using Layout XL, there is a Navigator available in the schematic window and one in the layout window. Selecting a component at the current level of hierarchy in either of the Navigators selects the corresponding component in the other Navigator and in the canvases. Similarly, selecting a component at the current level in either of the canvases selects the component in both the Navigators.

Similar to cross-selecting, cross probing across the two views only works for objects selected at the current level of hierarchy. Probes created elsewhere in the hierarchy still propagate through the hierarchy but they do not cross probe to the schematic. See Adding a Probe using the Right Mouse Button.

For a lab database and related workshop instructions to help you explore the features supported by the Navigator assistant, view or download the rapid adoption kit, *Virtuoso Navigator Assistant*.

Related Topics

The Navigator Assistant

Rapid Adoption Kit (*Virtuoso Navigator Assistant*)

Viewing Extended Chain in Navigator

Viewing Extended Selection for Instances, Pins, and Nets in Navigator

Viewing Ungenerated Instances In Navigator

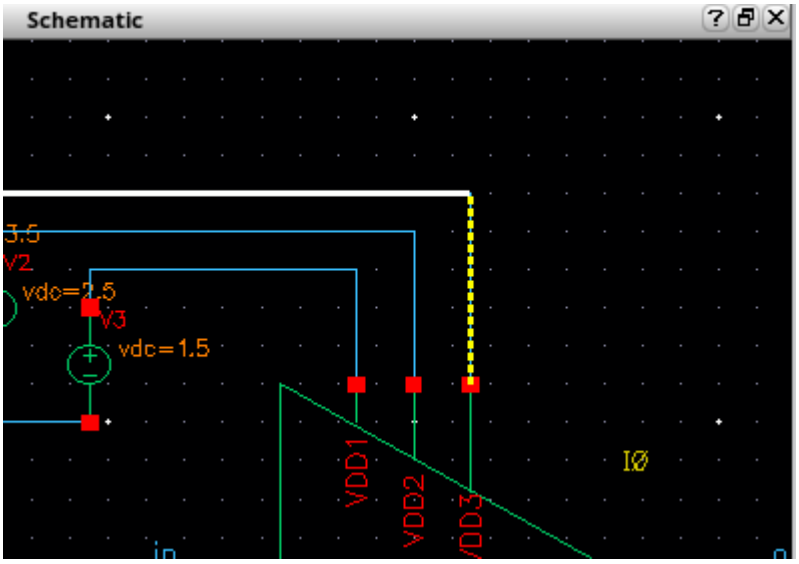
Selecting and Deselecting Nets in Navigator

Adding a Probe using the Right Mouse Button

© 2023 Cadence Design Systems, Inc.

Schematic Assistant

Use the Schematic assistant in Layout to view the current design as displayed in the Virtuoso Schematic Editor. You can move around and zoom in and out on the design as normal, and you can select parts of the design as if you were using the Virtuoso Schematic Editor. The Schematic assistant is useful, for example, for cross-probing, cross-selecting, or using it as a reference.



The Schematic assistant works only in preselect mode, that is, you must select an object before it can be interacted with.

The design must be open in the Virtuoso Schematic Editor in the same session for the Schematic assistant to display anything.

© 2023 Cadence Design Systems, Inc.