

Virtuoso Schematic Editor User Guide

**Product Version IC23.1
November 2023**

© 2023 Cadence Design Systems, Inc. All rights reserved.

Printed in the United States of America.

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

Open SystemC, Open SystemC Initiative, OSCI, SystemC, and SystemC Initiative are trademarks or registered trademarks of Open SystemC Initiative, Inc. in the United States and other countries and are used with permission.

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/or 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

Getting Started	39
Starting the Virtuoso Schematic Editor L/XL	40
Accessing Virtuoso Schematic Editor L/XL	40
Setting Schematics L or XL as the Default Application for Viewing Schematics	40
Quick Reference - Menus and Bindkeys	41
Menu Charts for the Virtuoso Schematic Editor	42
Menu Charts for the Virtuoso Symbol Editor	45
Bindkey Storage	49
Quick Reference - Menu Access Keys	51
Access Keys - Schematic Editor L	52
Access Keys - Virtuoso Symbol Editor	60
Quick Reference - Toolbars	66
Workspaces and Assistants in Schematics L/XL	70
Schematic Editor Workspaces	70
Schematic Editor Assistants	71
Additional Features of the Schematic Editor XL	74
Constraint Management in Schematics XL	74
Virtuoso ADE Integration in Virtuoso Schematic Editor	77
Photonics in Virtuoso Schematic Editor, ADE, and Layout XL (Virtuoso Photonics Option)	78
Help and Support Facilities	81
Contextual Cursors	84
Working with Designs	85
Opening a New Cellview	86
Opening an Existing Cellview	88
Using the Schematic Window	90
Using the Symbol Editor	92
Making Designs Editable	93
Making Designs Read Only	93
Saving Your Edits	94
Closing Editor Windows	95

Virtuoso Schematic Editor User Guide

<u>Quitting the Virtuoso Schematic Editor</u>	96
<u>Viewing a Design in Schematic Assistant of Virtuoso Studio Layout Suite</u>	97
<u>Working with Commands</u>	101
<u>Using Command Prompts</u>	102
<u>Canceling a Command</u>	103
<u>Undoing a Command</u>	103
<u>Redoing a Command</u>	103
<u>Selecting Objects</u>	104
<u>Dynamic Highlighting</u>	105
<u>Extending the Selection Area of a Net</u>	112
<u>Selecting One Object</u>	113
<u>Selecting Multiple Objects</u>	114
<u>Selecting Parts of Objects</u>	116
<u>Selecting Wires</u>	116
<u>Selecting a Wire Name and Its Associated Wire Segments</u>	117
<u>Selecting a Pin and Its Associated Wire Segments</u>	118
<u>Selecting an Instance and Its Associated Wire Segments</u>	119
<u>Deselecting Objects</u>	120
<u>Using Strokes</u>	121
<u>Working With the Design Canvas Window</u>	122
<u>Zooming</u>	123
<u>Panning</u>	130
<u>Using the Magnifier</u>	132
<u>Redrawing a Window</u>	139
<u>Viewing the Same Cell in Two Windows</u>	139
<u>Cycling Between Windows and Views</u>	139
<u>Saving View Area Settings in Memory</u>	140
<u>Restoring View Area Settings</u>	141
<u>Click-Lock</u>	142
<u>Showing and Hiding Labels and Notes</u>	143
<u>The Component Browser</u>	144
<u>Accessing and Using the Component Browser</u>	144
<u>Changing the Browser Type Option to the Component Browser</u>	145
<u>Opening the Component Browser</u>	146
<u>Opening the Add Instance Form Instead of the Component Browser</u>	147
<u>Listing Components from Several Libraries</u>	148

<u>Changing the Component Browser Display</u>	149
 2	
<u>Connectivity and Naming Conventions</u>	151
<u>Updating Connectivity</u>	152
<u>Connectivity with Objects</u>	155
<u>Instance Naming Conventions</u>	156
<u>System-Generated Names</u>	156
<u>User-Defined Names</u>	157
<u>Iterated Names</u>	158
<u>Wire Connections and Wire Naming Conventions</u>	160
<u>Wire-to-Wire Connections</u>	161
<u>Wire-to-Pin Connections</u>	163
<u>Wire-by-Name Connections</u>	164
<u>Wire-to-Iterated Instance Connections</u>	166
<u>Global Net Name Connections</u>	169
<u>Multiple-Bit Wire Connections and Naming Conventions</u>	170
<u>Multiple-Bit Wire Naming Conventions</u>	170
<u>Using Vector Expressions in Multiple-Bit Wire Names</u>	170
<u>Using Vector Expressions in Multiple Signals</u>	171
<u>Evaluating Vector Expressions in Multiple-Bit Wire Names</u>	171
<u>Using Prefix Repeat Operators in Multiple-Bit Wire Names</u>	172
<u>Bus Tapping</u>	172
<u>Tapping Multiple Bits of a Bus</u>	174
<u>Tapping Multiple Bits of a Bundle</u>	177
<u>Tapping Wire Intersections by Name</u>	178
<u>Designating Tap Size and Bit Order</u>	179
<u>System-Generated Net Names</u>	179
<u>Pin Connections and Pin Naming Conventions</u>	181
<u>Hierarchical and Offsheet Pin Names</u>	182
<u>Bus Pin Names</u>	182
<u>Pin-to-Pin Connections</u>	183
<u>Tapping Pins</u>	184
<u>Patch Cord Connections and Patch Cord Naming Conventions</u>	185
<u>Naming Single and Multiple Bits Using Patch Cords</u>	185

Virtuoso Schematic Editor User Guide

<u>Naming Multiple Nets Using Patch Cords</u>	187
<u>Inherited Connections</u>	188
<u>Inherited Connections in a Hierarchy</u>	189
<u>Using Physical Connections to Override Inherited Instance Pins</u>	190
<u>Defining Inherited Connections</u>	191
<u>The Syntax of an Inherited Net Expression</u>	192
<u>How Net Expressions Evaluate</u>	193
<u>Sample Net Expression Library</u>	195
<u>Inherited Connections Restrictions</u>	196
<u>Summary of Naming Conventions</u>	197
<u>Supply and Ground Sensitivity</u>	202
<u>Setting Supply and Ground Sensitivity (Schematic Window)</u>	202
<u>Setting Supply and Ground Sensitivity (Symbol Window)</u>	205
<u>Editing the Sensitivity Terminals of Schematic and Symbol Pins</u>	205
<u>Editing the Optical Signal Types</u>	206
<u>Checking Sensitivity Between Schematic and Symbol Cellviews</u>	207
<u>Propagating Optical Signal Types (Virtuoso Photonics Option)</u>	209
<u>Equivalent Connectivity Time</u>	213
<u>Automatically Signing Off Cosmetic Changes to Symbols</u>	213

3

<u>Creating Schematics</u>	215
<u>Getting Started with Schematics</u>	216
<u>Adding Instances</u>	217
<u>What Is an Instance?</u>	217
<u>Methods of Adding an Instance to Your Design</u>	218
<u>Adding Instances One by One</u>	219
<u>Adding Instances by Rows and Columns (an Array)</u>	225
<u>Adding Instances Using an Iterative Expression</u>	227
<u>Flattening an Instance</u>	229
<u>Changing the Default Instance Name Prefix</u>	235
<u>Editing Instance Masters</u>	236
<u>Specifying the View to Use While Placing the Instance</u>	236
<u>Changing the View after an Instance Is Already Placed</u>	236
<u>Renumbering Instances</u>	237

Virtuoso Schematic Editor User Guide

<u>Magnifying and Rotating Instances Using the symbol_xform View</u>	239
<u>Adding Blocks</u>	240
<u>Adding a Lower-Level Cell Block</u>	241
<u>Adding Wires and Pins to Blocks</u>	243
<u>Changing the Block Pin Direction Rule</u>	244
<u>Changing the Direction or Name of a Single Block Pin</u>	246
<u>Changing the Names and Directions of Multiple Block Pins</u>	247
<u>Deleting Block Pins</u>	249
<u>Disabling Automatic Block Pin Creation</u>	250
<u>Adding Pins</u>	251
<u>Adding Schematic and Offsheet Pins</u>	252
<u>Showing Pin Connections</u>	256
<u>Adding Wires</u>	257
<u>Adding Narrow or Wide Wires</u>	258
<u>Using Wire Snap and Gravity Options</u>	259
<u>Using Wire Draw, Width, and Route Options</u>	259
<u>Using Wire Flight Lines Option</u>	260
<u>Routing Flight Lines</u>	261
<u>Routing Named Connections</u>	262
<u>Manipulating Wires with Only the Mouse</u>	263
<u>Adding Wires to Schematic Instance Pins Using SKILL</u>	264
<u>Adding Wire Names</u>	265
<u>Naming a Wire</u>	266
<u>Adding Wire Names</u>	267
<u>Adding Multiple Wire Names</u>	270
<u>Registering Default Net Names</u>	271
<u>Creating Wire Stubs and Names</u>	273
<u>Verifying Invisible Wire Names</u>	275
<u>Using Net Name Prefixes</u>	276
<u>Automatic Justification of Wire Label Names</u>	277
<u>Adding Net Expression Labels to Create an Inherited Connection</u>	279
<u>Adding a Net Expression Label to a Wire</u>	280
<u>Adding an Inherited Supply Symbol to a Wire</u>	281
<u>Editing a Net Expression Label</u>	283
<u>Changing the Net Expression Label Display</u>	284
<u>Disabling the Automatic Usage of the Default Signal Name</u>	286

Virtuoso Schematic Editor User Guide

<u>Deleting a Net Expression Label</u>	286
<u>Adding netSet Properties to Create an Inherited Connection</u>	287
<u>Adding a netSet Property to an Instance</u>	288
<u>Modifying a netSet Property</u>	291
<u>Deleting a netSet Property</u>	291
<u>Analyzing Net Expression Property Names and Evaluated Names</u>	292
<u>Finding a Net Expression</u>	293
<u>Finding and Replacing a Net Expression</u>	294
<u>Manipulating Inherited Connections</u>	295
<u>Viewing the Evaluated Net Names</u>	297
<u>Viewing the Full List of all Hierarchical Paths</u>	299
<u>Checking the Net Expression</u>	300
<u>Adding Solder Dots</u>	301
<u>Adding Solder Dots</u>	302
<u>Deleting Solder Dots</u>	302
<u>Disabling Automatic Solder Dot Creation</u>	303
<u>Changing Solder Dot Size</u>	305
<u>Adding Notes</u>	306
<u>Adding Note Shapes</u>	307
<u>Adding Note Text</u>	309
<u>Importing an ASCII File for Note Text</u>	311
<u>Adding Patch Cords Using the Add Instances Form</u>	312
<u>Adding Patch Cords Using the Create Patchcord Form</u>	314
<u>Adding Probes</u>	318
<u>Adding Probes to a Net, Pin, or Instance</u>	319
<u>Showing Probe Information</u>	321
<u>Removing Probes from a Net, Pin, or Instance</u>	322
<u>Removing All Probes</u>	323
<u>Saving Probes</u>	323
<u>Loading Probes</u>	324
<u>Probing a Feedthrough Net</u>	325
<u>Tracing the Hierarchy of a Probed Net Using the Probes Assistant</u>	329
<u>Creating a Mapping Schematic</u>	330

4

<u>Creating a Multisheet Schematic</u>	335
<u>Browsing a List of Multisheets</u>	336
<u>Converting a Single-Sheet Schematic to a Multisheet Schematic</u>	337
<u>Adding New Sheets</u>	338
<u>Editing the Title Block</u>	339
<u>Editing the Sheet Border Size and Type</u>	340
<u>Understanding Offsheet Pins and Their Naming Conventions</u>	341
<u>Adding Offsheet Pins to Multisheet Schematics</u>	344
<u>Understanding the Index Schematic</u>	346
<u>Going to and from the Index Schematic</u>	347
<u>Repositioning a Schematic within the Sheet Borders</u>	348
<u>Deleting Sheets</u>	349
<u>Reordering Sheets</u>	350
<u>Renumbering Sheets</u>	351
<u>Working with Cross-References</u>	352
<u>Understanding Cross-Reference Zones</u>	353
<u>Understanding Pin Cross-References</u>	354
<u>Understanding Cross-Reference Reports</u>	355
<u>Preparing to Cross-Reference</u>	356
<u>Displaying Cross-References</u>	357
<u>Zooming In to View All Cross-Referenced Pins One by One</u>	359
<u>Going to a Specific Hierarchical Pin</u>	360
<u>Viewing Cross-Reference Reports</u>	361
<u>Editing a Cross-Reference Display</u>	363
<u>Controlling the Format String Syntax</u>	364
<u>Customizing the Format String Syntax</u>	365
<u>Customizing Multisheets</u>	366
<u>Using Customized Sheets</u>	367
<u>Customizing the Types and Sizes of Sheet Borders</u>	368
<u>Customizing the Cross Reference Options Form</u>	369
<u>Drawing a Sheet Border</u>	370
<u>Drawing Sheet Zones</u>	371
<u>Drawing Zone Labels</u>	372
<u>Adding the Sheet Zone Property</u>	373

Virtuoso Schematic Editor User Guide

<u>Adding a Title Block</u>	374
<u>Adding Title Block Values with Normal Labels</u>	375
<u>Adding Title Block Values with NLP and IL Labels</u>	376
<u>Verifying the Label Type</u>	377
<u>Adding Title Block Properties</u>	378
<u>Customizing Pin Cross-References</u>	379
<u>Presetting the Title Block Values</u>	380

5

<u>Creating Symbols</u>	383
<u>Setting Grid Options</u>	384
<u>Importing Symbols</u>	385
<u>Creating New Symbols</u>	386
<u>Drawing Lines</u>	387
<u>Drawing Rectangles</u>	388
<u>Drawing Polygons</u>	389
<u>Drawing Circles</u>	390
<u>Drawing Ellipses</u>	391
<u>Drawing Arcs</u>	392
<u>Adding Pins as Graphic Images</u>	393
<u>Adding Pins as Instances</u>	395
<u>Creating Custom Pins as Graphic Images</u>	397
<u>Adding a Selection Box</u>	399
<u>Adding Labels</u>	400
<u>Adding a Normal Label</u>	401
<u>Adding an NLP Label</u>	402
<u>Adding an IL Label</u>	406
<u>Store Default of CDF Parameter</u>	407
<u>Effect of Netlisting Mode on netSets</u>	407
<u>IL v. NLP labels</u>	407
<u>Effect of Netlisting Mode On Labels</u>	407
<u>Adding Note Text and Note Shapes</u>	410
<u>Adding a Net Expression Label</u>	410
<u>Setting the Symbol Origin</u>	411
<u>Creating Symbol Cellviews Using Generators</u>	413

6

Automatically Creating Cellviews	415
<u>Features of Creating Cellviews</u>	416
<u>Automatically Creating a Cellview from Another Cellview</u>	417
<u>Replacing an Existing Symbol Cellview</u>	419
<u>Modifying an Existing Symbol Cellview</u>	421
<u>Examples of Replacing and Modifying Pins on a Symbol</u>	424
<u>Automatically Creating a Cellview from a Pin List</u>	427
<u>Automatically Creating a Cellview from an Instance</u>	429
<u>Creating a Cellview from the Canvas (Make Cell)</u>	430
<u>Editing Symbol Generation Options</u>	435
<u>Adding, Changing, and Reordering Pin Names</u>	436
<u>Editing Pin Attributes</u>	439
<u>Excluding or Including Inherited Connection Pins</u>	441
<u>Editing Pin Label Defaults</u>	443
<u>Loading and Saving a TSG Template File</u>	445
<u>Editing Symbol Attributes</u>	447
<u>Editing Symbol Labels</u>	450
<u>Creating a New Label</u>	452
<u>Deleting Labels</u>	455
<u>Modifying Label Attributes</u>	456
<u>Editing Properties</u>	457
<u>Creating a New Property</u>	459
<u>Deleting Properties</u>	460
<u>Modifying Property Attributes</u>	461
<u>Specifying Symbol Generator Pin Sorting</u>	462

7

Editing Objects	463
<u>Using Direct Manipulation</u>	464
<u>Direct Manipulation Operations</u>	465
<u>Turning Off Direct Manipulation</u>	466
<u>Undoing and Redoing an Edit</u>	467
<u>Undo/Redo using DFII on OpenAccess</u>	467

Virtuoso Schematic Editor User Guide

<u>Changing the Undo Limit</u>	468
<u>Stretching</u>	469
<u>Stretching with Bindkeys</u>	470
<u>Stretching in the Schematic Window</u>	470
<u>Using the Reroute Wire Segments Connected To Option</u>	472
<u>Stretching Using Route Method Simple</u>	481
<u>Copying</u>	484
<u>Copying Single Objects</u>	485
<u>Copying Arrays of Objects</u>	485
<u>Copying Across Cellviews</u>	487
<u>Copying with Direct Manipulation</u>	487
<u>Copying with Bindkeys</u>	488
<u>Moving</u>	489
<u>Moving Objects</u>	490
<u>Moving with Direct Manipulation</u>	491
<u>Moving with Bindkeys</u>	491
<u>Snapping to Grid</u>	492
<u>Snapping Pre-Selected Objects to Grid</u>	495
<u>Snapping Post-Selected Objects to Grid</u>	495
<u>Limitations</u>	496
<u>Snapping Instances to Grid Using SKILL Functions</u>	499
<u>Deleting</u>	500
<u>Deleting Preselected Objects</u>	501
<u>Deleting Postselected Objects</u>	501
<u>Deleting Sheet Borders</u>	501
<u>Ignoring Instances</u>	502
<u>Ignoring Pre-Selected Instances</u>	507
<u>Ignoring Post-Selected Instances</u>	507
<u>Ignoring Instances Using SKILL Functions</u>	508
<u>Rotating</u>	509
<u>Rotating Using the Edit Toolbar</u>	509
<u>Rotating Preselected Objects</u>	510
<u>Rotating Postselected Objects</u>	511
<u>Rotating with Direct Manipulation</u>	511
<u>Rotating with Bindkeys</u>	512
<u>Aligning</u>	513

Virtuoso Schematic Editor User Guide

<u>Aligning Using the Edit Toolbar</u>	514
<u>Aligning Using SKILL Functions</u>	519
<u>Distributing</u>	520
<u>Distributing Using the Edit Toolbar</u>	521
<u>Distributing Using SKILL Functions</u>	522
<u>Discarding Edits</u>	523
<u>Alternating Symbol Views</u>	524
<u>Changing the View of a Preselected Object</u>	524
<u>Changing the View of a Postselected Object</u>	524
<u>Toggling Objects</u>	525
<u>Toggling Pin Direction Options</u>	525
<u>Toggling Wire Draw Mode Options</u>	525
<u>Toggling Instance Symbol View Options</u>	526
<u>Updating Pins Across Cellviews</u>	526
<u>Annotating Data on the Schematic Window</u>	527
<u>Annotation Context Menus on Schematic</u>	528
<u>Using Annotation Balloons</u>	530
<u>Using Annotation Setup Form</u>	534
<u>Annotation Balloon Options</u>	542
<u>Saving Annotation Settings</u>	544
<u>Loading Annotation Settings</u>	545
<u>Saving Data to User Cell-Level CDF</u>	549
<u>Loading Data from Effective CDF</u>	549
<u>Copying Annotation Settings</u>	549
<u>Applying Global Annotation Settings</u>	551

8

<u>Editing Properties</u>	555
<u>Using Command Shortcuts</u>	556
<u>Searching for Objects That Have a Specific Property</u>	557
<u>Search String Wildcards</u>	558
<u>Replacing Properties</u>	559
<u>Replacing String-Tagged Characters</u>	560
<u>Selecting Objects to Edit</u>	565
<u>Using Repeating Commands</u>	566

Virtuoso Schematic Editor User Guide

<u>Selecting Objects Automatically by Type</u>	567
<u>Selecting Objects by Property</u>	569
<u>Adding a Property Name to the Search List</u>	570
<u>Selecting Objects by Filtering</u>	571
<u>Object and Cellview Property Types and Values</u>	572
<u>Editing Object Properties</u>	574
<u>Applying Edits to Objects and Displaying Object Types</u>	575
<u>Applying One or More CDF Parameter Edits to Multiple Objects</u>	576
<u>Adding Object Properties</u>	578
<u>Adding Hierarchical Object Properties</u>	579
<u>Deleting Object Properties</u>	581
<u>Modifying Object Properties</u>	582
<u>Editing Specific Object Properties</u>	583
<u>Editing Object Properties (Text Edit) Directly on the Design Canvas</u>	584
<u>Editing Instance and Block Properties</u>	591
<u>Editing Pin Properties</u>	592
<u>Designating the Order of Pins</u>	593
<u>Copying a Pin Order from Another Cellview</u>	594
<u>Resolving Pin Mismatch</u>	594
<u>Verifying Pin Order</u>	595
<u>Editing Pin Name Properties</u>	595
<u>Editing Wire Properties</u>	597
<u>Editing Wire Name Properties</u>	598
<u>Editing Instance Pin Properties on an Instance of a Block</u>	599
<u>Editing Label Properties</u>	600
<u>Resetting Invisible Labels</u>	601
<u>Showing/Hiding Instance Labels</u>	602
<u>Editing netSet Properties</u>	603
<u>Editing Note Text Properties</u>	604
<u>Editing Note Shape Properties</u>	605
<u>Editing Symbol Shape Properties</u>	606
<u>Editing Symbol Pin Properties</u>	607
<u>Editing Symbol Label Properties</u>	608
<u>Editing Symbol Selection Box Properties</u>	609
<u>Editing User-Defined Object Properties</u>	610
<u>Passing Parameters in a Design</u>	611

Virtuoso Schematic Editor User Guide

<u>Editing Cellview Properties</u>	612
<u>Adding Cellview Properties</u>	613
<u>Deleting Cellview Properties</u>	614
<u>Modifying Cellview Properties</u>	615
<u>Editing VHDL Properties</u>	616
<u>Adding VHDL Properties</u>	617
<u>Deleting VHDL Properties</u>	618
<u>Modifying VHDL Properties</u>	619
<u>Setting the Tool Filter</u>	620
<u>Activating the Auto-Update Tool Filter</u>	621

9

Traversing the Design Hierarchy and Creating a Design

<u>Configuration View</u>	623
<u>About the Design Hierarchy</u>	624
<u>Exploring the Design Hierarchy</u>	625
<u>Top-Level Schematic View</u>	627
<u>Middle-Level Schematic View</u>	628
<u>Lower-Level Design Hierarchy</u>	629
<u>About the Configuration View of a Design Hierarchy</u>	630
<u>About the Hierarchy Editor</u>	631
<u>Traversing a Design Hierarchy</u>	632
<u>Descending Using the Descend Command</u>	633
<u>Editing an Instance in Place</u>	636
<u>Descending Using the View Name List</u>	637
<u>Setting the View Precedence Order</u>	638
<u>Returning Up the Design Hierarchy</u>	638
<u>Using Objective-Sensitive Menus to Traverse the Design Hierarchy</u>	639
<u>Using Bindkeys to Traverse the Design Hierarchy</u>	640
<u>Displaying the Current Hierarchical Scope</u>	641
<u>Displaying Cellview Listings Using Print Tree</u>	642
<u>Creating a New Configuration View for the Design Hierarchy</u>	644
<u>Opening an Existing Design Configuration View</u>	645
<u>About the Hierarchy Editor within the Virtuoso Schematic Editor</u>	646
<u>Working with a Design Configuration within the Virtuoso Schematic Editor</u>	647

Virtuoso Schematic Editor User Guide

<u>Editing a Design Configuration within the Virtuoso Schematic Editor</u>	648
<u>Setting Instance Bindings</u>	649
<u>Updating a Design Configuration</u>	651
<u>Building a New Inherited View List</u>	652
<u>Showing Views Found</u>	653

10

<u>Checking Designs</u>	655
<u>Features of the Check Commands</u>	656
<u>Highlighting Errors and Warnings in the CIW</u>	657
<u>Setting the Check Options</u>	658
<u>Setting User Preferences</u>	659
<u>Setting Schematic Rule Checks</u>	661
<u>Creating Custom Checks</u>	674
<u>Checking a Design</u>	676
<u>Checking and Saving a Design</u>	677
<u>Checking the Current Cellview</u>	679
<u>Checking a Design Hierarchy</u>	680
<u>Checking a Design Hierarchy in the Configuration Context</u>	682
<u>Checking Multisheet Schematics</u>	683
<u>Checking a Label Attachment</u>	684
<u>Cross-View Checking</u>	685
<u>Ignoring Explicitly Inherited Terminals for Text Cellviews</u>	686
<u>Checking Read-Only Cellviews</u>	687
<u>Bypassing Checks</u>	689
<u>Bypassing Floating Pin Checks</u>	689
<u>Bypassing Unconnected Wire Checks</u>	691
<u>Drawing a Custom noConn Symbol</u>	692
<u>Resolving Check Errors</u>	693
<u>Finding Errors and Warnings in a Design</u>	694
<u>Mapping an Error Message to a Marker</u>	695
<u>Managing the Error Message List</u>	695
<u>Assigning an ignoreCheck Property</u>	695
<u>Suppressing Message Displays</u>	696
<u>Deleting an ignoreCheck Property</u>	696

Virtuoso Schematic Editor User Guide

<u>Deleting All ignoreCheck Properties</u>	696
<u>Deleting Individual Markers</u>	697
<u>Deleting All Markers</u>	697

11

<u>Plotting Designs</u>	699
<u>Setting Up System Options to Plot</u>	700
<u>Setting Up Printer Options to Use Various Paper Sizes</u>	701
<u>Setting Various Paper Sizes Using the .cdsplotinit File</u>	701
<u>Plotting a Design</u>	702
<u>Plotting a Library</u>	703
<u>Plotting a Cellview</u>	705
<u>Plotting Part of a Schematic</u>	705
<u>Plotting Without a Header Page for Every Session</u>	706
<u>Adding Notes to a Plot</u>	706
<u>Setting Plot Options</u>	707
<u>Plotting a Color Schematic in Black and White</u>	711
<u>Changing the Background Color of a Schematic Window</u>	712
<u>Using a Plot Template File</u>	713
<u>Creating a Plot Template File</u>	713
<u>Loading a Plot Template File</u>	715
<u>Creating a Default Plot Template File</u>	716
<u>Checking the Queue Status</u>	717
<u>Canceling a Plot Job</u>	717

12

<u>Setting Virtuoso Schematic Editor Options</u>	719
<u>Overview of Virtuoso Schematic Editor Options</u>	720
<u>Changing User Preferences</u>	721
<u>Minimizing Mouse Clicks</u>	721
<u>Changing the Default Display Setting for Command Option Forms</u>	723
<u>Saving the Current Window Setup</u>	724
<u>Restoring a Saved Window Setup</u>	724
<u>Viewing Bindkey Settings</u>	726
<u>Changing Browser and CIW Preferences</u>	726

Virtuoso Schematic Editor User Guide

<u>Customizing Editor Options</u>	727
<u>Canceling the Repeat Command Option</u>	727
<u>Changing Display Options</u>	729
<u>Changing Canvas Color</u>	729
<u>Selecting Objects Using the Filter Option</u>	731
<u>Setting Plot Options</u>	731
<u>Setting Schematic Check Options and Rules</u>	731
<u>Setting the Tool Filter</u>	731
<u>Saving Form Field Default Settings</u>	731
<u>Loading Form Field Default Settings</u>	733

13

<u>Customizing the Virtuoso Schematic Editor</u>	735
<u>Setting Default Schematic Application</u>	736
<u>Customizing Pull-Down and Pop-Up Menus</u>	737
<u>Default Menu Definitions</u>	737
<u>Sample Menu File</u>	738
<u>Customizing Object-Sensitive Menus</u>	739
<u>Example of Customizing the Instance Object-Sensitive Menu</u>	739
<u>Understanding Initialization and Setup Files</u>	741
<u>Converting a Schematic from a 0.125 Grid to a 0.1 Grid</u>	741
<u>Customizing Global Editor Variables for Form Fields</u>	742
<u>Schematic Editor Variables</u>	742
<u>Symbol Editor Variables</u>	743
<u>Symbol Generation (tsg) Variables</u>	743
<u>HDL (Verilog and Verilog-A) Variables</u>	744
<u>AMS Netlisting Variables</u>	744
<u>schBlockPinAttrs</u>	745
<u>schBlockTemplate</u>	746
<u>schCycleViewNameExclusionList</u>	748
<u>schHdlEditAfterError</u>	749
<u>schHdlCrossViewCheck</u>	750
<u>schHdlCrossViewString</u>	751
<u>schHdlPrintCommand</u>	752
<u>schHdlUpdateSymbolAfterEdit</u>	753

Virtuoso Schematic Editor User Guide

<u>schPinMasters</u>	754
<u>schPinNetExprPlacement</u>	756
<u>schPinNetExprGenFunc</u>	758
<u>schPlotTemplate</u>	759
<u>schPropertiesFilter</u>	760
<u>schSelPropertyNameList</u>	761
<u>schSheetMasters</u>	763
<u>schSimProbeMasters</u>	765
<u>schSRCPackagedChecks</u>	766
<u>schSymbolLabelChoices</u>	767
<u>schSymbolPinMasters</u>	769
<u>schViewMasters</u>	771
<u>schViewMastersOptionFunc</u>	773
<u>schZoneFormatString</u>	774
<u>schZoneFormatValue</u>	775
<u>tsgConnectorMasters</u>	776
<u>tsgPinGraphicMasters</u>	778
<u>tsgTemplateMasters</u>	780
<u>Customizing the Plot Configuration</u>	781
<u>Customizing Schematic Object Attributes</u>	782
<u>Schematic Objects and LPPs</u>	782

14

<u>The Search Assistant and Toolbar</u>	787
<u>Introducing Search</u>	788
<u>Search Terminology</u>	789
<u>Accessing the Search Assistant and Toolbar</u>	794
<u>Hiding the Search Assistant and Toolbar</u>	794
<u>The Search User Interface</u>	795
<u>The Search Toolbar</u>	796
<u>The Search Assistant Pane</u>	810

15

<u>The Navigator Assistant</u>	819
<u>Introducing the Navigator</u>	820

Virtuoso Schematic Editor User Guide

<u>Accessing the Navigator</u>	822
<u>Hiding the Navigator</u>	823
<u>Navigator Keyboard Shortcuts</u>	824
<u>The Navigator User Interface</u>	825
<u>Selection and Display in the Navigator</u>	841
<u>Controlling Object Display in the Navigator with Sets</u>	841
<u>Controlling Object Display in the Navigator with Search</u>	844
<u>Updating the Navigator Tree</u>	846
<u>Being In-Context or Out-Of-Context</u>	846
<u>Cross Selecting Between the Navigator and the Current Cellview</u>	848
<u>Opening Instances and Cellviews from the Navigator</u>	853
<u>The Navigator Options Form</u>	856

16

<u>The Property Editor Assistant</u>	861
<u>Introducing the Property Editor</u>	862
<u>Accessing the Property Editor</u>	863
<u>Hiding the Property Editor</u>	864
<u>The Property Editor User Interface</u>	865
<u>Selecting Objects</u>	867
<u>Filtering by Property Type</u>	872
<u>The Property Editor Context-Menu</u>	874
<u>Property Editor Keyboard Shortcuts</u>	882
<u>Adding and Deleting Properties using the Property Editor</u>	883
<u>Adding a Property Using the Property Editor</u>	883
<u>Deleting a Property Using the Property Editor</u>	885
<u>Editing Property Values</u>	886
<u>Editing Single Object Property Values</u>	887
<u>Editing Multiple Object Property Values</u>	891
<u>Compound Properties</u>	893
<u>Extended Editing Forms</u>	894
<u>Property Status Visualization</u>	899
<u>Default Values</u>	899
<u>Read-Only Values</u>	900
<u>Differences In Property Values Across Selected Objects</u>	900

Virtuoso Schematic Editor User Guide

<u>The Property Editor and CDF Support</u>	901
<u>Callbacks</u>	901
<u>cdfgData</u>	902
<u>cdfgForm</u>	903
<u>Modgen Property Updates using the Property Editor Assistant</u>	905

17

<u>The Probes Assistant</u>	907
<u>Introducing the Probes Assistant</u>	908
<u>Benefits of using Probes Assistant</u>	908
<u>Accessing the Probes Assistant</u>	909
<u>Hiding the Probes Assistant</u>	909
<u>The Probes Assistant Graphical User Interface (GUI)</u>	909
<u>Displaying Probed Nets in the Probes Assistant</u>	910
<u>Context-Menu Options</u>	911
<u>Using Context-Menu Options in Probes Area</u>	912
<u>Using Context-Menu Options in Display Area</u>	912
<u>Controlling the Display in the Probes Assistant</u>	913
<u>Saving Probe Paths to a File</u>	916
<u>Removing a Probe or All Probes</u>	917

18

<u>The World View Assistant</u>	919
<u>World View Context-Menu</u>	921

19

<u>Exporting a Schematic Hierarchy to HTML</u>	923
<u>Schematic HTML Publisher Overview</u>	923
<u>Use Scenarios</u>	923
<u>Supported Browsers</u>	924
<u>Exporting a Design Hierarchy to HTML (non-configured schematic)</u>	925
<u>Exporting a Design Hierarchy to HTML (configured schematic)</u>	927
<u>Navigating Exported HTML in a Web Browser</u>	928
<u>The Dynamic Preview Pane</u>	930

Virtuoso Schematic Editor User Guide

<u>The Navigator Pane</u>	933
<u>The Bill of Materials</u>	935
<u>Printing an Exported Design</u>	936
<u>Customizing the HTML Files</u>	937

20

<u>RF Transmission Lines</u>	939
<u>Inflate</u>	942
<u>Create Net Topology</u>	943
<u>Create Stackup</u>	947
<u>Transmission Cells</u>	948
<u>Customizing the RF Toolbar</u>	949

21

<u>Using Split Symbols</u>	951
<u>Creating Split Symbols</u>	952
<u>Adding a Split Annotate Label</u>	952
<u>Marking a Symbol as a Split Symbol</u>	953
<u>Adding Split Instances to a Schematic</u>	955
<u>Viewing the Splits in a Design</u>	957
<u>Editing Properties of Split Instances</u>	959

A

<u>Environment Variables</u>	961
<u>Changing the Default Environment Variables</u>	961
<u>Copying Environment Variables from the Samples Directory to ~/.cdsenv</u>	961
<u>Checking the Value of an Environment Variable</u>	962
<u>Setting Environment Variables in the CIW</u>	962
<u>Changing Graphic Environment Variables That Are Cellview Properties</u>	962
<u>Retaining the Current Settings for Future Sessions</u>	963
<u>addStubs</u>	964
<u>aelEvalCDFParam</u>	965
<u>allTerms</u>	966
<u>annotationBalloonPinningStatus</u>	967

Virtuoso Schematic Editor User Guide

<u>annotationSetupFileList</u>	968
<u>autoDot</u>	969
<u>autoEquivalentConnectivityTime</u>	970
<u>autoLoadBindKeys</u>	971
<u>autoWireStubBusTerms</u>	972
<u>autoZoomMode</u>	973
<u>autoZoomPan</u>	974
<u>autoZoomScale</u>	975
<u>blockDirRules</u>	976
<u>broadcast</u>	977
<u>browserShowSchViews</u>	978
<u>browserType</u>	979
<u>bundleDisplayMode</u>	980
<u>checkAlignOverlaps</u>	981
<u>checkAlways</u>	982
<u>checkBeepOn</u>	983
<u>checkConstraints</u>	984
<u>checkHierOpenMode</u>	985
<u>checkHierSave</u>	986
<u>checkSymTime</u>	987
<u>comanageConstraintView</u>	988
<u>comanageFollowModeOnDescend</u>	989
<u>comanageFollowModeOnOpen</u>	990
<u>compBackTrace</u>	991
<u>condOp</u>	992
<u>copyDesignIntent</u>	993
<u>createCvOptions</u>	994
<u>createInstResetCDF</u>	995
<u>createInstShowCDF</u>	996
<u>createLabelAutoCenterSize</u>	997
<u>createLabelAutoJustify</u>	998
<u>createLabelAutoOffset</u>	999
<u>createLabelAutoSpacer</u>	1000
<u>createLabelFontHeight</u>	1001
<u>createLabelFontStyle</u>	1002
<u>createLabelHorzOffset</u>	1003

Virtuoso Schematic Editor User Guide

<u>createLabelHorzPosition</u>	1004
<u>createLabelHorzRotation</u>	1005
<u>createLabelJustify</u>	1006
<u>createLabelOffsetMode</u>	1007
<u>createLabelVertOffset</u>	1008
<u>createLabelVertPosition</u>	1009
<u>createLabelVertRotation</u>	1010
<u>createPinAutoNameFromNamedNet</u>	1011
<u>cvDisplay</u>	1012
<u>defaultsFile</u>	1013
<u>defWideWireWidth</u>	1014
<u>deleteInstPinsWire</u>	1015
<u>descendCanvasType</u>	1016
<u>descendDemoteViewNameList</u>	1017
<u>descendSheet</u>	1018
<u>descendTarget</u>	1019
<u>differential</u>	1020
<u>dimming</u>	1021
<u>dimmingIntensity</u>	1022
<u>dimmingMode</u>	1023
<u>directEditOnlySelected</u>	1024
<u>disablePortOrderPopup</u>	1025
<u>dotOverlap</u>	1026
<u>drawMode</u>	1027
<u>drawModeRoute</u>	1028
<u>useDynamicHighlightPacket</u>	1029
<u>echoToCIW</u>	1030
<u>editPropChangeAllConfirm</u>	1031
<u>editPropProtectCDF</u>	1032
<u>editPropReadOnlyFields</u>	1033
<u>editPropReadOnlyNetName</u>	1034
<u>editPropShowCDF</u>	1035
<u>editPropShowSystem</u>	1036
<u>editPropShowUser</u>	1037
<u>findAccessMode</u>	1038
<u>findZoomButton</u>	1039

Virtuoso Schematic Editor User Guide

<u>flattenConnectMode</u>	1040
<u>flightWidth</u>	1041
<u>gravityOn</u>	1042
<u>ignoreHilightColor</u>	1043
<u>ignoreHilightOn</u>	1044
<u>ignoreShowMessage</u>	1045
<u>incrementalConn</u>	1046
<u>incrementalConnTimeout</u>	1047
<u>inhConnDisableDefNetField</u>	1048
<u>inhConnDisablePinFields</u>	1049
<u>inhConnGenNetExprDefNet</u>	1050
<u>inhConnGenPinNetExpr</u>	1051
<u>instObjectSelectionMode</u>	1052
<u>invokeKanjiEditor</u>	1053
<u>labelsVisibleDuringDrag</u>	1054
<u>libList</u>	1055
<u>magAnchorPosition</u>	1056
<u>magConfined</u>	1057
<u>magCursorOffset</u>	1058
<u>magDisplayMode</u>	1059
<u>magFloatPosition</u>	1060
<u>magOffsetCorner</u>	1061
<u>magScale</u>	1062
<u>magSize</u>	1063
<u>makeCellFactorize</u>	1064
<u>makeCellPlaceSymbol</u>	1065
<u>mapCvMappingSchViewName</u>	1066
<u>mapCvTermDirectionCheck</u>	1067
<u>markerSeverity</u>	1068
<u>markerShowIgnore</u>	1069
<u>markerZoom</u>	1070
<u>matchedToIgnore</u>	1071
<u>maxLabelOffsetUU</u>	1072
<u>minMarkerSize</u>	1073
<u>modalCommands</u>	1074
<u>netExprDisplay</u>	1075

Virtuoso Schematic Editor User Guide

<u>netNameMethod</u>	1076
<u>netNamePrefix</u>	1077
<u>netNameRangeAuto</u>	1078
<u>netNameRangeDescend</u>	1079
<u>netNameSeparator</u>	1080
<u>noteFontHeight</u>	1081
<u>noteFontStyle</u>	1082
<u>noteJustify</u>	1083
<u>noteShape</u>	1084
<u>noteShapeLineStyle</u>	1085
<u>noteType</u>	1086
<u>objectInfoInStatusBar</u>	1087
<u>pinBackTrace</u>	1088
<u>pinLabelDrafting</u>	1089
<u>pinLabelFontHeight</u>	1090
<u>pinLabelFontStyle</u>	1091
<u>pinLabelJustify</u>	1092
<u>pinLabelOverbar</u>	1093
<u>pinLabelSetDrafting</u>	1094
<u>pinLabelSetFontHeight</u>	1095
<u>pinLabelSetFontStyle</u>	1096
<u>pinLabelSetJustify</u>	1097
<u>pinLabelSetOverbar</u>	1098
<u>pinNameSeed</u>	1099
<u>pinRefDuplicates</u>	1100
<u>pinRefFormat</u>	1101
<u>pinRefSorting</u>	1102
<u>postProcNetSigTypes</u>	1103
<u>probeConnectedInstances</u>	1104
<u>probeCycleBool</u>	1105
<u>probeDirection</u>	1106
<u>probeHiliteLppList</u>	1107
<u>probeHiliteLppString</u>	1108
<u>probeTraverseFeedThrough</u>	1109
<u>probeTraverseHierarchy</u>	1110
<u>probeNetDisplayHierarchyLevel</u>	1111

Virtuoso Schematic Editor User Guide

<u>propEditShowEOPSections</u>	1112
<u>recursionCheck</u>	1113
<u>replMasterSymPinName</u>	1114
<u>replSaveChanges</u>	1115
<u>runSRC</u>	1116
<u>runVIC</u>	1117
<u>saveAction</u>	1118
<u>saveDefaultPortOrder</u>	1119
<u>schCoalesceLimit</u>	1120
<u>schCursorShape</u>	1121
<u>schDisplayDepth</u>	1122
<u>schDrawAxesOn</u>	1123
<u>schDynamicHighlightOn</u>	1124
<u>schDynamicNetHighlightColor</u>	1125
<u>schDynamicNetHighlightColorAlt</u>	1126
<u>schDynamicNetHighlightColorMemNet</u>	1127
<u>schDynamicNetHighlightColorPre</u>	1128
<u>schDynamicNetHighlightIslands</u>	1129
<u>schDynamicNetHighlightMemNetIslands</u>	1130
<u>schDynamicNetHighlightOn</u>	1131
<u>schFilterSize</u>	1132
<u>schGridMultiple</u>	1133
<u>schGridSpacing</u>	1134
<u>schGridType</u>	1135
<u>schMaxDragFig</u>	1136
<u>schVerboseFind</u>	1137
<u>schPathType</u>	1138
<u>schShowDirectConnects</u>	1139
<u>schShowLabelOrigin</u>	1140
<u>schSnapSpacing</u>	1141
<u>schWindowBBox</u>	1142
<u>schematicAreaPartialSelect</u>	1143
<u>schematicSelectAllDefault</u>	1144
<u>schematicSelectFilter</u>	1145
<u>sensitiveMenu</u>	1146
<u>shapeLineType</u>	1147

Virtuoso Schematic Editor User Guide

<u>shapeLineWidth</u>	1148
<u>snapAlignDistribute</u>	1149
<u>snapEnabled</u>	1150
<u>spanLib</u>	1151
<u>srcCheckFloatingSignals</u>	1152
<u>srcConnByName</u>	1153
<u>srcFixedGlobalsCheck</u>	1154
<u>srcFloatingBIDIR</u>	1155
<u>srcFloatingInput</u>	1156
<u>srcFloatingIO</u>	1157
<u>srcFloatingIterTermStub</u>	1158
<u>srcFloatingNets</u>	1159
<u>srcFloatingOutput</u>	1160
<u>srcInhImplicitTerm</u>	1161
<u>srcInhInstPinPinConn</u>	1162
<u>srcInhPinNetCollision</u>	1163
<u>srcInheritedConn</u>	1164
<u>srcInstNameExpr</u>	1165
<u>srcInstNameSyntax</u>	1166
<u>srcInstOverlap</u>	1167
<u>srcInstOverlapValue</u>	1168
<u>srcIteratedInsts</u>	1169
<u>srcLogWarnLimit</u>	1170
<u>srcMatchedParameters</u>	1171
<u>srcMaxLabelOffset</u>	1172
<u>srcNameCollision</u>	1173
<u>srcNetNameExpr</u>	1174
<u>srcNetNameSyntax</u>	1175
<u>srcNoOverrideNet</u>	1176
<u>srcOffsheetConnCheck</u>	1177
<u>srcPinDirConflictCheck</u>	1178
<u>srcPinNetCollision</u>	1179
<u>srcReportIncompleteChecks</u>	1180
<u>srcSamePortCheck</u>	1181
<u>srcShortedOutputGlobal</u>	1182
<u>srcShortedOutputs</u>	1183

Virtuoso Schematic Editor User Guide

<u>srcSigTypeIntegrity</u>	1184
<u>srcSolderOnCrossover</u>	1185
<u>srcSparseIndexBus</u>	1186
<u>srcSparseIndexBusOnlyTerms</u>	1187
<u>srcSysVerilogSyntax</u>	1188
<u>srcTermNameExpr</u>	1189
<u>srcTermNameSyntax</u>	1190
<u>srcUnconnectedWires</u>	1191
<u>srcUnnamedConnection</u>	1192
<u>srcVHDLSyntax</u>	1193
<u>srcVerilogSyntax</u>	1194
<u>ssgSortPins</u>	1195
<u>statusBanner</u>	1196
<u>statusBannerXY</u>	1197
<u>stickyWireLabelRange</u>	1198
<u>stubLength</u>	1199
<u>symCursorShape</u>	1200
<u>symDisplayDepth</u>	1201
<u>symDrawAxesOn</u>	1202
<u>symDynamicHighlightOn</u>	1203
<u>symGridMultiple</u>	1204
<u>symGridSpacing</u>	1205
<u>symGridType</u>	1206
<u>symMaxDragFig</u>	1207
<u>symPathType</u>	1208
<u>symShowDirectConnects</u>	1209
<u>symShowLabelOrigin</u>	1210
<u>symSnapSpacing</u>	1211
<u>symWindowBBox</u>	1212
<u>symbolAreaPartialSelect</u>	1213
<u>symbolLabelChoice</u>	1214
<u>symbolLabelChoiceText</u>	1215
<u>symbolLabelFontHeight</u>	1216
<u>symbolLabelFontStyle</u>	1217
<u>symbolLabelJustify</u>	1218
<u>symbolLabelType</u>	1219

Virtuoso Schematic Editor User Guide

<u>symbolPinFlatten</u>	1220
<u>symbolPinIncrCount</u>	1221
<u>symbolPinLocate</u>	1222
<u>symbolPinUsage</u>	1223
<u>symbolSelectAllDefault</u>	1224
<u>symbolSelectFilter</u>	1225
<u>symbolShape</u>	1226
<u>symbolShapeFillStyle</u>	1227
<u>transToSheetType</u>	1228
<u>tsgAnnotationsForModifiedPins</u>	1229
<u>tsgExcludeInhConnPins</u>	1230
<u>tsgExcludeInhConnPinsNames</u>	1231
<u>tsgTemplateType</u>	1232
<u>updateConn</u>	1233
<u>updateInstResetCDF</u>	1234
<u>vicCheckPinOrder</u>	1235
<u>vicExactPinMatch</u>	1236
<u>vicIgnoreInhTermAll</u>	1237
<u>vicIgnoreInhTermList</u>	1238
<u>vicIgnoreInhTermViewList</u>	1239
<u>vicIgnoreNonExistingTerms</u>	1240
<u>vicIgnoreViewsWithNoSetPinOrder</u>	1241
<u>vicInheritedPinMatch</u>	1242
<u>vicInheritedPins</u>	1243
<u>vicLogWarnLimit</u>	1244
<u>vicSeverity</u>	1245
<u>vicTermSigType</u>	1246
<u>vicTermSupplySens</u>	1247
<u>vicViewList</u>	1248
<u>viewNameList</u>	1249
<u>wireBackTrace</u>	1250
<u>zoomOutButton</u>	1251
<u>Photonics Environment Variables (Virtuoso Photonics Option)</u>	1252
<u>opticalNetColor</u>	1253
<u>opticalNetColoring</u>	1254
<u>opticalNetLineStyle</u>	1255

Virtuoso Schematic Editor User Guide

<u>opticalPinColor</u>	1256
<u>opticalPinColoring</u>	1257
<u>opticalPinFillColor</u>	1258
<u>opticalPinFillStyle</u>	1259
<u>opticalPinLineStyle</u>	1260
<u>opticalPinStipple</u>	1261
<u>opticalSigTypePropagation</u>	1262
<u>srcOpticalElectricalConnection</u>	1263
<u>srcOpticalMultiToSingle</u>	1264
<u>srcOpticalSingleToMulti</u>	1265
<u>srcOpticalTooManyConnections</u>	1266
<u>Navigator Environmental Variables</u>	1267
<u>autoResizeColumns</u>	1268
<u>compactBusDisplay</u>	1269
<u>defaultOpenIn</u>	1270
<u>defaultOpenMode</u>	1271
<u>fullSizeNameColumn</u>	1272
<u>showSummaryPage</u>	1273
<u>Net Tracing Environmental Variables</u>	1274
<u>alternateHierSeparator</u>	1275

B

<u>Symbol and Simulation Library Generator</u>	1277
<u>Library Command File</u>	1278
<u>Running the Symbol and Simulation Library Generator</u>	1279
<u>Running the Symbol and Simulation Library Generator in the Cadence Graphic Environment</u>	1279
<u>Running the Symbol and Simulation Library Generator in the Cadence Nongraphic Environment</u>	1280
<u>Variables That Control Command Execution</u>	1283
<u>Properties That Modify Values</u>	1283
<u>Sample Library Command File</u>	1284

C

<u>Text-to-Symbol Generator</u>	1289
<u>Symbol Shape Generated by TSG</u>	1290
<u>TSG Symbol Description File</u>	1291
<u>TSG Template File</u>	1292
<u>Clock and Negation Indicators</u>	1292
<u>Backannotation</u>	1293
<u>Simple File Example</u>	1293
<u>Complex File Example</u>	1294
<u>Extended File Example</u>	1296
<u>TSG Symbol Description File Structure</u>	1298
<u>Basic File Structure</u>	1298
<u>Syntax</u>	1299
<u>Arguments</u>	1300
<u>Executing TSG</u>	1307
<u>TSG Processing Order</u>	1308
<u>Creating Symbols Automatically from a Schematic</u>	1308

D

<u>Form Descriptions</u>	1309
<u>Add Custom Pin</u>	1310
<u>Add Generic – VHDL Properties</u>	1311
<u>Add Instance</u>	1312
<u>Add Kanji Note</u>	1314
<u>Add New Property for Search</u>	1316
<u>Add New Property to Find</u>	1317
<u>Add New Property to Find/Replace</u>	1318
<u>Add Property – Edit Object Properties</u>	1319
<u>Add Selection Box</u>	1320
<u>Add Symbol Label</u>	1321
<u>Add Symbol Shape</u>	1324
<u>Add Wire</u>	1326
<u>Annotation Setup</u>	1328
<u>Build Inherited View List</u>	1329

Virtuoso Schematic Editor User Guide

<u>Cellview From Cellview</u>	1330
<u>Cellview From Instance</u>	1331
<u>Cellview From Pin List</u>	1332
<u>Cellviews with Evaluated Net Expression Name</u>	1334
<u>Change Sheet Border Size</u>	1335
<u>Check Hierarchy</u>	1336
<u>Check Label Attachment</u>	1338
<u>Component Browser</u>	1339
<u>Component Browser Preferences</u>	1340
<u>Copy</u>	1341
<u>Create a Mapping Schematic Step 1 of 2</u>	1342
<u>Create a Mapping Schematic Step 2 of 2</u>	1344
<u>Create Block</u>	1346
<u>Create Cellview Dialog Box – Replace or Modify</u>	1347
<u>Create Net Expression</u>	1348
<u>Create Note Shape</u>	1350
<u>Create Note Text</u>	1353
<u>Create Patchcord</u>	1355
<u>Create Pin – Schematic</u>	1356
<u>Create Pin – Symbol</u>	1359
<u>Create Schematic</u>	1362
<u>Create Schematic Sheet</u>	1363
<u>Create Wire Name</u>	1364
<u>Cross Reference Options</u>	1366
<u>Cross-View Checker</u>	1368
<u>Delete All Markers - OpenAccess version</u>	1370
<u>Delete Generic – VHDL Properties</u>	1372
<u>Delete Marker</u>	1373
<u>Delete Property – Edit Object Properties</u>	1374
<u>Delete Schematic Sheet</u>	1375
<u>Descend Edit/Read</u>	1376
<u>Display Options</u>	1377
<u>Edit Cellview Properties</u>	1382
<u>Edit Object Properties – Basic</u>	1384
<u>Edit Object Properties – Instance and Block</u>	1386
<u>Edit Object Properties – Instance Pin</u>	1390

Virtuoso Schematic Editor User Guide

<u>Edit Object Properties – Label</u>	1392
<u>Edit Object Properties – Master Property</u>	1394
<u>Edit Object Properties – Net Expression</u>	1396
<u>Edit Object Properties – Note Shapes</u>	1398
<u>Edit Object Properties – Note Text</u>	1400
<u>Edit Object Properties – Other Object</u>	1402
<u>Edit Object Properties – Pin</u>	1404
<u>Edit Object Properties – Pin Name</u>	1407
<u>Edit Object Properties – Selection Box</u>	1409
<u>Edit Object Properties – Symbol Label</u>	1411
<u>Edit Object Properties – Symbol Pin</u>	1413
<u>Edit Object Properties – Symbol Shape</u>	1415
<u>Edit Object Properties – Wire Name</u>	1417
<u>Edit Object Properties – Wire Segment</u>	1419
<u>Edit Pin Order</u>	1421
<u>Edit Schematic Sheet Number</u>	1422
<u>Editor Options</u>	1423
<u>Export Schematic Hierarchy</u>	1429
<u>Find Marker</u>	1431
<u>Flatten</u>	1433
<u>Go To Pin</u>	1434
<u>Go To Sheet</u>	1435
<u>Import Symbol</u>	1436
<u>Left, Right, Top, Bottom Pin Attributes</u>	1437
<u>Load Label Display</u>	1438
<u>Load Schematic Defaults</u>	1439
<u>Make Cell</u>	1440
<u>Modify Generic – VHDL Properties</u>	1441
<u>Modify Property – Edit Object Properties</u>	1442
<u>Move</u>	1443
<u>Net Expression Available Property Names</u>	1444
<u>Net Expression Evaluated Names</u>	1446
<u>New File</u>	1447
<u>Occurrence Paths to Evaluated Name</u>	1448
<u>Open File</u>	1449
<u>Plot Options</u>	1450

Virtuoso Schematic Editor User Guide

<u>Register Net Name per Terminal</u>	1452
<u>Renumber Instances</u>	1453
<u>Reset Invisible Labels</u>	1455
<u>Rotate</u>	1456
<u>Route Flight Line</u>	1457
<u>Route Named Connection</u>	1458
<u>Save a Copy</u>	1459
<u>Save Label Display</u>	1460
<u>Save Schematic Defaults</u>	1461
<u>Save Schematics with Errors</u>	1462
<u>Schematic Check Options</u>	1463
<u>Schematic Find</u>	1467
<u>Schematic Replace</u>	1469
<u>Schematic Rules Checks Setup</u>	1472
<u>Schematic Select By Filter</u>	1479
<u>Schematic Select By Property</u>	1480
<u>Schematic Selection Filter</u>	1482
<u>Schematic Title Block Properties</u>	1483
<u>Set Instance Binding</u>	1484
<u>Set Label Display Simulation Data Directory</u>	1486
<u>Show Views Found</u>	1487
<u>Stretch – Schematic View</u>	1488
<u>Stretch – Symbol View</u>	1490
<u>Submit Plot</u>	1491
<u>Symbol Find</u>	1495
<u>Symbol Generation Options</u>	1497
<u>Symbol Replace</u>	1501
<u>Symbol Select By Filter</u>	1504
<u>Symbol Select By Property</u>	1505
<u>Symbol Selection Filter</u>	1506
<u>Update Pins</u>	1507
<u>VHDL Properties</u>	1508

E

<u>Configuring and Using Kanji Notes</u>	1509
<u>Configuring Your System</u>	1510
<u>Specifying the Kanji Editor</u>	1510
<u>Configuring the Kanji Menu Command</u>	1511
<u>Using Kanji Notes</u>	1512
<u>Formats, Files, and Views</u>	1512
<u>Adding a New Note</u>	1513
<u>Editing an Existing Note</u>	1516
<u>Limitations</u>	1519

F

<u>Bindkeys in Virtuoso Schematic Editor</u>	1521
<u>Introduction</u>	1521
<u>Arranged as a Keyboard Layout</u>	1522
<u>Arranged by Functions</u>	1523
<u>Related Topics</u>	1524

G

<u>Using opticalLib to Create PDK (Virtuoso Photonics Option)</u> .	1525

H

<u>Troubleshooting Schematic Connectivity</u>	1529
<u>Wire Naming</u>	1531
<u>SCH-7060, SCH-7071, SCH-7072, SCH-7088: Incorrect Wire Label Syntax</u>	1532
<u>SCH-7064: Conflicting Labels on the Same Wire</u>	1533
<u>SCH-7065: Syntax Error in Wire Label</u>	1534
<u>SCH-7067: Incorrect Alias Label Syntax</u>	1535
<u>SCH-7068: Alias Width Does Not Match Net Width</u>	1536
<u>SCH-7069 and SCH-7084: Alias Label Not Glued to a Wire</u>	1537
<u>SCH-7070 and SCH-7077: Unconnected Wire Labels and Net Expression</u>	1538
<u>SCH-7078: Base Name Already in Use by Another Net</u>	1539

Virtuoso Schematic Editor User Guide

<u>SCH-7902: System-generated Net Prefix Cannot Be Used</u>	1540
<u>SCH-7903: Unexpected Format for Wire Label Glue Distance Setting</u>	1541
<u>Shorts and Misconnections</u>	1542
<u>SCH-7003: Short between Nets</u>	1543
<u>SCH-7005 to 7008: Shorted Signals across a Feed-through Pin</u>	1544
<u>Incompatible Connection Widths</u>	1547
<u>SCH-7004 and SCH-7080: Misconnection between Unrelated Nets of Different Widths</u>	1547
<u>Tapping and Repeating</u>	1549
<u>SCH-7020 to SCH-7035, SCH-7050 and SCH-7051: Tapped Bits Out of Range</u>	1550
<u>SCH-7061: Unexpected Repeat Label</u>	1553
<u>SCH-7062: Unexpected Tap Label</u>	1554
<u>SCH-7116: Ambiguous Connection at Tap</u>	1555
<u>Inherited Connections</u>	1556
<u>SCH-7015: Conflicting Net Expressions on the Same Net</u>	1557
<u>SCH-7074 and SCH-7075: Conflict between Net Expression and Wire Label</u>	1558
<u>SCH-7079 to SCH-7087: Incomplete Net Expression Label</u>	1559
<u>SCH-7111 to SCH-7115: Circular References</u>	1560
<u>SCH-7280: Non-inherited Pin with Net Expression</u>	1562
<u>SCH-7281: Global Schematic Pin Name Inconsistent with Net Expression Default</u>	1563
<u>SCH-7282 to SCH-7284: Net Expression with Non-global Default Net</u>	1564
<u>Patch Cords</u>	1565
<u>SCH-7090 to SCH-7092: Incorrectly Constructed Patch Cord Symbol</u>	1566
<u>SCH-7093 to SCH-7094: Incorrectly Connected Patch Cord Instance</u>	1567
<u>SCH-7095 to SCH-7102: Problems with Patch Cord Connection Expressions</u>	1568
<u>SCH-7103 to SCH-7110 and SCH-7286 Ripper Property Issues</u>	1570
<u>Indirect Shorts between Global Signals</u>	1572
<u>SCH-7200 to SCH-7203: Shorted Global Signals through a Patch Cord</u>	1573
<u>SCH-7220 to SCH-7223: Shorted Global Signals through an Alias Label</u>	1575
<u>SCH-7240 to SCH-7242 and SCH -7260: Shorted Globals across a Feed-through Pin</u>	1577
<u>Indirect Shorts between Pins</u>	1578
<u>SCH-7204 to SCH-7207: Shorted Pins through Patch Cord</u>	1579
<u>SCH-7224 to SCH-7227: Shorted Pins through Alias Label</u>	1581
<u>SCH-7243 to SCH-7245: Shorted Pins across a Feed-through Pin</u>	1582
<u>SCH-7270: Shorted Pins on Different Parts of the Same Net</u>	1583

Virtuoso Schematic Editor User Guide

<u>Indirect Shorts between Pins and Global Signals</u>	1584
<u>SCH-7208 to SCH-7215: Shorted Pin and Global through Patch Cord</u>	1585
<u>SCH-7228 to SCH-7235: Shorted Pin and Global through Alias Label</u>	1587
<u>SCH-7246 to SCH-7251: Shorted Pin and Global across a Feed-through Pin</u> ...	1589
<u>SCH-7271: Non-global Pin Connected to Global Net</u>	1591
<u>SCH-7299: Short between Global Nets or Pins</u>	1592
<u>Unexpected Schematic Content</u>	1593
<u>SCH-7066: Unexpected Missing Instance Terminal</u>	1594
<u>SCH-7285: Unexpected Pins on Sheet Border Instance</u>	1595
<u>SCH-7286: Unsupported Ripper Configuration</u>	1596
<u>SCH-7287: No-connection Annotation with Global Pin</u>	1597
<u>SCH-7288: No-connection Annotation with Net Expression</u>	1598
<u>SCH-7289: No-connection Annotation is Not Ignored by Netlister</u>	1599
<u>Other Errors</u>	1600
<u>Ungrouped Messages</u>	1601
<u>Finding the Range of Bits in a Bus</u>	1602
 <u>Glossary</u>	 1603

Getting Started

The Virtuoso® Schematic Editor L and XL are design entry tools that support the work of logic and circuit design engineers, including drafters. Physical layout designers and printed circuit board designers can use the information as background material to support their work.

This user guide describes the functionality of the L and XL versions of the schematic editor:

- L version: describes the core schematic editing functionality and features that are available in the L version of the schematic editor, with additional reference to the Virtuoso® Symbol Editor.
- XL version: describes schematic functionality and features that are available in the XL version of the schematic editor and are contained within a single chapter.

This chapter discusses the following topics:

- [Starting the Virtuoso Schematic Editor L/XL](#)
- [Accessing Virtuoso Schematic Editor L/XL](#)
- [Quick Reference - Menus and Bindkeys](#)
- [Quick Reference - Menu Access Keys](#)
- [Quick Reference - Toolbars](#)
- [Workspaces and Assistants in Schematics L/XL](#)
- [Additional Features of the Schematic Editor XL](#)
- [Virtuoso ADE Integration in Virtuoso Schematic Editor](#)
- [Photonics in Virtuoso Schematic Editor, ADE, and Layout XL \(Virtuoso Photonics Option\)](#)
- [Help and Support Facilities](#)
- [Working with Designs](#)
- [Working with Commands](#)

- [Selecting Objects](#)
- [Working With the Design Canvas Window](#)
- [The Component Browser](#)
- [Additional Features of the Schematic Editor XL](#)

Starting the Virtuoso Schematic Editor L/XL

To start the Virtuoso Schematic Editor L you must run the `virtuoso` executable:

1. At a UNIX prompt in an xterm window type:

```
virtuoso &
```

Note: The ampersand (&) starts the software in the background so you can use the xterm window for other tasks while the software is running.

2. Press the `Return` key.

The Command Interpreter Window (CIW) appears.

For details on the CIW, refer to [Virtuoso Design Environment User Guide](#).

Accessing Virtuoso Schematic Editor L/XL

You can launch *Schematics L/XL* by [Opening a New Cellview](#) or [Opening an Existing Cellview](#).

Note: Access to *Schematics L* and *Schematics XL* will require the correct license. For more information see [Licensing Requirements](#).

Setting Schematics L or XL as the Default Application for Viewing Schematics

You can set *Schematics L* or *Schematics XL* to be your default application by selecting *File – Set Default Application* and ensuring that the application required is set as the default application when viewing schematics.

For more information on setting your default application see [Setting the Default Application for a Cellview](#) in the [Virtuoso Design Environment User Guide](#).

Quick Reference - Menus and Bindkeys

Bindkeys are used to tie commonly-used commands to single keys, key combinations, or mouse sequences. You can use the bindkeys to automate repetitive tasks and speed up your work.

For information on the default bindkeys defined for VSE L, refer to [Bindkeys in Virtuoso Schematic Editor](#).

Note: For more information on bindkeys, see [Bindkey Storage](#) and [Configuring Application Bindkeys](#).

Virtuoso Schematic Editor User Guide

Getting Started

Menu Charts for the Virtuoso Schematic Editor

Launch	File	Edit	
☒ ADE Explorer	☒ New	☒ Undo (U)	☒ Properties
☒ ADE Assembler	☒ Open	☒ Redo (Shift+U)	☐ Objects (Q)
☒ Layout XL	☒ Open Symbol	☒ Move (Shift+M)	☐ Cellview (Shift+Q)
☒ Layout EXL	☒ Close	☒ Copy (C)	☐ Pin Order
☒ Layout MXL	☒ Save	☒ Stretch (M)	☐ VHDL
☒ Layout SiP with Template	☒ Save a Copy (Ctrl+S)	☒ Delete (Del)	☐ Propagate Pin Signal Types
☒ Layout SiP	☒ Discard Edits	☒ Ignore Instances	☐ Register As Special Cell
☒ Schematics L	☒ Check and Save (Shift+X)	☒ Rotate	☒ Net Expression
☒ Schematics XL	☒ Make Read Only / Editable	☒ Align	☐ Available Properties
☒ Pcell IDE	☒ Export HTML (grayed out in VSE L)	☐ Align Left	☐ Evaluated Names
☒ Plugins	☒ Export Image	☐ Align Vertical	☒ Select
☐ Debug CDF	☒ Print	☐ Align Right	☐ Select All
☐ EAD	☒ Print Status	☐ Align Top	☐ Deselect All
☐ Hierarchy Editor	☒ Bookmarks	☐ Align Horizontal	☐ Select By Area
☐ Migrate	☐ Add Bookmark	☐ Align Bottom	☐ Select By Line
☐ Mixed Signal Options	☐ Manage Bookmarks	☐ Snap to Grid	☐ Select By Filter
☐ AMS	☒ Recently viewed files	☒ Distribute	☐ Select By Property
☐ Configuration	☐ Set Default Application	☐ Distribute Vertical	☐ Filter (Ctrl+F)
☐ SimVision	☒ Close All	☐ Distribute Horizontal	☒ Route Flight (5)
☐ Parasitics	- For information on opening and closing views see Working with Cellviews in the Virtuoso Design Environment User Guide .	☒ Text	☒ Renumber Instances
☐ Simulation		☒ Increase Text Size	☒ Component Display
☐ NC-Verilog		☒ Decrease Text Size	☒ Alternate View
☐ Other		☒ Sheet Size	☒ Update Pins From View
☐ Spectre		☒ Sheet Title	☒ Find
☐ SystemVerilog		☒ Hierarchy	☒ Replace
☐ UltraSim	- For information on opening and closing views see Using Bookmarks and Views in the Virtuoso Design Environment User Guide .	☐ Descend Edit (Shift+E)	
☐ VHDL Toolbox		☐ Descend Read (E)	
		☐ Edit In Place	
		☐ Show Scope	
		☐ Return (Ctrl+E)	
		☐ Return To Top	
		☐ Print Tree (Shift+T)	
		☐ Make Cell (J)	
		☐ Flatten (Shift+J)	

Virtuoso Schematic Editor User Guide

Getting Started

View

- ☒ Zoom In (I)
- ☒ Zoom Out (O)
- ☒ Zoom To Area (Z)
- ☒ Zoom To Selected (Ctrl+T)
- ☒ Zoom To Fit
- ☒ Pan
- ☒ Redraw (F6)
- ☒ Reset Invisible Labels
- ☒ Show/Hide Instance Labels
- ☒ Show/Hide Notes
- ☒ Show/Hide Terminal Labels
- ☒ Show/Hide Parasitic Notes
- ☒ Annotations
 - ☐ Design Defaults
 - ☐ Net Names
 - ☐ Pin Names
 - ☐ DC Voltages
 - ☐ DC Currents
 - ☐ Transient Voltages
 - ☐ Transient Currents
 - ☐ DC Voltages, Currents
 - ☐ Transient Voltages, Currents
 - ☐ Component Parameters
 - ☐ Model Parameters
 - ☐ DC Operating Points
 - ☐ Transient Operating Points
 - ☐ Setup...
- ☒ Dynamic Zooming
- ☒ Net Highlighting
- ☒ Display Dimming
- ☒ Magnifier
- ☒ Info Balloons
- ☒ Save/Restore
 - ☐ Previous View (Shift+Z)
 - ☐ Next View (Ctrl+Shift+Z)
 - ☐ Save View
 - ☐ Restore View

- For information on saving and restoring views, see [Using Bookmarks and Views](#) in the *Virtuoso Design Environment User Guide*.

Create

- ☒ Instance (I)
- ☒ Wire (narrow) (W)
- ☒ Wire (wide) (Shift+W)
- ☒ Wire Name (L)
- ☒ Wire Stubs and Names (Space)
- ☒ Net Expression
- ☒ Pin (P)
- ☒ Block (B)
- ☒ Mapping Schematic
- ☒ Cellview
 - ☐ From Cellview
 - ☐ From Pin List
 - ☐ From Instance
- ☒ Solder Dot
- ☒ Note
 - ☐ Text (Shift+N)
 - ☐ Shape (N)
 - ☐ *Add Constraint Notes
 - ☐ *Remove Constraint Notes
 - ☐ *Update Constraint Notes
 - ☐ *Add All Constraint Notes in Hierarchy
 - ☐ *Remove All Constraint Notes in Hierarchy
 - ☐ *Update All Constraint Notes in Hierarchy
- ☒ Patchcord
- ☒ Probe
 - ☐ Add Net (9)
 - ☐ Create Pin
 - ☐ Add Instance
 - ☐ Explain
 - ☐ Remove Net
 - ☐ Remove Pin
 - ☐ Remove Instance
 - ☐ Remove All (Shift+9)
 - ☐ Load
 - ☐ Save
- ☒ **MultiSheet

Note: *Grayed out in VSE L. See [Creating Constraint Notes](#) in the *Virtuoso Unified Custom Constraints User Guide*.

Note: **Creates a **Sheet** menu with: Go To, Cross Reference, Go To Pin, Delete, Rename, and Resequencing menu options.

Check

- ☒ Current Cellview (X)
- ☒ Hierarchy
- ☒ Cross-View
- ☒ Label Attachment
- ☒ Rules Setup
- ☒ Find Marker (G)
- ☒ Delete Marker (Ctrl+G)
- ☒ Delete All Markers

Options

- ☒ Editor (Shift+O)
- ☒ Display (O)
- ☒ Magnifier (')
- ☒ Register Net Name per Terminal
- ☒ Select Filter (Ctrl+F)
- ☒ Check
- ☒ Tool Filter
- ☒ Save Defaults
- ☒ Load Defaults

Window

- ☒ Assistants
 - ☐ *Annotation Browser
 - ☐ Probes
 - ☐ Property Editor
 - ☐ Search
 - ☐ Navigator
 - ☐ World View
 - ☐ Show All
 - ☐ Hide All
- ☒ Workspaces
 - ☐ Basic
 - ☐ Classic
 - ☐ Explore
 - ☐ Save As
 - ☐ Delete
 - ☐ Load
 - ☐ Set Default
 - ☐ Revert to Saved
 - ☐ Show/Hide Assistant (F11)
- ☒ Toolbars
 - ☐ File
 - ☐ Edit
 - ☐ View
 - ☐ Create
 - ☐ Search
 - ☐ Select
 - ☐ Go
 - ☐ Bookmarks
 - ☐ Show All
 - ☐ Hide All
 - ☐ Customize
- ☒ Tabs
 - ☐ Open tab list
 - ☐ Close Current Tab
 - ☐ Close Other Tabs
- ☒ Copy Window

*- For more information on the Annotation Browser, see the [Annotation Browser](#) in the *Virtuoso Layout Suite XL User Guide*.

- For more information on window components see the [Virtuoso Design Environment User Guide](#).

Virtuoso Schematic Editor User Guide

Getting Started

RF-Module (Virtuoso RF Solution Option)

- Assign Project
- Edit Project
- Schematic to SiP Layout
- Launch SiP Layout
- Update Schematic from SiP Layout
- Schematic Vs SiP Layout
- Reports
 - ☐ Bill Of Materials
 - ☐ Open Physical Nets
 - ☐ Unplaced Components
 - ☐ Design Rules Check
 - ☐ Short Physical Nets
- Logs
 - ☐ Chips
 - ☐ Property
 - ☐ Net
 - ☐ Pin
 - ☐ Property Conflict
 - ☐ Backannotation
 - ☐ Schematic Package
 - ☐ SiP Creation
 - ☐ Property Backannotation
 - ☐ SiP Netlisting
 - ☐ Verilog Netlister
- SiP Implementation Utilities
 - ☐ SiP Package Connectivity Import
 - ☐ Place Die with Connectivity
 - ☐ Archiver
 - ☐ Convert DE-HDL Library to Virtuoso
 - ☐ Uprev Entity to 64-bit...
 - ☐ Write CDF from PTF...
 - ☐ Library Verification...
 - ☐ Add Simulation Views...

- Import Allegro Layout
- Create Extracted View
- Annotate from Extracted View
- Allegro Analysis Utilities
 - ☐ Display Component Selector
 - ☐ Place Package
 - ☐ Route All Flight Lines
 - ☐ Create Test-bench Symbol
 - ☐ Bind to Schematic Hierarchy
 - ☐ Bind to Subckt Hierarchy
 - ☐ Layout vs Extracted View
 - ☐ Schematic vs Extracted View
- Transmission Lines
 - ☐ Create Stackup
 - ☐ Create Net Topology
 - ☐ Inflate

- For more information on the RF-Module menu options, see the [Virtuoso RF Solution Guide](#).

Help

- Schematics L User Guide
- Schematics L What's New
- Schematics L KPNS
- Virtuoso Documentation Library
- Virtuoso Video Library
- Virtuoso Rapid Adoption Kits
- Virtuoso Learning Map
- Virtuoso Custom IC Community
- Cadence Online Support
- Cadence Training
- Cadence Community
- Cadence OS Platform Support
- Contact Us
- Cadence Home
- About Virtuoso

- For more information on window components, such as Workspaces and Toolbars, see the [Virtuoso Design Environment User Guide](#).

Virtuoso Schematic Editor User Guide

Getting Started

Menu Charts for the Virtuoso Symbol Editor

Launch

- Symbol L
- Symbol XL
- Pcell IDE
- Plugins
 - ☐ Debug CDF

File

- New
- Open
- Open Schematic
- Close
- Save
- Save a Copy (Ctrl+S)
- Discard Edits
- Check and Save (Shift+X)
- Make Read Only / Editable
- Export HTML (grayed out in VSE L)
- Export Image
- Print
- Print Status
- Bookmarkst
- *Recently viewed files*
- Set Default Application
- Close All

Edit

- Undo (U)
- Redo (Shift+U)
- Move (Shift+M)
- Copy (C)
- Stretch (M)
- Delete (Del)
- Ignore Instances
- Rotate
- Align
 - ☐ Align Left
 - ☐ Align Vertical
 - ☐ Align Right
 - ☐ Align Top
 - ☐ Align Horizontal
 - ☐ Align Bottom
 - ☐ Snap to Grid
- Distribute
 - ☐ Distribute Vertical
 - ☐ Distribute Horizontal
- Text
- Increase Text Size
- Decrease Text Size
- Hierarchy
 - ☐ Descend Edit
 - ☐ Descend Read
 - ☐ Edit In Place
 - ☐ Show Scope
 - ☐ Return (Ctrl+E)
 - ☐ Return To Top
 - ☐ Print Tree
- Properties
 - ☐ Objects (Q)
 - ☐ Cellview (Shift+Q)
 - ☐ Pin Order
 - ☐ VHDL

View

- Zoom In (I)
- Zoom Out (O)
- Zoom To Area (Z)
- Zoom To Selected (Ctrl+Z)
- Zoom To Fit (F)
- Pan
- Redraw (F6)
- Reset Invisible Labels
- Show/Hide Instance Labels
- Show/Hide Notes
- Show/Hide Terminal Labels
- Show/Hide Parasitic Notes
- Annotations
 - ☐ Design Defaults
 - ☐ Net Names
 - ☐ Pin Names
 - ☐ DC Voltages
 - ☐ DC Currents
 - ☐ Transient Voltages
 - ☐ Transient Currents
 - ☐ DC Voltages, Currents
 - ☐ Transient Voltages, Currents
 - ☐ Component Parameters
 - ☐ Model Parameters
 - ☐ DC Operating Points
 - ☐ Transient Operating Points
 - ☐ Setup...
- Dynamic Zooming (grayed out)
- Net Highlighting (grayed out)
- Display Dimming
- Magnifier
- Info Balloons

Virtuoso Schematic Editor User Guide

Getting Started

- For information on opening and closing views see [Working with Cellviews](#) in the *[Virtuoso Design Environment User Guide](#)*.

- For information on bookmarks see [Using Bookmarks and Views](#) in the *[Virtuoso Design Environment User Guide](#)*.

- Select
 - ☐ Select All
 - ☐ Deselect All
 - ☐ Select By Area
 - ☐ Select By Line
 - ☐ Select By Filter
 - ☐ Select By Property
 - ☐ Filter (Ctrl+F)

- Origin
- Update Pins From View
- Find
- Replace

- Save/Restore
 - ☐ Previous View (Shift+Z)
 - ☐ Next View (Ctrl+Shift+Z)
 - ☐ Save View
 - ☐ Restore View

- For information on saving and restoring views, see [Using Bookmarks and Views](#) in the *[Virtuoso Design Environment User Guide](#)*.

Virtuoso Schematic Editor User Guide

Getting Started

Create

- Shape
 - ☐ Line
 - ☐ Rectangle
 - ☐ Polygon
 - ☐ Circle
 - ☐ Ellipse
 - ☐ Arc
- Pin (P)
- Label (L)
- Selection Box
- Cellview
 - ☐ From Cellview
 - ☐ From Pin List
 - ☐ From Instance (grayed out)
- Note
 - ☐ Text (Shift+N)
 - ☐ Shape (N)
 - ☐ Kanji (Shift+K)
- Net Expression
- Custom Pin
- Import Symbol
- Probe
 - ☐ Add Net (9)
 - ☐ Create Pin
 - ☐ Explain
 - ☐ Remove Net
 - ☐ Remove Pin
 - ☐ Remove All
 - ☐ Load
 - ☐ Save

Check

- Current Cellview
- Cross-View (X)
- Rules Setup
- Find Marker (G)
- Delete Marker (Ctrl+G)
- Delete All Markers

Options

- Editor (Shift+O)
- Display (O)
- Select Filter (Ctrl+F)
- Check
- Tool Filter
- Save Defaults
- Load Defaults

Window

- Assistants
 - ☐ Annotation Browser
 - ☐ Circuit Prospector
 - ☐ Constraint Manager
 - ☐ Probes
 - ☐ Property Editor
 - ☐ Search
 - ☐ Navigator
 - ☐ World View
 - ☐ Show All
 - ☐ Hide All
- Workspaces
 - ☐ Basic
 - ☐ Classic
 - ☐ Constraints
 - ☐ Constraints-Helper
 - ☐ Design-Intent
 - ☐ EAD
 - ☐ Explore
 - ☐ Power
 - ☐ RF (Virtuoso RF Solution Option)
 - ☐ Save As
 - ☐ Delete
 - ☐ Load
 - ☐ Set Default
 - ☐ Revert to Saved
 - ☐ Show/Hide Assistants (F11)
 - ☐ Hide All
 - ☐ Customize...

Toolbars

- ☐ File
- ☐ Edit
- ☐ View
- ☐ Create
- ☐ Search
- ☐ Select
- ☐ Design Intent
- ☐ RF (Virtuoso RF Solution Option)
- ☐ Bookmarks
- ☐ Go
- ☐ Workspaces
- ☐ Show All
- Tabs
 - ☐ Close Current Tab
 - ☐ Close Other Tabs
- Copy Window

Virtuoso Schematic Editor User Guide

Getting Started

Help

- Schematics L User Guide
- Schematics L What's New
- Schematics L KPNS
- Virtuoso Documentation Library
- Virtuoso Video Library
- Virtuoso Rapid Adoption Kits
- Virtuoso Learning Map
- Virtuoso Custom IC Community
- Cadence Online Support
- Cadence Training
- Cadence Community
- Cadence OS Platform Support
- Contact Us
- Cadence Home
- About Virtuoso

- For more information on window components, such as Workspaces and Toolbars, see the *Virtuoso Design Environment User Guide*.

Bindkey Storage

Cadence-provided bindkeys, which are loaded by default, are located in the following directory:

```
<cds_install>/share/cdssetup/dfII/bindkeys/
```

Here, the following definition files are available, and loaded, when the respective application is launched:

- `Symbol.il`
- `Schematics.il`
- `TextEditor.il`
- `Layout.il`

You can control the loading of default bindkeys through the use of the following environment variable:

```
schematic autoLoadBindKeys boolean t/nil
```

Customizing Bindkeys

You can set new bindkeys, or override default bindkeys, by creating a corresponding SKILL (.il) file, and placing it in the search path that is defined by the default `setup.loc` file and/or the CSF mechanism.

The following locations are recommended for customization at the design level, user level, project level, or company level, respectively:

- `./.cadence/dfII/bindkeys/`
- `~/.cadence/dfII/bindkeys/`
- `$CDS_PROJECT/.cadence/dfII/bindkeys/`
- `$CDS_SITE/.cadence/dfII/bindkeys/`

Note: The bindkey files are searched for in each of those defined locations. The files found will then be loaded in reverse order (for example, using the above example, `$CDS_SITE` will be loaded first, then `$CDS_PROJECT`, and so on).

Virtuoso Schematic Editor User Guide

Getting Started

For more information, see:

- Specifying CSF Search for Additional Files and Cadence Setup Search File: setup.loc in the *Cadence Application Infrastructure User Guide*
- Bindkeys and Access Keys in the *Virtuoso Design Environment User Guide*
- hiSetBindKey in the *Cadence User Interface SKILL Reference*

Quick Reference - Menu Access Keys

Menu access keys provide keyboard access to functionality and application menus without the need to use mouse selections.

For example, selecting the `Alt + F` access keys together will display the contents of the *File* banner menu.

Menu access keys should not be confused with bindkeys, for example those detailed in the [Quick Reference - Menus and Bindkeys](#), as bindkeys do not require a menu to be displayed before a bindkey “shortcut” can be used.

For example, if you want to open the Editor Options form directly you can use the “O” (`Shift + O`) bindkey. It is not necessary to use the *Edit* menu access keys (`Alt + E`) to first of all display the contents of the *Edit* menu.

Note: When you have displayed a menu’s content further menu access keys may be used to run any of the sub-menu options listed.



When a menu’s content is on display you cannot use bindkeys.

Once you have used access keys to display the contents of the *Launch* menu you can then use a further access key to select an application menu to be added to the banner menu. After an application menu has been added to the banner menu you can then use their associated access keys to display their menu content.

For example, to display the content of the *AMS* menu on the banner menu you would:

1. Select the access keys `Alt + L` (this displays the content of the *Launch* menu).
2. Select the access key `R` (this adds the *Parasitics* menu to the banner menu)
3. Select the access keys `Alt + R` (this displays the content of the *Parasitics* menu on the banner menu).

Virtuoso Schematic Editor User Guide

Getting Started

Access Keys - Schematic Editor L

The following table details the menu access keys for the Virtuoso Schematic Editor L:

Menu/Application to be Added or Displayed	Add Access Key	Display Access Key
Launch		Alt + L
<i>ADE Explorer</i>	E	
<i>ADE Assembler</i>	A	
<i>Layout XL</i>	Y	
<i>Layout GXL</i>	G	
<i>Layout EXL</i>	E	
<i>Schematics L</i>	L	
<i>Schematics XL</i>	X	
<i>Pcell IDE</i>	P	Alt + P
<i>Plugins</i>	P	
☐ <i>Debug CDF</i>		
☐ <i>Diva</i>	I	Alt + I
☐ <i>Hierarchy Editor</i>	H	Alt + D
☐ <i>Mixed Signal Options</i>	M	Alt + X
☐ <i>AMS</i>	A	
☐ <i>Parasitics</i>	R	Alt + R
☐ <i>Simulation</i>	S	Alt + S
☐ <i>NC-Verilog</i>	N	
☐ <i>Other</i>	O	
☐ <i>Spectre</i>	S	
☐ <i>SystemVerilog</i>		
☐ <i>UltraSim</i>	U	
☐ <i>VHDL Toolbox</i>	T	
Note: All of the above require you to select Alt + L to first of all display the <i>Launch</i> menu.		
File		Alt + F
<i>New</i>	N	
<i>Open</i>	O	

Virtuoso Schematic Editor User Guide

Getting Started

<i>Open Symbol</i>	E	
<i>Close</i>	C	
<i>Save</i>	S	
<i>Save a Copy</i>	A	
<i>Discard Edits</i>	T	
<i>Check and Save</i>	K	
<i>Make Read Only / Make Editable</i>	M	
<i>Export HTML</i>	H	
<i>Export Image</i>	X	
<i>Print</i>	P	
<i>Bookmarks</i>	B	
○ <i>Add Bookmarks</i>	A	
○ <i>Manage Bookmarks</i>	M	
<i>Print Status</i>	I	
<i>Set Default Application</i>	D	
<i>Close All</i>	L	
Edit		Alt + E
<i>Undo</i>	U	
<i>Redo</i>	R	
<i>Move</i>	M	
<i>Copy</i>	C	
<i>Stretch</i>	S	
<i>Delete</i>	D	
<i>Ignore Instances</i>		Shift+Del
<i>Rotate</i>	O	
<i>Align</i>		
○ <i>Align Left</i>	L	
○ <i>Align Vertical</i>	V	
○ <i>Align Right</i>	R	
○ <i>Align Top</i>	T	
○ <i>Align Horizontal</i>	H	
○ <i>Align Bottom</i>	B	
○ <i>Snap to Grid</i>	G	
<i>Distribute</i>		

Virtuoso Schematic Editor User Guide

Getting Started

☐ <i>Distribute Vertical</i>	V
☐ <i>Distribute Horizontal</i>	H
<i>Text</i>	T
<i>Increase Text Size</i>	
<i>Decrease Text Size</i>	
<i>Sheet Size</i>	Z
<i>Sheet Title</i>	T
<i>Hierarchy</i>	H
☐ <i>Descend Edit</i>	E
☐ <i>Descend Read</i>	R
☐ <i>Edit In Place</i>	P
☐ <i>Show Scope</i>	S
☐ <i>Return</i>	U
☐ <i>Return To Top</i>	T
☐ <i>Print Tree</i>	N
<i>Properties</i>	P
☐ <i>Objects</i>	O
☐ <i>Cellview</i>	C
☐ <i>Pin Order</i>	P
☐ <i>VHDL</i>	H
<i>Net Expression</i>	X
☐ <i>Available Properties</i>	P
☐ <i>Evaluated Names</i>	N
<i>Select</i>	
☐ <i>Select All</i>	Ctrl + A
☐ <i>Deselect All</i>	Ctrl + D
☐ <i>Select By Area</i>	Shift + A
☐ <i>Select By Line</i>	Shift + Ctrl + A
☐ <i>Select By Filter</i>	
☐ <i>Select By Property</i>	
☐ <i>Filter (Ctrl+F)</i>	Ctrl + F
<i>Route Flight</i>	G
<i>Renumber Instances</i>	I
<i>Component Display</i>	N

Virtuoso Schematic Editor User Guide

Getting Started

<i>Alternative View</i>	A	
<i>Update Pins From View</i>	V	
<i>Find</i>	F	
<i>Replace</i>	E	
View		Alt + V
<i>Zoom In</i>	I	
<i>Zoom Out</i>	O	
<i>Zoom To Area</i>	Z	
<i>Zoom To Selected</i>	S	
<i>Zoom To Fit</i>	F	
<i>Pan</i>	P	
<i>Redraw</i>	R	
<i>Reset Invisible Labels</i>	I	
<i>Show/Hide Instance Labels</i>	B	
<i>Show/Hide Notes</i>	O	
<i>Show/Hide Terminal Labels</i>	E	
<i>Show/Hide Parasitic Notes</i>	A	
<i>Dynamic Zooming</i>	D	
<i>Net Highlighting</i>	N	
<i>Display Dimming</i>	G	
<i>Magnifier</i>	M	
<i>Save/Restore</i>	V	
○ <i>Previous View</i>	P	
○ <i>Next View</i>	N	
○ <i>Save View</i>	S	
○ <i>Restore View</i>	R	
Create		Alt + C
<i>Instance</i>	I	
<i>Wire (Narrow)</i>	W	
<i>Wire (Wide)</i>	D	
<i>Wire Name</i>	M	
<i>Wire Stubs and Names</i>	E	
<i>Net Expression</i>	X	
<i>Pin</i>	P	

Virtuoso Schematic Editor User Guide

Getting Started

<i>Block</i>	B	
<i>Mapping Schematic</i>	G	
<i>Cellview</i>	C	
○ <i>From Cellview</i>	C	
○ <i>From Pin List</i>	P	
○ <i>From Instance</i>	I	
<i>Solder Dot</i>	S	
<i>Note</i>	N	
○ <i>Text</i>	T	
○ <i>Shape</i>	S	
○ <i>Kanji</i>	K	
○ <i>Add Constraint Notes</i>	A	
○ <i>Remove Constraint Notes</i>	R	
○ <i>Update Constraint Notes</i>	U	
<i>Patchcord</i>	T	
<i>Probe</i>	O	
○ <i>Add Net</i>	N	
○ <i>Create Pin</i>	P	
○ <i>Add Instance</i>	I	
○ <i>Explain</i>	X	
○ <i>Remove Net</i>	R	
○ <i>Remove Pin</i>	M	
○ <i>Remove Instance</i>	V	
○ <i>Remove All</i>	A	
○ <i>Load</i>	L	
○ <i>Save</i>	S	
<i>MultiSheet</i>	U	
Check		Alt + K
<i>Current Cellview</i>	C	
<i>Hierarchy</i>	H	
<i>Cross-View</i>	V	
<i>Label Attachment</i>	L	
<i>Rules Setup</i>	R	
<i>Find Marker</i>	F	

Virtuoso Schematic Editor User Guide

Getting Started

<i>Delete Marker</i>	M	
<i>Delete All Markers</i>	A	
Sheet (<i>displayable after selecting Create – MultiSheet</i>)		Alt + S
<i>Go To</i>	G	
<i>Cross Reference</i>	R	
<i>Go To Pin</i>	P	
<i>Delete</i>	D	
<i>Renumber</i>	N	
<i>Resequence</i>	S	
Options		Alt + P
<i>Editor</i>	E	
<i>Display</i>	D	
<i>Register Net Name per Terminals</i>	R	
<i>Select Filter</i>	I	
<i>Check</i>	C	
<i>Tool Filter</i>	T	
<i>Save Defaults</i>	S	
<i>Load Defaults</i>	L	
Window		Alt + W
<i>Assistants</i>	A	
☐ <i>Annotation Browser</i>		
☐ <i>Property Editor</i>		
☐ <i>Search</i>		
☐ <i>Navigator</i>		
☐ <i>World View</i>		
☐ <i>Show All</i>	A	
☐ <i>Hide All</i>	H	
<i>Workspaces</i>	W	
☐ <i>Basic</i>		
☐ <i>Classic</i>		
☐ <i>Explore</i>		
☐ <i>Save As</i>	S	
☐ <i>Delete</i>	D	
☐ <i>Load</i>	L	

Virtuoso Schematic Editor User Guide

Getting Started

☐	<i>Set Default</i>	E	
☐	<i>Revert to Saved</i>	R	
☐	<i>Show/Hide Assistants</i>	H	
<i>Toolbars</i>		T	
☐	<i>File</i>		
☐	<i>Edit</i>		
☐	<i>View</i>		
☐	<i>Create</i>		
☐	<i>Search</i>		
☐	<i>Select</i>		
☐	<i>Go</i>		
☐	<i>Bookmarks</i>		
☐	<i>Workspaces</i>		
☐	<i>Show All</i>	A	
☐	<i>Hide All</i>	H	
<i>Tabs</i>		B	
☐	<i>First cellview</i>	1	
☐	<i>Second cellview</i>	2	
☐	<i>etc.</i>	3	
☐	<i>Close Current Tab</i>	C	
☐	<i>Close Other Tabs</i>	O	
<i>Copy Window</i>		C	
Help			Alt + H
	<i>Schematics L User Guide</i>	G	
	<i>Schematics L What's New</i>	N	
	<i>Schematics L KPNS</i>	K	
	<i>Virtuoso Documentation Library</i>	D	
	<i>Virtuoso Video Library</i>	V	
	<i>Virtuoso Rapid Adoption Kits</i>	R	
	<i>Virtuoso Learning Map</i>	L	
	<i>Virtuoso Custom IC Community</i>	I	
	<i>Cadence Online Support</i>	O	
	<i>Cadence Training</i>	T	
	<i>Cadence Community</i>	C	

Virtuoso Schematic Editor User Guide

Getting Started

<i>Cadence OS Platform Support</i>	P
<i>Contact Us</i>	U
<i>Cadence Home</i>	H
<i>About Virtuoso</i>	A

Note: It is acceptable to have the same access keys to display multiple application menu content, as once an application has been added to the menu bar (using unique access keys) other applications that share the same display access keys cannot be added. For example, you cannot add the Virtuoso ADE and AMS menus to the menu bar at the same time, so there is no conflict over them sharing the same menu display access keys.

Virtuoso Schematic Editor User Guide

Getting Started

Access Keys - Virtuoso Symbol Editor

The following table details the menu access keys for the symbol editor:

Menu/Application to be Added or Displayed	Add Access Key	Display Access Keys
Launch		Alt + L
Symbol L	L	
Symbol XL	X	
Pcell IDE		
Plugins	P	
○ Debug CDF		
File		Alt + F
New	N	
Open	O	
Open Schematic	E	
Close	C	
Save	S	
Save a Copy	A	
Discard Edits	T	
Check and Save	K	
Make Read Only / Make Editable	M	
Export HTML	H	
Export Image	X	
Print	P	
Print Status	I	
Bookmarks	B	
○ Add Bookmarks	A	
○ Manage Bookmarks	M	
Set Default Application	D	
Close All	L	
Edit		Alt + E
Undo	U	
Redo	R	

Virtuoso Schematic Editor User Guide

Getting Started

<i>Move</i>	M
<i>Copy</i>	C
<i>Stretch</i>	S
<i>Delete</i>	D
<i>Rotate</i>	O
<i>Align</i>	
☐ <i>Align Left</i>	L
☐ <i>Align Vertical</i>	V
☐ <i>Align Right</i>	R
☐ <i>Align Top</i>	T
☐ <i>Align Horizontal</i>	H
☐ <i>Align Bottom</i>	B
☐ <i>Snap to Grid</i>	G
<i>Distribute</i>	
☐ <i>Distribute Vertical</i>	V
☐ <i>Distribute Horizontal</i>	H
<i>Text</i>	T
<i>Increase Text Size</i>	
<i>Decrease Text Size</i>	
<i>Hierarchy</i>	H
☐ <i>Descend Edit</i>	E
☐ <i>Descend Read</i>	R
☐ <i>Edit In Place</i>	P
☐ <i>Show Scope</i>	S
☐ <i>Return</i>	U
☐ <i>Return To Top</i>	T
☐ <i>Print Tree</i>	N
<i>Properties</i>	P
☐ <i>Objects</i>	O
☐ <i>Cellview</i>	C
☐ <i>Pin Order</i>	P
☐ <i>VHDL</i>	H
<i>Select</i>	
☐ <i>Select All</i>	Ctrl + A

Virtuoso Schematic Editor User Guide

Getting Started

☐ <i>Deselect All</i>	Ctrl + D	
☐ <i>Select By Area</i>	Shift + A	
☐ <i>Select By Line</i>	Shift + Ctrl + A	
☐ <i>Select By Filter</i>		
☐ <i>Select By Property</i>		
☐ <i>Filter (Ctrl+F)</i>	Ctrl + F	
<i>Origin</i>	G	
<i>Update Pins From View</i>	V	
<i>Find</i>	F	
<i>Replace</i>	E	
View		Alt + V
<i>Zoom In</i>	I	
<i>Zoom Out</i>	O	
<i>Zoom To Area</i>	Z	
<i>Zoom To Selected</i>	S	
<i>Zoom To Fit</i>	F	
<i>Pan</i>	P	
<i>Redraw</i>	R	
<i>Reset Invisible Labels</i>	I	
<i>Hide Instance Labels</i>	B	
<i>Dynamic Zooming</i>	D	
<i>Net Highlighting</i>	N	
<i>Display Dimming</i>	G	
<i>Save/Restore</i>	V	
☐ <i>Previous View</i>	P	
☐ <i>Next View</i>	N	
Create		Alt + C
<i>Shape</i>	S	
☐ <i>Line</i>	L	
☐ <i>Rectangle</i>	R	
☐ <i>Polygon</i>	P	
☐ <i>Circle</i>	C	
☐ <i>Ellipse</i>	E	
☐ <i>Arc</i>	A	

Virtuoso Schematic Editor User Guide

Getting Started

<i>Pin</i>	P	
<i>Label</i>	L	
<i>Selection Box</i>	B	
<i>Cellview</i>	C	
○ <i>From Cellview</i>	P	
○ <i>From Pin List</i>	I	
○ <i>From Instance</i>	I	
<i>Note</i>	N	
○ <i>Note Text</i>	T	
○ <i>Shape</i>	S	
○ <i>Kanji</i>	K	
<i>Net Expression</i>	X	
<i>Custom Pin</i>	U	
<i>Import Symbol</i>	Y	
<i>Probe</i>	O	
○ <i>Add Net</i>	N	
○ <i>Create Pin</i>	P	
○ <i>Explain</i>	X	
○ <i>Remove Net</i>	R	
○ <i>Remove Pin</i>	M	
○ <i>Remove All</i>	A	
○ <i>Load</i>	L	
○ <i>Save</i>	S	
Check		Alt + K
<i>Current Cellview</i>	C	
<i>Cross-View</i>	V	
<i>Rules Setup</i>	R	
<i>Find Marker</i>	F	
<i>Delete Marker</i>	M	
<i>Delete All Markers</i>	A	
Options		Alt + P
<i>Editor</i>	E	
<i>Display</i>	D	
<i>Select Filter</i>	I	

Virtuoso Schematic Editor User Guide

Getting Started

<i>Check</i>	C	
<i>Tool Filter</i>	T	
<i>Save Defaults</i>	S	
<i>Load Defaults</i>	L	
Window		Alt + W
<i>Assistants</i>	A	
☐ <i>Property Editor</i>		
☐ <i>Search</i>		
☐ <i>Navigator</i>		
☐ <i>World View</i>		
☐ <i>Show All</i>	A	
☐ <i>Hide All</i>	H	
<i>Workspaces</i>	W	
☐ <i>Basic</i>		
☐ <i>Classic</i>		
☐ <i>Explore</i>		
☐ <i>first workspace</i>	1	
☐ <i>second workspace</i>	2	
☐ <i>etc.</i>	3	
☐ <i>Save As</i>	S	
☐ <i>Delete</i>	D	
☐ <i>Load</i>	L	
☐ <i>Set Default</i>	E	
☐ <i>Revert to Saved</i>	R	
☐ <i>Show/Hide Assistants</i>	H	
<i>Toolbars</i>	T	
☐ <i>File</i>		
☐ <i>Edit</i>		
☐ <i>View</i>		
☐ <i>Create</i>		
☐ <i>Search</i>		
☐ <i>Go</i>		
☐ <i>Bookmarks</i>		
☐ <i>Workspaces</i>		

Virtuoso Schematic Editor User Guide

Getting Started

☐	<i>Show All</i>	A	
☐	<i>Hide All</i>	H	
	<i>Tabs</i>	B	
☐	<i>First cellview</i>	1	
☐	<i>Second cellview</i>	2	
☐	<i>etc.</i>	3	
☐	<i>Close Current Tab</i>	C	
☐	<i>Close Other Tabs</i>	O	
	<i>Copy Window</i>	C	
Help			Alt + H
	<i>Contents</i>	C	
	<i>What's New</i>	N	
	<i>Known Problems and Solutions</i>	K	
	<i>Virtuoso Documentation</i>	D	
	<i>Cadence Video Library</i>	V	
	<i>Cadence Online Support</i>	O	
	<i>Cadence User Forums</i>	U	
	<i>Web Collaboration</i>	W	
	<i>About Schematics</i>	A	

Quick Reference - Toolbars

The following toolbars are available for selection in both the Virtuoso Schematic (and Symbol) Editor L and XL (unless specifically stated).

■ *File toolbar*



Options: *New / Open / Check and Save / Save*

■ *Edit toolbar*



Options: *Move / Copy / Stretch / Delete / Edit Properties / Direct Text Edit / Repeat Last Command / Undo / Redo / *Rotate / Increase Text Size / Decrease Text Size / Align / Distribute*

Note: * The *Rotate* option has a pull-down where you can choose to either *Rotate Right*, *Rotate Left*, *Flip Horizontal*, or *Flip Vertical*. See also [Rotating](#).

■ *View toolbar*



Options: *Zoom In / Zoom Out / Zoom to Fit / Zoom to Selected*

■ *Create (Schematics) toolbar*



Options: *Create Instance / Create Narrow Wire / Create Wide Wire / Create Wire Name / Create Pin / Command Options*

Virtuoso Schematic Editor User Guide

Getting Started

■ Create (Symbol) toolbar



Options: *Create Rectangle / Create Label / Create Pin / Create Circle / Create Polygon / Create Ellipse / Create Line / Create Arc / Create Note*

■ Search toolbar



Options: *Search Categories / Search / Search History / Advanced Search*

■ Select toolbar



Options: *Default Selection Filter / Instance Select Filter / Wire Select Filter / Pin Select Filter / Text Select Filter / Open Selection Filter*

■ Design Intent toolbar

Note: This is available in Schematics XL only.



Options: *Annotation Show/Hide / Glyph Show/Hide / Halo Show/Hide / Delete all DI in Cellview / Report*

For more details see, [Virtuoso Design Intent User Guide](#).

■ RF toolbar (Virtuoso RF Solution Option)



Options: *Inflate / Create Net Topology / Create Stackup / analogLib cap symbol / analogLib capq symbol / analogLib ind symbol / analogLib indq symbol / analogLib res symbol / analogLib resistor symbol* and includes shortcuts to access

Virtuoso Schematic Editor User Guide

Getting Started

the transmission line components from the library. For more details, see [RF Transmission Lines](#).

■ *Bookmarks* toolbar



Options: (*list of bookmarked views*)

For more information on the *Bookmarks* toolbar see [Using Bookmarks and Views](#) in the *Virtuoso Design Environment User Guide*.

Virtuoso Schematic Editor User Guide

Getting Started

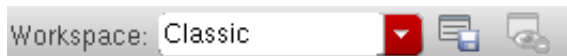
■ Go toolbar



Options: *Back / Forward / Up / Top*

For more information on the *Go* toolbar and design navigation see [Navigating Cellviews and Hierarchies](#) in the [Virtuoso Design Environment User Guide](#).

■ Workspaces toolbar



Options: *Workspace Configuration / Save Workspace / Toggle Assistants Visibility*

For more information on the *Workspaces* toolbar see [Working with Workspaces](#) in the [Virtuoso Design Environment User Guide](#).

See also [Toolbars and Assistants Panes](#) in the [Virtuoso Design Environment User Guide](#).

Workspaces and Assistants in Schematics L/XL

You can use the Cadence provided workspaces as well as your own, new, workspaces to customize your schematics' working environment to improve your day-to-day working practices.

Workspaces and assistants are available in both in VSE L and VSE XL (there are also additional Constraint-related workspaces in VSE XL).

Schematic Editor Workspaces

Schematics L/XL workspaces can be accessed from:

- The *Workspace Configuration* toolbar pull-down



or by

- Selecting *Window – Workspaces*

Schematics L provides access to the following workspaces:

- The *Basic* workspace which is the default workspace contains The Navigator Assistant and The Property Editor Assistant.
- The *Classic* workspace where selection will remove all assistants from the current session window.

Note: Selecting **F11** will also toggle on/off all assistant display, while **Ctrl+F11** will toggle all toolbar display, and **Shift+F11** will toggle both all toolbar and all assistant display.

- The *Explore* workspace which contains The Navigator Assistant, The Search Assistant and The Property Editor Assistant.

Additionally, as well as displaying the standard toolbars (*File*, *Edit*, *View*, *Select* and *Create*), the *Explore* workspace will also ensure that the *Search*, *Go*, and *Bookmark* toolbars are displayed.

Note: For more information on all of these toolbars see Quick Reference - Toolbars.

Virtuoso Schematic Editor User Guide

Getting Started

Additionally, to all of the above workspaces, the Virtuoso Schematic Editor XL also has the following workspaces:

- The *Constraint-Helper* workspace

Contains the Constraint Manager Assistant and the Circuit Prospector Assistant.

For more information on constraints and their related workspaces, see the *Virtuoso Unified Custom Constraints User Guide*.

- The *Constraints* workspace

Contains The Navigator Assistant, The Search Assistant and the Constraint Manager Assistant.

- The *Design Intent* workspace

Note: This is available in Schematics XL only.

Contains the Design Intent toolbar, The Navigator Assistant, The Navigator Assistant and The Property Editor Assistant.

For more information on design intent and the related workspace, see the *Virtuoso Design Intent User Guide*.

- The *Power* workspace

Contains the The Navigator Assistant and The Property Editor Assistant.

- The *RF* workspace (**Virtuoso RF Solution Option**)

Contains the RF-Module Menu, RF toolbar, The Navigator Assistant and The Property Editor Assistant. For more details, see the *Virtuoso RF Solution Guide*.

Note: For more information on workspaces in general and how to customize and create your own workspace configurations in Virtuoso® Schematic Editor see Working With Workspaces in the *Virtuoso Design Environment User Guide*.

Schematic Editor Assistants

Schematic Editor assistants can be accessed by selecting *Window – Assistants*.

The *Schematic L* and *XL* have the following assistants:

- The Navigator Assistant
- The Search Assistant
- The Property Editor Assistant

Virtuoso Schematic Editor User Guide

Getting Started

- [The World View Assistant](#)
- [The Annotation Browser](#)

Note: For information on the *Annotation Browser*, see the [Annotation Browser](#) in the *Virtuoso Layout Suite XL User Guide*.

In addition to these assistants, *Schematics XL* also has the following assistants:

- [The Constraint Manager Assistant](#)
- [The Circuit Prospector Assistant](#)

Note: For more information on the use of assistants, see [Working With Workspaces](#) in the *Virtuoso Design Environment User Guide*.

Assistant Pane Keyboard Shortcuts

The following table lists all the (non-configurable) keyboard shortcut controls you can use when working with assistant panes in the Virtuoso Schematic Editor.

Key Combination	Action	Key Combination	Action
Backspace	Deletes character to left of cursor	Ctrl+A	Moves cursor to beginning of line
Ctrl+B	Moves cursor one character to left	Ctrl+Backspace	Deletes word to left of cursor
Ctrl+Insert	Copies selected text to clipboard	Ctrl+D	Deletes character to right of cursor
Ctrl+Delete	Deletes word to right of cursor	Ctrl+E	Moves cursor to end of line
Ctrl+equals (=)	Expand all tree branches	Ctrl+F	Moves cursor one character to left
Ctrl+H	Deletes character to left of cursor	Ctrl+Insert	Copies selected text to clipboard
Ctrl+K	Deletes to end of line	Ctrl+Left Arrow	Collapse currently selected branch
Ctrl+Minus	Collapse all branches	Ctrl+Plus	Expand all branches
Ctrl+Right Arrow	Expand currently selected branch	Ctrl+V	Pastes clipboard text into line edit
Ctrl+X	Deletes selected text and copies it to clipboard	Ctrl+Y	Redoes the last undone operation
Ctrl+Z	Undoes the last operation	Delete	Deletes character to right of cursor
End	Moves cursor to the end of line	Home	Moves cursor to beginning of line

Virtuoso Schematic Editor User Guide

Getting Started

Left Arrow	Moves cursor 1 character to left	Right Arrow	Moves cursor 1 character to right
Shift+Delete	Deletes selected text and copies it to clipboard	Shift+Insert	Pastes clipboard text into line edit
Shift+Left Arrow	Moves and selects text 1 character to the left	Shift+Right Arrow	Moves and selects text 1 character to the right.

Additional Features of the Schematic Editor XL

Schematics XL provides the following additional features:

■ Schematic HTML Publisher

Virtuoso Schematic Editor XL provides the ability to export a schematic design hierarchy to HTML.

For more information, see [Exporting a Schematic Hierarchy to HTML](#).

■ Constraint Management

Constraint management functionality allows for the addition, modification, and deletion of constraints from/to the Cadence constraint storage system.

For more information see [Constraint Management in Schematics XL](#) and the [Virtuoso Unified Custom Constraints User Guide](#).

■ Design Intent

Design intent is a collaboration tool that uses annotations as a means for schematic designers to capture and communicate design goals in Schematics XL and provide layout designers with the freedom to decide how to implement those goals in Layout XL.

For more information see the [Virtuoso Design Intent User Guide](#).

Constraint Management in Schematics XL

Schematics XL also provides advanced constraint management functionality through two constraint assistant panes:

■ [The Constraint Manager Assistant](#)

■ [The Circuit Prospector Assistant](#)

The Constraint Manager Assistant

The *Constraint Manager* assistant pane allows for the addition, modification, and deletion of constraints from/to the constraint storage system. These constraints consist of the likes of matching, symmetry, orientation, relative orientation, IR drop, clustering, alignment, area, distance, boundaries, power structure (guard rings), custom module generators (modgens), cell plans, and analog groups.

The *Constraint Manager* also keeps track of all added constraints by grouping constraints by instance or constraint type. Each constraint can be modified, for example symmetry can be changed from mirror to simple using the Constraints Editor.

As well as the above, you can also add notes, set priority and change the status of a constraint.

You can access the *Constraint Manager* by

- Selecting *Window – Assistants – Constraint Manager*.
- Selecting *Constraint Manager* from the *Toolbars* context-sensitive menu.
- Selecting the *Constraints* or *Constraint-Helper* workspaces from the *Workspace Configuration* pull-down menu on the toolbar (or selecting *Window – Workspaces Constraints/Constraint-Helper*).

For more information see [The Constraints Manager Assistant](#) in the [Virtuoso Unified Custom Constraints User Guide](#).

The Circuit Prospector Assistant

The *Circuit Prospector* assistant pane provides three main elements of functionality to assist the creation of constraints in [The Constraint Manager Assistant](#):

1. The ability to find structures in a schematic view.

The current set of structures automatically recognized include devices in the same well, symmetric devices, devices of the same cell name, MOS current mirrors, MOS differential pairs, MOS common gates, and so on.

2. The ability to modify the structure finders.

This means that you can modify an existing finder in order to customize it. For example, the finder for MOS common gate can be modified to create a finder for BJT common base.

3. The ability to create a customized finder based on selected topologies.

Here, you can select devices and nets in the schematic view which represent a custom structure. You would capture this structure, name the finder for the structure, and then use this finder to locate all structures of this type.

You can access the *Circuit Prospector* by:

- Selecting *Window – Assistants – Circuit Prospector*.

Virtuoso Schematic Editor User Guide

Getting Started

- Selecting *Circuit Prospector* from the *Toolbars* context-sensitive menu.
- Selecting the *Constraint-Helper* workspace from *Windows – Workspaces* or the *Workspace Configuration* toolbar pull-down.

For more information see The Circuit Prospector Assistant in the *Virtuoso Unified Custom Constraints User Guide*.

Virtuoso ADE Integration in Virtuoso Schematic Editor

You can launch Virtuoso ADE Explorer or Virtuoso ADE Assembler from Virtuoso Schematic Editor (VSE) or vice versa by using the VSE menu. Virtuoso ADE toolbars are then displayed along with the VSE toolbars. The VSE window title bar is updated to state whether you are working in ADE Explorer or ADE Assembler mode.

For more information, see [Opening ADE Explorer From Virtuoso Schematic Editor](#).

Photonics in Virtuoso Schematic Editor, ADE, and Layout XL (Virtuoso Photonics Option)

The EPDA Framework has been built by using the three main Virtuoso applications: Virtuoso Schematic Editor, Virtuoso Analog Design Environment, and Virtuoso Layout Suite. Spectre is also used to enable true Electrical - Photonics co-design.

Optical pins, which are an essential part of a photonics solution, are created by defining the signal types as optical. It further contributes in creating the optical ports and waveguides that can be identified by assigning a distinctive color and line style.

VSE and VLS support the optical signal types `optical`, `singleModeOptical`, and `multiModeOptical` during editing, connectivity extraction, and checking as part of this feature. For example,

- Preserving optical signal types during extraction.
- Propagating optical signal types from instance terminals.
- Resolving mixed optical signal types during propagation.
- Applying the optical wire style after connectivity extraction.
- Running optical checks after connectivity extraction.

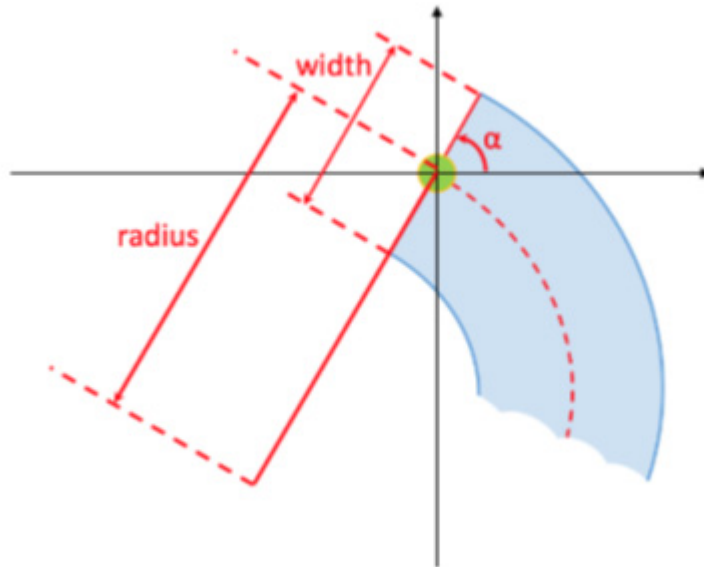
In photonics, waveguides are also one of the basic building blocks. They are used to guide light as optical modes and can be curved. Each photonic port is associated with an access to an optical (photonic) waveguide.

There are three properties associated with every photonic port:

- **Width** width of the waveguide associated to the port
- **Radius** curvature of the waveguide associated to the port
- **Angle** access direction to the photonic port

In photonics processes, waveguides are necessary in order to guide light along a path. In Virtuoso, waveguides can be generated as connected Pcells or fixed layouts.

Figure 1-1 Photonics Waveguide



Waveguide Editor allows editing of the waveguides. It can be accessed from VSE as well as VLS.

Refer to the following links to know more about the photonics solution in Virtuoso:

VSE

- [Editing the Optical Signal Types](#)
- [Propagating Optical Signal Types \(Virtuoso Photonics Option\)](#)
- [Optical Checks \(**Virtuoso Photonics Option**\)](#)
- [Editing the Optical Signal Types in Property Editor \(**Virtuoso Photonics Option**\)](#)
- [Photonics Environment Variables \(Virtuoso Photonics Option\)](#)
- [Using opticalLib to Create PDK \(Virtuoso Photonics Option\)](#)

VLS XL

- [Rotating Objects](#)
- [Create Pin Options](#)
- [Edit Properties Options](#)

Virtuoso Schematic Editor User Guide

Getting Started

- [Generating Optical Pins for the Virtuoso Photonics Solution](#)
- [Editing Composite Waveguides](#)
- [Composite Waveguide Editor](#)
- [Abutment in Virtuoso Photonics Solution](#)
- [Photonic Interprocess Communication Functions](#)

Help and Support Facilities

The following help and support facilities are available as *Help* menu options:

Help menu option	Description
<i>Search</i>	A text field that lets you enter a search string. Press <code>Enter</code> to view the search results. Note: Do not enclose the search string in double quotes.
<i>Schematics L User Guide</i>	Opens VSE L User Guide (at the section that provides information about using CIW) in Cadence Help.
<i>Schematics L What's New</i>	Opens the VSE L What's New document in Cadence Help.
<i>Schematics L Known Problems and Solutions</i>	Opens the VSE L Known Problems and Solutions document in Cadence Help.
<i>Virtuoso Documentation Library</i>	Opens the Cadence Help home page, which provides quick access links to the following local and online resources: ■ What's New ■ Video Demos and Tutorials ■ Featured Content ■ Known Problems and Solutions ■ Other web resources
<i>Virtuoso Video Library</i>	Opens the Video Library page available on Cadence Online Support (COS). This page lists the videos available for various Virtuoso products. Note: You must have a COS account to access the content available on COS. Note: Contact your IT support to ensure that the Internet ports required for video playback are enabled.
<i>Virtuoso Rapid Adoption Kits</i>	Opens the Rapid Adoption Kits page on COS. This page lists Rapid Adoption Kits (RAKs) available for various Virtuoso products. A RAK contains design databases and instructions on how to run the design flow.

Virtuoso Schematic Editor User Guide

Getting Started

<i>Virtuoso Learning Map</i>	Lists domain-specific training available on Cadence Training Services. Cadence Training Services learning maps provide a comprehensive visual overview of the learning opportunities for Cadence customers. They provide recommended course flows as well as tool experience and knowledge levels to guide customers through a complete learning plan.
<i>Virtuoso Custom IC Community</i>	Opens the Virtuoso Custom IC Community web page. This page provides access to the latest blogs and discussion threads on various Virtuoso products and design topics, information about software downloads and support and training, and other related information. You too can contribute to the community forum by creating a Cadence account. This gives you additional benefits such as alerts about topics of interest and access to online webinars.
<i>Cadence Online Support</i>	Opens COS, which you can use to access information about Cadence products, documentation, videos, RAKs, application notes, troubleshooting information, alerts, and so on. Improvements are regularly made to COS to enable you to look up the information you want. We recommend that you bookmark this web site and use it as your first point of reference for any Virtuoso-related information.



Tip

You can also access COS by clicking the Cadence logo available in the upper-right banner in each Virtuoso window.



<i>Cadence Training</i>	Opens the Cadence training web page. You can find on this page information about the training courses available in different regions. Information is available about both classroom and online courses.

Virtuoso Schematic Editor User Guide

Getting Started

<i>Cadence Community</i>	Opens the Cadence Community web page. This page provides access to the latest blogs and discussion threads on various Cadence products and solutions, and EDA Industry Insights. You too can contribute to the community forum by creating a Cadence account. This gives you additional benefits such as alerts about topics of interest and access to online webinars.
<i>Cadence OS Platform Support</i>	Provides information about the current Cadence software releases and the supported platforms.
<i>Contact Us</i>	Opens the Cadence Customer Support web page, which provides customer support contact information for different regions.
<i>Cadence Home</i>	Opens the Cadence corporate web site.
<i>About Virtuoso</i>	Displays Virtuoso Design Environment version information.

Contextual Cursors

Each enter function in the schematic editor has a specific cursor to distinguish which action is currently being performed (for example, direct text edit, selection, delete, add instance, and so on).

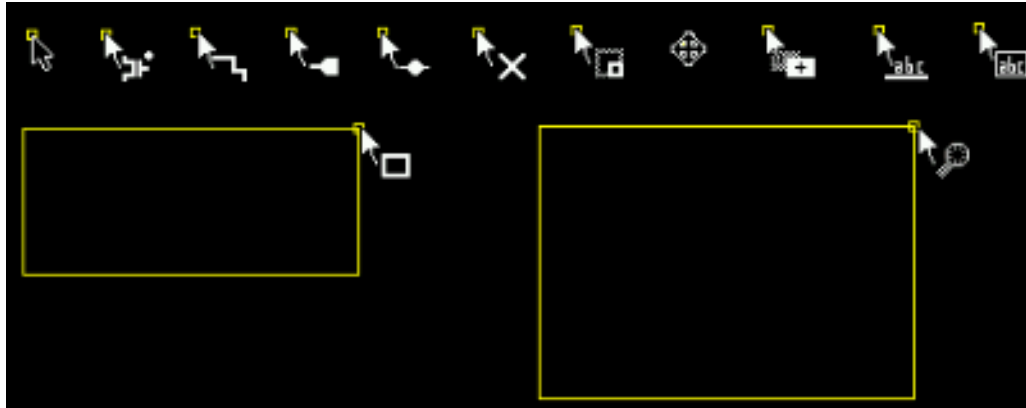


Figure 1-2 Examples of Contextual Cursors Used in the Schematic Editor

See also [hiSetCursor](#) in the *Cadence User Interface SKILL Reference*.

Working with Designs

The following section describes the basics about the schematic editor and the symbol editor.

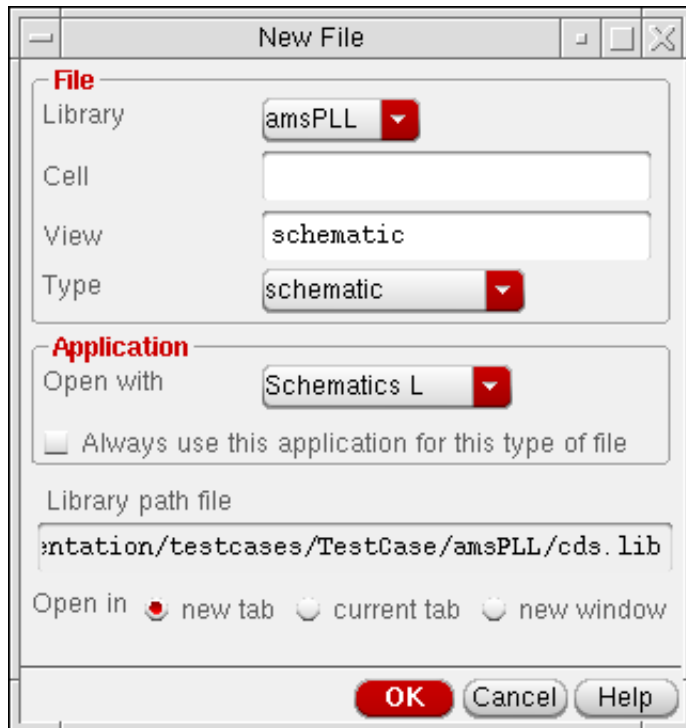
- [Opening a New Cellview](#)
- [Opening an Existing Cellview](#)
- [Using the Schematic Window](#)
- [Using the Symbol Editor](#)
- [Making Designs Editable](#)
- [Making Designs Read Only](#)
- [Saving Your Edits](#)
- [Closing Editor Windows](#)
- [Quitting the Virtuoso Schematic Editor](#)
- [Viewing a Design in Schematic Assistant of Virtuoso Studio Layout Suite](#)

Opening a New Cellview

To open a new cellview from the schematic editor:

1. Choose *File – New*.

The New File form appears.



2. In the *Library* pull-down choose the library in which to store the new cellview.
3. In the *Cell* field enter a new cell name or leave as-is with the current cell name entered.
4. In the *View* field enter a name for the new cellview.
5. In the *Type* pull-down choose the view type, for example *schematic* or *config*, that is to be associated with the new cellview.
6. In the *Open with* field choose the application that the new cellview should automatically be opened with.
7. Optionally, check the *Always use this application for this type of file* checkbox to associate the selection made in the *Open with* field with the current *View* type selected.

This will save you in future having to select what application should be opened with a selected cellview *Type*.

Virtuoso Schematic Editor User Guide

Getting Started

8. Choose whether you want to open the new cellview in a *new tab*, the *current tab*, or in a completely *new window*.

Note: The *Library path file* displays the location where the new cellview will be stored.

9. Click *OK*.

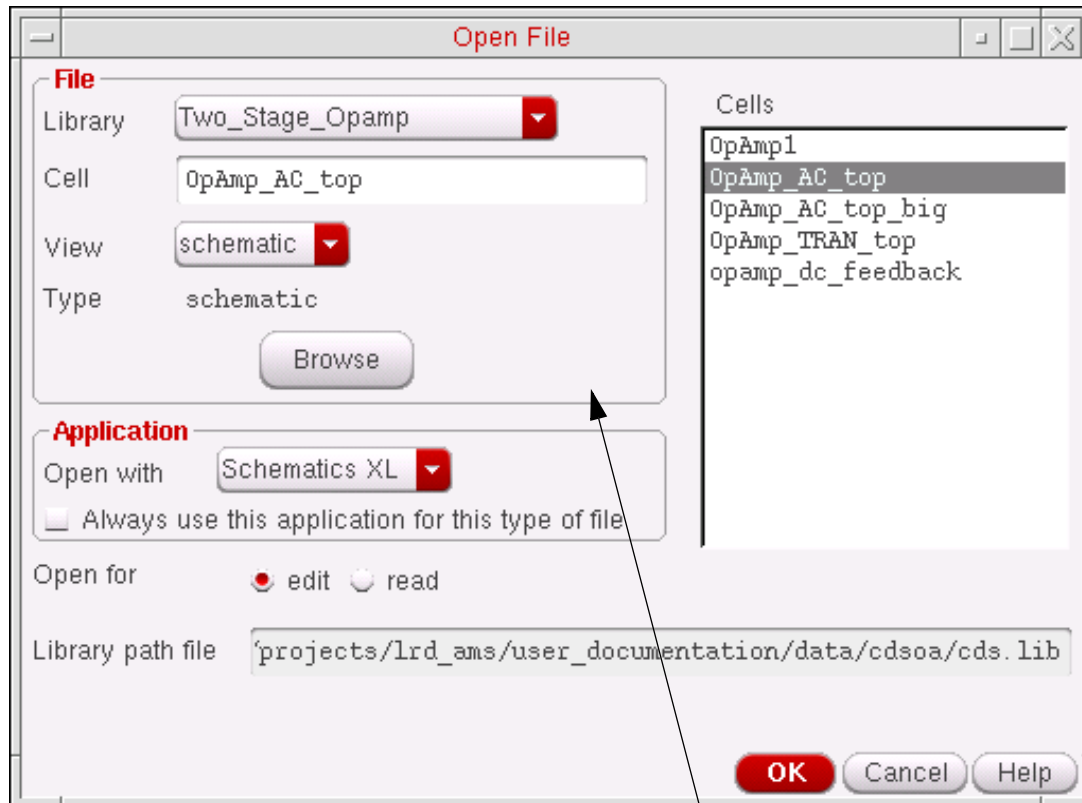
Note: Schematics XL can also be launched directly from Schematics L by selecting *Launch – Schematics XL*.

Opening an Existing Cellview

To open an existing cellview in the schematic editor:

1. Choose *File – Open* from an editor window or the CIW.

The Open File form appears.



Thumbnail of selected view is displayed (if found) to provide a visual selection aid

2. In the *Library* pull-down choose the library that contains the existing cellview.
3. In the *Cell* field specify a cell name to open by doing one of the following:
 - ☐ Type the cell name
 - ☐ Click a cell name in the *Cells* list box
 - ☐ *Browse* for the cell in the Library Browser

Virtuoso Schematic Editor User Guide

Getting Started

4. Choose the cellview to be opened from the pull-down.

The *Type* field will update to display the type of view of the selected cell view.

Note: If a thumbnail of the selected cellview is available, this will be displayed in the Open File form as an aid to selection. If no thumbnail is available, the space will be blank.

5. In the *Open with* field choose the application that the cellview should be opened with.
6. Optionally, check the *Always use this application for this type of file* checkbox to associate the selection made in the *Open with* field with the current *View* type selected.

This will save you in future having to select what application should be opened with a selected cellview *Type*.

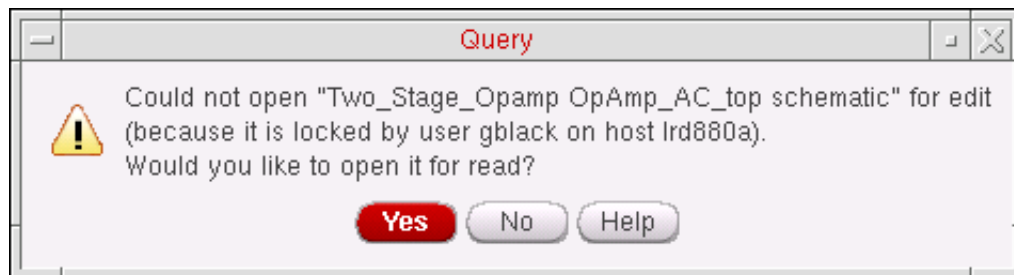
7. Choose to open the cellview in *edit* or *read* mode.
8. Choose to open the cellview in a *new tab* in the current session window, the *current tab* in the current session window, or a *new window* in a new session.
9. Click *OK*.

Note: For more information on cellview tabs see [Working with Cellviews](#) in the [Virtuoso Design Environment User Guide](#).

Cellview Already in Use

If you attempt to open a cellview that is already being used (that is, the cellview file is locked by another user), a warning message will be displayed informing you of the current user of the cellview and the current host machine name.

From here you can choose to either open the cellview as read-only or not to open the cellview.



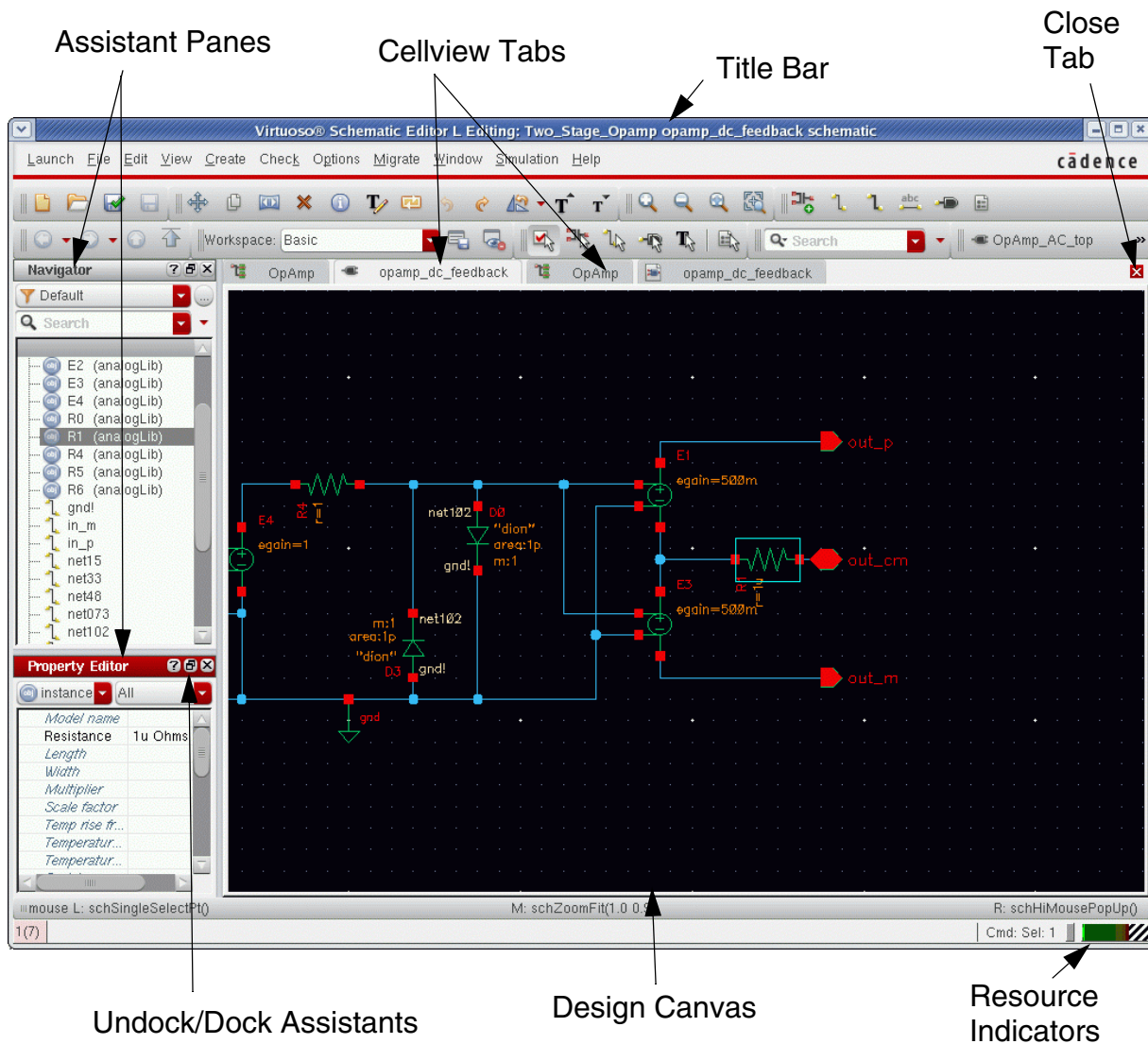
Note: This warning message will also be displayed if you attempt to descend into, or edit in place, a cellview currently in use by another user.

Virtuoso Schematic Editor User Guide

Getting Started

Using the Schematic Window

A schematic design is a graphical representation of an electronic circuit design. The connectivity information is derived from the placement of pins, wires, and instances. To create a schematic design, you use the schematic window, which is shown in the following graphic.

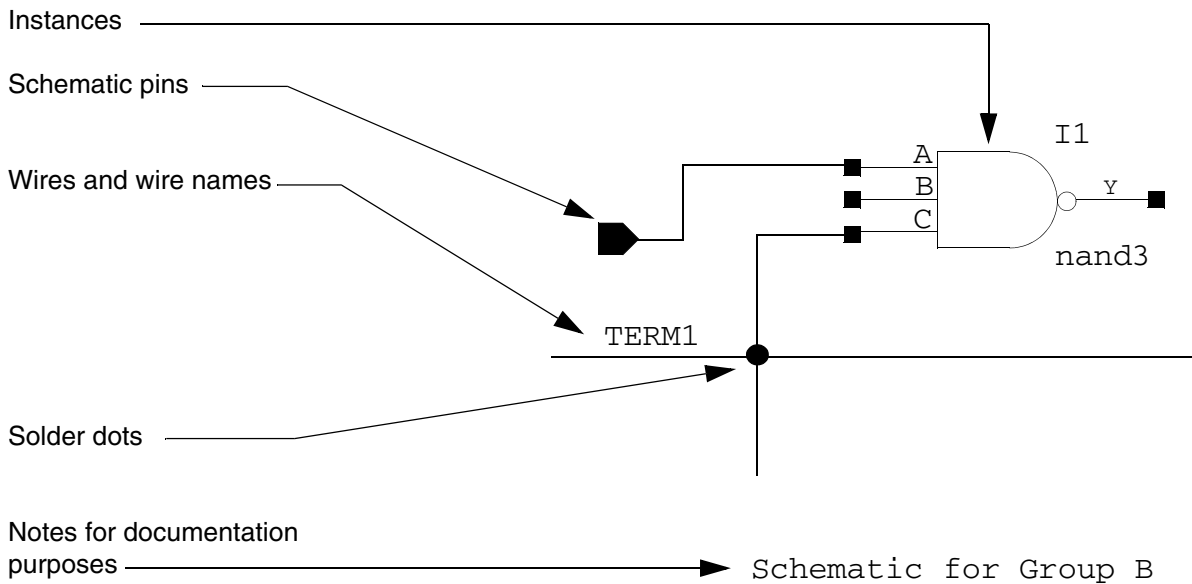


Note: For more information on how to track license access and memory usage in the current session, see Resource Indicators in the *Virtuoso Design Environment User Guide*.

Virtuoso Schematic Editor User Guide

Getting Started

You can then to choose add a range of objects to the schematic canvas including the following:



You should use the symbol editor to draw symbols that you place in the schematic window. For more information see [Using the Symbol Editor](#).

Note: You can use multisheet schematics to partition a large design among several sheets (pages), which improves readability and produces better plot output.

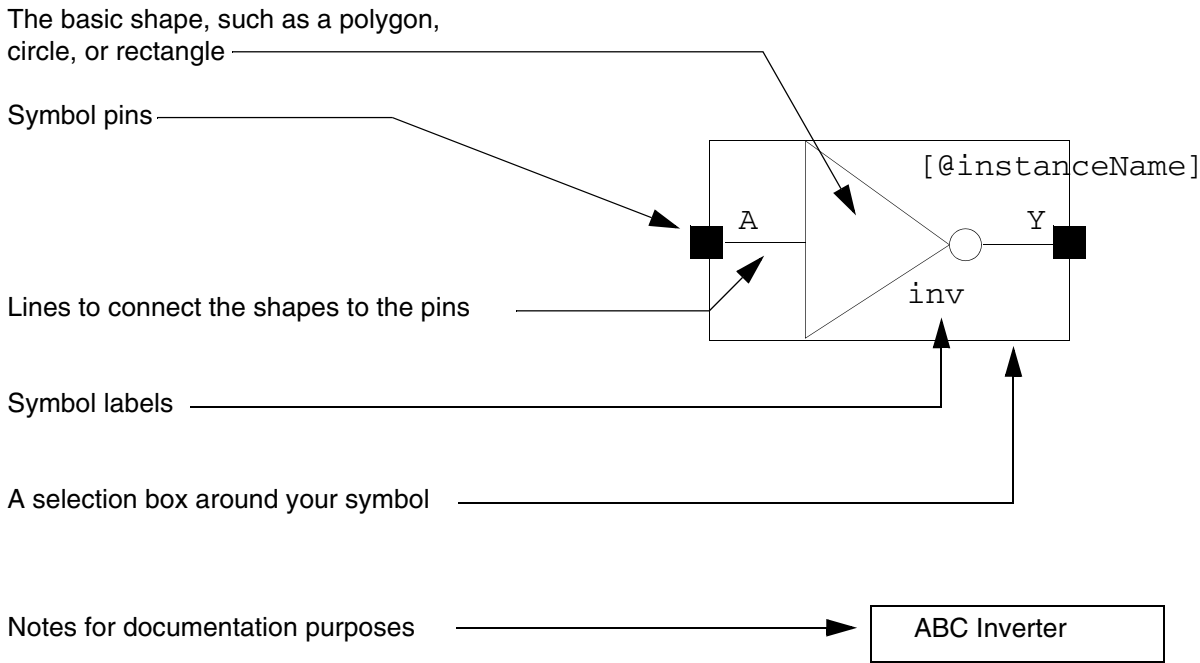
Using the Symbol Editor

When a schematic design requires a symbol that does not have a corresponding symbol in a Cadence library, you can create your own symbol using the symbol editor. After you create the master symbol, you can add a copy of it (known as an instance) to your design using the schematic editor.

If you later change a master symbol, the schematic editor automatically applies the changes to all the existing instances of that symbol, as well as to any new instances you might add.

The only differences visually between the schematic editor and the symbol editor are the commands.

You use the symbol editor to add the following graphics.



Making Designs Editable

If you opened your schematic or symbol in read-only mode, you can change the mode to make it editable.

- Choose *File – Make Editable*.

This command toggles with *File – Make Read Only*.

Note: Some libraries might have read-only file permissions, which you cannot override with this command.

Making Designs Read Only

To make a design open in read-only mode,

- Choose *File – Make Read Only*.

If you have made any changes to the current cellview but have not saved them, the Change Edit Mode dialog box appears and prompts you to either save the changes or discard them

- ☐ Yes saves your edits.
- ☐ No discards your edits.

After you click *Yes* or *No*, the mode becomes read only.

This command toggles with *File – Make Editable*.

Note: Both the *Make Read Only* and *Make Editable* options can be applied to a selected constraint in the *Constraint Manager* assistant pane. The *Constraint Manager* is only available in Schematics XL. For more information see [Virtuoso Unified Custom Constraints User Guide](#).

Saving Your Edits

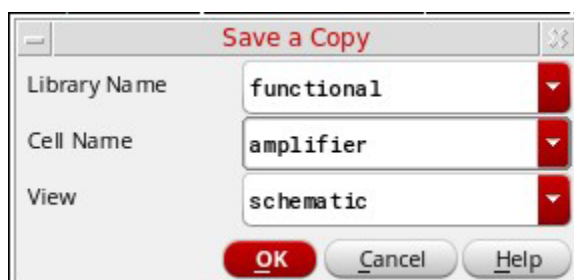
To save a design:

- Choose *File – Save*.

To save a design to a different name,

1. Choose *File – Save a Copy*.

The Save a Copy form appears.



2. Change the form fields (in the Library Name, Cell Name, and View fields) to a filename of your choice.
3. Click *OK*.

The *Save* options can also be used to store any changes made, for example the creation or removal of constraints, in the *Constraint Manager* assistant pane. You may therefore, when using constraints management, see a *Save (Constraints)* option. The *Constraints Manager* is only available in Schematics XL. For more information see [Virtuoso Unified Custom Constraints User Guide](#).

Thumbnail images, where they have been set up in the current view for pre-viewing cellviews, are updated automatically when you choose to *Save* or *Save a Copy* of that cellview. This is actioned by an automatic call to the `hiGenerateThumbnails` command.

To disable auto-generation of thumbnails, set the following environment variable in `.cdsenv`:

```
ui.thumbnails generate boolean t
```

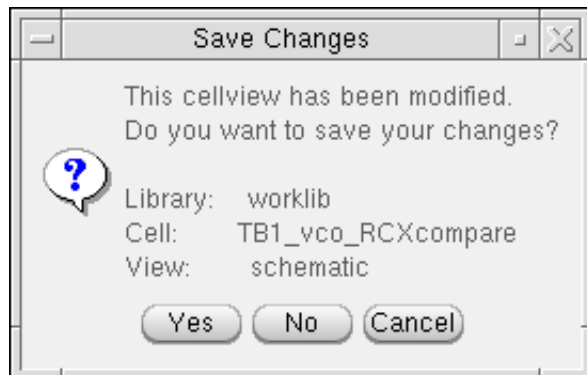
For more information on the use of thumbnails see [Thumbnail View Functions](#) in the *Cadence User Interface SKILL Reference* in the first instance.

Closing Editor Windows

To close an editor window,

- Choose *File – Close*.

If you edited the current cellview since the last time you saved your data, the Save Changes dialog box appears and prompts you to save the changes, discard them, or cancel the Close operation.



Yes saves your edits and closes the window.

No discards your edits and closes the window.

Cancel cancels the *Close* command but does not discard the changes. It returns you to the cellview you were editing.

Quitting the Virtuoso Schematic Editor

To quit the Virtuoso schematic editor

1. From the schematic editor session window choose *File – Close/Close All*.

If you have not saved your edits, the Save Changes form appears.

2. Specify one of the save options by clicking on the corresponding button.

- ☐ Click *All* to reselect all cellviews and save all your changes.
- ☐ Click *None* to deselect all cellviews and quit without saving your changes.

3. Click *OK*.

If you click *Cancel* on the Save Changes form, you cancel both the save and the quit operations, and the software continues to run.

For information about saving your environment, saving your session, or viewing your log file, refer to *Virtuoso Design Environment User Guide*.

Virtuoso Schematic Editor User Guide

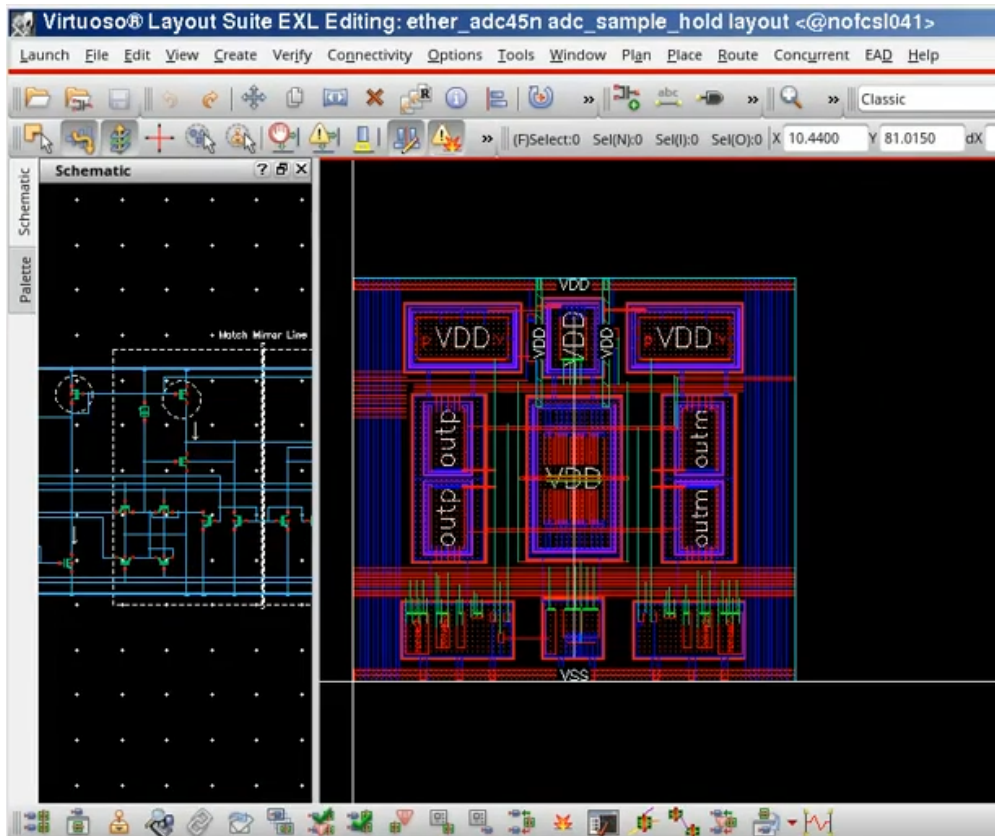
Getting Started

Viewing a Design in Schematic Assistant of Virtuoso Studio Layout Suite

While working with the layout, a layout designer often needs to refer to the design components in the schematic. Either you can keep the design open in the Virtuoso Schematic Editor window along with the Layout Editor window and switch between two, or you can use the *Schematic* assistant in Virtuoso Studio Layout Suite to view the schematic in read-only mode.

The *Schematic* assistant is an assistant in the layout editor that displays the current design as displayed in the schematic editor. In this assistant, 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 schematic editor. This assistant also lets to use the cross-probing and cross-selection features of the schematic editor.

To start the Schematic Assistant, using Virtuoso Studio Layout Suite XL or higher, select *Window – Assistants – Schematic*.



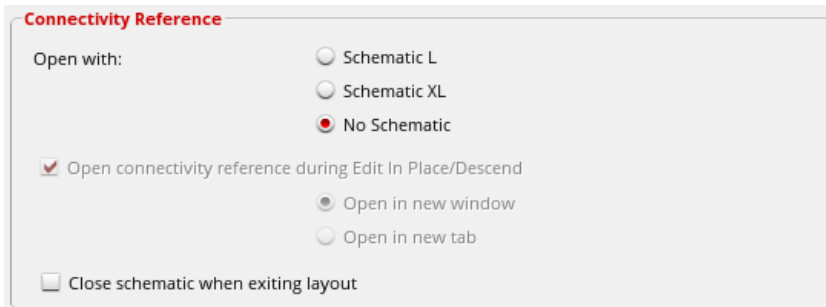
To make the assistant part of the default view in Virtuoso Studio Layout Suite:

1. Select *Options – Connectivity* to open the *Connectivity* form.

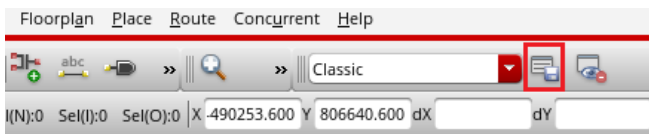
Virtuoso Schematic Editor User Guide

Getting Started

2. In the *Connectivity Reference* area, set *Open with:* to *No Schematic*.



3. Click *OK*.
4. On the *Workspaces* toolbar, select *Save Workplace*.



5. Enter a name for your new workspace.
6. Click *OK* to save the workspace.

The new workspace now contains the Schematic Assistant by default.

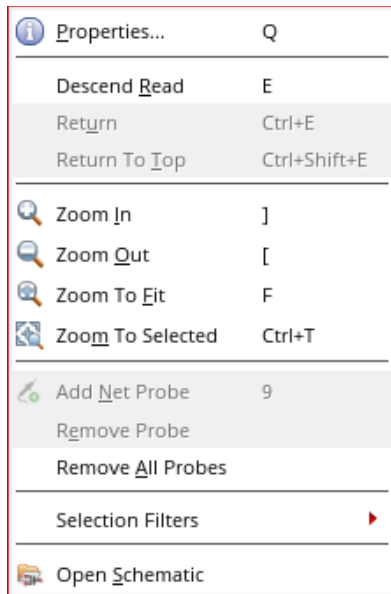
The *Schematic* assistant works only in preselect mode, that is, you must select an object in the assistant or in the schematic editor before it can be interacted with.

Related Topics

Schematic Assistant

Schematic Assistant Right-Click Menu

When you right-click inside the Schematic Assistant, the following pop-up menu appears:



The various options available are:

- *Properties*: Opens the Edit Object Properties form.
- *Descend Read / Return / Return To Top*: Navigates around the schematic hierarchy.
- *Zoom In / Zoom Out / Zoom To Fit / Zoom To Selected*: Zooms on the schematic canvas.
- *Add Net Probe*: Adds a net probe to the selected object.
- *Remove Probe*: Removes selected net probe.
- *Remove All Probes*: Removes all net probes.
- *Selection Filters*: Lets you define the selection filters.
- *Open Schematic*: Opens the schematic in the Virtuoso Studio Schematic Editor.



When using the bindkeys of the Schematic assistant, if you try to use a bindkey that is not specified, the assistant checks if the same bindkey is specified for the layout editor and uses that instead.

Virtuoso Schematic Editor User Guide

Getting Started

Related Topics

Edit Object Properties – Basic

Working with Commands

The following sections provide information about working with commands.

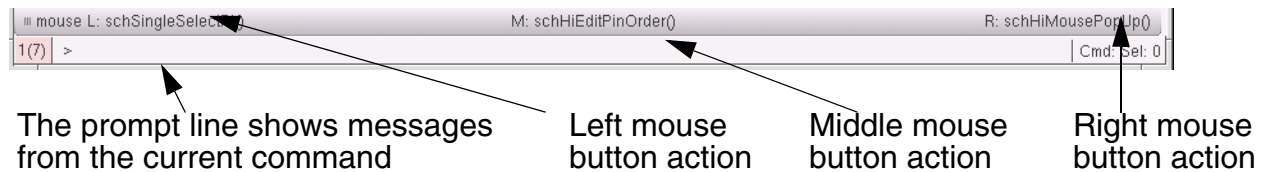
- [Using Command Prompts](#)
- [Canceling a Command](#)
- [Undoing a Command](#)
- [Redoing a Command](#)

Virtuoso Schematic Editor User Guide

Getting Started

Using Command Prompts

After you start a command the prompts at the bottom of the schematic editor window will direct you as to what to do next.



The mouse settings line shows the current settings for the mouse buttons. The settings change as you move the mouse in and out of design windows or start and stop commands.

For example:

`mouse L:schSingleSelectPt` = enter a point

`M:schHiEditPinOrder` = repeats last command (open Pin Order form)

`R:schHiMousePopUp` = popup menu

Canceling a Command

To cancel a command, do one of the following:

- Press the `Esc` key.
- Click *Cancel* on a form.
- Select another command.

Undoing a Command

To undo a command, do one of the following:

- Press the `u` key.
- Choose *Edit – Undo*.

Redoing a Command

To redo a command (that is, undo an undo), do one of the following:

- Press the `Shift-u` keys.
- Choose *Edit – Redo*.

Selecting Objects

This section describes the following topics:

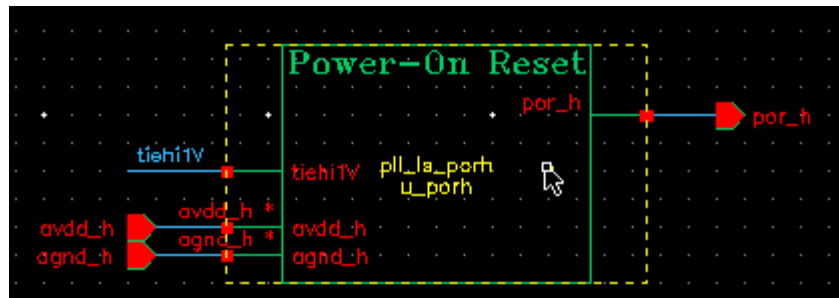
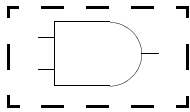
- [Dynamic Highlighting](#)
- [Extending the Selection Area of a Net](#)
- [Selecting One Object](#)
- [Selecting Multiple Objects](#)
- [Selecting Parts of Objects](#)
- [Selecting Wires](#)
- [Selecting a Wire Name and Its Associated Wire Segments](#)
- [Selecting a Pin and Its Associated Wire Segments](#)
- [Selecting an Instance and Its Associated Wire Segments](#)
- [Deselecting Objects](#)

Dynamic Highlighting

When you move the pointer over an object, the object is automatically highlighted. This behavior is called dynamic highlighting. It helps you in identifying the object that will be:

- selected on left-click
- affected by a direct manipulation command
- affected by a command when infix is selected on the User Preferences form

The object is highlighted with a yellow dashed line.



Note: To turn off dynamic highlighting, use the Dynamic Highlighting option available in the Display Options form. Also, when a Cadence® SKILL language enter function is run, the dynamic highlighting of the objects in the schematic window is turned off.

Virtuoso Schematic Editor User Guide

Getting Started

Dynamic Net Highlighting

The dynamic net highlighting feature can be enabled by selecting the *Net Highlighting* menu option on the *View* menu. Consequently, when you move the mouse pointer over a selectable net, or block, any other associated objects are also highlighted and the names of these objects are displayed on the status bar.



Figure 1-3 Net Highlighting Option in the View Menu

Alternatively, you can enable this feature by selecting the "on" radio button corresponding to *Dynamic Net Highlighting* in the *Display Options*.

You can also use `schDynamicNetHighlightOn` to specify whether the schematic editor dynamically highlights all objects of the same net when the mouse pointer is placed over objects.

Virtuoso Schematic Editor User Guide

Getting Started

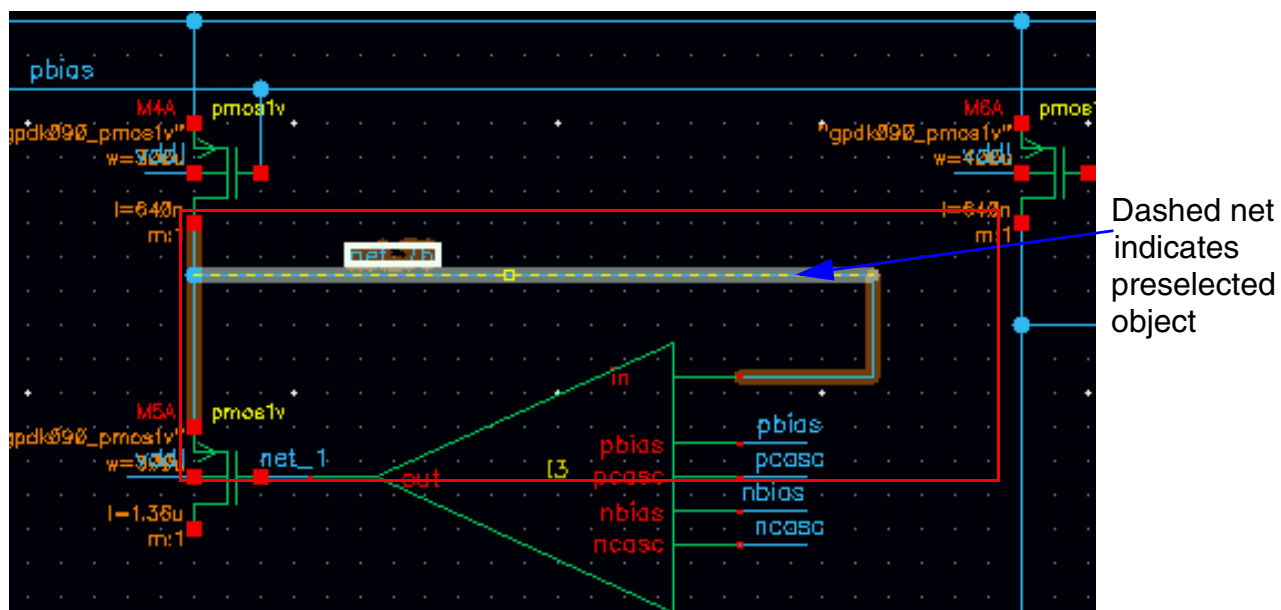


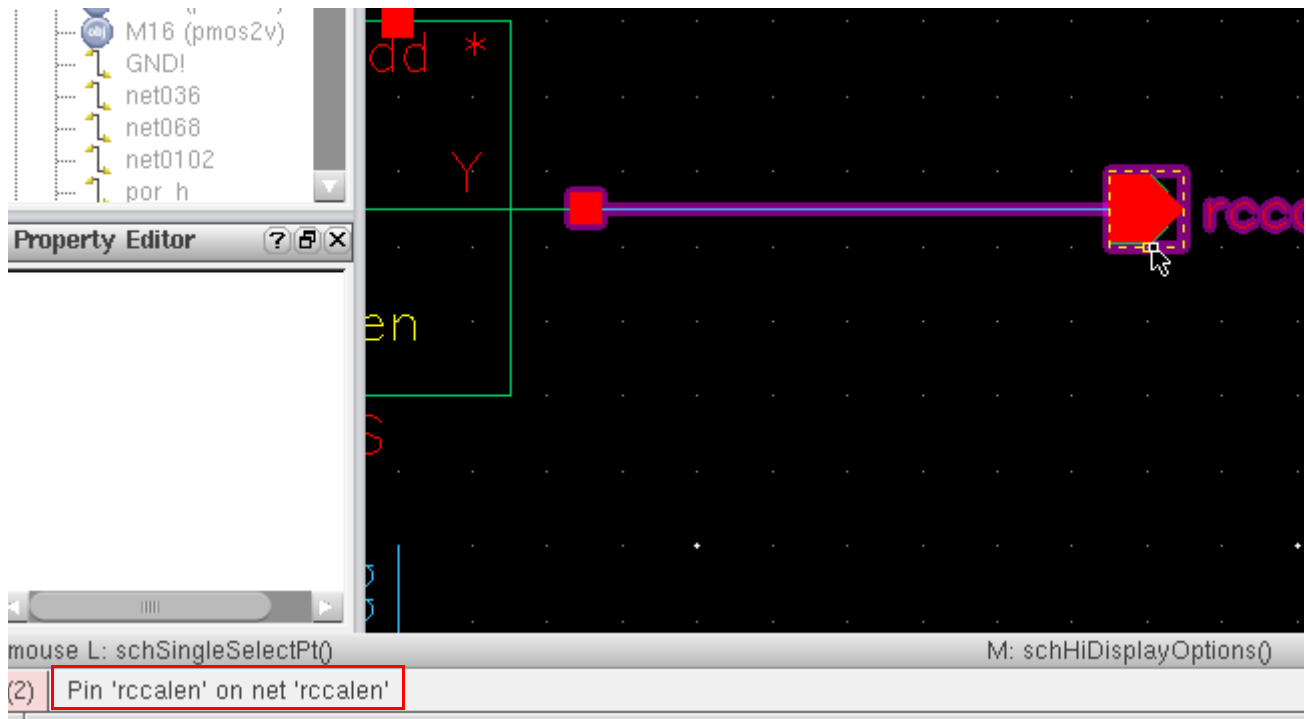
Figure 1-4 Net Highlighting Switched On

Virtuoso Schematic Editor User Guide

Getting Started

Highlighted Object Names on Status Bar

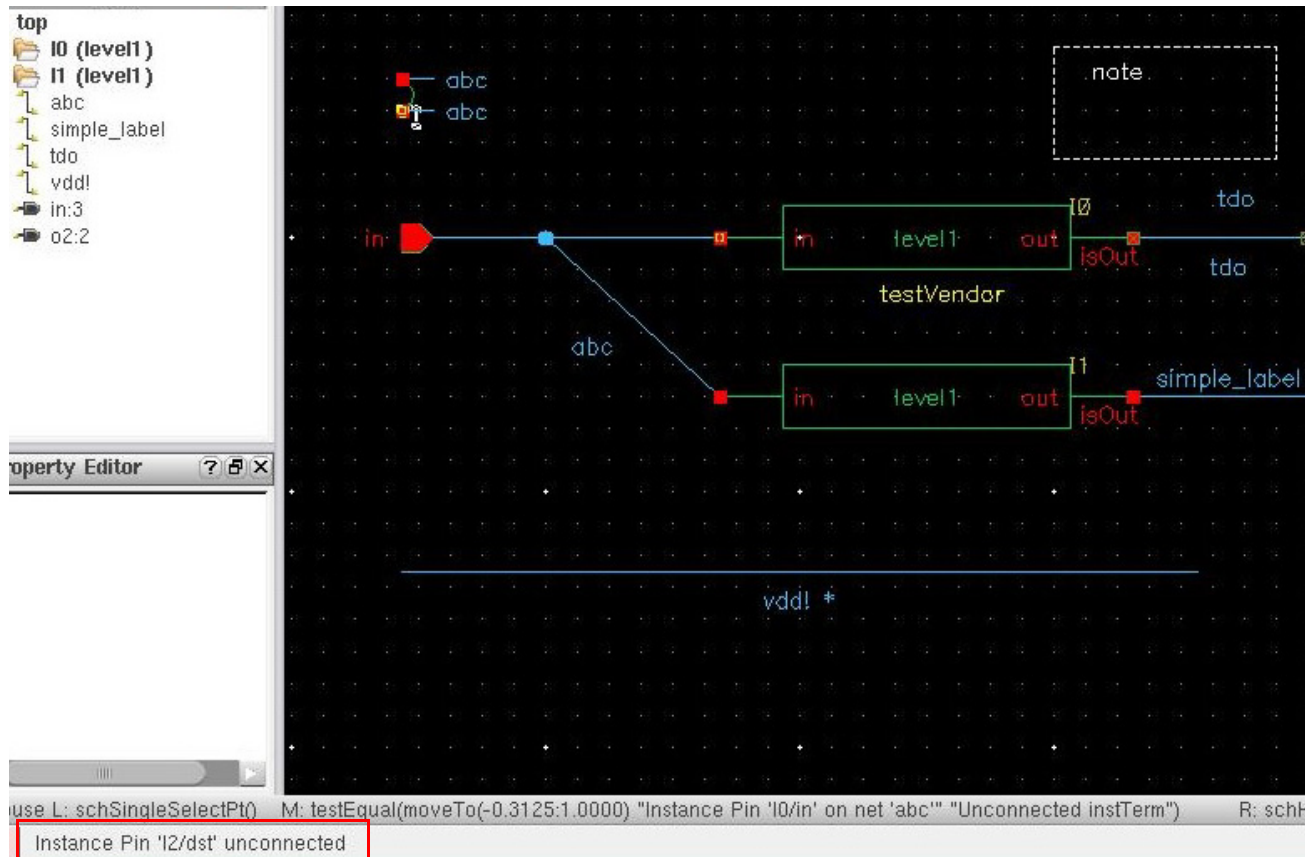
The name of the highlighted objects are displayed on the status bar along with the connected net name. You can use `objectInfoInStatusBar` to control the display of information about the selected objects on the status bar.



Virtuoso Schematic Editor User Guide

Getting Started

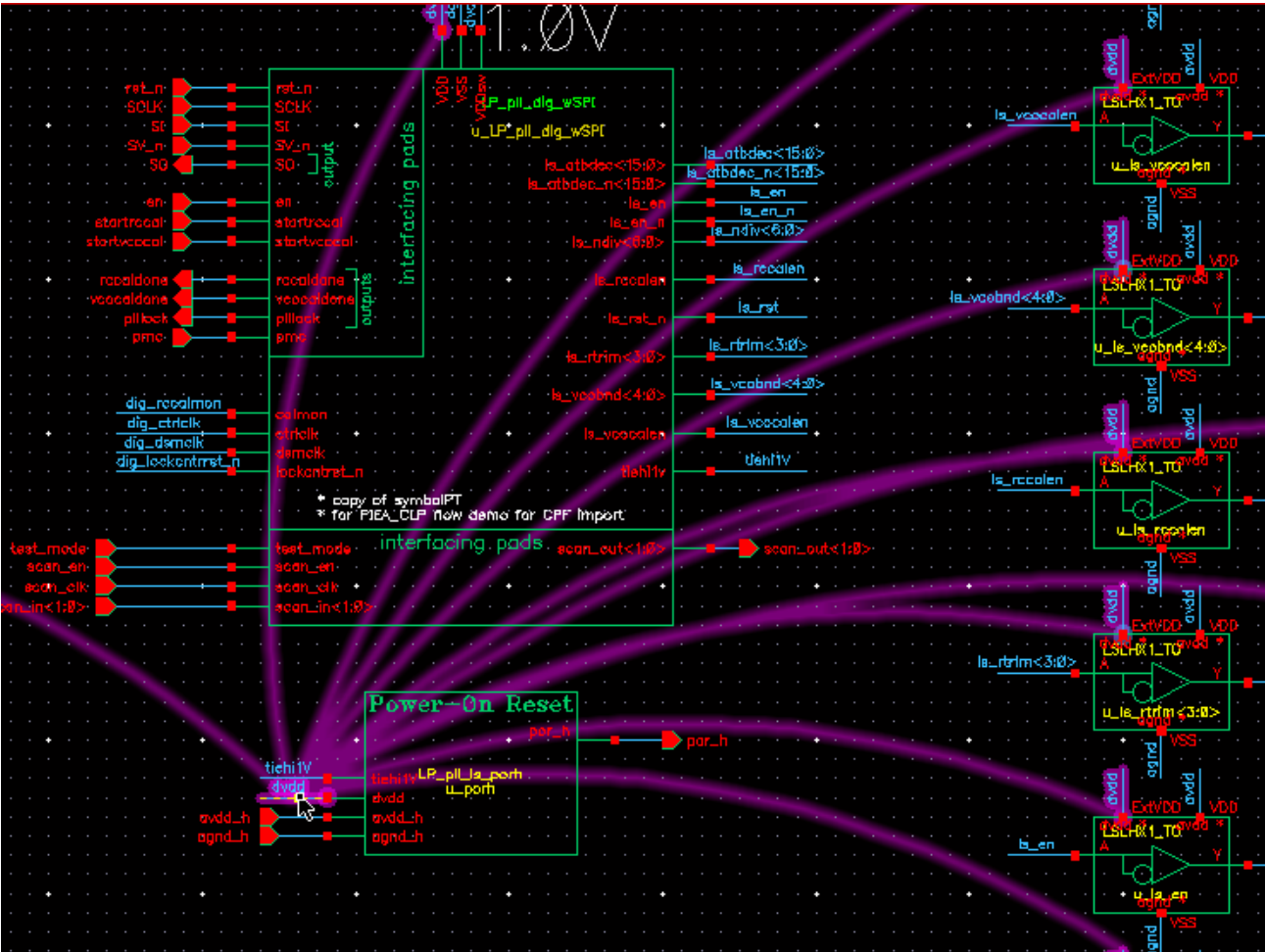
When the pin does not have any connectivity, *unconnected* is displayed on the status bar along with the pin name, as shown below.



If the same net is used in different areas of a design and has several associated objects, arcs are drawn between islands when this common net is highlighted. When you move the mouse pointer over an associated object, all the other objects on the same net are also highlighted.

Getting Started

In addition, a reference dynamic arc is drawn from the object on which the mouse pointer is currently placed to all the other same net islands.



Highlight Colors

When the design is up to date and has not been modified since last save, by default, the following colors are used for highlighting the objects:

- `magenta` is used for highlighting the objects under the pointer as well as the objects attached to the objects under the pointer.
- `cadetBlue` is used for highlighting the member nets and the objects on the member nets.

Virtuoso Schematic Editor User Guide

Getting Started

When the design is not up to date, the default colors for highlighting the objects are the same as those listed above. However, `orange` color is used to highlight the objects attached to the object under the pointer.

You can use the following environment variables to set the default color for highlighting the objects:

■ `schDynamicNetHighlightColor`

Specifies the color for the objects on the same net as the preselected object when the connectivity is up to date. The default color for highlighting is `magenta`.

■ `schDynamicNetHighlightColorAlt`

Specifies the color for the objects on the same net as the preselected object when the connectivity is *not* up to date. The default color for highlighting is `orange`.

■ `schDynamicNetHighlightColorPre`

Specifies the color for the preselected highlighted object that is under the cursor. The default color for highlighting is `magenta`.

■ `schDynamicNetHighlightColorMemNet`

Specifies the color for the objects on the member net of the preselected object. The default color for highlighting is `cadetBlue`.

Extending the Selection Area of a Net

When you want to copy, move, stretch, or delete an entire section of a net, you need to extend the selection area of all physically connected wire segments.

An entire section of a net ends at

- Any three-way or four-way wire connection point
- An instance pin
- A schematic pin
- A change in wire width (bus)

You can use extended selection to select the following:

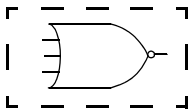
- Several wire segments
- A wire name and its associated wire segments
- A pin and its associated wire segments
- An instance and its associated wire segments

Selecting One Object

To select one object to copy, move, stretch, or delete,

1. Move your cursor over the object.

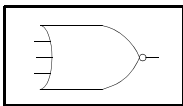
Dynamic highlighting shows you which object will be selected when you click the mouse button.



Dynamic highlighting

2. Click the mouse button.

The selected object is highlighted.



Highlighted object

Selecting Multiple Objects

There are three ways to select multiple objects:

- **Shift-click** (hold down the `Shift` key and click) on each object that you want to add to the set

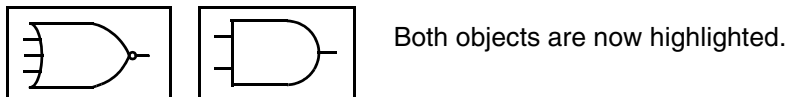
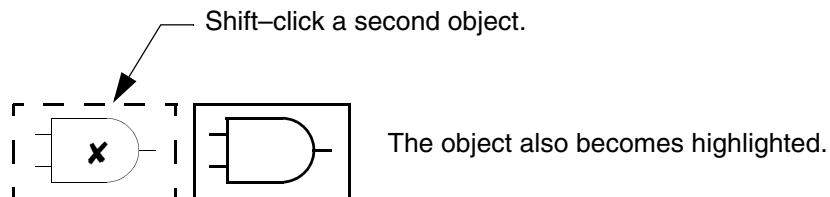
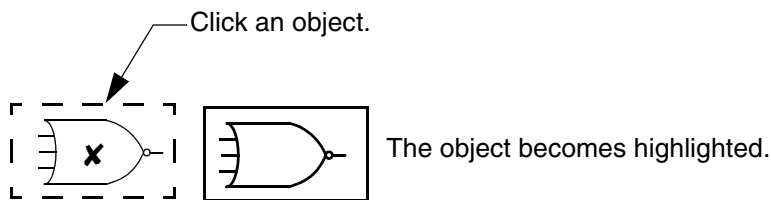
- Drag a box completely around all the objects that you want to select

Note: Partially selected objects won't be copied.

- Select one object, and drag a box completely around the other objects that you want to add to the selected set

Example 1

- Hold down the `Shift` key and click each object you want to add to the set.

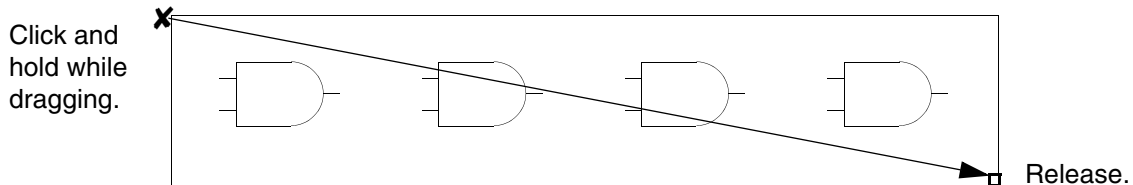


Virtuoso Schematic Editor User Guide

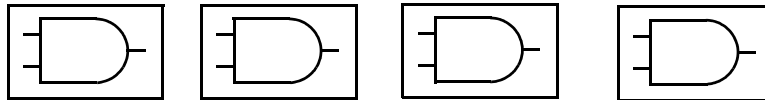
Getting Started

Example 2

- Hold down the left mouse button while you drag a box completely around all the objects you want to select.

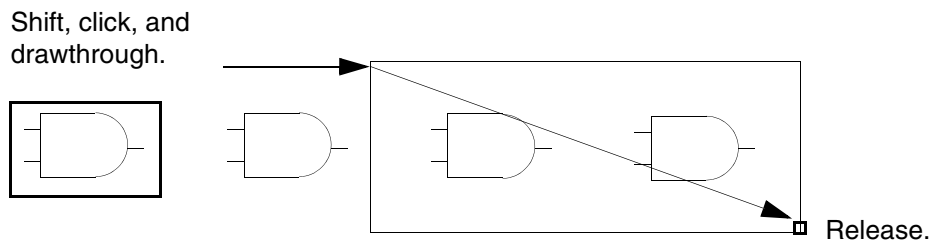


All the objects are now highlighted.

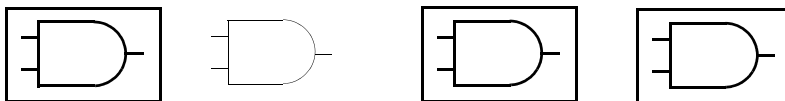


Example 3

- Click the first object and then hold down the `Shift` key and the left mouse button and drag a box completely around the objects you want to add to a selected set.



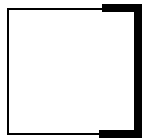
The first and subsequent objects are now highlighted.



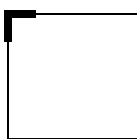
Selecting Parts of Objects

Selecting a part of an object is important when you want to stretch only that part.

Edge of an object



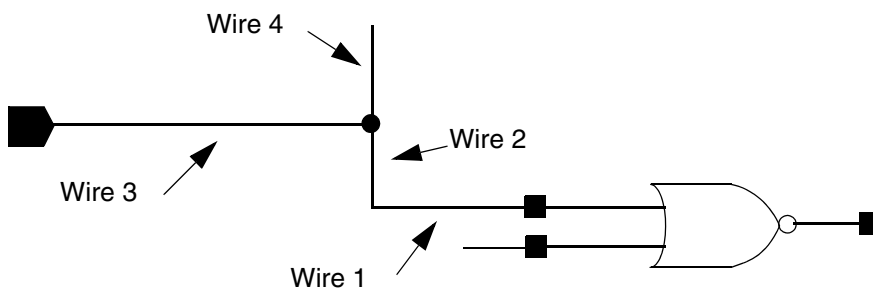
Corner of an object



Wire or line segment endpoint

You can select parts of schematic wires, note shapes, symbol lines, and symbol devices.

Selecting Wires



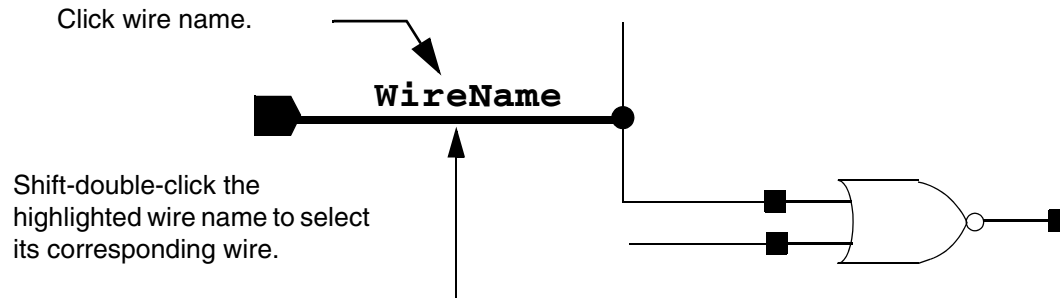
To select several physically connected segments of one wire,

1. Click Wire 1 to select the first wire segment.
2. Shift-double-click Wire 1 to select both Wire 1 and Wire 2.

The extended selection grows to include all segments up to a branch.

3. Shift-double-click again on Wire 1 to select all the connected wires (Wires 2, 3, and 4).
All segments of the network that are physically connected are selected and highlighted.

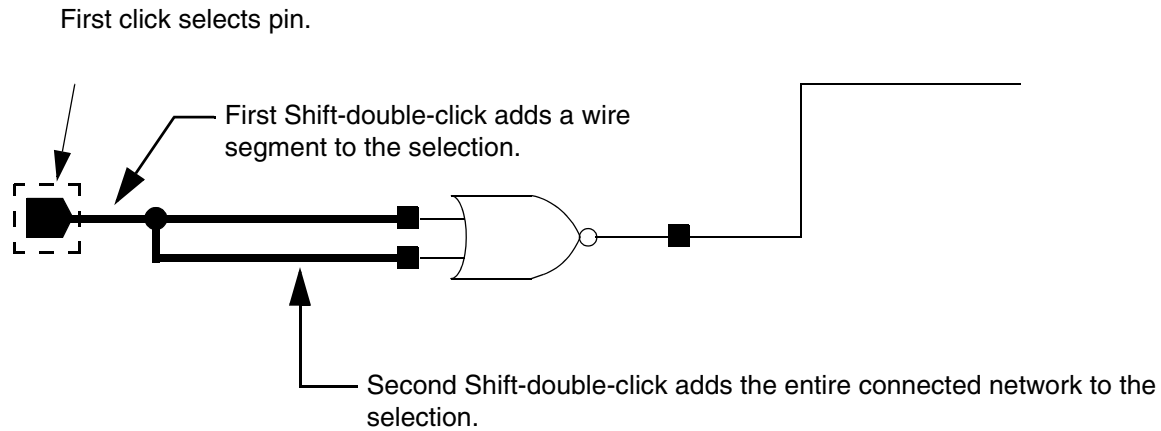
Selecting a Wire Name and Its Associated Wire Segments



To select a wire name and its associated wire segments,

1. Click the wire name to select it.
2. Shift-double-click the already selected wire name to select its associated wire segment.

Selecting a Pin and Its Associated Wire Segments



To select a pin and its associated wire segments,

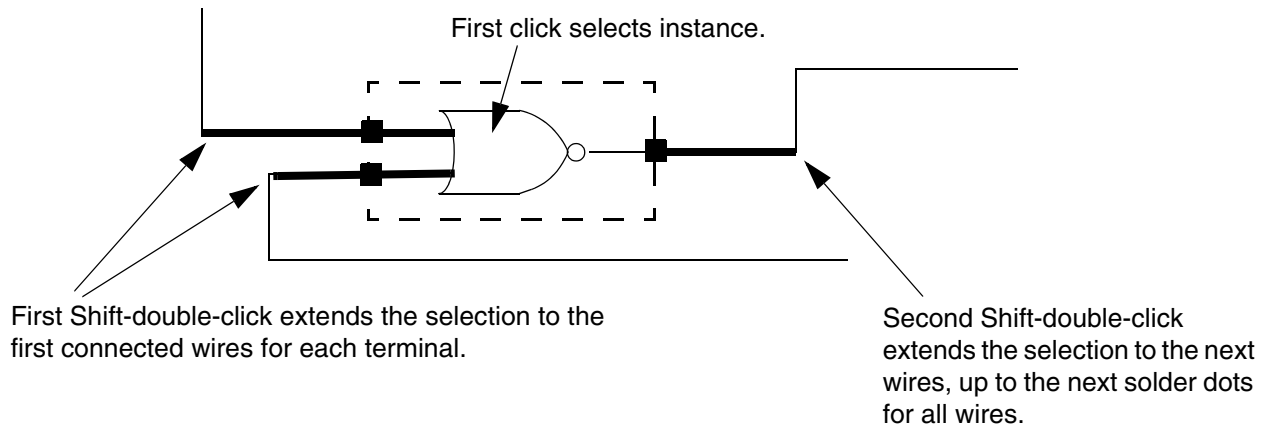
1. Click a pin to select it.
2. Shift-double-click the selected pin.

This selects the wire segment connected to the pin.

3. Shift-double-click again on the selected pin.

This extends the selection area to all physically connected wire segments.

Selecting an Instance and Its Associated Wire Segments



To select an instance and its associated wire segments,

1. Click an instance to select it.
2. Shift-double-click the selected instance.

Extends the selection to the first connected wires for each terminal.

3. Shift-double-click again on the instance.

Extends the selection to the next wires, up to the next solder dots for all wires.

Note: The next Shift-double-click will proceed further up to the next solder dots, unless all wires that are connected to each terminal are already selected. In this case, the next Shift-double-click would de-select all wires, or cycle through the above behaviors of wire selected and de-selection.

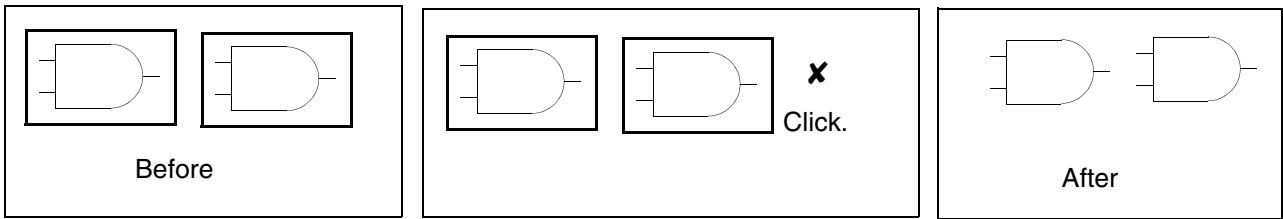
Deselecting Objects

You can use the mouse to deselect objects.

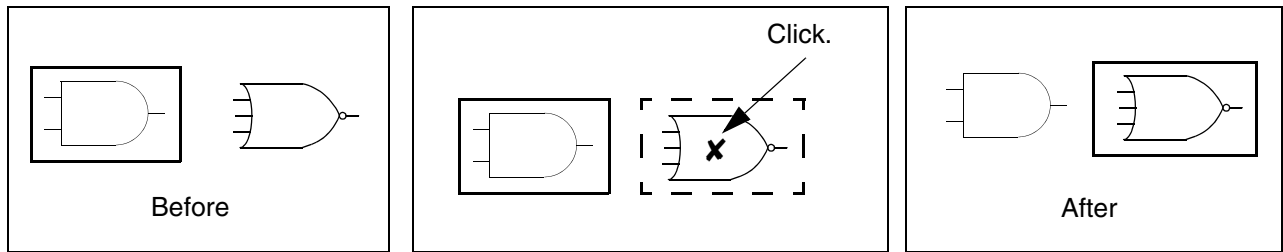
The following table summarizes how you can use your mouse to deselect objects in a design window:

Action	Result
Click once in an open area	Deselects all objects if not within a command. If in a repeat command, use <code>Ctrl-d</code> .
Ctrl-click	Deselects the object under the pointer.

To deselect all objects, click in an open area of your design.



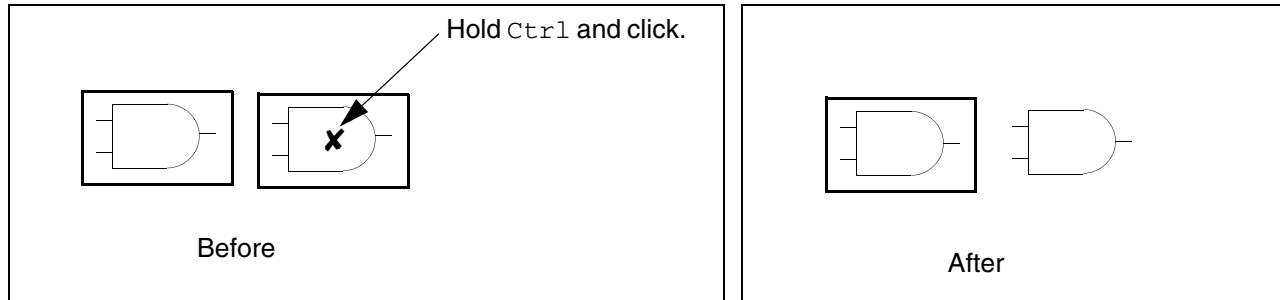
To deselect one object and select another, click the second object.



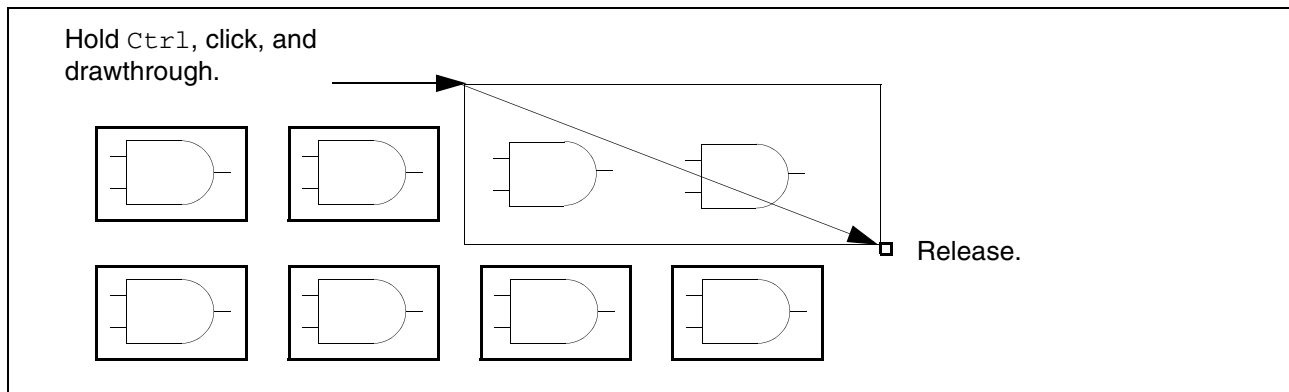
Virtuoso Schematic Editor User Guide

Getting Started

To deselect only one object from a selected set, hold down the `Ctrl` key and click the object you want to deselect.



To deselect more than one object from the selected set, hold down both the `Ctrl` key and the left mouse button while dragging a box around the objects you want to deselect.



Using Strokes

A stroke is a unique shape that you can draw in a cellview by pressing and holding down the right mouse button.

Cadence ships a set of pre-defined strokes for Virtuoso that you can load from the Cadence installation hierarchy. This loading must be done before attempting to use strokes.

For more information, see *Loading and Unloading Strokes* in the [Virtuoso Layout Suite L User Guide](#).

Working With the Design Canvas Window

The following commands describe how to quickly navigate around the main canvas window and view and manipulate windows.

- [Zooming](#)
- [Panning](#)
- [Using the Magnifier](#)
- [Redrawing a Window](#)
- [Viewing the Same Cell in Two Windows](#)
- [Saving View Area Settings in Memory](#)
- [Restoring View Area Settings](#)

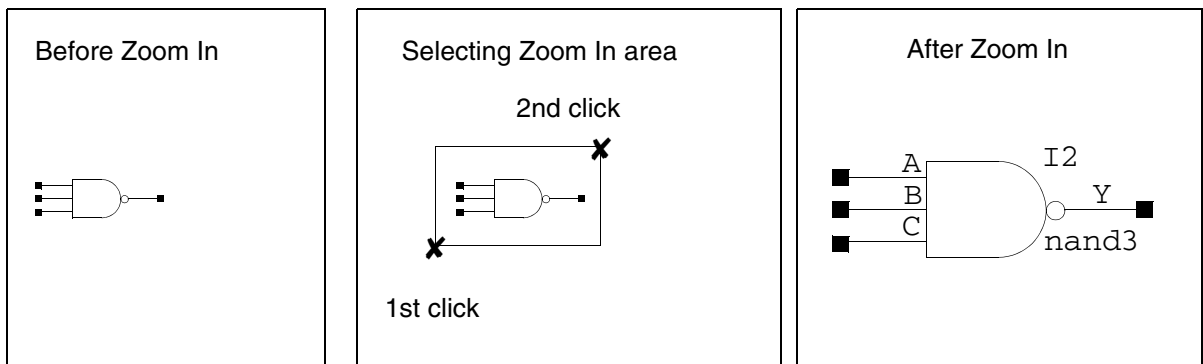
Zooming

Zooming is helpful as you add and edit objects. For example, you can view objects in two windows at the same time and use a different zoom factor in each.

Zooming In Manually

To zoom in (decrease the display area) manually,

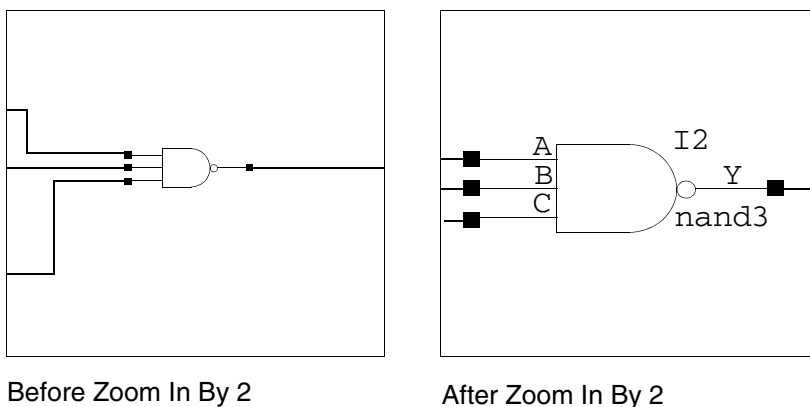
1. Choose *View – Zoom In*.
2. Click your cursor to mark one corner of the zoom area.
3. Click your cursor to mark the other corner of the zoom area.



Zooming In Automatically

To zoom in automatically by 2 (decrease the display area by a factor of 2),

- Choose *View – Zoom In By 2*.



Zooming to Selected Objects

1. Select the object(s) that you want to look closer at.

Note: If no objects have been selected prior to performing the next step, Step 2, you will be prompted to draw a rectangular selection area. This area will then be zoomed in on when Step 2 is performed.

2. Choose *View – Zoom To Sel Set*.

When this option is selected, the window zooms to a maximum magnification so that all the selected objects are visible in the window.

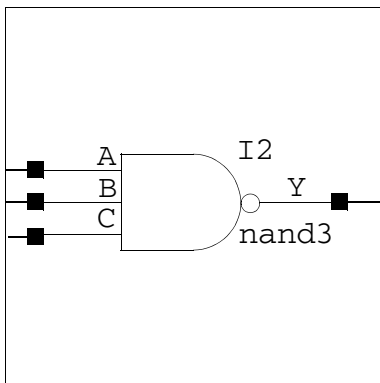
You can also access this functionality from the Multiple Selection and Instance Pop-Up menus, and via the CIW using the `schHiZoomToSelSet` command.

Note: The bindkey for this function is `Ctrl+T`.

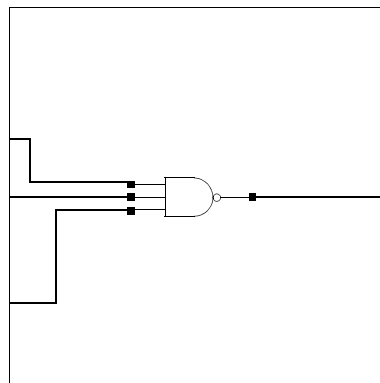
Zooming Out Automatically

To zoom out by 2 (increase the display area by a factor of 2),

- Choose *View – Zoom Out By 2*.



Before Zoom Out By 2

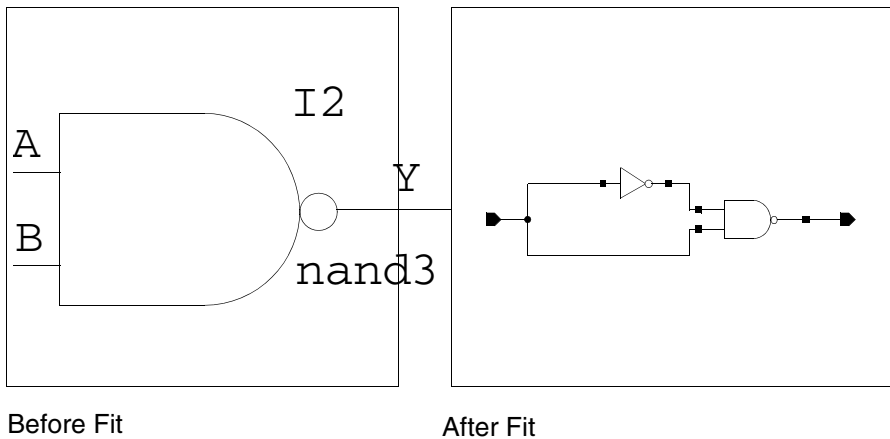


After Zoom Out By 2

Zoom to Fit Automatically

To zoom to fit (resize the display area to show the entire cellview in the window),

- Choose *View – Fit* (or bindkey F).



Dynamic Zoom and Pan

You can use dynamic zoom and pan to go directly to any object/constraint selections made in, for example, the *Navigator* and *Constraint Manager* assistants.

To activate dynamic zoom and pan, to objects on the design canvas, ensure that the *View – Dynamic Zooming* option is active. The default for this option is *on*.

The dynamic zoom setting can be set to one of the following modes: *Pan To Selected*, *Zoom To Selected*, or *Zoom To Fit*.

See also [autoZoomPan](#), [autoZoomMode](#), and [autoZoomScale](#).

To set the preferred dynamic zoom setting:

1. Select *Options – Display*.

The Display Options form is displayed.

Virtuoso Schematic Editor User Guide

Getting Started

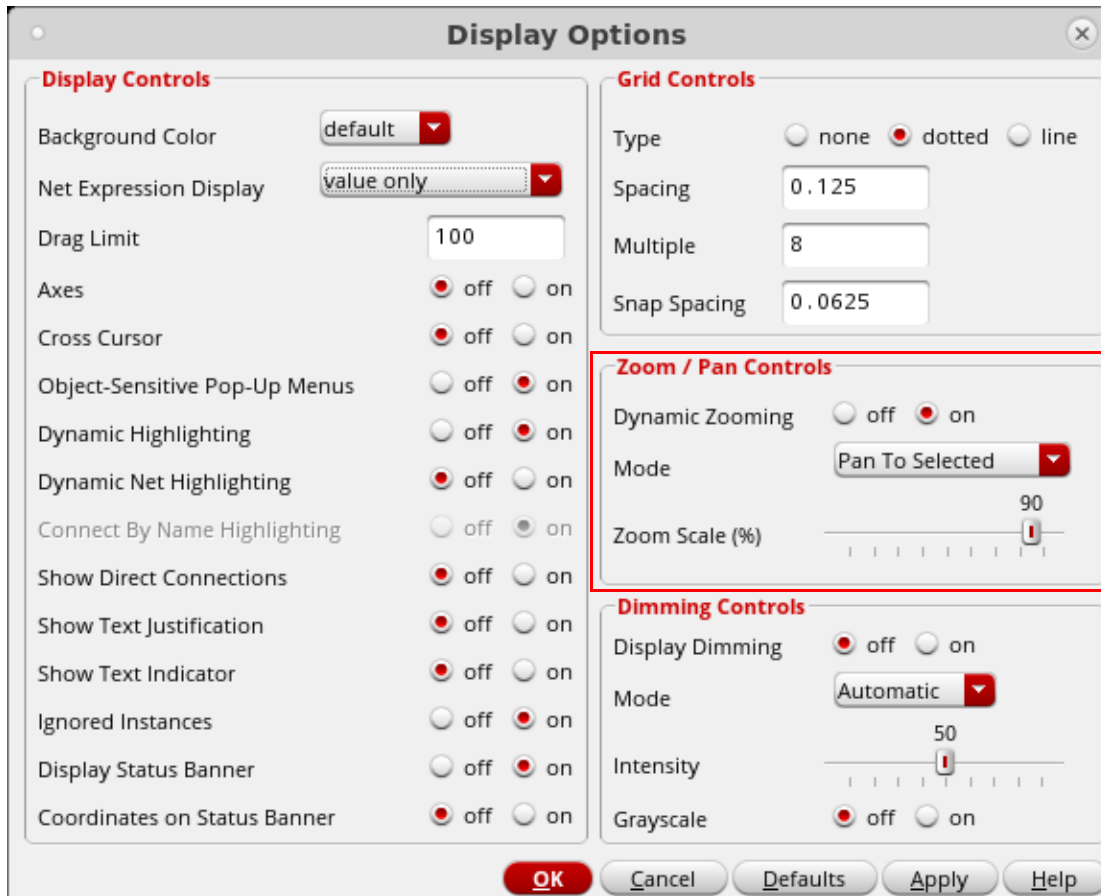


Figure 1-5 Display Options Form with Zoom/Pan Controls for Dynamic Zooming Highlighted

2. Ensure that the *Dynamic Zooming* option is set to *on*.
3. Choose the dynamic zoom *Mode* that you want to use:

☐ *Pan To Selected*

This is equivalent to *View – Pan*. Where, whenever an object, or a set of objects, is selected/cross-selected in an assistant, the view will pan to have the first object visible in the center of the canvas without changing the zoom factor. If the selected object is already visible in the canvas, the view will not pan to center.

☐ *Zoom To Selected*

If this mode is selected, dynamic zoom will perform an auto-zoom (the equivalent of *View – Zoom To Selected*) that ensures that all selected objects are visible in the design canvas window display.

Figure 1-6 Display Dimming Off (Default)

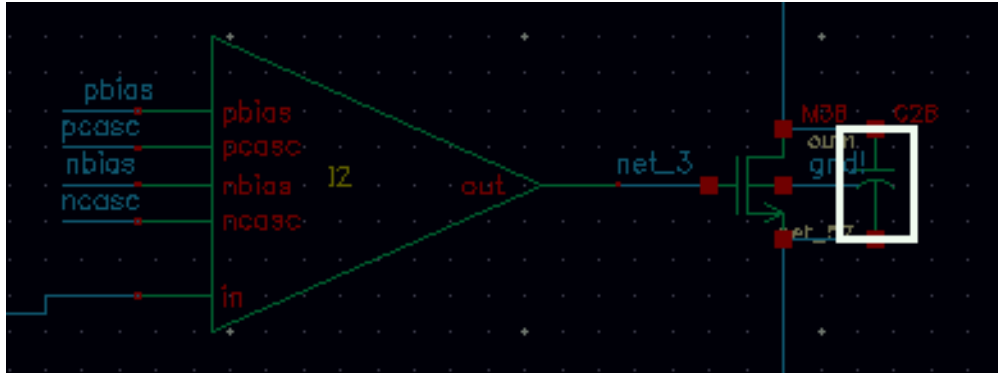


Figure 1-7 Display Dimming On

Display dimming allows you then to dim the display of all objects in the canvas except for those objects that are either currently selected, highlighted, or being probed.

Virtuoso Schematic Editor User Guide

Getting Started

You can control dimming display using the *Dimming Controls* in the Display Options.

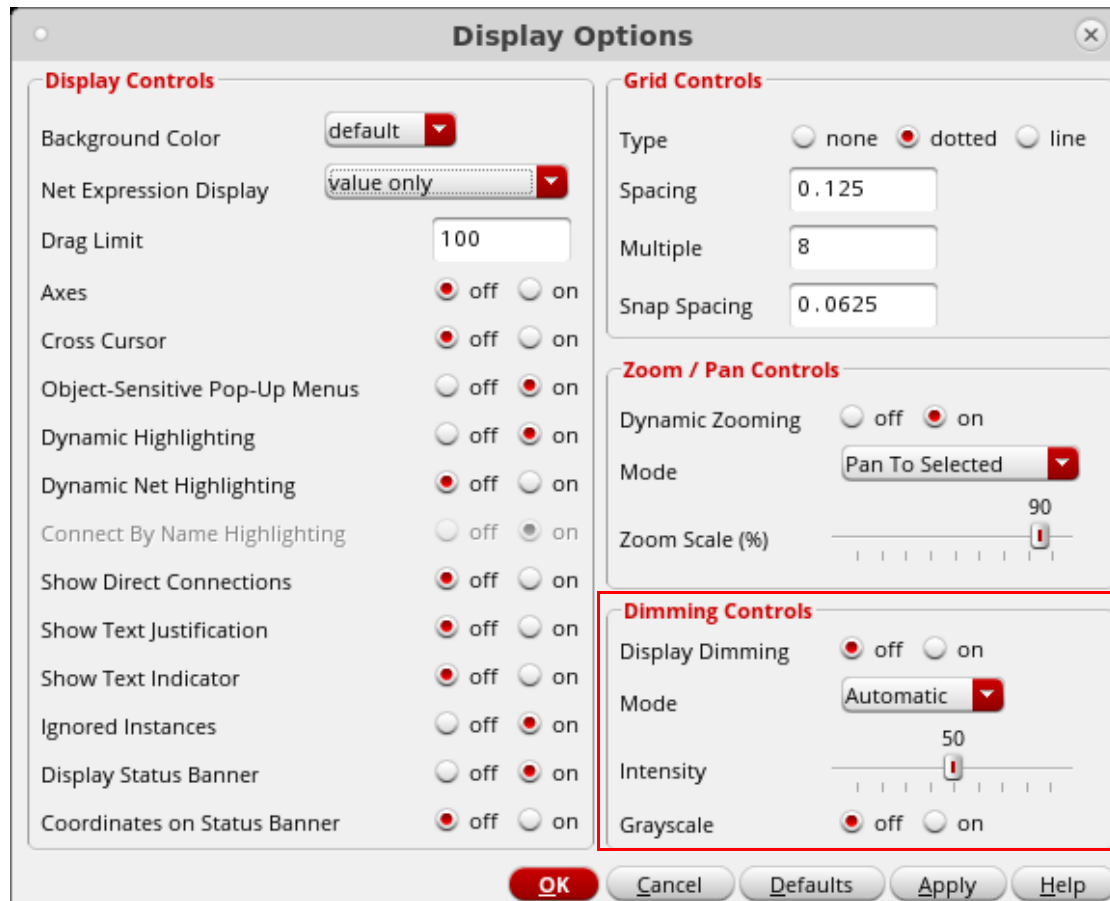


Figure 1-8 Dimming Controls in the Display Options Form

Here:

- *Display Dimming* lets you set dimming *on* or *off*. When *on* is selected, the *View – Display Dimming* option is activated which controls the current dimming setting. See also [dimming](#).
- *Mode* lets you choose whether dimming be *Automatic* (the default); where dimming will only be enabled when at least one object is currently selected, highlighted, or being probed. Or, choose to set dimming to *Always* which will enable dimming permanently (until the option setting is changed). See also [dimmingMode](#).
- *Intensity* lets you set the intensity (brightness) display of the dimmed objects, where 0 equals the dimmed object display being the same as the selected object, up to 100 which will cause the dimmed objects to be “invisible” on the canvas. The default setting is 50. See also [dimmingIntensity](#).

Panning

Panning lets you reposition your design within the editor window, which is helpful when you want to add, view or edit objects.

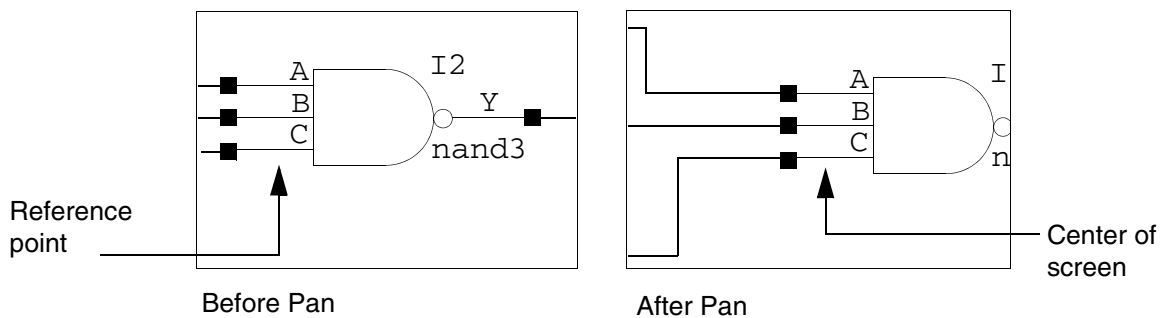
To pan, you can use

- Your cursor and the *View – Pan* command
- The arrow keys on your keyboard
- The editor window scroll bars
- Assigned bindkeys

Panning Using Your Cursor and the View – Pan Command

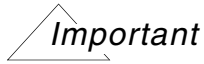
To pan,

1. Choose *View – Pan*.
2. Click the point that you want to appear in the center of the editor window (the reference point).



The image moves and places the reference point in the center of the editor window.

Panning Using Zoom to Selected



See also [autoZoomPan](#).

To automatically pan to an object in the design canvas:

1. Ensure that the *View – Zoom To Selected* menu option is checked.

This option is “on” by default.

2. Select an object in either the *Navigator* and *Search* assistants.

If the object is located outside of the visible pane, VSE will now automatically pan to that object in the design canvas.

Note: *Zoom To Selected* will have no visible effect if the selected object is already visible in the canvas.

Panning Using the Arrow Keys

To pan using the arrow keys on your keyboard,

- Press the arrow keys corresponding to the direction you want to pan.

Panning Using the Editor Window Scroll Bars

To pan using the vertical and horizontal scroll bars on the editor window,

1. From the CIW, choose *Options – User Preferences*.

The User Preferences form appears.

2. Set *Scroll Bars* to *on*.

3. Click *OK*.

Note: This feature works only if you turn on the scroll bars before you open a design.

Using the Magnifier



Video

For a short video overview of the magnifier, see [Using the Magnifier](#).

The magnifier allows you to zoom into a specific area of the design canvas, rather than the design canvas.

When the magnifier is first launched, it will display in accordance with the current, related, variable settings in the `.cdsenv` file for the current root application type (for information on how these settings can be amended for future invocations of the magnifier, see [Magnifier Bindkeys](#) and [Customizing Magnifier Settings](#)).

Note: The magnifier is currently only supported for schematic, symbol, and layout views, with accessibility controlled by the root application type that is current, for example “maskLayout”, `schematicSymbol`, or “schematic” (as returned by `deGetRootType`). The magnifier `.cdsenv` setting values are also stored based on the root application type, therefore if a sub-application of, for example, `schematic` has its own `.cdsenv` variables, then it would still need to have them set at the root application level for them to take effect.

To activate the magnifier on the design canvas:

1. Choose *View – Magnifier*.

A squared magnification area attaches itself to the cursor.

Note: See [Customizing Magnifier Settings](#) for how this area can be altered.

2. Place your cursor over the area in the design canvas that you want to magnify.

Note: By default, the magnifier does not magnify over the design canvas area that it is currently floating over, rather it will magnify the area where the cursor is currently placed (again, see [Customizing Magnifier Settings](#) for information on how to amend this setting).

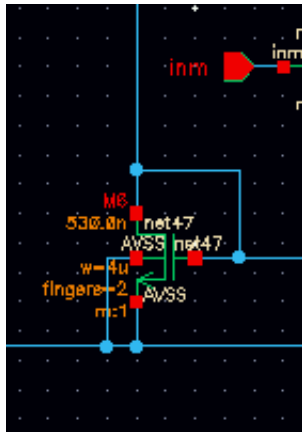
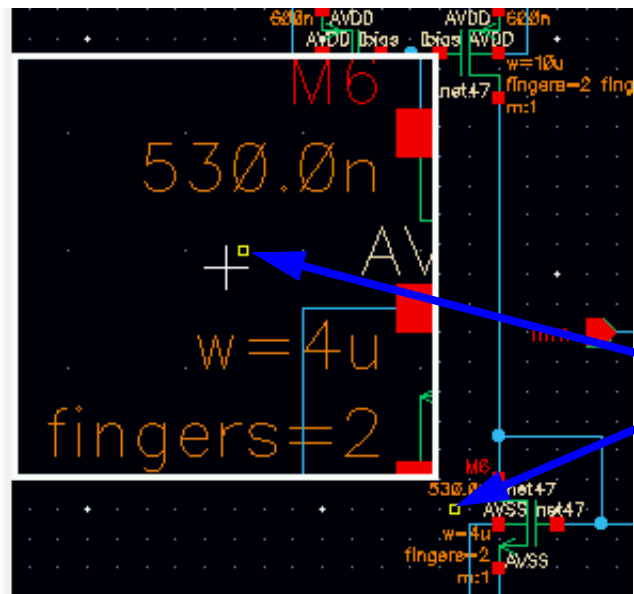


Figure 1-9 Magnifier Currently Off



Magnifier will magnify the area where the cursor is currently placed

Figure 1-10 Magnifier Switched On

To switch design canvas magnification *off*, again choose *View – Magnifier* to uncheck the feature.

Note: See also [Magnifier Functions](#), in the *Cadence User Interface SKILL Reference*, for information on SKILL APIs that can be used for canvas magnification procedures at any time during a Virtuoso session.

Virtuoso Schematic Editor User Guide

Getting Started

Magnifier Bindkeys

The following bindkeys can be used inconjunction with the magnifier:

Action	Bindkey	hiSetBindKey Setting	Related API
Toggle magnifier on/off	<Key>.	<code>hiSetBindKey("Schematic s" "<Key>." "hiToggleMagnifier()")</code>	hiToggleMagnifier
Toggle magnifier anchor on/off	<Key>#	<code>hiSetBindKey("Schematic s" "<Key>#" "hiToggleAnchorMagnifier()")</code>	hiToggleAnchorMagnifier
Update magnifier options	<Key>'	<code>hiSetBindKey("Schematic s" "<Key>'" "hiUpdateMagOptions()")</code>	hiUpdateMagOptions
Scale Magnifier	Ctrl+Shift<Btn4Down> Ctrl+Shift<Btn5Down> Ctrl+Shift<Key>Up Ctrl+Shift<Key>Down	<code>hiSetBindKey("Schematics" "Ctrl Shift<Btn4Down>" "hiScaleMagnifier(1.25)")</code> <code>hiSetBindKey("Schematics" "Ctrl Shift<Btn5Down>" "hiScaleMagnifier(0.80)")</code> <code>hiSetBindKey("Schematics" "Ctrl Shift<Key>Up" "hiScaleMagnifier(1.25)")</code> <code>hiSetBindKey("Schematics" "Ctrl Shift<Key>Down" "hiScaleMagnifier(0.80)")</code>	hiScaleMagnifier
ZoomInMagnifier	Ctrl+shift+scroll up	<code>hiSetBindKey("Schematics" "Ctrl Shift<Btn1Down>"</code>	
ZoomOutMagnifier	Ctrl+Shift+scroll down	<code>hiSetBindKey("Schematics" "Ctrl Shift<Btn2Down>"</code>	

For more details, see [hiSetBindKey](#).

Customizing Magnifier Settings

- ➡ To customize the use of the magnifier select *Options – Magnifier*.

This will display the Magnifier Options form, which will be synchronized to reflect the magnifier settings currently displayed in the window.

Virtuoso Schematic Editor User Guide

Getting Started

Note: You can also access the Magnifier Options form by selecting the bindkey '. If the magnifier options have been altered while the magnifier is active, the form will reflect the changes made through bindkey use (see [Magnifier Bindkeys](#)).

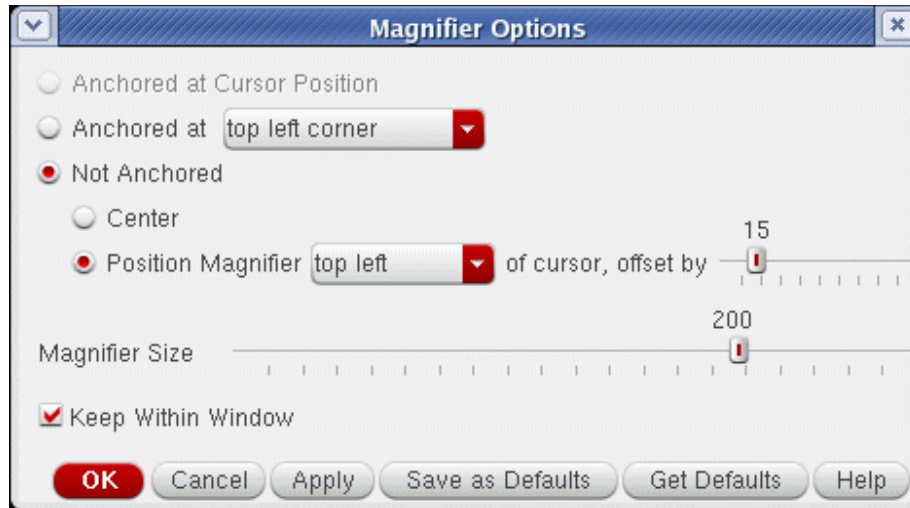
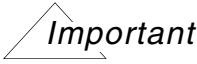


Figure 1-11 The Magnifier Options Form

Option	Description
<i>Anchored at Cursor Position</i>	 Important This option cannot be activated directly from the Magnifier Options form, as doing so would remove the cursor from the canvas. Consequently, it would have no meaning in relation to the magnifier being placed <i>at the cursor position</i> in the canvas. This option is however displayed for consistency purposes, so that the Magnifier Options form values are always synchronized with the state of the magnifier in the current window.
<i>Anchored at</i>	Specifies where the magnifier should be positioned in relation to the cursor tip. The default is <i>top left corner</i>. Choose from: <i>top left corner</i>, <i>top right corner</i>, <i>bottom left corner</i>, and <i>bottom right corner</i>. Note: Selecting <i>Anchored at</i> will disable the <i>Not Anchored</i> options.
<i>Not Anchored</i>	Specifies that the magnifier is not anchored to the cursor tip. Instead, it can be set so that the cursor is either placed in the <i>Center</i> of the magnification area, or customized as per the current <i>Position Magnifier</i> settings. Note: Selecting <i>Not Anchored</i> will disable the <i>Anchored at</i> option.
☐ <i>Center</i>	Specifies that the cursor is placed in the center of the magnification area.
☐ <i>Position Magnifier...</i>	Choose where you want the magnifier to be positioned on the design canvas in relation to the current position of the cursor, along with its proximity to the cursor. Choose from: <i>top left</i>, <i>top right</i>, <i>bottom right</i>, and <i>bottom left</i>, and choose an <i>offset by</i> setting in a range of 0-100 (default 15). Note: An <i>offset by</i> setting of 0 will position the magnification area directly at the cursor tip.

Option	Description
Magnifier Size	Sets the pixel size of the magnification area that is to be attached to your cursor when the magnifier is active. The range is 50 to 250, with a default of 200. Note: The specified size is the length, in pixels, of one side of the square magnifier.
Keep Within Window	Specifies that the magnifier will always display in full within the confines of the design canvas area. When this option is de-selected, the magnifier will be free to only partly display within the canvas window.

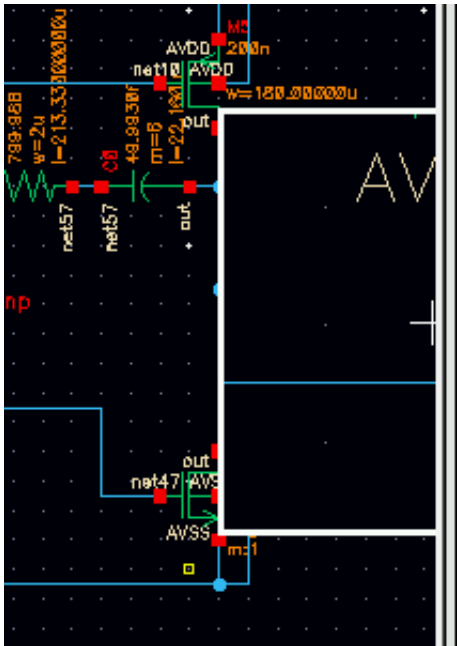


Figure 1-12 Magnifier Not Confined to Canvas Area

	Note: The current rectangular shape of the magnifier indicates that part of it is “off” canvas.
OK	Saves the current magnifier settings for the current window, and will also affect the magnifier if it is currently displayed (or the next time it is displayed). The form is also closed.
Cancel	Closes the form without applying any setting changes that may have been made.

Virtuoso Schematic Editor User Guide

Getting Started

Option	Description
<i>Apply</i>	Saves the current magnifier settings for the current window, and will also affect the magnifier if it is currently displayed (or the next time it is displayed). The form is not closed.
<i>Save as Defaults</i>	Saves the latest magnifier option settings to the <code>.cdsenv</code> file, for the current root application. These newly stored defaults can then be retrieved at any time when <i>Get Defaults</i> is selected. Note: The saved <code>.cdsenv</code> values will be reflected the first time that a magnifier for the given route application is displayed in a window.
<i>Get Defaults</i>	Restores magnifier settings to the defaults for the current root application, ignoring any amendments that may have been made since <i>Save as Defaults</i> was last selected. Note: <i>OK</i> or <i>Apply</i> must be selected in order for retrieved defaults to affect the magnifier in the canvas.
<i>Help</i>	Displays this help information on the magnifier.

Redrawing a Window

To redraw a window,

- Choose *View – Redraw*.

Viewing the Same Cell in Two Windows

Use the *Copy Window* command to see the same cellview from different zoom and pan settings simultaneously. This feature lets you work in one area of the cellview and still have a reference window showing the entire cellview.

To view the same cellview in two windows,

- Choose *Window – Copy Window*.

A copy of the current cellview opens in a new window.

The copy has the same title as the original. Any change you make in either window affects both windows.

Note: Commands started in one window are also active in the other window. For example, when you use a *Create – Wire* command, you create wires inside both windows.

Cycling Between Windows and Views

You can cycle between the last ten window zoom or pan views that are in memory using either:

- *View – Save/Restore – Previous View*
- *View – Save/Restore – Next View*

Note: You can also use the equivalent [`hiPrevWinView`](#) and [`hiNextWinView`](#) in the CIW.

For more information on retrieving previously created views see [Using Bookmarks and Views](#) in the *Virtuoso Design Environment User Guide*.

For more information on moving through a design hierarchy see [Navigating Cellviews and Hierarchies](#) in the *Virtuoso Design Environment User Guide*.

Saving View Area Settings in Memory

You can temporarily store your current viewing area. This procedure is useful when you want to look at one section of the design, move to a new section, and then return to the first section.

To temporarily save your viewing area,

1. Choose *View – Save/Restore – Save View*.

The Save View form appears.

2. In the *Name* field, type the name you want to assign to your current view.
3. Click *OK*.

Restoring View Area Settings

To restore a saved viewing area,

1. Choose *View – Save/Restore – Restore View*.

The Restore View form appears, listing the names you assigned with the *View – Save/Restore – Save View* command.

2. Click once on the name.
3. Click *OK*.

The previously saved viewing area appears.

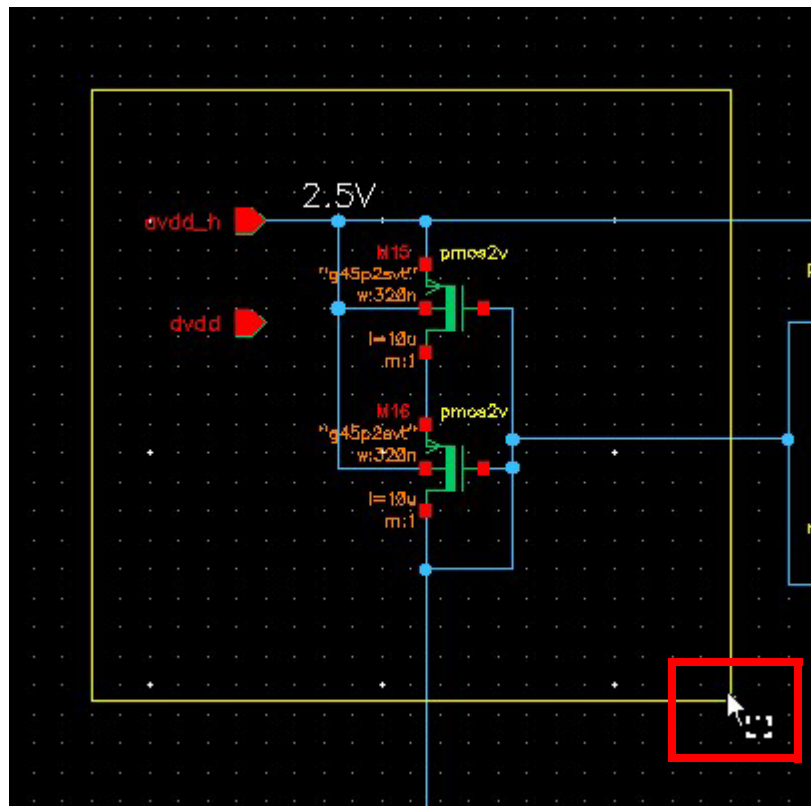
Click-Lock

This functionality allows you to switch between the LMB and RMB operation. You can perform various RMB commands and switch back to the LMB commands, such as zooming the selected area on the canvas. When click-lock is enabled, you do not need to hold the left mouse button anymore to keep the drag command active for the selection box. Instead, you can release the left mouse button and complete the command.

This functionality is useful for commands that involve left mouse drag, such as drawing a rectangle shape. To set the opposite corner of the selected box or rectangle, click the left mouse button again.

Refer to the following steps for creating a selection box using click-lock.

1. Click the left mouse button and drag to create a rectangle.
2. Right-click while holding the left mouse button.
3. Release the left mouse button. Lock has been activated now, as shown below.



4. Perform the required RMB action on the canvas.

Showing and Hiding Labels and Notes

You can use a number of options in the *View* menu to determine (toggle on and off) what note and label information is to be shown on the design canvas.

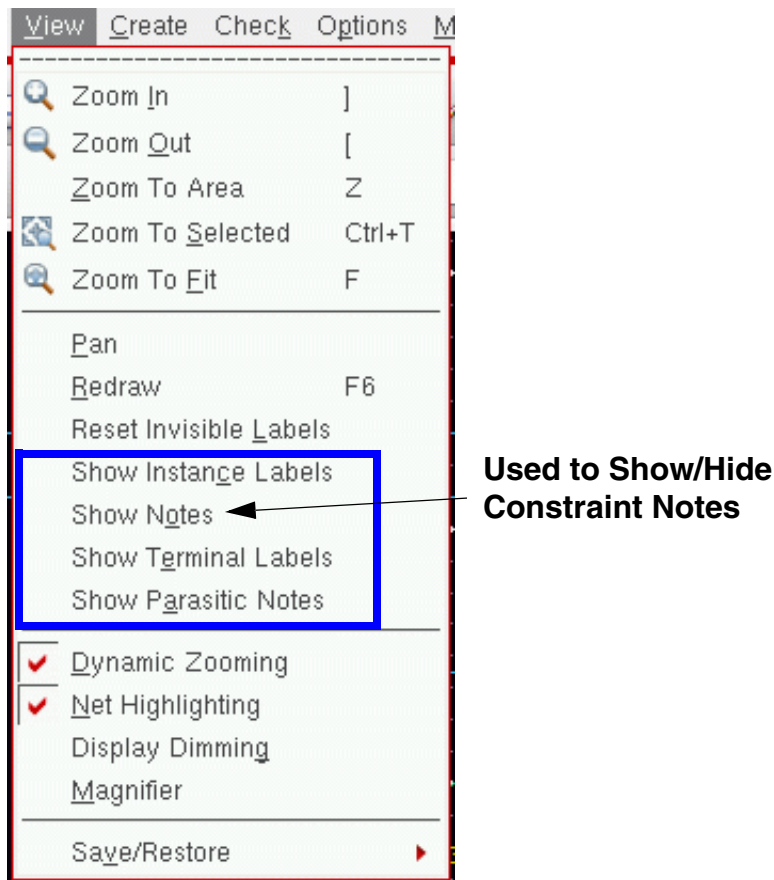


Figure 1-13 Show/Hide Labels and Notes

Note: The visibility of design intent annotation labels and shapes is controlled by the Design Intent toolbar in Schematics XL. For more details, see the [Virtuoso Design Intent User Guide](#).

The Component Browser

You can use the Component Browser to view and select the contents of the libraries you specify in your `cds.lib` file. As an alternative, you can use the Library Browser.

The following sections provide information about the Component Browser.

- [Accessing and Using the Component Browser](#)
- [Changing the Browser Type Option to the Component Browser](#)
- [Opening the Component Browser](#)
- [Opening the Add Instance Form Instead of the Component Browser](#)
- [Listing Components from Several Libraries](#)
- [Changing the Component Browser Display](#)

Accessing and Using the Component Browser

Note: The Component Browser is available only in conjunction with the *Create – Instance* command, and only after you change the default setting on the Editor Options form. By default, the browser type is set to the Library Browser.

You can use the Component Browser:

- To view the contents of the libraries specified in your `cds.lib` file
- To select the components you want to add to your design

Changing the Browser Type Option to the Component Browser

To access the Component Browser,

1. Choose *Options – Editor* to display the Editor Options form.

The screenshot shows the 'Editor Options' dialog box with the 'Command Controls' tab selected. The 'Add Instance Browser Type' option is set to 'component'. Other options include 'Show Schematic Views in Browser' (off), 'Block Pin Direction Rule' (edge), 'Wire Snapping' (on), 'Automatic Solder Dots' (on), 'Solder Dot Wire Overlap' (0.03125), 'Symbol Generator Pin Sorting' (geometric), 'Add Symbol Pins as Instances' (off), 'Repeat Commands' (on), 'Cross Selection' (off), 'Gravity' (on), 'Minimum Marker Size' (0.05), 'View Name List' (schematic symbol), and 'Auto Wire Stub for Bus Terminals' (wide). The Constraint Notes Controls section is also visible at the bottom.

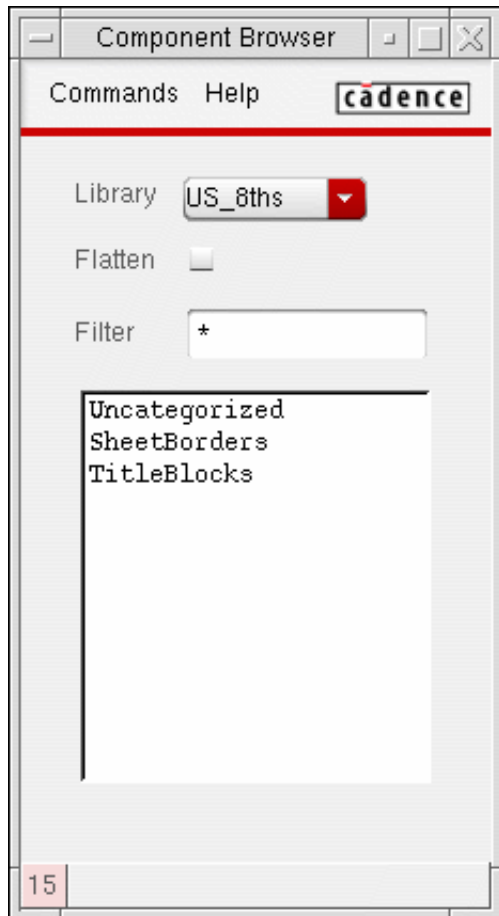
2. In the *Add Instance Browser Type* option, select *component*.
3. Click *OK*.

Opening the Component Browser

To open the Component Browser to add component instances from a library,

- Choose *Create – Instance*.

The Component Browser form appears.



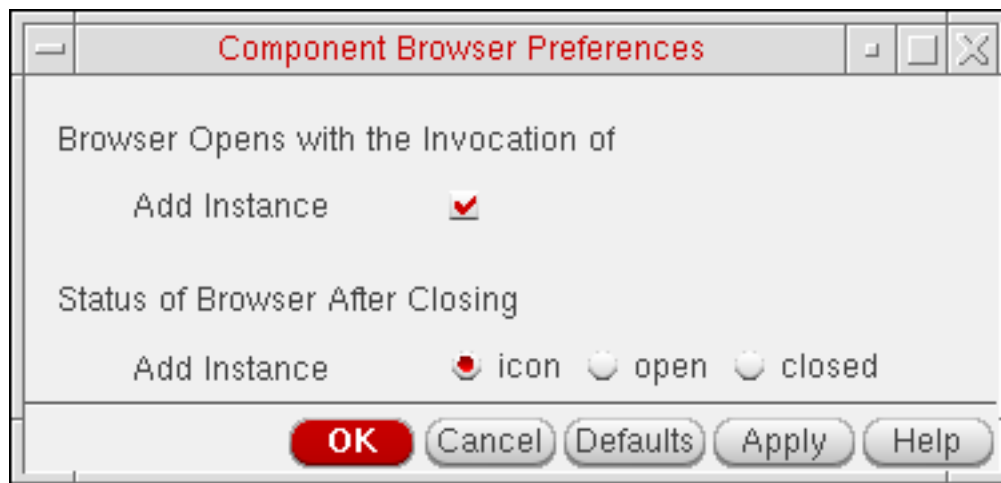
The form shows the top-level categories for the US_8ths library.

Opening the Add Instance Form Instead of the Component Browser

If you have specified the Component Browser as the default browser but want to open the Add Instance form first when you choose *Create – Instance*,

1. From the Component Browser, choose *Commands – Preferences*.

The Component Browser Preferences form appears.



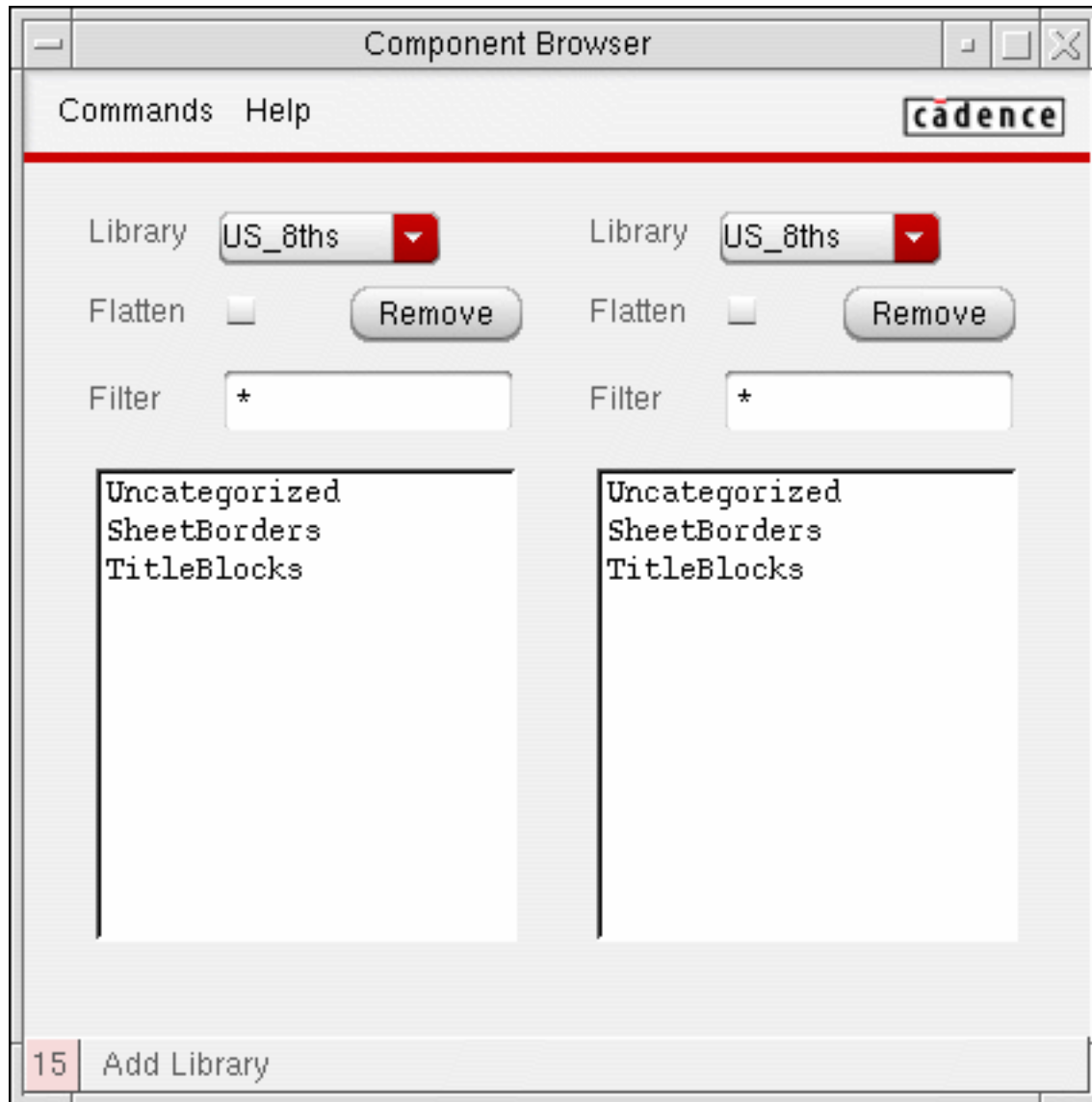
2. Turn off the *Browser Opens with the Invocation of Add Instance* option.
3. Click *OK*.

Listing Components from Several Libraries

To list components from several libraries,

1. From the Component Browser, choose *Commands – Add Library*.

The Component Browser form expands to display the contents of another library.



2. From the *Library* cyclic fields, choose the libraries that you want to browse.

Changing the Component Browser Display

To display all components within categories of a library,

- Choose the *Flatten* option.

By default, *Flatten* is deselected, so the Component Browser displays a hierarchical list of components grouped by categories.

To limit the display to specific components,

- Type a selection pattern in the *Filter* field.

By default, the *Filter* field contains an asterisk, so the Component Browser displays all library components.

The following table describes filter patterns.

Key	Filter Pattern Description
<code>c</code>	Matches any ordinary character (not a special character listed here).
<code>[c...]</code>	Matches the string enclosed in the square brackets (called a set). If the first character in the set is <code>^</code> , it matches a character not in the set. Specify an inclusive set of characters by separating the characters with a dash.
<code>\$</code>	At the end of a filter string, matches the end of a pattern. Used elsewhere in the pattern, is treated as an ordinary character.
<code>*</code>	When used alone, lists all components. When used as the last character in a pattern, matches zero or more occurrences of a pattern. For example, to list all components beginning with the letter <i>c</i> , use the filter <code>c*</code> .
<code>+</code>	When used as the last character in a pattern, matches one or more occurrences of a pattern.
<code>\(...\)</code>	Saves the enclosed expression in a numbered register (from 1 to 9).
<code>.</code>	Matches any character.
<code>\</code>	Preceding a special character, matches the character literally.
<code>\n</code>	Matches the contents of the <i>n</i> th register from the current regular expression.
<code>\<...\></code>	Restricts the pattern matching to the beginning and/or end of a word.
<code>rs</code>	A composite regular expression. Matches the longest match of <i>r</i> followed by a match for <i>s</i> .
<code>^</code>	At the beginning of a filter string, matches the beginning of a pattern. Used elsewhere in the pattern, is treated as an ordinary character.

Virtuoso Schematic Editor User Guide

Getting Started

Connectivity and Naming Conventions

Connectivity refers to the way in which components and pins are linked together in a non-ambiguous manner. The connectivity in a schematic is represented by wires linking the component pins to the pins of other components. Connectivity can also be established and represented by assigning a common name to multiple pins.

This chapter discusses the following topics:

- [Updating Connectivity](#)
- [Connectivity with Objects](#)
- [Instance Naming Conventions](#)
- [Wire Connections and Wire Naming Conventions](#)
- [Multiple-Bit Wire Connections and Naming Conventions](#)
- [Pin Connections and Pin Naming Conventions](#)
- [Patch Cord Connections and Patch Cord Naming Conventions](#)
- [Inherited Connections](#)
- [Summary of Naming Conventions](#)
- [Supply and Ground Sensitivity](#)
- [Equivalent Connectivity Time](#)

Updating Connectivity

The schematic extractor updates connectivity such that only the minimum necessary changes to the database connectivity are made.

- There are two modes in which the connectivity of a design can be updated: incremental and batch.

The default is incremental connectivity mode, which means that the connectivity in your design is updated automatically as you create and edit the design. In batch connectivity mode, you must update the connectivity of the design explicitly by running one of these check commands: *File – Check and Save*, *Check – Current Cellview*, or *Check – Hierarchy*.

You switch modes using the *Incrementally update connectivity while editing* check box on the *Connectivity* tab of the Editor Options form or by setting the corresponding environment variable incrementalConnTimeout.

When you update the connectivity in your design, if issues are found that prevent the connectivity from being made, error and warning messages are generated on the canvas, Annotation Browser, and CIW. For more information on these messages and what you can do to resolve them, see Troubleshooting Schematic Connectivity.

Incremental Connectivity Extraction Status

The current status of incremental connectivity mode is indicated on the status bar at the bottom of the schematic window.

- *On* means that incremental connectivity is enabled.
- *Off* means that incremental connectivity is disabled.
- *Paused* means that incremental connectivity has been paused.

Resuming a Paused Incremental Connectivity Extraction

The status of incremental connectivity mode changes to *Paused* when a batch connectivity check is required, for example, when an existing schematic is opened for the first time in incremental connectivity mode. Running a batch extraction sets a baseline for the subsequent incremental updates. A batch connectivity check is also required to re-establish the connectivity baseline if incremental connectivity mode was disabled and then re-enabled.

Incremental connectivity extraction can also be automatically paused if you have made a large or complex edit to the design that is taking longer than the permitted maximum time to

Virtuoso Schematic Editor User Guide

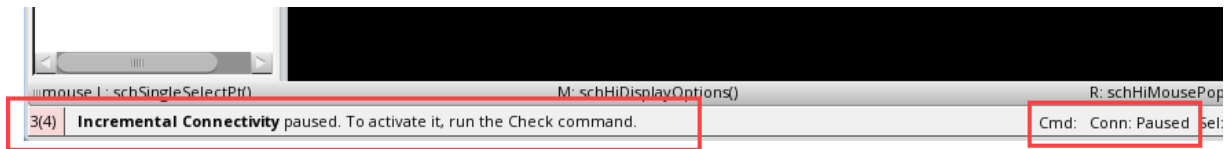
Connectivity and Naming Conventions

extract. By default, any extraction requiring longer than 0.1 seconds is deemed too long and causes incremental connectivity extraction to be paused. You can adjust this timeout value using the `incrementalConnTimeout` environment variable.

To resume incremental connectivity extraction:

1. Open an existing cellview.

A message similar to the following appears on the status bar at the bottom of the schematic window and the incremental status is indicated as *Paused*.



2. Choose *File – Check and Save* or *Check – Current Cellview* to run a batch connectivity extraction.

The message clears from the status bar, and the incremental status changes from *Paused* to *On*.



Cellviews Ignored by Incremental Connectivity Extraction

The schematic connectivity extractor ignores schematic Pcell supermasters and submasters. This is because a schematic Pcell can create internal connectivity that does not match its representation on the canvas. For example, a Pcell might directly connect instance terminals and terminals to nets without any of the corresponding graphical wires and pins. Extracting this connectivity in a way that matches the canvas representation risks overwriting the intended result.

The extractor also ignores schematics where the `schType` property is set to `noSchEdit`. This property indicates that a schematic should not be processed by the schematic checker, geometry management system, and so on and is useful for generated schematics with special or incomplete connectivity or graphics.

Understanding when Incremental Connectivity Extraction Runs

This section describes how incremental connectivity extraction behaves when you are running Virtuoso in non-graphical mode, for example, by using the SKILL interface or by launching Virtuoso with the `-replay` option.

Incremental Connectivity Extraction and SKILL

Incremental connectivity extractions run when Virtuoso becomes idle after finishing each top-level SKILL command. For example, if you have a SKILL procedure that builds a schematic:

```
// create.il
procedure(createSchematic()
  ...build a schematic using various SKILL functions...
  schCheck(cv)
  cv
)
```

And you run the following SKILL command:

```
load("create.il")
sch = createSchematic()
```

In this case, incremental connectivity extraction runs only after the `sch = createSchematic()` line finishes. It does not run after each individual command inside `createSchematic()`. Additionally, incremental connectivity extraction only runs if there is something to extract. Because `createSchematic()` ends with a call to `schCheck()`, no further extraction is required. To update connectivity inside a SKILL procedure, call `schCheck()` explicitly.

Incremental Connectivity Extraction and Replay Files

If you have a replay file containing individual editing commands, for example:

```
// replay.il
\i schHiCreateInst()
...
\i schHiCreateWire()
...
```

And you run the following command:

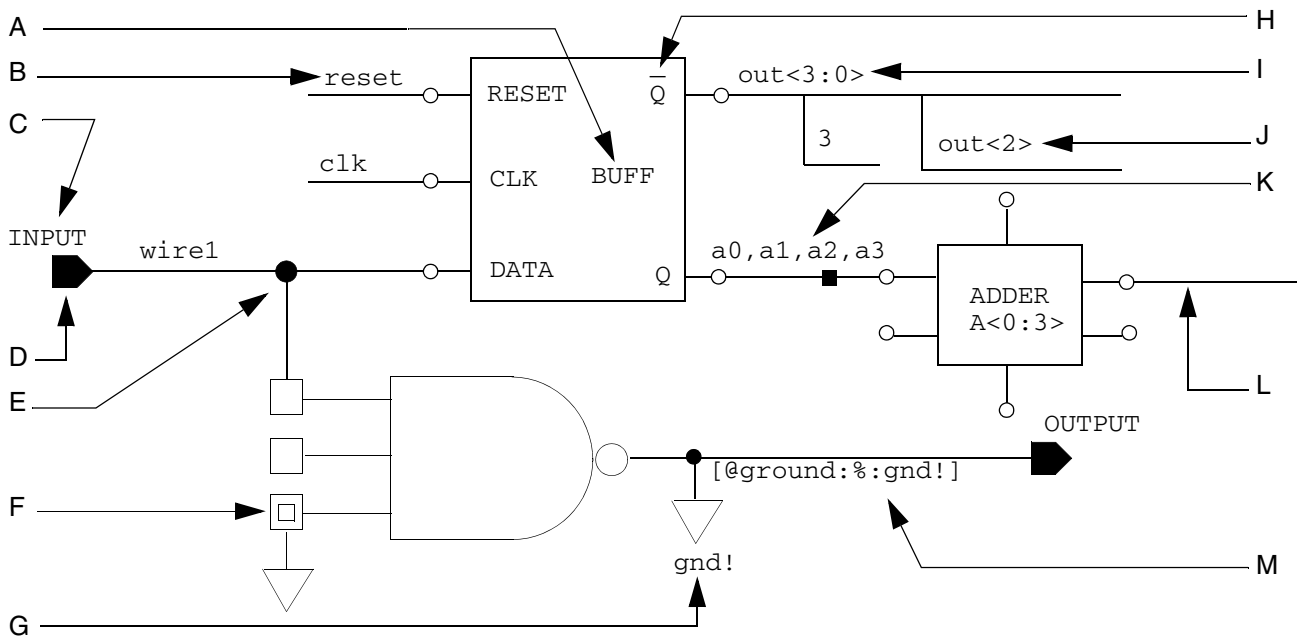
```
virtuoso -replay replay.il
```

In this case, incremental connectivity extraction runs after each line, because all lines are considered to be top-level commands.

Connectivity with Objects

Connectivity is the logical connection between signals in a design. Connectivity is derived (extracted) from the physical connections found in a schematic. Pins and wires implement physical connections. Terminals and signals implement logical connections. Pins and terminals indicate which signals in a cellview are imported from or exported to other levels of a design hierarchy.

A complete schematic design provides input data for a variety of design tools, such as simulation and layout. It is essential, therefore, that you establish connectivity in your schematic designs. The following diagram shows connectivity and naming conventions.



Key	Connectivity / Name
A	<u>Instance name</u>
B	<u>Single wire names</u>
C	<u>Pin name</u>
D	<u>Wire-to-pin connection</u>
E	<u>Wire-to-wire connection</u>
F	<u>Pin-to-pin connection</u>
G	<u>Global net name</u>

Key	Connectivity / Name
H	<u>Overbars in a name</u>
I	<u>Wire name with multiple bits</u>
J	<u>Bus tap</u>
K	<u>Multiple-bit naming convention</u>
L	<u>Wire-to-iterated instance connection</u>
M	<u>Net expression label</u>

Instance Naming Conventions

You can name instances using any one of the following naming conventions:

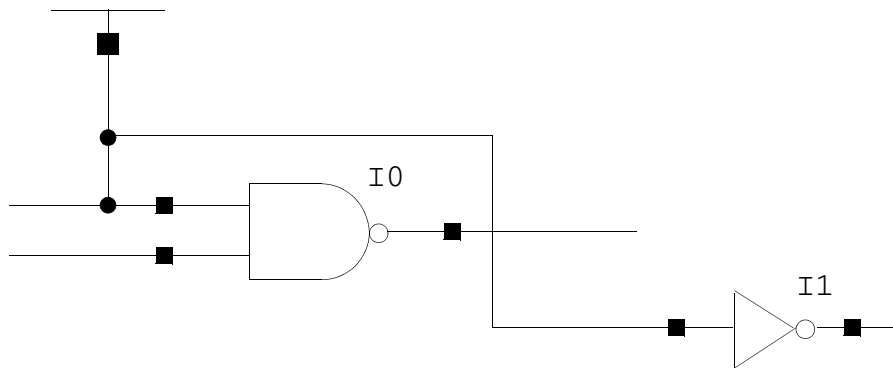
- [System-Generated Names](#)
- [User-Defined Names](#)
- [Iterated Names](#)

Related Topics

[Summary of Naming Conventions](#)

System-Generated Names

If you do not name an instance explicitly, the schematic editor assigns a default instance names by using **I** plus a number. The schematic editor automatically increments the instance number starting from 0.



The advantage of system-generated instance names is that you never have to type a name on the Add Instance form. You can add many instances very quickly and automatically.

You can also change the instance name prefix (**I**) to another letter. For more details, see [Changing the Default Instance Name Prefix](#).

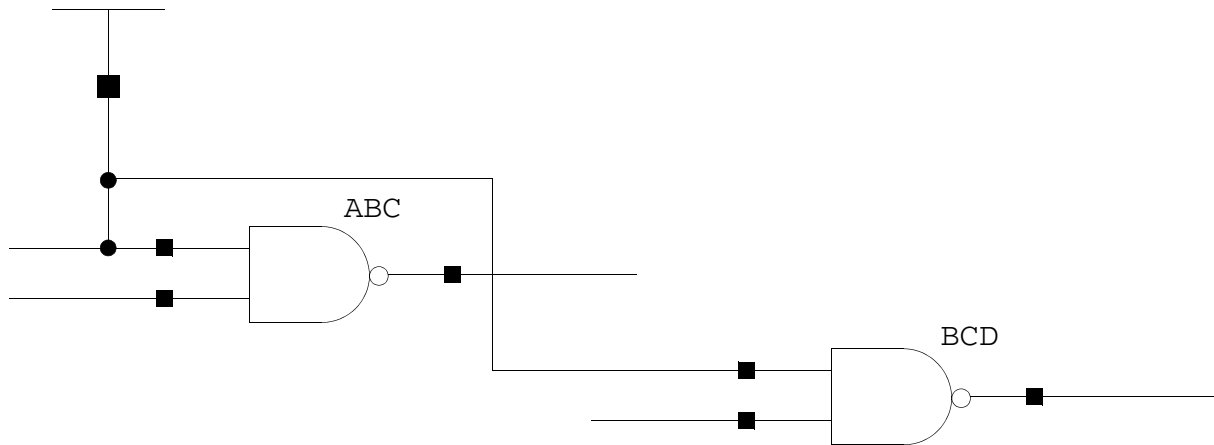
User-Defined Names

The advantage of user-defined names is that you can explicitly name instances in your design. Examples of user-defined names are as follows:

```
A0 A1 A2 A3  
ABC DEF HIJ  
IOLATCH FLIPFLOP
```

Each user-defined name you add must be a string of one or more characters. You must name an instance when you create it using the Add Instance form. If you do not name the instance explicitly, the system generates a name for you.

On the Add Instance form, separate each instance name by a space; for example, ABC BCD.



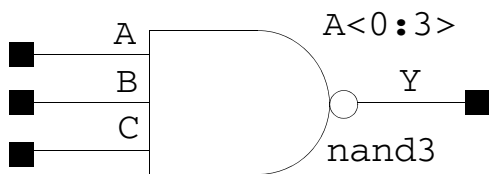
Related Topics

[Summary of Naming Conventions](#)

Iterated Names

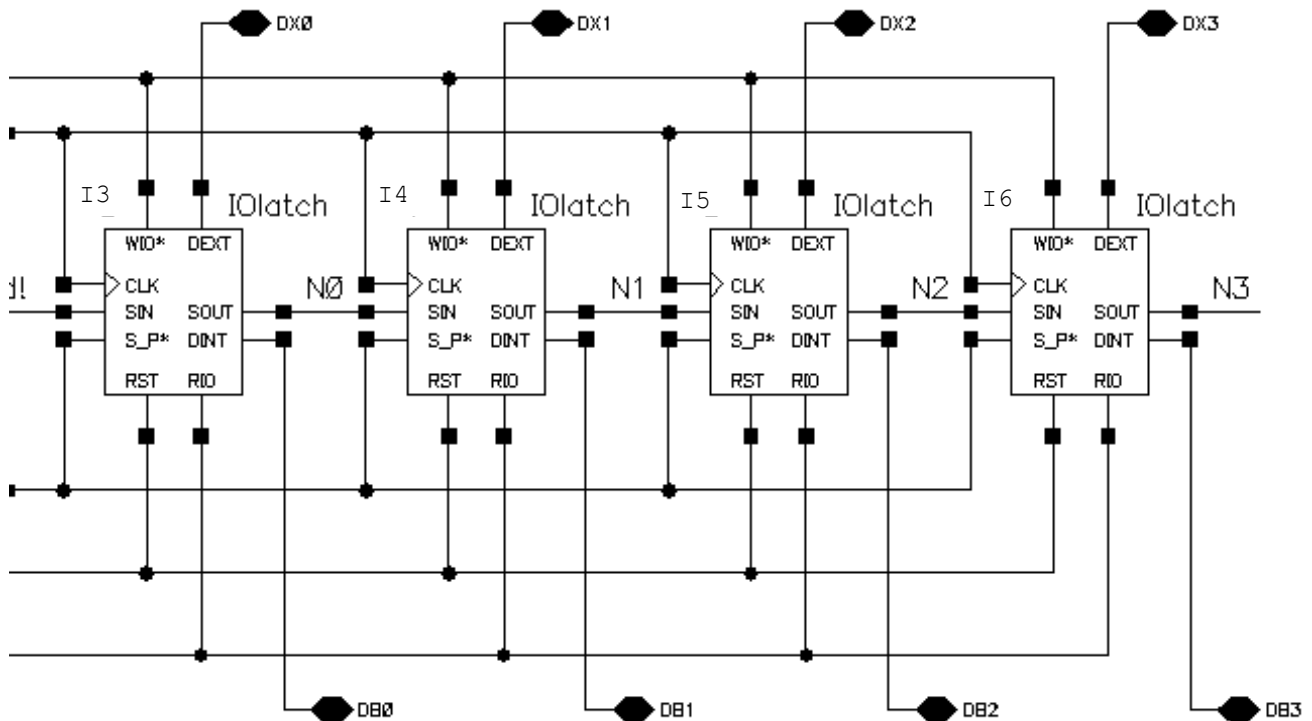
An iterated name is a compact way to describe many similar instances. This is useful in bus-type or data flow architectures that have identical structures to handle each bit on the bus. It also saves space in your design.

This example uses the base name A. The 0 to the left of the colon and the 3 to the right of the colon indicate the numbered range (0 through 3), which names instances A<0>, A<1>, A<2>, and A<3>.



Iterated instances save space. The instances in the following two figures represent identical circuitry, but the second figure is easier and faster to create.

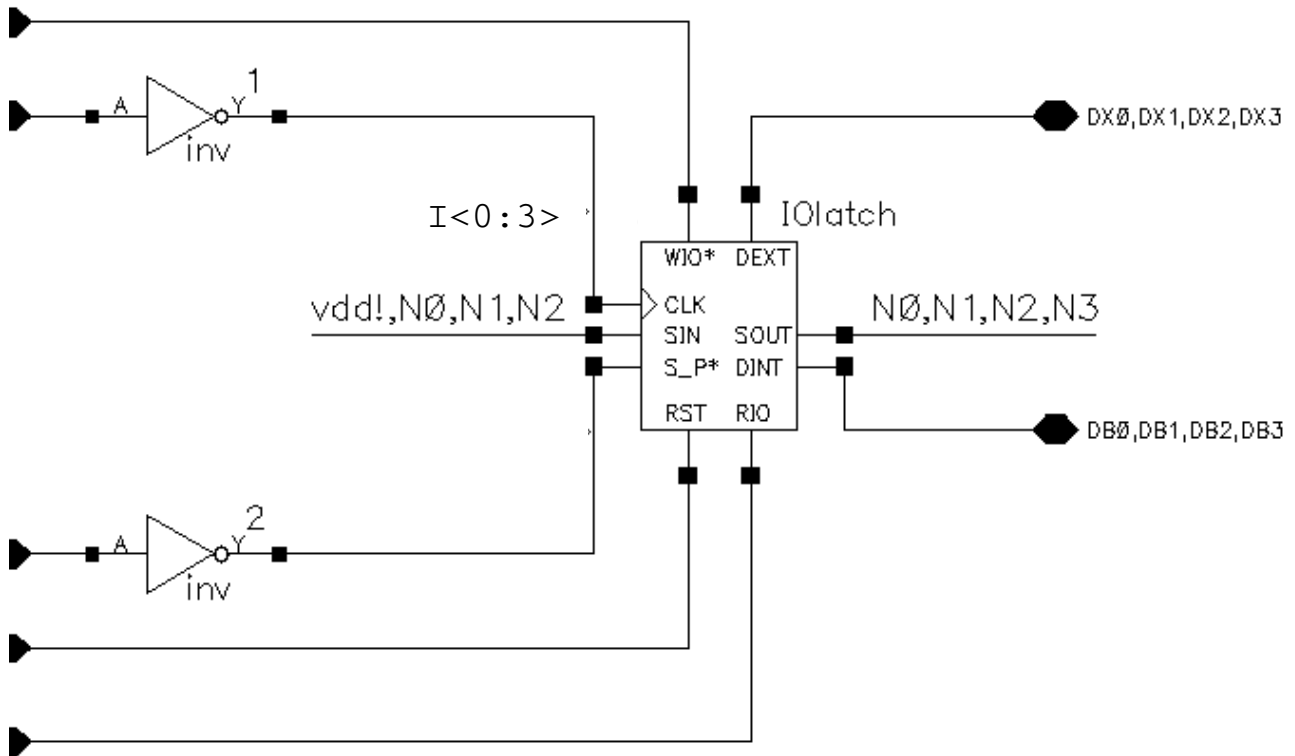
The first schematic shows four `IOlatch` instances in a design: I3, I4, I5, I6. Compare this with the next schematic.



Virtuoso Schematic Editor User Guide

Connectivity and Naming Conventions

This schematic shows one `IOlatch` iterated four times: `I<0:3>`. Compare this with the previous schematic.



Wire Connections and Wire Naming Conventions

You can connect single-bit wires in your design using any one of the following conventions:

- Wire-to-Wire Connections
- Wire-to-Pin Connections
- Wire-by-Name Connections
- Wire-to-Iterated Instance Connections
- Global Net Name Connections

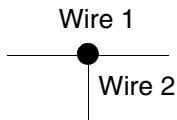
Important

When creating wires, there are mechanisms to help you make accurate connections:

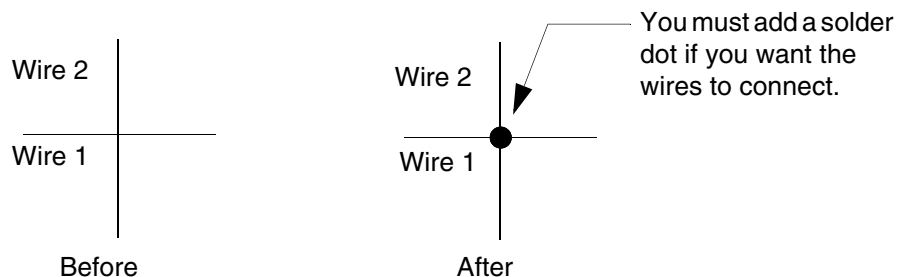
- ☐ As you move the pointer over another wire or near a pin while you draw the wire, a small diamond appears to show you the nearest object with which you can establish electrical connectivity.
- ☐ Press `s` to snap the wire to the nearest object that shows a diamond shape.
- ☐ You can zoom in on your design to see if you are making an accurate connection.

Wire-to-Wire Connections

If you draw two intersecting wires in a T configuration, the schematic editor automatically connects the wires.



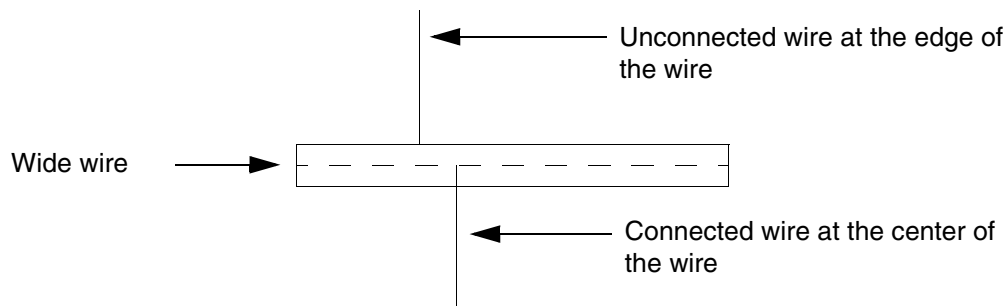
If you draw two wires that cross, you must explicitly add a solder dot to make the connection at the crossover.



If you draw two (right-angle) wires in an L configuration, or if you draw an angled wire and a wire that intersect at their end points, the wires are connected, but the schematic editor does not add a solder dot.



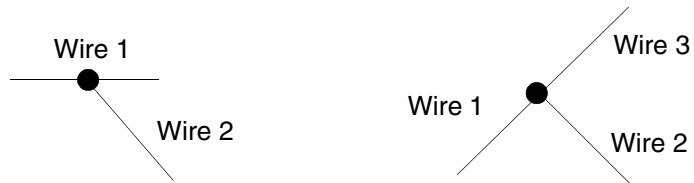
If you connect a narrow wire to a wide wire (for example, in a bus), try to intersect the centerline of the wide wire.



Virtuoso Schematic Editor User Guide

Connectivity and Naming Conventions

If you want to connect to an angled wire, you must connect to one of its endpoints. The schematic editor automatically adds a solder dot after you add the last wire in each of these examples.

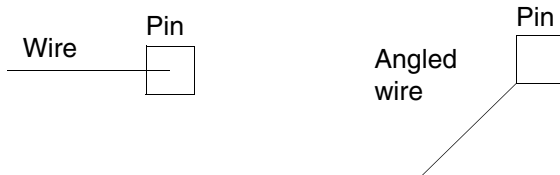


Note: Make connections for all angled wide wires, narrow wires, and flight lines at their endpoints.

Wire-to-Pin Connections

To make wire-to-pin connections, the wire must intersect the pin.

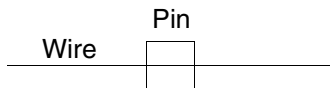
- Draw the wire so it ends in the body of a pin.



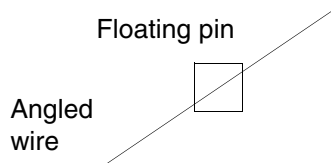
- Draw the wire so it ends on the border or perimeter of a pin.



- Draw the wire so it passes through a pin.

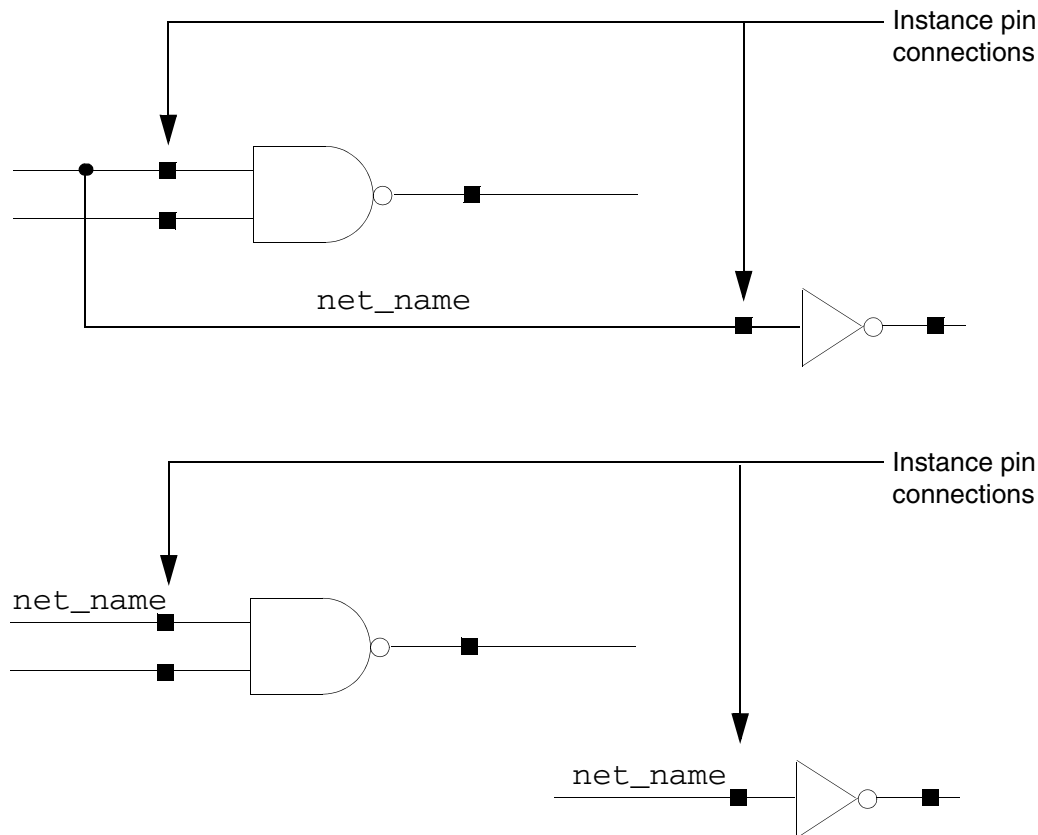


- You cannot make a wire-to-pin connection by drawing an angled wire that passes through a pin.



Wire-by-Name Connections

A signal is said to be “connected by name” if more than one physically disjoint wire or group of wires exist that carry the signal.



To connect instance pins, assign the same name to each wire. The wires do not have to be physically connected. The system recognizes when a wire and pin have identical names and connects them by name. For example, a wire labeled `P` connects by name to a schematic pin named `P`. If you use a local net name, you can connect by name on the same page. If you use a global net name, you can connect by name across all levels of the hierarchy.

When a wire and pin are connected but have different names, the net will take the name of the wire. For example, when a wire labeled `x` is connected to a pin named `P`, the pin is placed on a net named `x`.

You can use the Route Named Connection form to physically connect disjoint wires with the same name.

Be careful not to inadvertently connect a signal across a schematic by using the same name. You can catch this error before your design has been netlisted by checking your design. For

more details, see the *Instance/Net Name Collision* and *Pin/Net Name Collision* checks on the Schematic Rules Checks Setup form.

Related Topics

[Schematic Rules Checks Setup](#)

[Routing Named Connections](#)

[Route Named Connection](#)

Wire-to-Iterated Instance Connections

Before the schematic editor lets you connect a wire to an instance, it checks to see that the net width or pin width equals one of the following:

- Net Width Equal to the Connection Width
- Net Width Equal to the Pin Width
- Net Width Equal to 1
- Pin Width Equal to 1

If the connection cannot be made, an error message is generated when you check your design.

Note: The net width has no limit.

By default, if you do not specify a net name, the connection width will be equal to the net width. This means that the connection width will be the product of the instance width by the pin width. To set the connection width differently, you must add a wire label to the net or schematic pin. The width of this wire label is used unless a width incompatibility error is generated.

Related Topics

Adding Wire Names

Net Width Equal to the Connection Width

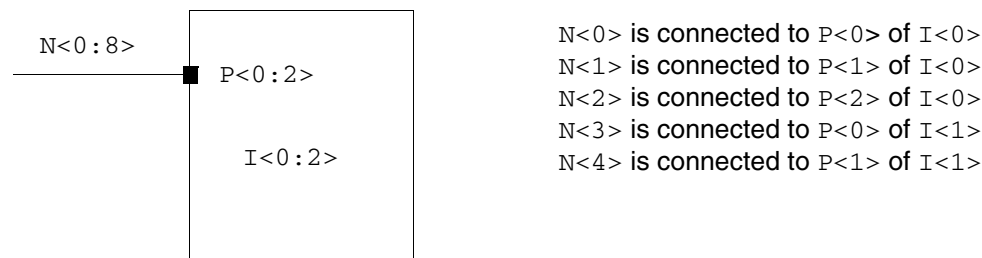
If the net width is equal to the connection width, the schematic editor allows the connection without modifying the design. To determine whether net width equals connection width, the editor compares the net width to a computed connection width. The connection width is computed as follows:

*instanceIterationWidth * instancePinWidth*

Virtuoso Schematic Editor User Guide

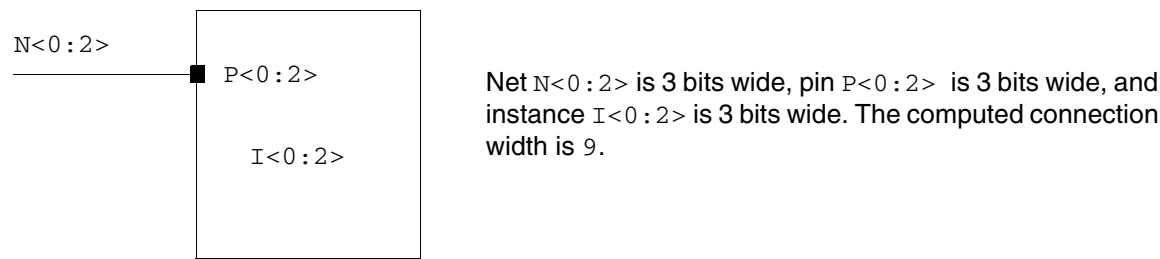
Connectivity and Naming Conventions

In the following example, the width of instance `I<0:2>` is 3 and the width of pin `P<0:2>` is 3. The computed connection width equals 9. The net width `N<0:8>` also equals 9. Therefore, the connection can be made.

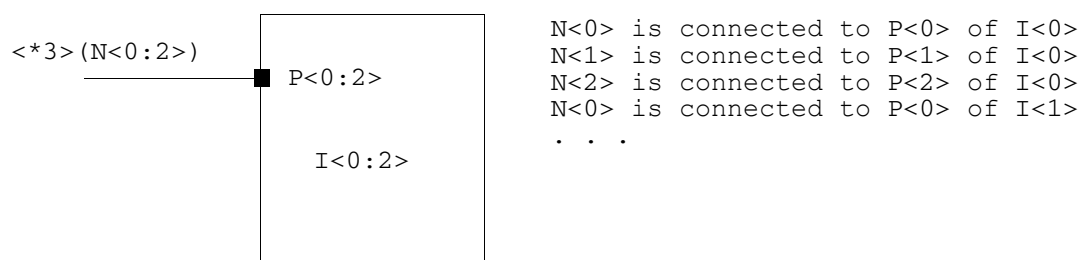


Net Width Equal to the Pin Width

If the net width is equal to the pin width, the schematic editor adds a net prefix repeat operator `<*n>` to the net width before it allows the connection. If the net width equals the pin width but does not equal the connection width, the editor computes a net prefix repeat operator that equals the instance iterations.

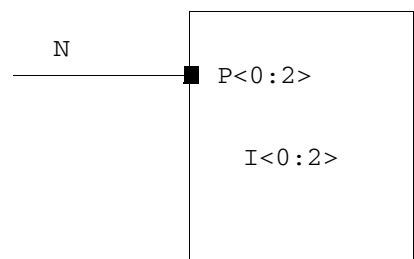


The net width (3) equals the pin width but does not equal the connection width (9). Therefore, the schematic editor generates a net prefix repeat operator, `<*3>`, for the net in order to allow the connection.



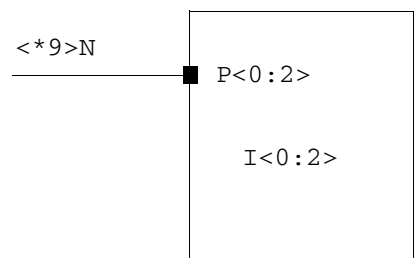
Net Width Equal to 1

If the net width is equal to 1, the schematic editor adds a net prefix repeat operator to the net width before it allows the connection. If the connection width is 1, the schematic editor computes a net prefix repeat operator that equals the computed connection width.



The net width equals 1, the pin width equals 3, and the instance width equals 3. The computed width is 9.

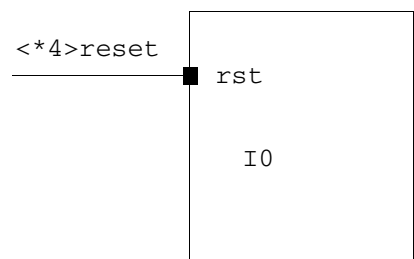
The net width does not equal either the connection width (9) or the pin width (3), but it does equal 1.



A net prefix repeat operator, <*9>, is generated for the net in order to allow the connection.

Pin Width Equal to 1

If the pin width is equal to 1, and the net repeats a single bit, the schematic editor connects the pin to the net by extracting its repeating element.

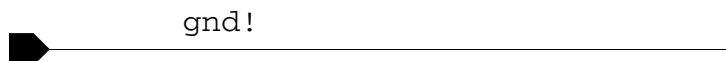


A net named <*4>reset connects to a pin named rst on an instance named I0 because the pin width is equal to 1 and the net is a repeat of the bit net reset. The net is automatically named reset.

Global Net Name Connections

Global nets make connections across the design hierarchy without using pins and wires to explicitly pass the signal through the hierarchy. Global nets are used primarily for power and ground connections.

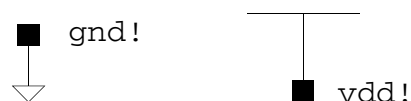
To create a global net, add an exclamation mark to the net name.



To make a bus global, add an exclamation mark to the base name of the bus.



Another way to make a net global is to connect it to a pin or instance pin that is global. The net inherits the name of the global pin or instance pin. The *basic* library has various global symbols you can use to indicate global nets.



Note: When labeling a wire with a net name that is partially global, for example `gnd! , net3`, a separate net `gnd!` is automatically created for the global part.

In summary, you can create a global signal by

- Using the *Create – Wire Name* command
- Using the *Create – Instance* command of normal power and ground symbols

Creating a net expression works in a similar way.

Related topics

Adding Wires

Multiple-Bit Wire Connections and Naming Conventions

A multiple-bit wire name can be a bundle, a bus, or a combination of the two.

This section describes the following information about multiple-bit wire connections and naming conventions:

- [Multiple-Bit Wire Naming Conventions](#)
- [Bus Tapping](#)

Multiple-Bit Wire Naming Conventions

A bus is a wire or pin sharing a common base name. A bus is represented by the base name followed with a vector expression; for example, `A<0:7>`.

A bundle is a wire or pin name consisting of simple names separated by commas; for example, `A, B, Data<0:15>`. Spaces are not allowed after the commas. The signals of a bundle do not have the same base name.

You can connect multiple-bit wires in your design using any of the following naming conventions:

- [Using Vector Expressions in Multiple-Bit Wire Names](#)
- [Using Vector Expressions in Multiple Signals](#)
- [Evaluating Vector Expressions in Multiple-Bit Wire Names](#)
- [Using Prefix Repeat Operators in Multiple-Bit Wire Names](#)
- [Using Suffix Repeat Operators in Multiple-Bit Wire Names](#)

Using Vector Expressions in Multiple-Bit Wire Names

A multiple-bit wire name can be a bundle, a bus, or a combination of the two. You can improve readability in your designs by shortening multiple-bit wire names.

To indicate that multiple bits of one wire carry similar information, give each bit the same base name. Then add a suffix to each bit name to distinguish the signals.

For example, to show four bits of a wire with a common base name (`DATA`), but with individual suffixes (`<0>`, `<1>`, `<2>`, and `<3>`), assign the following names: `DATA<0>`, `DATA<1>`, `DATA<2>`, and `DATA<3>`.

Using Vector Expressions in Multiple Signals

To name a wire that contains multiple signals with the same base name, specify the base name followed by a vector expression. The vector expression can be

- A list of the individual bit numbers separated by commas and enclosed with angle brackets.

For example, `DATA<2, 1, 0>` represents `DATA<2>`, `DATA<1>`, and `DATA<0>`.

- A range of numbers containing a lower and upper bound and, optionally, an increment value. The numbers are separated with colons and enclosed with angle brackets.

For example, `DATA<1:7:2>` represents

`baseName <lowerBound : upperBound : [incrValue]>`

Use a positive integer for *incrValue*. If you do not specify an increment value, the editor uses 1.

The ordering of the bits in a bus is important when you are connecting the bus to a pin that has a width greater than 1.

Evaluating Vector Expressions in Multiple-Bit Wire Names

The schematic editor evaluates vector expressions in multiple-bit wire names as follows:

- The vector expression `DATA<0:3:2>` names a 2-bit bus containing `DATA<0>` and `DATA<2>`.
- The vector expression `DATA<1:3:2>` names a 2-bit bus containing `DATA<1>` and `DATA<3>`.
- The vector expressions `DATA<0:3>` and `DATA<0:3:1>` are equivalent names that generate a 4-bit bus containing `DATA<0>`, `DATA<1>`, `DATA<2>`, and `DATA<3>`.

If the lower bound is larger than the upper bound, the schematic editor generates the bit numbers in descending order, as follows:

- The vector expression `DATA<2:0>` generates a 3-bit bus that contains `DATA<2>`, `DATA<1>`, and `DATA<0>`.

Using Prefix Repeat Operators in Multiple-Bit Wire Names

You can repeat a single signal name, a group of signal names, or a vector name any number of times in the wire name by placing a prefix repeat operator $\langle *n \rangle$ in front of the name, where n is a positive integer that defines the number of times to repeat each bit in the vector name.

- Use the prefix repeat operator $\langle *n \rangle$ to repeat a single-signal name. The following equivalent wire names both name the same four-bit wire:

```
 $\langle *2 \rangle A, B, C$   
 $A, A, B, C$ 
```

- Use the prefix repeat operator $\langle *n \rangle$ and parentheses to repeat a group of signal names. The following two wire names are equivalent:

```
 $\langle *2 \rangle (A, B), C$   
 $A, B, A, B, C$ 
```

- Use combinations of the prefix repeat operator $\langle *n \rangle$ and parentheses to nest parenthetical expressions to any required depth. The schematic editor expands nested expressions from the innermost expression outward. For example, a name with the expression $\langle *2 \rangle (A, \langle *2 \rangle (X, Y)) , B$ expands to $A, X, Y, X, Y, A, X, Y, X, Y, B$.

Using Suffix Repeat Operators in Multiple-Bit Wire Names

A suffix repeat operator is a number, $\langle *n \rangle$, after a vector name, where n is a positive integer that defines the number of times to repeat each bit in the vector name.

- Use the suffix repeat operator $\langle *n \rangle$ to repeat each bit in a group of bit names before expanding the vector name. For example, the following three names all describe the same six-bit wire:

```
 $A \langle 0:2 *2 \rangle$   
 $A \langle 0 *2, 1 *2, 2 *2 \rangle$   
 $A \langle 0, 0, 1, 1, 2, 2 \rangle$ 
```

- Use the suffix repeat operator $\langle *n \rangle$ and parentheses to repeat the sequence of bit names. In this case, the vector name is expanded before the bits are repeated. The following names all describe the same six-bit wire:

```
 $A \langle (0:2) *2 \rangle$   
 $A \langle 0:2, 0:2 \rangle$   
 $A \langle 0, 1, 2, 0, 1, 2 \rangle$ 
```

Bus Tapping

Tapping extracts specified bits from a named net and propagates the bits along the design. Tapping a subset of signals from a bus or a bundle is called bus tapping.

Virtuoso Schematic Editor User Guide

Connectivity and Naming Conventions

You can connect multiple-bit wires in your design using any one of the following naming conventions:

- Tapping Multiple Bits of a Bus
- Tapping Multiple Bits of a Bundle
- Tapping Wire Intersections by Name
- Designating Tap Size and Bit Order
- System-Generated Net Names

To create a bus tap

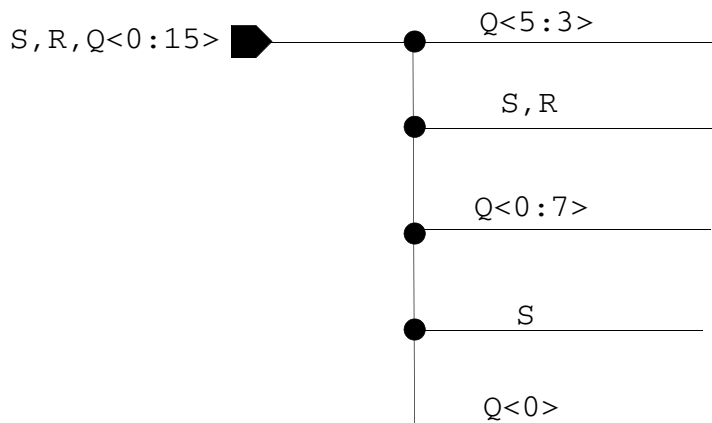
1. Connect a wire that carries the subset of signals to the bus or bundle that contains the signals.

You can use one of the many different configurations of wire-to-wire or pin-to-wire connections to create the bus tap.

2. Add the name of the bus tap that specifies the tapped signals.

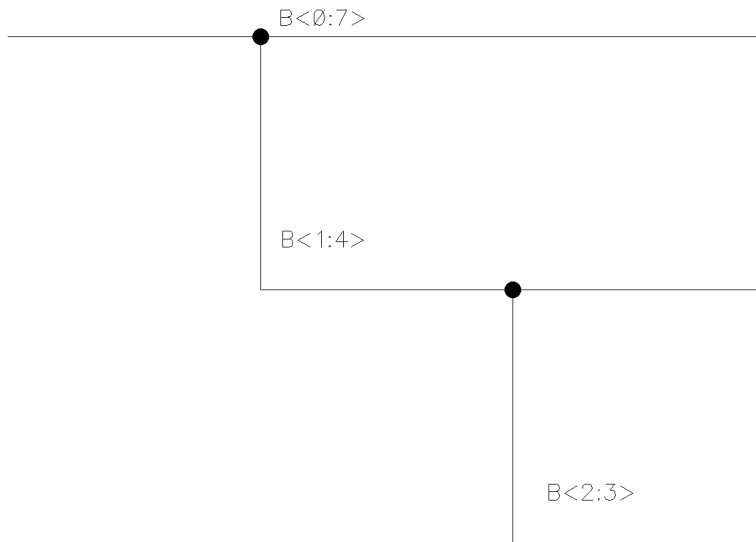
The name depends on whether you tap from a bus or from a bundle.

Tapping bits of a bus or bundle gives the tapping wire segments names that are members of the base bus or bundle. For example, in the following illustration, all names on the right are taps from the bundle $S, R, Q<0:15>$.



Tapping Multiple Bits of a Bus

In the previous section, it explains how to tap a wire from a bus. A wire that taps a bus, can itself also be tapped by using vector expressions. For example, bus B<0:7> can be tapped by wire B<1:4>, and wire B<1:4> can be tapped by wire B<2:3>.

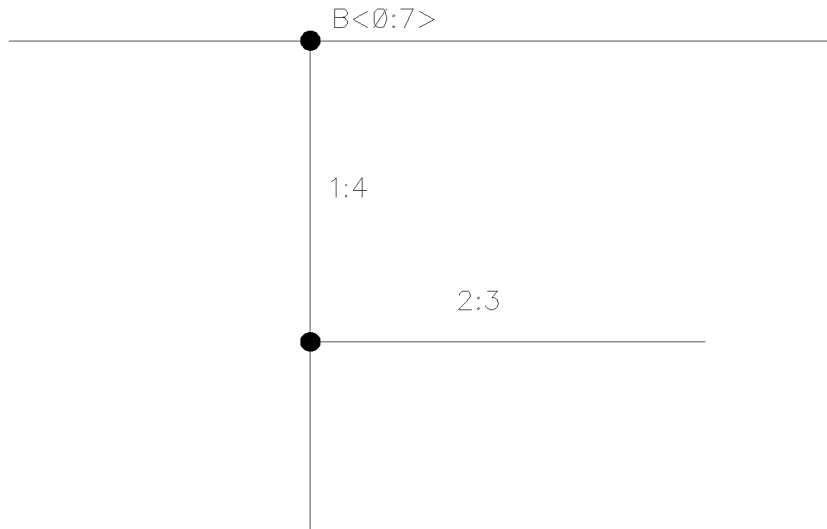


To improve readability in your design, rather than repeating the base name of the bus for each tapped wire, you can just name each wire using the vector expression. Any wires or pins of instances connected to tap wires receive the name of the tapped signal. This process is called “cascading” bus taps. If a tap wire carries more than one bit from the bus, the schematic editor can tap that wire.

Virtuoso Schematic Editor User Guide

Connectivity and Naming Conventions

The previous example, would look like this:

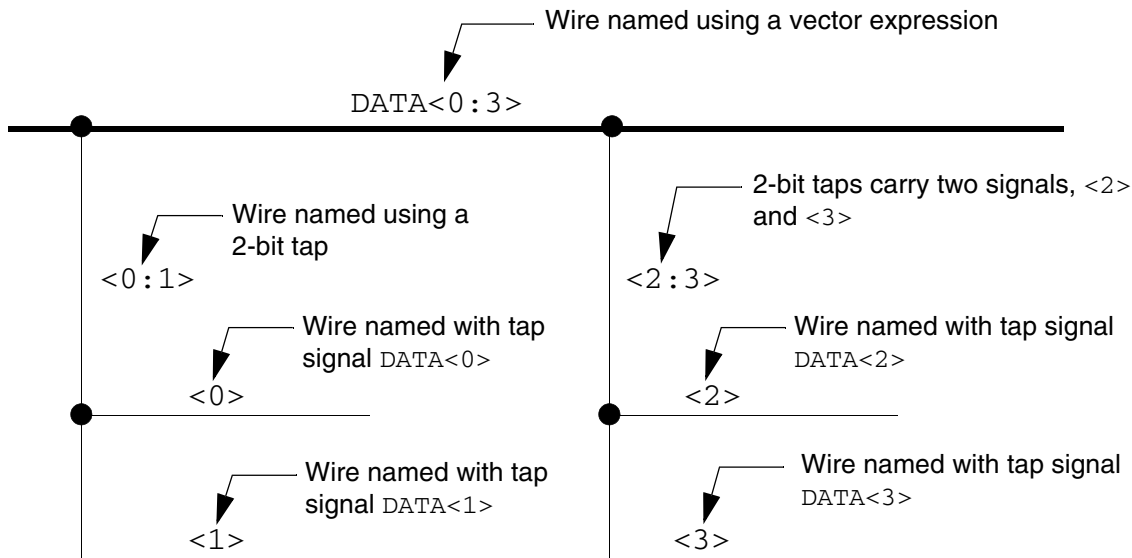


Note: A wire cannot be named with a vector expression unless it connects to a bus. By naming each tap using vector expressions ensures that nets are not created in error. Although it is possible to create a tap without a vector expression, it can cause ambiguity. For example, if a wire named 3 is connected to a bus that contains the same number within its base name $D<2 : 5>$, the wire is recognized as a tap. If there is not a bus to connect to, the same wire is recognized as a net.

Virtuoso Schematic Editor User Guide

Connectivity and Naming Conventions

You can create a geometric configuration when a tap wire carries more than one bit from the bus. The single-bit taps $\langle 0 \rangle$, $\langle 1 \rangle$, $\langle 2 \rangle$, and $\langle 3 \rangle$ are drawn from the two-bit taps $\langle 0 : 1 \rangle$, $\langle 2 : 3 \rangle$ rather than directly from the $\text{DATA}\langle 0 : 3 \rangle$ bus.

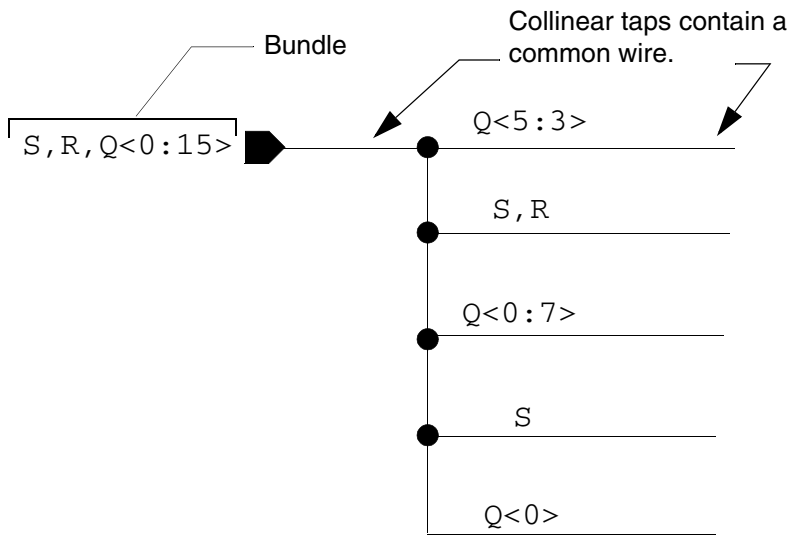


As shown in the above example, if you intersect a bus wire named $\text{DATA}\langle 0 : 7 \rangle$ with a tap wire named $\langle 0 : 3 \rangle$, you can intersect the tap wire with a second tap wire named $\langle 0 : 1 \rangle$. The name of the first tap wire becomes $\text{DATA}\langle 0 : 3 \rangle$. The name of the second tap wire becomes $\text{DATA}\langle 0 : 1 \rangle$.

You can also intersect the second tap wire with other tap wires to individually tap bit $\langle 0 \rangle$, bit $\langle 1 \rangle$, and so on. In general, you can intersect any bus with a tap wire, even if the bus itself is a tap of a larger bus. Taps can cascade down any number of steps.

Tapping Multiple Bits of a Bundle

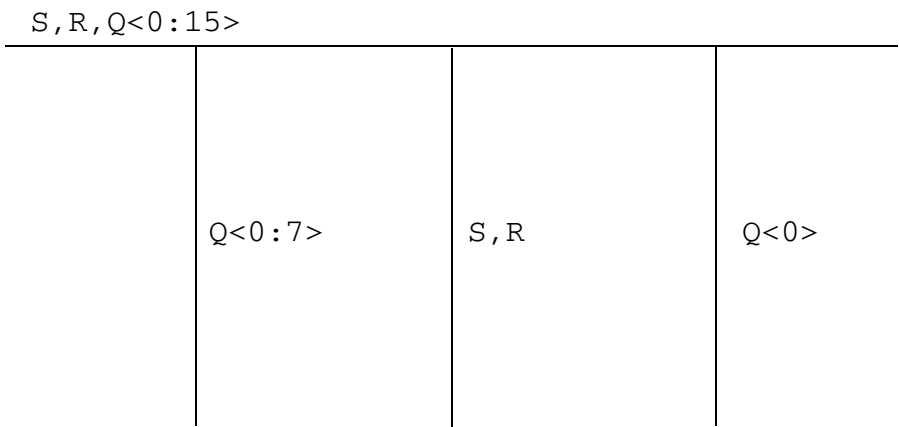
You tap bits of a bundle when you connect wires to a bundle and name the wires accordingly. For example, in the following illustration, all multiple-bit wires use signal *S*, signal *R*, or bus *Q*.



You can also use enclosed names to tap from a bundle. For example, *<S, R>* can be used to tap from the bundle *S, R, Q<0:15>*. However, *<S, R>* cannot be used to name a wire by itself.

To tap signals from a bundle (that is, a multiple-bit wire with signals that do not have the same base name), you must specify the basename of the signals in the wire name to indicate numbered bits.

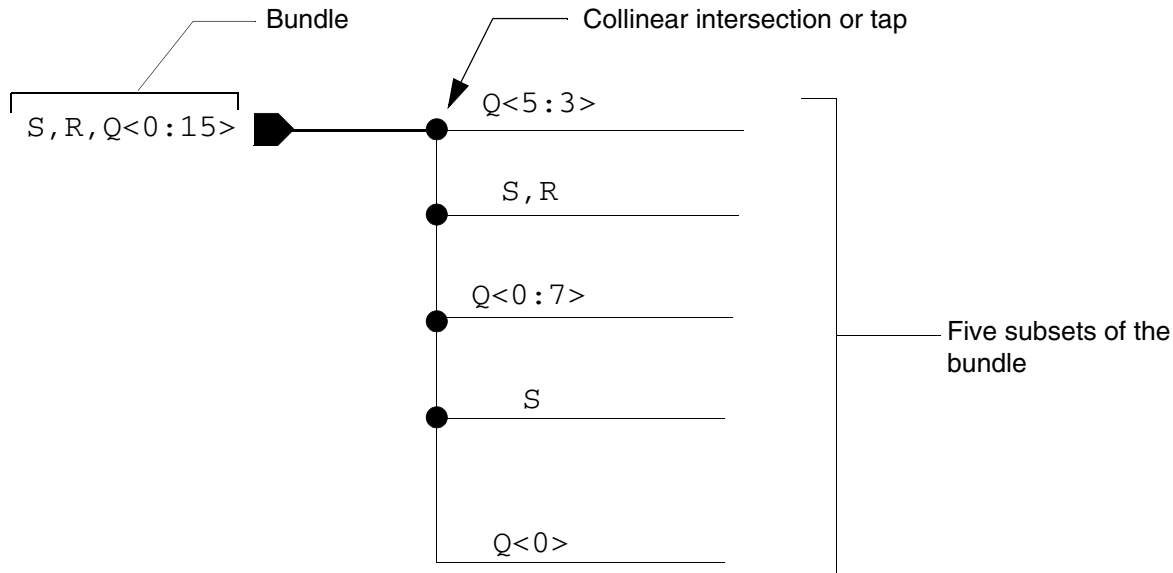
In the following example, bus *Q<0:7>* taps eight bits from bus *Q<0:15>*, which is part of a bundle. In this case, the tap name contains a base name with a vector expression. Wire *S, R* taps two signals with different names from the bundle. In this case, the wire name is a list of names separated by commas.



Virtuoso Schematic Editor User Guide

Connectivity and Naming Conventions

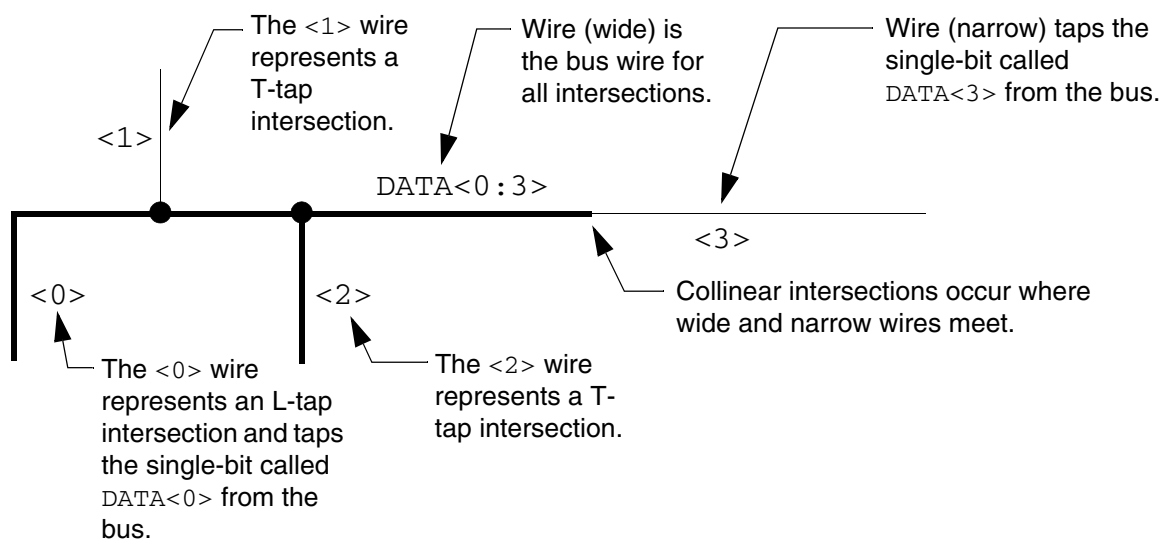
The following example shows the same multiple bits of a bundle with a pin and a collinear intersection or tap.



Like with buses, you can intersect any bundle with a tap wire, even if the bundle itself is a tap of a larger bundle. Taps can cascade down any number of steps.

Tapping Wire Intersections by Name

The following illustration summarizes how to specify bus tap intersections.



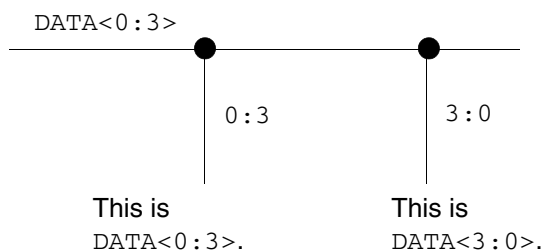
Any other wires or pins of instances connected to tap wires receive the name of the tapped signal. This process is called “cascading” bus taps. Any narrow or wide wire connected to the wire named `<3>` is connected to the `DATA<3>` bit of the bus. Any wire connected to the wire named `<2>` is connected to the `DATA<2>` bit of the bus, and so on.

Designating Tap Size and Bit Order

A tap wire from a bus can contain any number of bits with the constraint that each bit must appear somewhere in the bus. This can be achieved with or without repeat operators in the tap expression. For example, you can intersect a 4-bit bus wire named `A<0 : 3>` with an 8-bit tap wire named `<0 : 1 * 2, (0 : 1) * 2>`. The resulting name of the tap wire is `A<0, 0, 1, 1, 0, 1, 0, 1>`. Alternatively, you can intersect a bus named `A<2 : 0>` with a tap wire named `A<0>, A<0>, A<0>`.

The schematic editor orders bits in a bus just as they appear in the bus name. Similarly, the editor orders bits in a tap as they appear in the tap name.

For example, you can tap the four bits of a bus named `DATA<0 : 3>` in order from `DATA<0>` to `DATA<3>`. However, a tap wire named `DATA<3 : 0>` that intersects with bus `DATA<0 : 3>` taps the bits in reverse order, as shown in the following figure.



System-Generated Net Names

Tapping starts from the bus or bundle with the largest number of unique bits, and cascades outward from there.

Any unnamed wire segments that intersect are grouped and placed onto the same unnamed net. The named nets that intersect with this net are used to derive a system generated name.

The following items explain how the system determines a name for unnamed wire segments.

- The schematic editor places all unnamed intersecting wires in the same unnamed net.

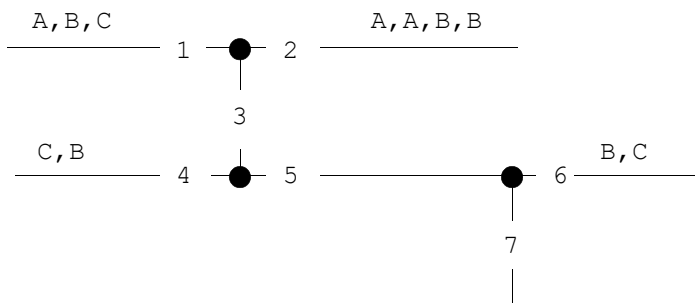
Virtuoso Schematic Editor User Guide

Connectivity and Naming Conventions

- To determine the name of the unnamed net, the schematic editor chooses a name from among the named nets that intersect with the unnamed net. Named nets are those nets whose names are derived from wire names, schematic pins, or tap names.
- The chosen name for the unnamed net is the one with the largest number of unique members. For example, a net named A, B, C has three unique members, but a net named A, A has only one unique member.
- At all wire-wire and wire-pin intersections, the schematic editor verifies that all names present at the intersection are members of the name that contains the most unique members.

For example, in the following illustration

- The schematic editor places wire segments 3, 5, and 7 in the same net because these segments are intersecting and unnamed.
- This unnamed net is then named A, B, C because it takes the names of all the intersecting named nets. The name A, B, C is the one with the most unique member names; that is, three.
- The schematic editor then verifies that all intersecting named nets (B, C, C, B and A, A, B, B) are taps of A, B, C .



Pin Connections and Pin Naming Conventions

You can name pins in a design to declare which nets in the schematic are connected to a higher level of the design hierarchy or which nets span across sheets in a multisheet schematic.

Each pin name must be unique and observe the rules for reserved characters.

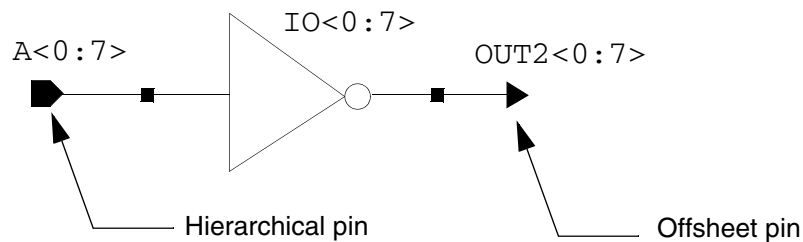
This section describes the following pin connections and naming conventions:

- Hierarchical and Offsheet Pin Names
- Bus Pin Names
- Pin-to-Pin Connections
- Tapping Pins

Hierarchical and Offsheet Pin Names

If you are designing a multisheet schematic, your sheets contain hierarchical pins and offsheet pins.

Hierarchical pins are pins that also appear on the symbol of the design. The hierarchical pins from each sheet become the hierarchical pins of the multisheet. Each hierarchical pin in a multisheet design must have a unique name.



Offsheet pins connect signals across the sheets of a multisheet schematic only. Use the same name for the offsheet pins on each sheet.

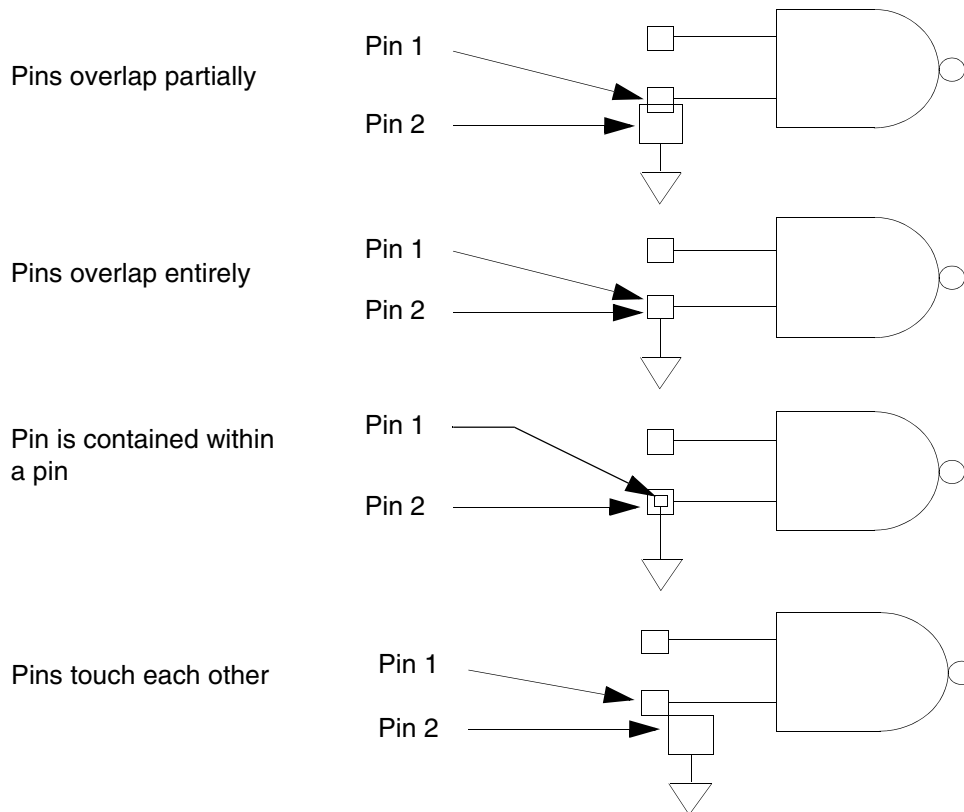
If you want a signal to appear on a multiple sheet that is also exported by a hierarchical pin, place a hierarchical pin for the first usage and offsheet pins on the other sheets.

Bus Pin Names

Use a base name with a vector expression. For example, `IN<16:10:2>` expands to these four pins: `IN<16>`, `IN<14>`, `IN<12>`, and `IN<10>`.

Pin-to-Pin Connections

A direct connection between two instance pins without an intermediate wire is called a pin-to-pin connection. There are four ways to make pin-to-pin connections:

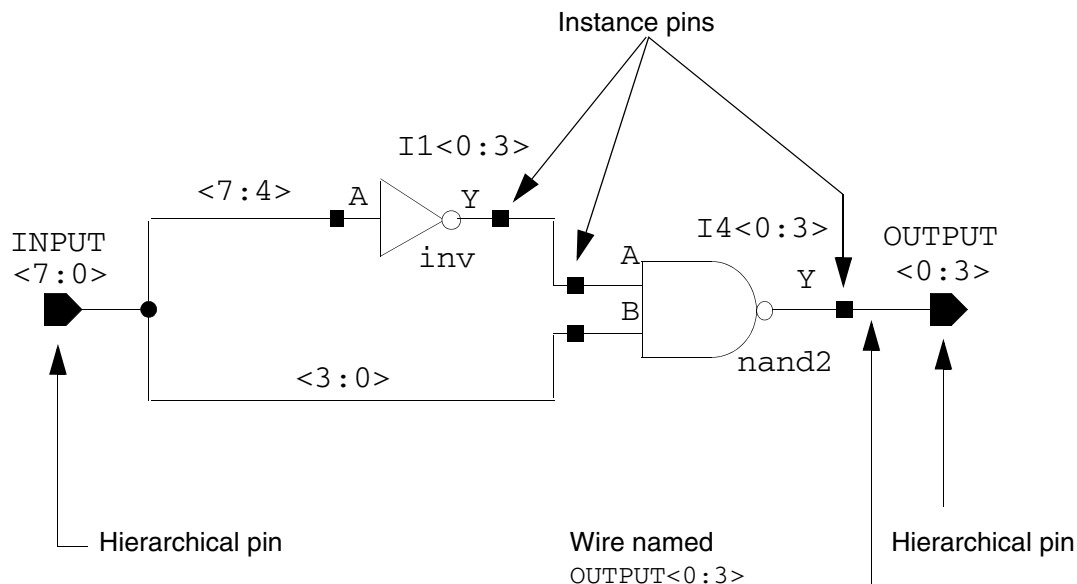


Tapping Pins

Tapping pins is similar to tapping a bus or bundle. The ‘tap’ is a subset of signals extracted from a pin and any wire or instance pin connected to the tapped bits receive the tapped signal. When labeling the tap, the label must reflect the name of the bits that are being tapped from the pin.

If a schematic pin has a bus name, you can draw a tap directly from the pin using a vector expression to name a wire that intersects with the pin.

The following figure shows how to tap pins in a design.



In the example above, the wire named **<7:4>** is a tap of the schematic pin named **INPUT<7:0>** and connects a four-bit bus called **INPUT<7:4>** to the input of the inverter (a four-bit iterated instance).

The wire named **<3:0>** is also a tap of the **INPUT<7:0>** pin and connects a four-bit bus called **INPUT<3:0>** to the input of the four-bit **nand2** gate.

If you do not apply a name to a net that is attached to a pin, the schematic editor gives the net the same name as the pin. For example, the wire that connects the output of the **nand2** gate to the **OUTPUT<0:3>** pin is called **OUTPUT<0:3>**.

Patch Cord Connections and Patch Cord Naming Conventions

A patch cord is a special connection symbol. You can use a patch cord to:

- establish aliases between the signals of two different nets
- copy one net to another (this is different to tap names, where you extract bits of a single net)
- map bits from one net to different bits of another net
- connect multiple nets (merge two buses)

Note: You cannot create a patch cord on a schematic pin (I/O pin). For example, you cannot connect an input pin to an output pin even though you use a patch cord in the middle.

The value (connection expression) of the NLP label describes how to map the bits of the net attached to the `src` (source) pin to the bits of the net attached to the `dst` (destination) pin.

The syntax you use for naming the vector expression is the same as the syntax for wires and multiple-bits. The number of bits in the `src` vector expression must equal the number of bits in the `dst` vector expression.

Related Topics

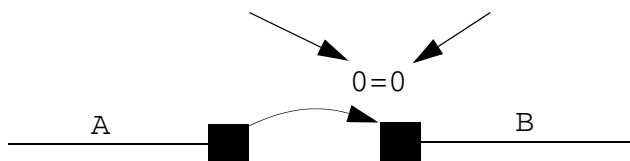
[Adding Patch Cords Using the Add Instances Form](#)

[Adding Patch Cords Using the Create Patchcord Form](#)

Naming Single and Multiple Bits Using Patch Cords

The `schPatchExpr` connection expression (value) for patch cords has the following form:

`srcVectorExpression = dstVectorExpression`



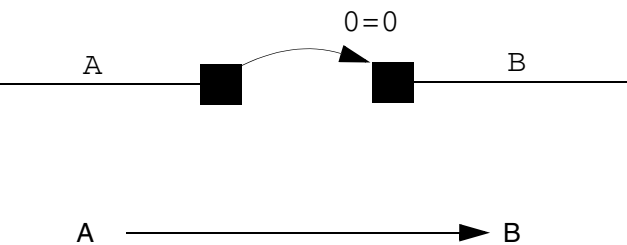
where `srcVectorExpression` is the source and `dstVectorExpression` is the destination.

Virtuoso Schematic Editor User Guide

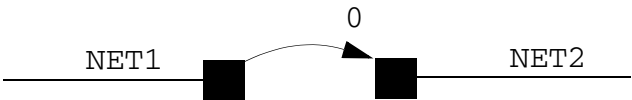
Connectivity and Naming Conventions

Naming conventions for patch cords using vector expressions have the same syntax as the vector expressions for wires with multiple bits. The number of bits in *srcVectorExpression* must equal the number of bits in *dstVectorExpression*.

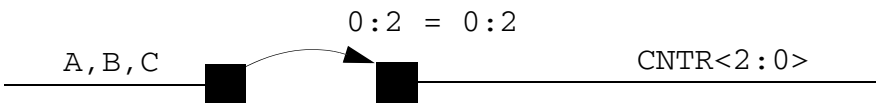
In the following example, nets A and B are aliased together. Logically, each object that A is connected to is also connected to B, and the patch cord connects the first bit named in *srcVectorExpression* to the first bit named in *dstVectorExpression*.



If you want to connect two single-bit nets called NET1 and NET2 with a patch cord, a connection expression of 0 is sufficient:



In the following example, signal A is aliased to CNTR<2>, signal B is aliased to CNTR<1>, and signal C is aliased to CNTR<0>.

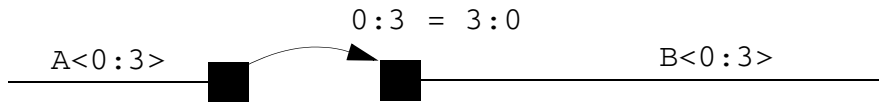


From	To
A	CNTR<2>
B	CNTR<1>
C	CNTR<0>

Virtuoso Schematic Editor User Guide

Connectivity and Naming Conventions

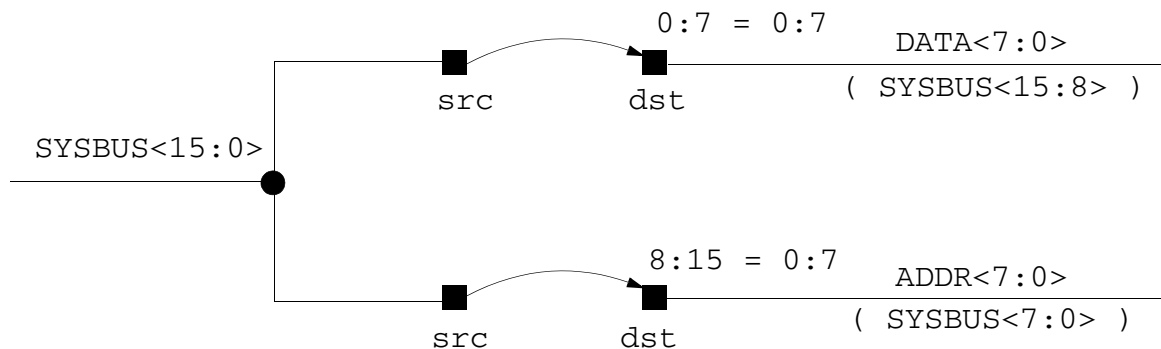
In the following example, A<0> is aliased to B<3>, A<1> is aliased to B<2>, A<2> is aliased to B<1>, and A<3> is aliased to B<0>.



From	To
A<0>	B<3>
A<1>	B<2>
A<2>	B<1>
A<3>	B<0>

Naming Multiple Nets Using Patch Cords

The following figure shows how to use patch cords to merge two buses.



The upper patch cord connects the eight-bit bus SYSBUS<15:8> to the eight-bit bus DATA<7:0>. The lower patch cord connects the eight-bit bus SYSBUS<7:0> to the eight-bit bus ADDR<7:0>.

The range of bits named in the lower patch cord using *srcVectorExpression* <8:15> does not refer to the bits of SYSBUS. It refers to the order of the bits attached to the *src* pin of the patch cord. Therefore, according to the connection expression, bits 8 through 15 of bus SYSBUS<15:0> (corresponding to the bits named SYSBUS<7:0>) are connected to bit 0 through 7 of bus ADDR<7:0> (corresponding to the bits named ADDR<7:0>).

Inherited Connections



When you are creating a block (or the components of a block) that will be shared, distributed, or used elsewhere, Cadence recommends defining all inherited connections as explicit pins rather than as implicit terminals.

If you have a need to change the value of a global signal in your design, you can do so without recreating a complete hierarchical branch. You can build one library containing components potentially requiring different power and ground connections in different parts of a design. This is referred to as an inherited connection.

Inherited connections let you selectively override global signals in designs originated in the Virtuoso® Schematic Editor. To have this override information available to other Cadence® tools across the design flow, the override information is communicated to the software by means of net expressions and a class of properties called `netSet` properties.

Note: For information on how to retrieve details of inherited connections and manipulate their settings see [Manipulating Inherited Connections](#) on page 295.

Inherited connections is an extension to the connectivity model that allows you to create special global signals and override their names for selected branches of the design hierarchy. This flexibility allows you to use

- Multiple power supplies in a design
- Overridable substrate connections
- Parameterized power and ground symbols

The inherited connections feature is recognized by many tools throughout the IC design flow.

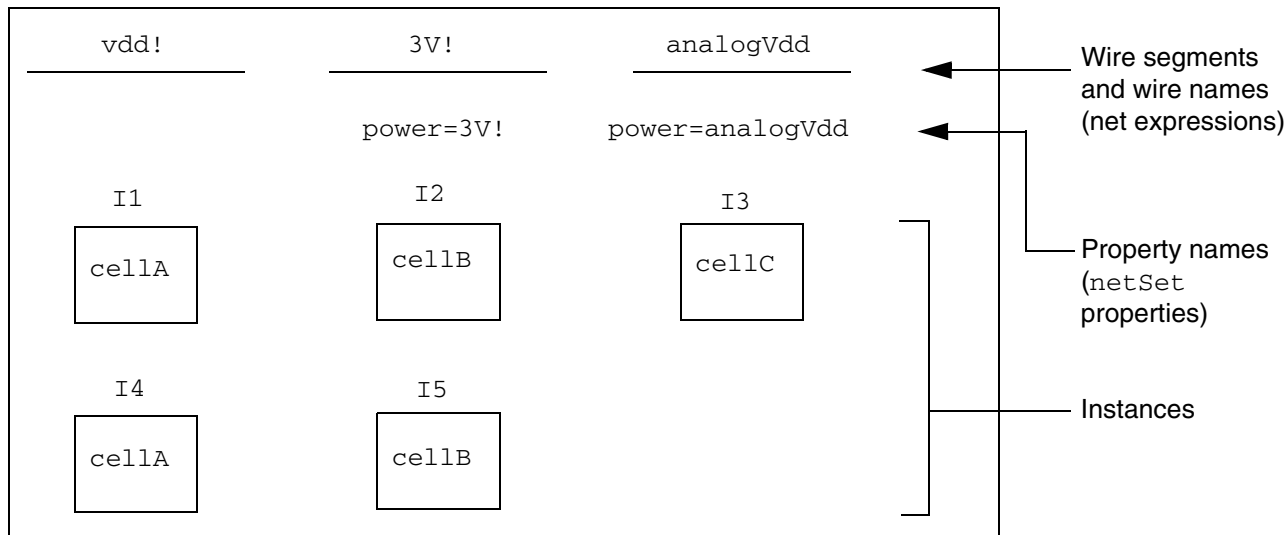
This section describes the following information about inherited connections:

- [Inherited Connections in a Hierarchy](#)
- [Defining Inherited Connections](#)
- [The Syntax of an Inherited Net Expression](#)
- [How Net Expressions Evaluate](#)
- [Sample Net Expression Library](#)
- [Inherited Connections Restrictions](#)

Inherited Connections in a Hierarchy

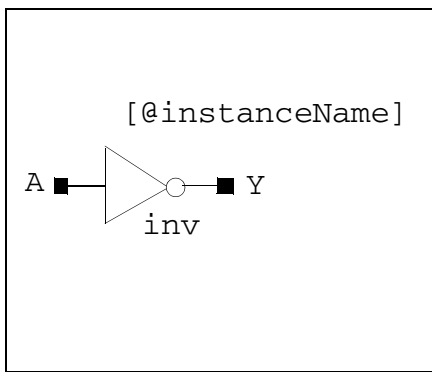
You use inherited connections to selectively override global signals within your design. Consider the following example and explanation:

High-Level Schematic

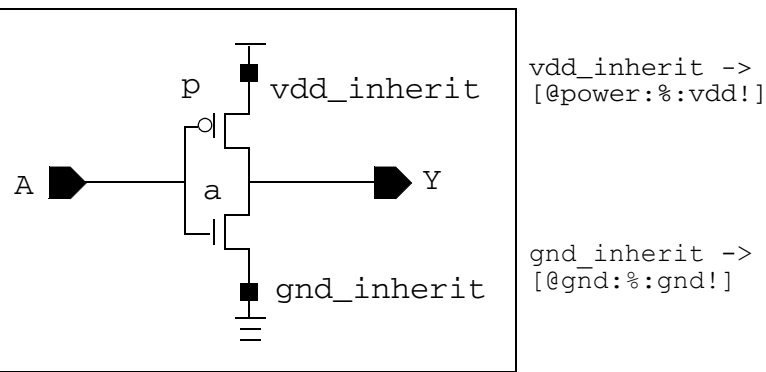


Lower-Level Schematic

Inverter Symbol



Inverter Schematic



The high-level schematic shows five instances: I1, I2, I3, I4, and I5. Each instance represents a portion of the design, each of which eventually references the inverter shown in the lower-level schematic.

In the inverter schematic, the power and ground wires are defined using the net expressions `[@power:%:vdd!]` and `[@gnd:%:gnd!]`, respectively. By default, all inverter pmos transistors are connected to `vdd!` and all nmos transistors are connected to `gnd!`.

Virtuoso Schematic Editor User Guide

Connectivity and Naming Conventions

In the high-level schematic, all the inverter `pmos` transistors below instance `I2` of `cellB` are to use `3V!` as the power supply, and the inverters in instance `I3` of `cellC` are to use `analogVdd`.

To selectively override the default global signal `vdd!`, create a `netSet` property on instance `I2` named `power` with the value `3V!` and a `netSet` property on instance `I3` named `power` with the value `analogVdd`. Notice that the inverter `pmos` transistors below instances `I1`, `I4`, and `I5` are still connected to `vdd!`.

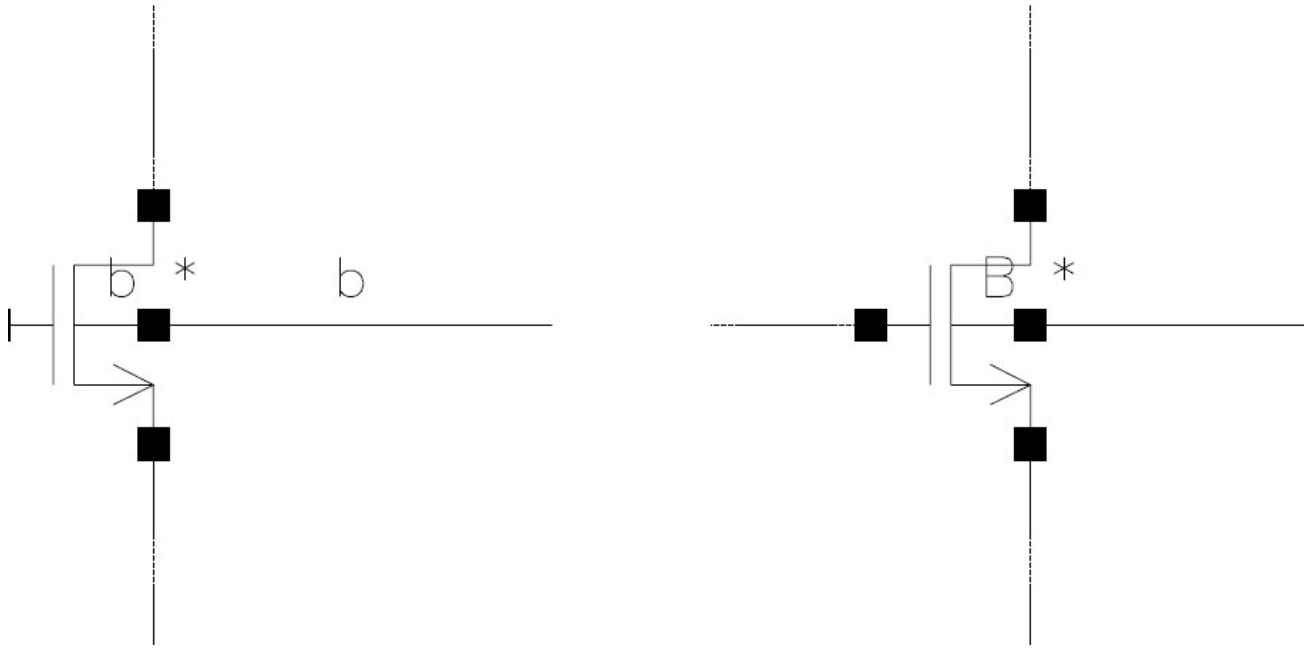
In other words, you place a `netSet` property on an instance representing the branch of hierarchy where it is to be applied. You can create the `netSet` property on any instances at any level above the cellviews with net expressions. For example, if a large hierarchical design has seven levels of hierarchy, you can place a `netSet` property `power = 3V!` on an `I2` instance in the top-level schematic. This affects all the logic below instance `I2` all the way to the bottom in all cellviews that contain a net expression, such as `[@power:%:vdd!]`. All cellviews that contain this net expression would use `3V!` instead of `vdd!` for that branch of the design.

However, if on a lower-level cellview there was an `I7` instance with a `netSet` property `power = 2V!`, then `2V!` would always be used below the `I7` instance.

Using Physical Connections to Override Inherited Instance Pins

You can override an inherited instance pin by connecting a labeled wire or a schematic pin. For example, in the below-mentioned design, the wire labeled `b` overrides a

[@BULK:%:gnd!] net expression on the left instance's bulk terminal. Pin B overrides the right instance's bulk pin net expression.



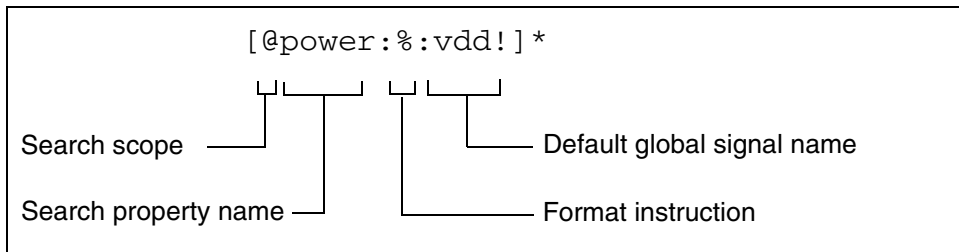
Defining Inherited Connections

In the schematic editor, you define an inherited connection by adding a net expression label to either a wire or a pin. A net expression defines the default global signal name for the connection and the name of a property that can be used to override that default global signal name.

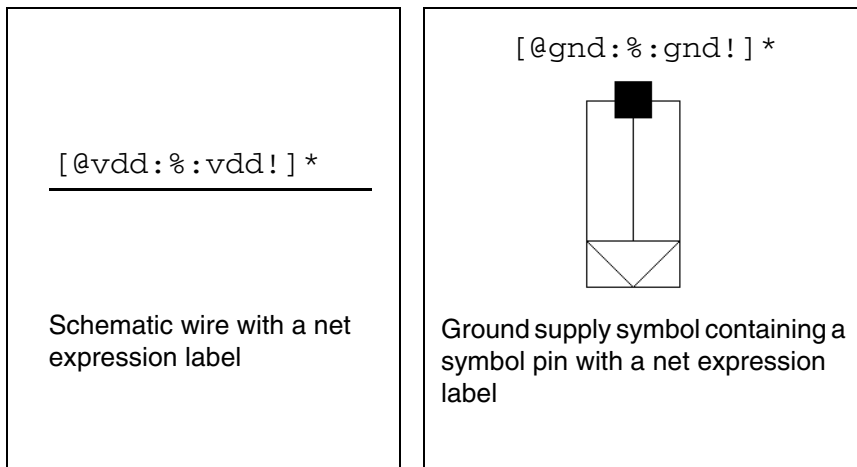
The Syntax of an Inherited Net Expression

The syntax of an inherited net expression is a subset of that of an NLP (Natural Language Processing) expression. The four components are the search scope, search property name, format instruction, and default global signal name.

Example Net Expression



Example Net Expression Labels



The asterisk (*) after a global name shows that this is not a regular wire name but a name that is an overridable net expression. The default global signal name specifies what the wire or pin is connected to by default.

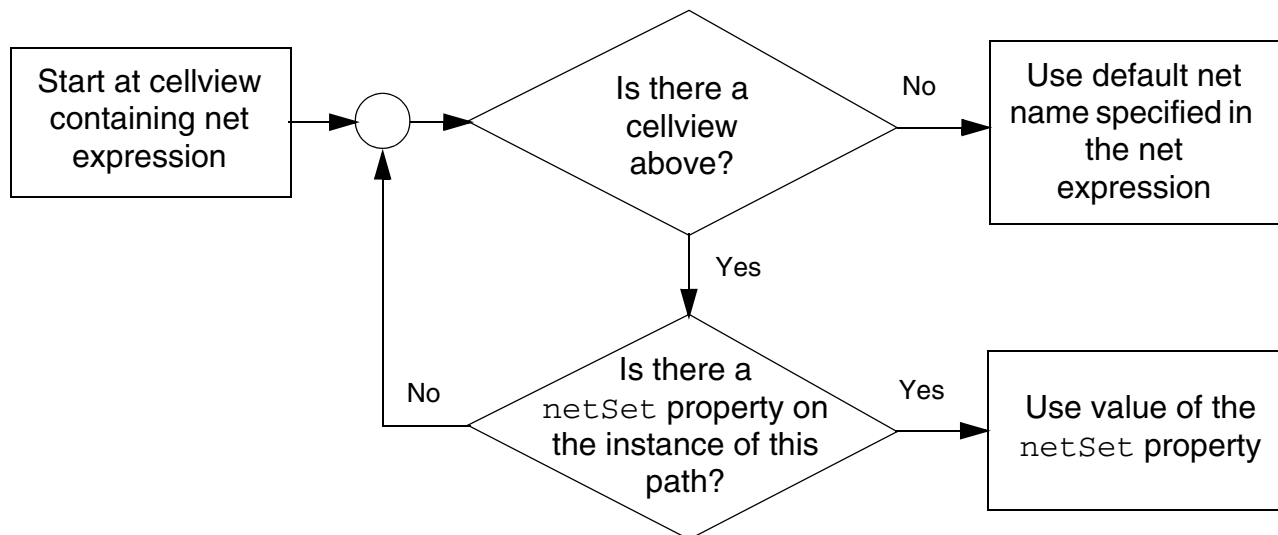
Note: There are several options for displaying net expression labels available. For more details, see *Net Expression Display* on the [Display Options](#) form.

How Net Expressions Evaluate

The search scope is limited to the @ search, which requires an instance lineage. The search travels up the instance hierarchy looking for an instance property of the given search property name until either an override is found or the instance lineage is exhausted. The default signal name must be global in nature and serves two important purposes: it names the net locally and is the inherited signal when no override is found during the search. The format instruction (%) is retained here in compliance with the syntax of an NLP expression.

The lookup property is a `netSet` type and is used only for an inherited connection. The `netSet` property value can be one of the two possibilities: an override signal name or an overriding, redirecting net expression. The former is the inherited signal name while the latter case redirects the search to a new property name and may introduce a new default signal name to replace the one in the original net expression in the hierarchy below it. The search for the new lookup property starts with the current instance where the `netSet` property with the override inherited net expression is found.

Net expressions are evaluated for each hierarchical path. The schematic editor uses the property name specified in the net expression label to search upward from the cellview containing the net expression label (one instance at a time) to the top of the design. The first instance that is found that has a matching property name terminates the search. If the property is of type `netSet` and is a legal net name, its value is used as the connecting net rather than the default net name specified in the expression.



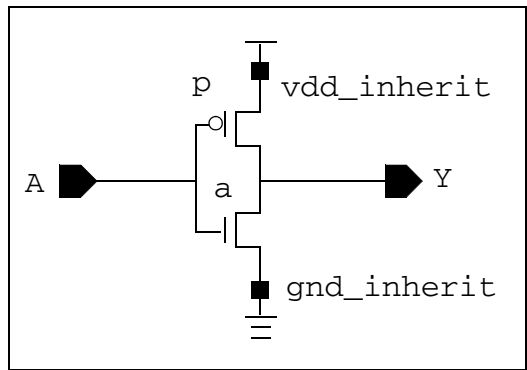
The `netSet` property can be placed on any instance and it will affect all net expression labels with matching property names at all levels below that instance unless overridden by another `netSet` property on a lower-level instance.

Virtuoso Schematic Editor User Guide

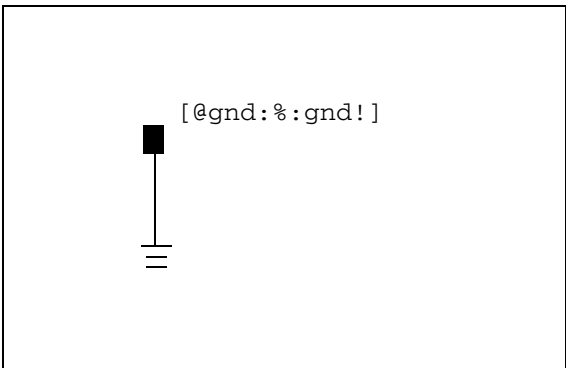
Connectivity and Naming Conventions

You can create an inherited connection in a schematic by placing an instance of a symbol where one of the symbol pins has a net expression label. When the extractor runs on the schematic, the net expression label from the symbol pin is propagated onto the net within the schematic.

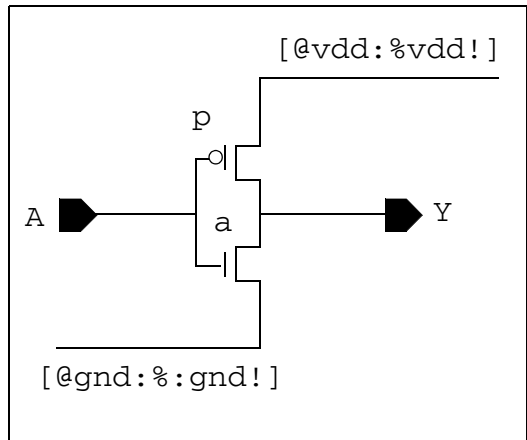
Inverter Schematic



gnd_inherit Symbol



Inverter Schematic



Net expression labels can also be placed on wires to achieve the same effect.

The *basic* library contains sample parameterized power and ground supply symbols called *vcc_inherit*, *vdd_inherit*, *gnd_inherit*, and *vss_inherit*.

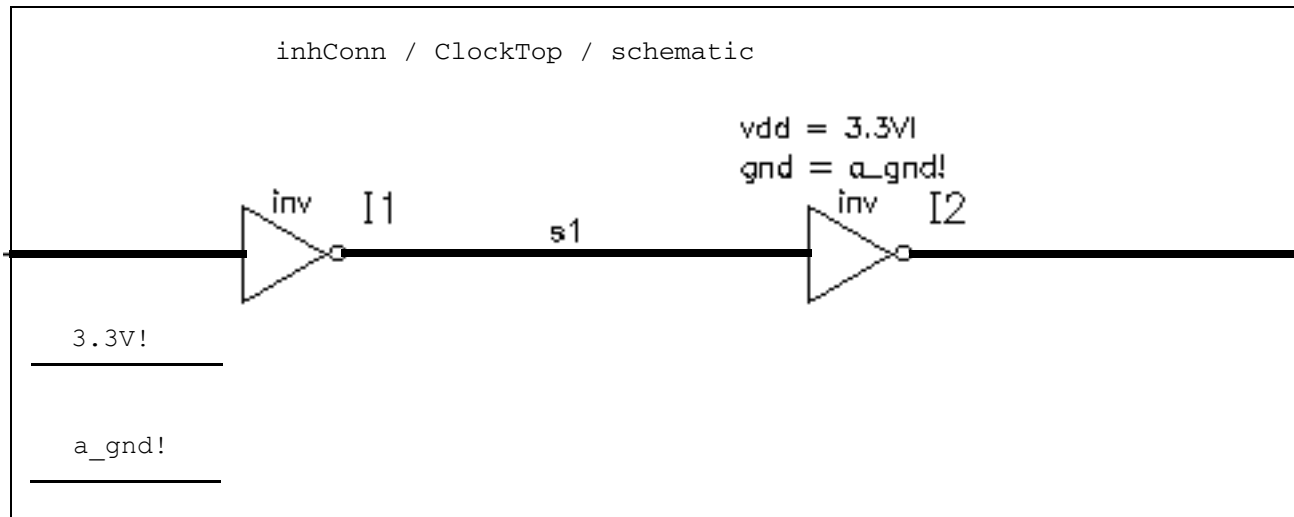
Inherited Supply Symbol	Net Expression Label
vcc_inherit	[@vcc:%:vcc!]
vdd_inherit	[@vdd:%:vdd!]
gnd_inherit	[@gnd:%:gnd!]
vss_inherit	[@vss:%:vss!]

Sample Net Expression Library

The location of a sample design that uses inherited connections is

`your_install_dir/tools/dfII/samples/tutorial/inhConn/inhConn/schematic`

The following design shows the top-level schematic cellview.



Inherited Connections Restrictions

You should be aware of some restrictions when using net expression labels and `netSet` properties:

- When you are creating a block (or the components of a block) that will be shared, distributed, or used elsewhere, Cadence recommends defining all inherited connections as explicit pins rather than as implicit terminals.
- The default name in a net expression label must denote a single-bit (scalar), global name. For example, `vdd!` and `mygnd!` are legal; `bus!<0:15>` and `vdd` are illegal.
- The property you use to override an inherited connection must be of type `netSet`.
- The `netSet` property value must represent an existing single-bit signal name (for example, `A`, `clock`, or `myPower!`) or another net expression.

Note: The signal name used in a `netSet` property must be the name of an existing net in that cellview.

- Within one cellview, you cannot use two different net expression labels for the same default global signal. Therefore, using both `[@power:%:vdd!]` and `[@vdd:%:vdd!]` in the same cellview is not allowed.

Note: The [Schematic Rules Checks Setup](#) form contains Inherited Connections checks which you can define to check existing or missing inherited connections.

Related topics

[Adding Net Expression Labels to Create an Inherited Connection](#)

[Adding netSet Properties to Create an Inherited Connection](#)

[Analyzing Net Expression Property Names and Evaluated Names](#)

Summary of Naming Conventions

The following tables detail the naming conventions.

Note: Any character that is not reserved or illegal can be used in names.

Character	Symbol	Description	Example
comma	,	Separates the elements of a wire name	A, B<0 : 1>, C
angle brackets	< >	Enclose expressions for multiple-bit wires; used for iterators (<u>prefix repeat operator</u>) and bus taps	<*2>B, A<0 : 3> , <2>
parentheses	()	Enclose related multiple-bit wires within an expression	A< (0 : 2) *2>
colon	:	Separates a range of numbers within an expression	A<0 : 2>
exclamation point	!	Global signal for a power supply	vdd!, gnd!

Table 2-1 Reserved Characters for Instances

Character	Symbol	Description	Example
angle brackets	< >	Enclose a range of numbers	A<0>
colon	:	Separates a range of numbers expressed by a starting and ending number	A<1 : 3>

Table 2-2 Illegal Characters in Wire and Net Names

Character	Symbol
back/forward slashes	\ /

Virtuoso Schematic Editor User Guide

Connectivity and Naming Conventions

Table 2-3 Illegal Characters in Instance Names

Character	Symbol
comma	,
parentheses	()
colon	:
back/forward slashes	\ /

Table 2-4 Syntax Conventions for Wire, Net, and Pin Names

Name	Description
<i>baseName</i>	A string of printable ASCII characters that does not include <u>reserved characters</u> . NET, R, C
<i>nameTerm</i>	<i>baseName</i> optionally followed by a <u>vector expression</u> enclosed in angle brackets. <i>nameTerm</i> can also be a name expression enclosed in parentheses. Must be in the form { <i>baseName</i> [< <i>vectExpr</i> >] (<i>nameExpr</i>) }
<i>nameExpr</i>	A string consisting of <i>nameTerm</i> , optionally prefixed with the < * <i>n</i> > construct. The < * <i>n</i> > construct is also referred to as a <u>prefix repeat operator</u> , which means that <i>nameTerm</i> is to be repeated <i>n</i> times. Name expressions can be concatenated to form a comma-separated list. Must be in the form [< * <i>n</i> >] <i>nameTerm</i> { , <i>nameExpr</i> }

Virtuoso Schematic Editor User Guide

Connectivity and Naming Conventions

Table 2-4 Syntax Conventions for Wire, Net, and Pin Names

<i>vectTerm</i>	A single number or a range of numbers. The range can be expressed by a starting, ending, and increment number, separated by colons. <i>vectTerm</i> can also be a <u>vector expression</u> enclosed in parentheses. In this case, the innermost term is expanded first, and then the <u>repeat operator</u> repeats the entire expanded term enclosed in parentheses. Must be in the form $n[: n[: n]] \mid (vectExpr)$
<i>vectExpr</i>	A single or repeated <i>vectTerm</i>. The optional $\langle *n \rangle$ suffix on a <i>vectTerm</i> is called a <u>suffix repeat operator</u> and indicates n repetitions of each bit of <i>vectTerm</i>. The value of n must be greater than zero. Vector expressions can be concatenated to form a comma-separated list. Must be in the form $vectTerm [*n] \{ , vectExpr \}$

Table 2-5 Syntax Conventions for Instances

Name	Description
<i>baseName</i>	A string of printable ASCII characters that does not include reserved characters. I0, R, C
<i>instName</i>	<i>baseName</i> optionally followed by a number or a range of numbers separated by a colon. Single-bit and iterated instance names must be unique. A single-bit name cannot imply more than one member. An iterated instance name must imply the same number of members as iterations. Must be in the form $baseName [< n [: n] >]$ where n is a positive integer.

Virtuoso Schematic Editor User Guide

Connectivity and Naming Conventions

Table 2-6 Expanded Names for Wire, Net, and Pin Names

Sample Name	Members	Expanded Form
clk	1	clk
data<2>	1	data<2>
<*1>base	1	base
<*2>term	2	term, term
<*2>(a,b),c	5	a,b,a,b,c
<*2>(a,<*2>b)	6	a,b,b,a,b,b
b<0:2>	3	b<0,1,2> or b<0>,b<1>,b<2>
b<0:2:1>	3	b<0,1,2>
b<3:0:2>	2	b<3,1>
b<0:2*2>	6	b<0,0,1,1,2,2,>
b<(0:2)*2>	6	b<0,1,2,0,1,2>
b<0,2*2>	3	b<0,2,2>
b<(0,2)*2>	4	b<0,2,0,2>
b<0,1:3:4*1,2:2>	3	b<0,1,2>
b<0:1,2:2>	3	b<0,1,2>

Note: You can use up to 1,024 characters for a name, but the schematic editor ignores any blank or tab characters found before or after a name. To determine the *n*th member name, use a zero-based counting method to find the *n*th element in the expanded form. For example,

- Member zero of <*2>(a,<*2>b) is a
- Member one of <*2>(a,<*2>b) is b
- Member two of <*2>(a,<*2>b) is b, and so on

If the name contains a vector expression, such as b<0:1,2:2>, then

- Member zero in this case is called b<0>
- Member one in this case is called b<1>

Virtuoso Schematic Editor User Guide

Connectivity and Naming Conventions

- Member two in this case is called `b<2>`

Table 2-7 Expanded Names for Instances

Name	Members	Expanded Form
H10	1	H10
J20<0:1>	2	J20<0>, J20<1>
K30<2:1>	2	K30<2>, K30<1>

Note: You can use up to 1,024 characters for a name, but the schematic editor ignores any blank or tab characters found before or after a name.

Supply and Ground Sensitivity

When a digital net connects to an analog net, the logic value on the digital net requires to be converted to a voltage value on the analog net for simulation. This is reversed when a analog net connects to a digital net, the voltage value on the analog net requires to be converted to a logic value. With multiple supply designs, you need to specify the applicable supply (out of many) to be used for the conversion. You can do this by adding two properties;

`supplySensitivity` and `groundSensitivity` to the cellview terminals. These properties will subsequently distinguish the power and ground supplies applicable to the net.

Note: The power and ground terminals must exist before they can be used in the sensitivity fields.

Setting Supply and Ground Sensitivity (Schematic Window)

You can specify the supply and ground terminal names that are to be used to make a pin's terminals sensitive to specified supply terminals. When editing, this allows the sensitivity terminal attributes to be set correctly at the time of creation.

Note: Terminal sensitivity is applied through a module terminal (port), not directly to a global net. If you choose to set supply sensitivity, ground sensitivity, or both to a pin's terminals, all of the terminals have this attribute set.

The Create Pin – Schematic and Edit Object Properties – Pin forms have supply sensitivity options with corresponding drop-down fields to select appropriate terminals.

For example, the terminal `pin1` could be set so that it would be sensitive to the power terminal `vdd` and the ground terminal `gnd`. If there is no pre-established supply sensitivity of the pin's terminal, then these fields will be left blank. In such cases, you can type in the names of the power and ground terminals, or use the *value* option to populate the field.

The pin name and the specified power and ground terminals must be single-bit terminals (scalar) or single-bit terminals with vector elements (vector bit). They must pre-exist in the database before the sensitivity relationship is established. If you specify terminals that do not already exist in the database, or if the terminal name is not a single-bit, this will invoke an error condition. Subsequently, no pin placement will take place, the error field will be highlighted, and an error message will be displayed in the CIW.

Note: The same behavior applies to the Edit Object Properties – Symbol Pin form in the Symbol Editor.

Virtuoso Schematic Editor User Guide

Connectivity and Naming Conventions

To set supply and ground sensitivity:

1. In the Virtuoso Schematic Editing window, select *Create – Pin* to display the Create Pin form.
2. Provide a pin name. The pin name should be a single-bit terminal or single-bit terminal with vector elements for a pin to be created with pwr or gnd sensitivity. If the name is not a single-bit, a signal type can only be added if the Expand Busses option is also selected which will apply to the array of pins.

Create Pin

Names

Direction

Usage

Signal Type

☐ Expand busses

☐ Place multiple pins

▼ **Net Expression**

☐ Attach to pin

☐ Derive from pin name

Property Name

Default Net Name

Font

Height

Style

Justification

▼ **Supply Sensitivity**

Attach To

Power Pin

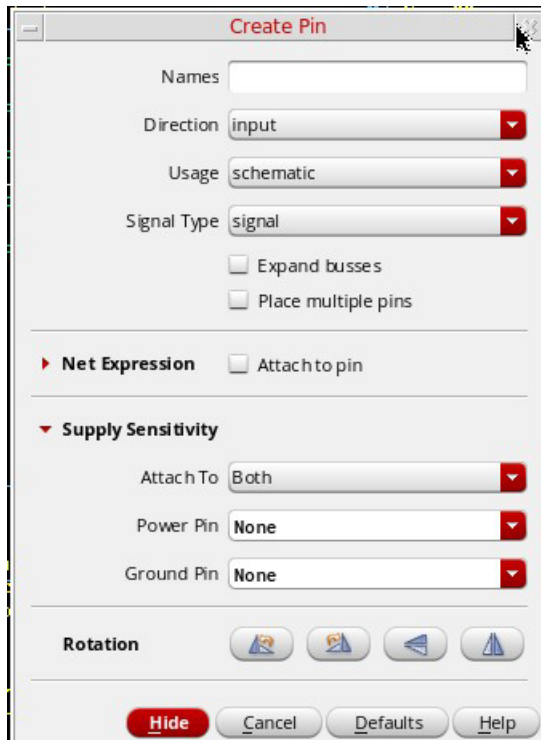
Ground Pin

Rotation

Virtuoso Schematic Editor User Guide

Connectivity and Naming Conventions

3. Expand *Supply Sensitivity* to reveal three drop-down lists: *Attach To*, *Power Pin*, and *Ground Pin*.



4. Select the required option from the *Attach To* drop-down list to apply to the next pin.

For example, if you set *Attach To* to *Both*, *Power Pin* to *vdd*, and *Ground Pin* to *gnd*, the terminals for the next pin (*pin1*) will be sensitive to the power pin *vdd* and the ground pin *gnd*. Likewise, if you set *Attach To* to *Power*, *pin1* will only be sensitive to the power pin *vdd* even if there is an entry in the *Ground Pin* field.

5. In the *Power Pin* field, select from the drop-down list, or type in, the name of the power pin that the next pin is to be sensitive to.

Note: The drop-down menu lists all the single-bit power (or ground) terminals in the cellview. This scrollable list is automatically (alphanumerically) updated when a terminal on the schematic is added, removed, or its signal type is changed. This also applies to the *Ground Pin* drop-down list.

6. In the *Ground Pin* field, select from the drop-down list, or type in, the name of the ground pin that the next pin is to be sensitive to.

The default is the first or most recently used ground pin.

7. If required, collapse the *Supply Sensitivity* section to hide the options you have set.

Related Topics

[Create Pin – Schematic \(form\)](#)

[Setting Supply and Ground Sensitivity \(Symbol Window\)](#)

[Editing the Sensitivity Terminals of Schematic and Symbol Pins](#)

[Editing the Optical Signal Types](#)

Setting Supply and Ground Sensitivity (Symbol Window)

The behavior of the Create Pin form in the symbol window is as described for the schematic window (see [Setting Supply and Ground Sensitivity \(Schematic Window\)](#)), in that you would also expand *Supply Sensitivity* to expand the Create Pin form.

The specified power and ground terminals must be single-bit terminals and pre-exist in the database before the sensitivity relationship can be established. If you specify terminals that do not already exist in the database, or if the terminal name is not a single-bit, a warning is given and the selection will revert back to the previous choice.

Related Topics

[Create Pin – Symbol \(form\)](#)

[Setting Supply and Ground Sensitivity \(Schematic Window\)](#)

[Editing the Sensitivity Terminals of Schematic and Symbol Pins](#)

[Editing the Optical Signal Types](#)

Editing the Sensitivity Terminals of Schematic and Symbol Pins

Use the Edit Object Properties form to edit the sensitivity terminal attributes of the schematic and symbol pins.

1. With the pin to be edited selected, choose *Edit – Properties – Objects*.

The [Edit Object Properties – Pin](#) form appears.

2. Select the appropriate terminal for *Pwr Sensitivity* using the corresponding value drop-down field.

Note: The specified power and ground terminals must be single-bit terminals and pre-

exist in the database before the sensitivity relationship is established. If you specify terminals that do not already exist in the database, or if the terminal name is not a single-bit, this will invoke an error condition. Subsequently, no pin placement will take place, the error field will be highlighted, and an error message will be displayed in the CIW.

3. Select the appropriate terminal for *Gnd Sensitivity* using the corresponding value drop-down field.

For example, we could set the terminals for `pin1` so that it would be sensitive to the power terminal `vdd` and the ground terminal `gnd`. If there is no pre-established supply sensitivity of the pin's terminal, then these fields will be left blank. In such a case, you can type in the names of the power and ground terminals, or use the *value* option to fill the field.

4. Click *OK*.

Note: The same behavior applies to the Edit Properties - Pin form in the Symbol Editor.

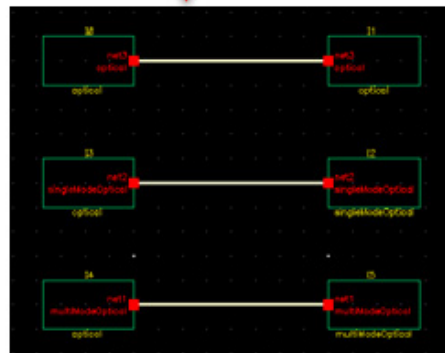
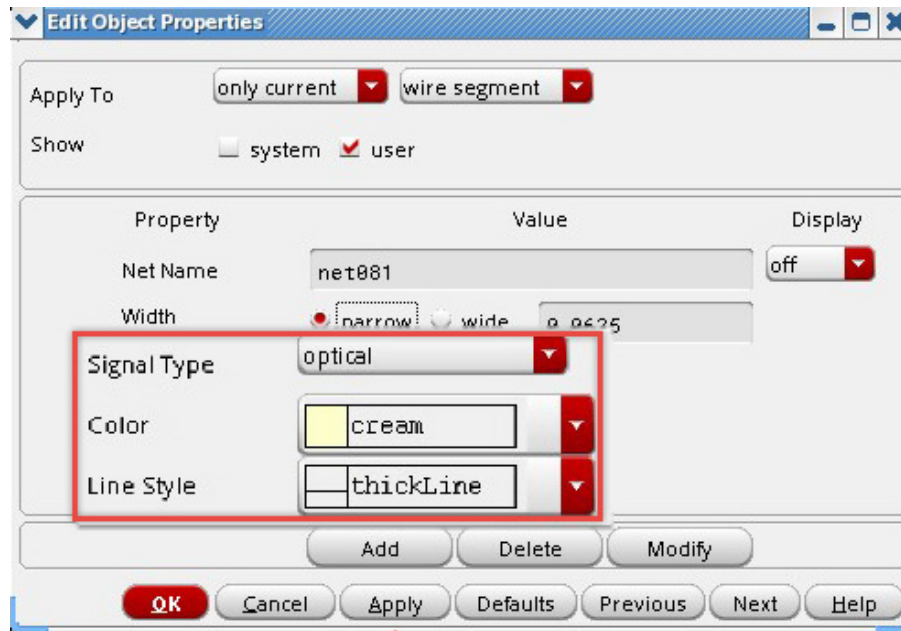
Editing the Optical Signal Types

Note: Virtuoso Photonics Solution - You need the *Virtuoso_Photonics_Option* license to use the functionality discussed in this section. For information about obtaining the required license, contact your local Cadence representative.

Virtuoso Schematic Editor User Guide

Connectivity and Naming Conventions

Use the Edit Object Properties – Pin form to view or edit optical signal types `optical`, `singleModeOptical`, and `multiModeOptical` on pins and wires. You can set a distinctive color and line style to identify the optical wires.



Checking Sensitivity Between Schematic and Symbol Cellviews

The sensitivity of a terminal should be consistent between the schematic and symbol cellviews. You can ensure that this consistency is validated by selecting the *Match Terminal*

Virtuoso Schematic Editor User Guide

Connectivity and Naming Conventions

Supply Sensitivity check box in the Schematic Check Options form (*Options – Check*). This option is *on* by default.

The screenshot shows the 'Schematic Check Options' dialog box with the following settings:

- Connectivity**
 - ☒ Extract connectivity during schematic checking
 - ☒ Beep When 0 Errors/Warnings
 - ☐ Set Signal Type from Net Name and Type Registration
 - Check and Save Action on Error: ☐ Save ☐ No Save ☒ Ask Me
 - From a Multisheet Index, Check: ☐ all ☒ only if needed
 - Wire Name Glue Distance:
- Schematic Rules Checks**
 - ☒ Run Schematic Rules Checks
 - Rules Setup ...
- Cross View Checks**
 - Cross View Check Severity: ☐ ignored ☒ warning ☐ error
 - Views To Check:
 - ☐ Match Pin Names Exactly
 - ☒ Check Pin Order
 - ☒ Only check views with explicit pin order
 - ☒ Match Inherited Terminals
 - ☒ Ignore Terminals with Net Expression and no Terminal in the Other View
 - Ignore Terminals for Specified Views
 - View Names:
 - ☒ All Terminals ☐ Terminal Names
 - ☒ Match Terminal Net Expressions
 - ☒ Match Terminal Supply Sensitivity
 - ☒ Match Terminal Signal Type

Buttons at the bottom: **OK**, Cancel, Defaults, Apply, Help.

Related Topics

[Schematic Check Options](#) (form)

Propagating Optical Signal Types (Virtuoso Photonics Option)

Note: Virtuoso Photonics Solution - You need the *Virtuoso_Photonics_Option* license to use the functionality discussed in this section. For information about obtaining the required license, contact your local Cadence representative.

The Optical options let you propagate the optical signals and set the optical wire style.

Optical signal types are propagated from an instTerm to a net when the signal type of the net has been specified as `optical` and has only optical connections. A net that already has an optical signal type is not changed.

For matching optical connections, the common optical signal type is used. Mixed optical connections are resolved by using the specific signal types.

`optical + singleModeOptical => singleModeOptical`

`optical + multiModeOptical => multiModeOptical`

A single + multi connection remains unchanged and is processed by the checker.

Environment Variable: [`opticalSigTypePropagation`](#)

Virtuoso Schematic Editor User Guide

Connectivity and Naming Conventions

You can disable the optical signal type propagation by clearing the *Propagate Optical Signal Type Values* check box. For more information about the other options on the Schematic Check Options form, see [Schematic Check Options](#).

Virtuoso Schematic Editor User Guide

Connectivity and Naming Conventions

Schematic Check Options

Connectivity

- ☒ Extract connectivity during schematic checking
- ☒ Beep When 0 Errors/Warnings
- ☐ Set Signal Type from Net Name and Type Registration

Check and Save Action on Error: ☐ Save ☐ No Save ☒ Ask Me

From a Multisheet Index, Check: ☐ all ☒ only if needed

Wire Name Glue Distance:

Optical

- ☒ Propagate Optical Signal Type Values
- ☒ Set Optical Wire Style:
- ☒ Set Optical Pin Color:

Schematic Rules Checks

- ☒ Run Schematic Rules Checks

Cross View Checks

Cross View Check Severity: ☐ ignored ☒ warning ☐ error

Views To Check:

- ☐ Match Pin Names Exactly
- ☒ Check Pin Order
 - ☒ Only check views with explicit pin order
- ☒ Match Inherited Terminals
 - ☒ Ignore Terminals with Net Expression and no Terminal in the Other View
- Ignore Terminals for Specified Views
 - View Names:
 - ☒ All Terminals ☐ Terminal Names
- ☒ Match Terminal Net Expressions
- ☒ Match Terminal Supply Sensitivity
- ☒ Match Terminal Signal Type

Virtuoso Schematic Editor User Guide

Connectivity and Naming Conventions

Equivalent Connectivity Time

You can avoid the need to re-run schematic Check and Save or netlisting if a cellview has equivalent connectivity to the previous version. Before updating a symbol, check the Equivalent Connectivity Time to ensure that it is less than or equal to the time set with the `lastSchematicExtraction` timestamp property.

You can use the following SKILL functions to specify Equivalent Connectivity Time information in a cellview:

- [dbCellViewHasEquivalentConnectivityTime](#)
- [dbGetCellViewEquivalentConnectivityTime](#)
- [dbSetCellViewEquivalentConnectivityTime](#)
- [dbUnsetCellViewEquivalentConnectivityTime](#)

The [autoEquivalentConnectivityTime](#) environment variable lets you automatically sign off cosmetic changes to a symbol cellview. For more details, see [Automatically Signing Off Cosmetic Changes to Symbols](#).

Automatically Signing Off Cosmetic Changes to Symbols

The [autoEquivalentConnectivityTime](#) environment variable lets you automatically sign off cosmetic changes to a symbol cellview without triggering a connectivity check in all schematics that instantiate the symbol.

A change is considered to be cosmetic when it does not modify the symbol interface or impact the connectivity of instantiating schematics. For example, adding a label, adding a shape or line, and so on. Modifying, moving, adding, or deleting a symbol or its port impact the connectivity and are not automatically signed off.

Enabling `autoEquivalentConnectivityTime` is effective only if the changes do not impact the interface definitions of a symbol.

The interface of a symbol is defined as the information provided by the SKILL function [schSymbolToPinList](#). Such information includes all cellview, terminal and pin attributes and properties, including pin names, directions, and bounding boxes.

Virtuoso Schematic Editor User Guide

Connectivity and Naming Conventions

Creating Schematics

This chapter provides information about creating a schematic.

- [Getting Started with Schematics](#)
- [Adding Instances](#)
- [Adding Blocks](#)
- [Adding Pins](#)
- [Adding Wires](#)
- [Adding Wire Names](#)
- [Adding Net Expression Labels to Create an Inherited Connection](#)
- [Adding netSet Properties to Create an Inherited Connection](#)
- [Analyzing Net Expression Property Names and Evaluated Names](#)
- [Adding Solder Dots](#)
- [Adding Note Text](#)
- [Adding Patch Cords Using the Add Instances Form](#)
- [Adding Patch Cords Using the Create Patchcord Form](#)
- [Adding Probes](#)
- [Creating a Mapping Schematic](#)

Getting Started with Schematics

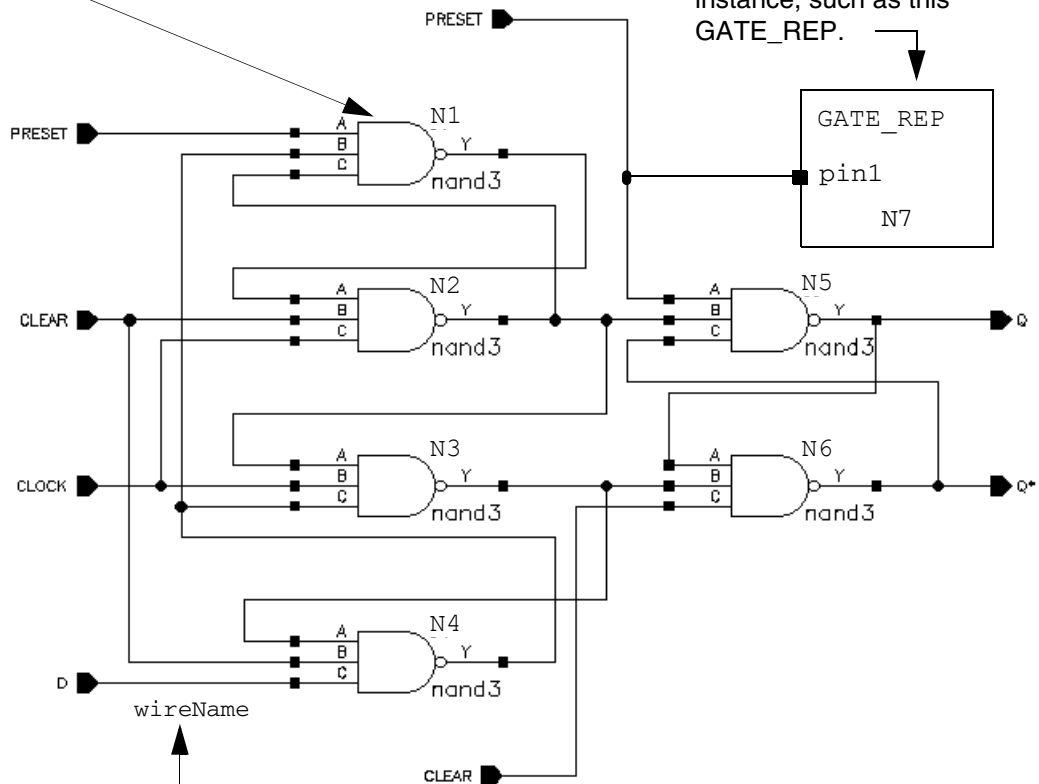
One of the best ways to begin a schematic in an empty window is by adding objects in a certain order, as indicated below:

1. Add instances from your library, such as this nand3 gate.

2. Add blocks to hold the place of an undetermined instance, such as this GATE_REP.

3. Add schematic pins, such as this input pin.

4. Add wires and wire names to connect schematic pins to instances.



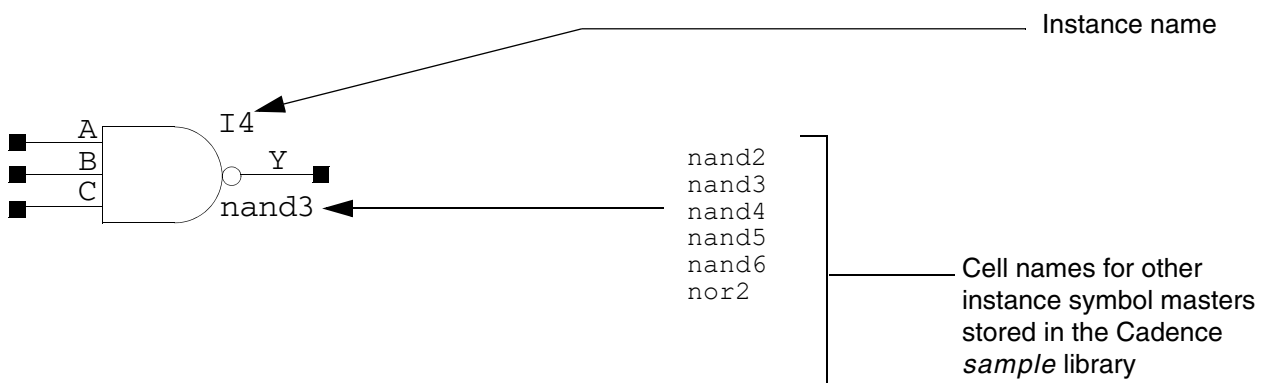
Adding Instances

You can add instances to your design using several methods with different options.

- [What Is an Instance?](#)
- [Methods of Adding an Instance to Your Design](#)
- [Adding Instances One by One](#)
- [Adding Instances by Rows and Columns \(an Array\)](#)
- [Adding Instances Using an Iterative Expression](#)
- [Changing the Default Instance Name Prefix](#)
- [Editing Instance Masters](#)
- [Specifying the View to Use While Placing the Instance](#)
- [Changing the View after an Instance Is Already Placed](#)
- [Renumbering Instances](#)
- [Magnifying and Rotating Instances Using the symbol_xform View](#)

What Is an Instance?

An instance is a database object that represents a master symbol cellview. When you add an instance to your design, you create a reference to the master symbol from a reference library.



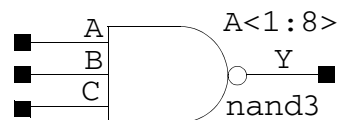
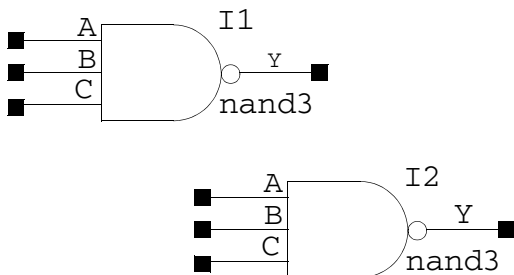
You can have several instances of the same master symbol cellview in a design, but the instance name must be different. For example, the `nand3` gate is a master symbol stored in the Cadence® *sample* library. Each time you add a `nand3` gate to your design, you add an instance of the `nand3` master. The instance names can be `I1`, `I2`, `I3`, and so on.

Methods of Adding an Instance to Your Design

Plan to place instances

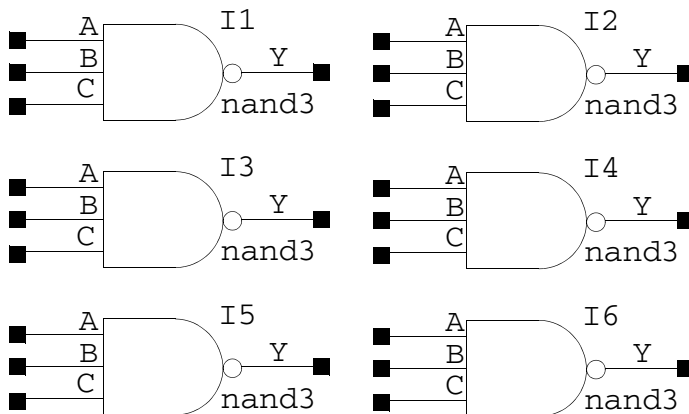
- One by one
- By rows and columns (an array)
- By using iterative expressions

One by one. Use this method to place instances singly.



Using an iterative expression. To save space in your design window when you need to add several instances of the same type, you can express multiple unique names with an iterative expression.

By rows and columns (an array). Use this method to place instances in an array.



Adding Instances One by One

The *Create – Instance* command lets you place multiple copies of a single instance in a schematic, based on a single *lib/cell/view* name.

To add instances singly in your design window,

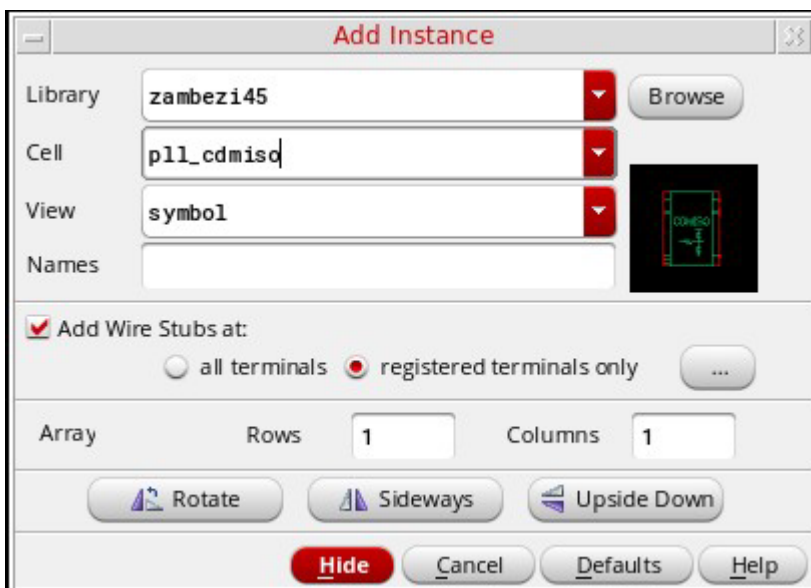
1. From the schematic window, choose *Create – Instance*.

The Add Instance form appears.

Important

This form contains the *Library*, *Cell*, and *View* text fields, which can be auto-completed. When you type the characters, the drop-down list shows the names containing the matching characters at any position in the name (beginning, middle, or end of the name). The non-editable fields appear as disabled. Any incorrect input for a field highlights the same. Select from the drop-down list and specify the name in the text field. The drop-down list is retrieved from `cds.lib`. The fields are highlighted in magenta if there are no matching library or cell or view names for the characters mentioned in the field.

Note: A thumbnail will also be displayed if there is one available for the current cellview. For more information on the generation and use of thumbnails see Thumbnail View Functions in the *Cadence User Interface SKILL Reference* in the first instance.



2. In the *Library*, *Cell*, and *View* fields, specify the cellview of the instance.

Virtuoso Schematic Editor User Guide

Creating Schematics

You can also click the [Browse](#) button to assist you in finding the cellview you want to add.

You can designate the *Browse* button to open either the [Component Browser](#) or the [Library Browser](#). If the cell you want does not exist, you need to create it using the [symbol editor](#).

If you do not add component/instance names in the *Names* field, default instance names prefixed with the letter `I` are generated.

3. Choose whether or not to *Add Wire Stubs* at either *all terminals* or for *registered terminals* only.

Clicking on the *Browse (...)* button will display the Net Name Registration form.

For more information on this option, see [Registering Terminals for Use With Add Wire Stubs](#).

4. Move the pointer into your cellview.

The instance appears attached to the pointer.

5. Click in your cellview to place the instance.

The *Create – Instance* command is repeatable (if repeat command mode is turned on) and continues to prompt you to add another instance. To stop generating instances, press the `ESC` key or choose another command.

Adding Wire Stubs

In the Add Instance you can choose to *Add Wire Stubs at:* either

- *all terminals*

Selecting this option will create wire stubs for all terminals in the cellview.

Note: Any wire stubs that are created for unregistered terminals will not have a wire label added.

or

- *registered terminals only* (see also Registering Terminals for use with Add Wire Stubs and Registering Default Net Names)

Selecting this option will create wire stubs only for those terminals that are registered with a net. This is the default option.

Note: Wire labels (the same used by the default net name) will automatically be added to each new wire stub. Label justification is set according to the terminal position.

To select one of the above options, ensure that the *Add Wire Stubs at:* check box is first activated.

Choosing to add a wire stub, to one or more terminals, creates a wire stub that is orthogonal to the edge of the selection box for all terminals, or only those that have been registered, if preferred.

Virtuoso Schematic Editor User Guide

Creating Schematics

Add Instance

Library:

Cell:

View:

Names:

☒ Add Wire Stubs at:

☒ all terminals ☐ registered terminals only

Array: Rows: Columns:

Macro name:

Fall time:

Rise time:

Level 1 voltage:

Level 0 voltage:

Level X voltage:

Level Z voltage:

Figure 3-1 Adding Wire Stubs for Terminals in the Add Instance Form

If you have chosen to add wire stubs to an instance, the instance drag image on placement will preview the instance terminals that are connected to the wire stubs.

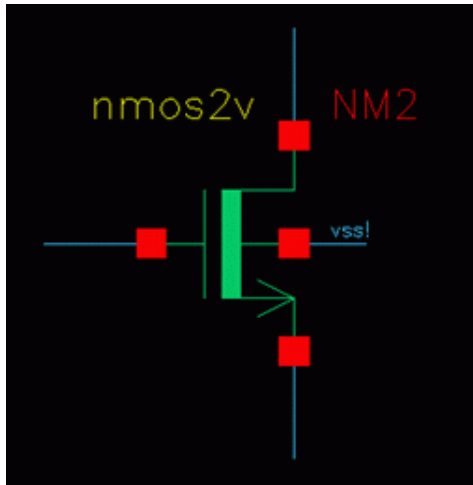


Figure 3-2 Registered Terminal Connected to Wire Stub with the Net “vss!”

Registering Terminals for Use With Add Wire Stubs

Note: See also [Registering Default Net Names](#).

To register terminals for use with the [Add Wire Stubs](#) option:

1. Select the ... (net name registration) option to display the Net Name Registration form.

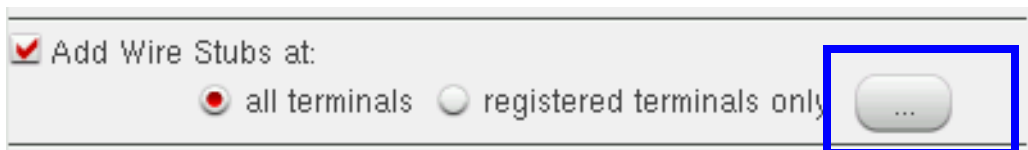
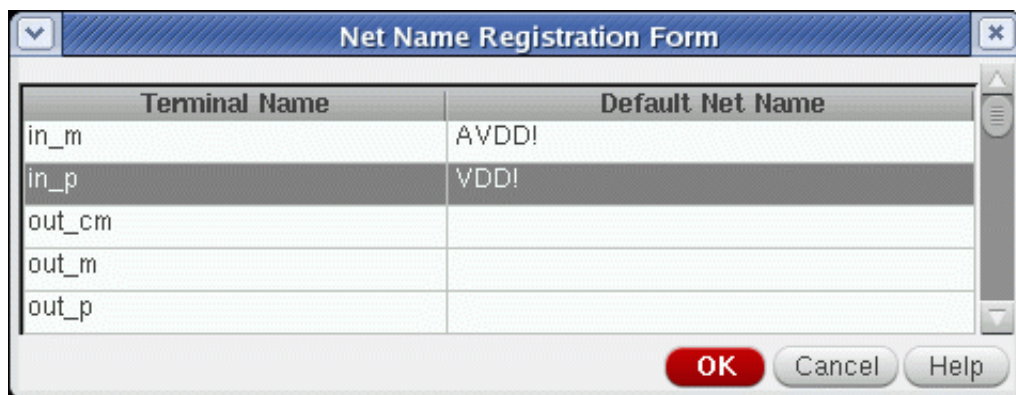


Figure 3-3 Net Name Registration for Adding Wire Stubs

The Net Name Registration form displays terminals (in the *Terminal Name* column) and if there is a *Default Net Name* registered to them.

From here, you can change the default net name.

2. Double-click the *Default Net Name* column of the instance whose terminal that is to be updated, and enter the new default net name.



The image shows a dialog box titled "Net Name Registration Form". It contains a table with two columns: "Terminal Name" and "Default Net Name". The table has six rows. The first row has "in_m" in the first column and "AVDD!" in the second. The second row has "in_p" in the first column and "VDD!" in the second. The third row has "out_cm" in the first column and is empty in the second. The fourth row has "out_m" in the first column and is empty in the second. The fifth row has "out_p" in the first column and is empty in the second. At the bottom right of the dialog are three buttons: "OK" (highlighted in red), "Cancel", and "Help".

Terminal Name	Default Net Name
in_m	AVDD!
in_p	VDD!
out_cm	
out_m	
out_p	

Figure 3-4 The Net Name Registration Form

3. Click the *OK* button after you have finished updating the *Default Net Name* for one or more terminals.

The chosen terminals will now be registered for use with the Add Wire Stubs option.

Adding Instances by Rows and Columns (an Array)



See also [Adding Instances One by One](#).

To add an array of a single instance using rows and columns,

1. From the schematic window, choose *Create – Instance*.

The [Add Instance](#) form appears.

2. In the *Library*, *Cell*, and *View* fields, specify the cellview of the instance.

You can also click the [Browse](#) button to assist you in finding the cellview you want to add.

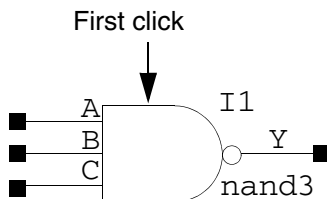
If you do not add instance names in the *Names* field, default instance names prefixed with the letter **I** are generated.

3. Fill in the *Rows* and *Columns* fields.

4. Move the pointer into your cellview.

The first instance appears, attached to the pointer.

5. Click to place the first instance.

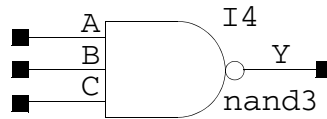
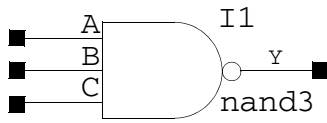


6. Drag the mouse vertically and click a location for the first instance of the second row.

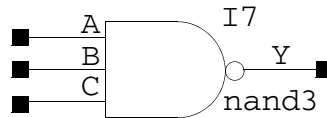
Virtuoso Schematic Editor User Guide

Creating Schematics

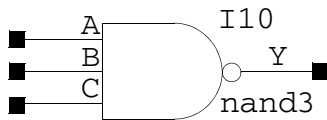
The rest of the instances follow the cursor.



← First instance of the second row

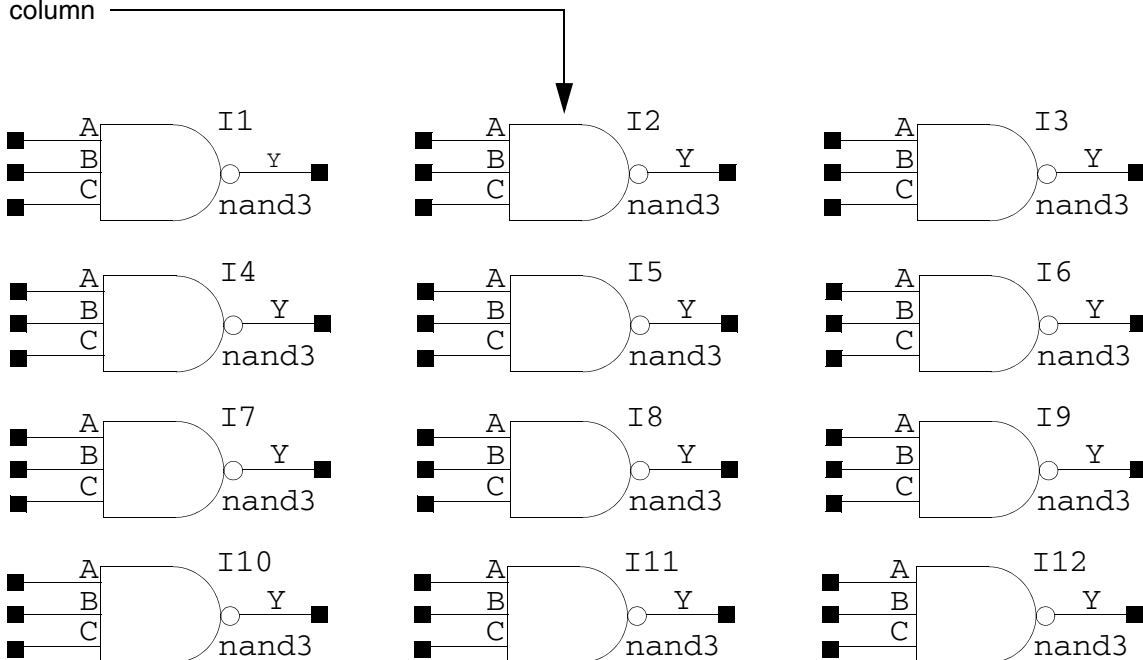


The rest of the instances follow the cursor.



7. Drag the mouse horizontally and click a location of the first instance of the second column. The rest of the instances follow the cursor.

First instance of the second column

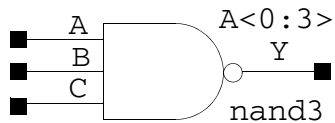


The *Create – Instance* command is repeatable and continues to prompt you to add another instance. To stop generating instances, press the `ESC` key or choose another command.

Adding Instances Using an Iterative Expression

Instance iteration is a compact way of displaying repeated instances of a symbol in your schematic. It is particularly useful in bus-type or data-flow architectures that have identical structures to handle each bit on the bus.

To save space in your design window when you need to add several instances of the same type, you can express multiple unique names with an iterative expression. For example, `A<0:3>` generates one graphic representing four instances: A0, A1, A2, and A3.



Using iterated instances not only saves space in your design, it makes your design easier to read.

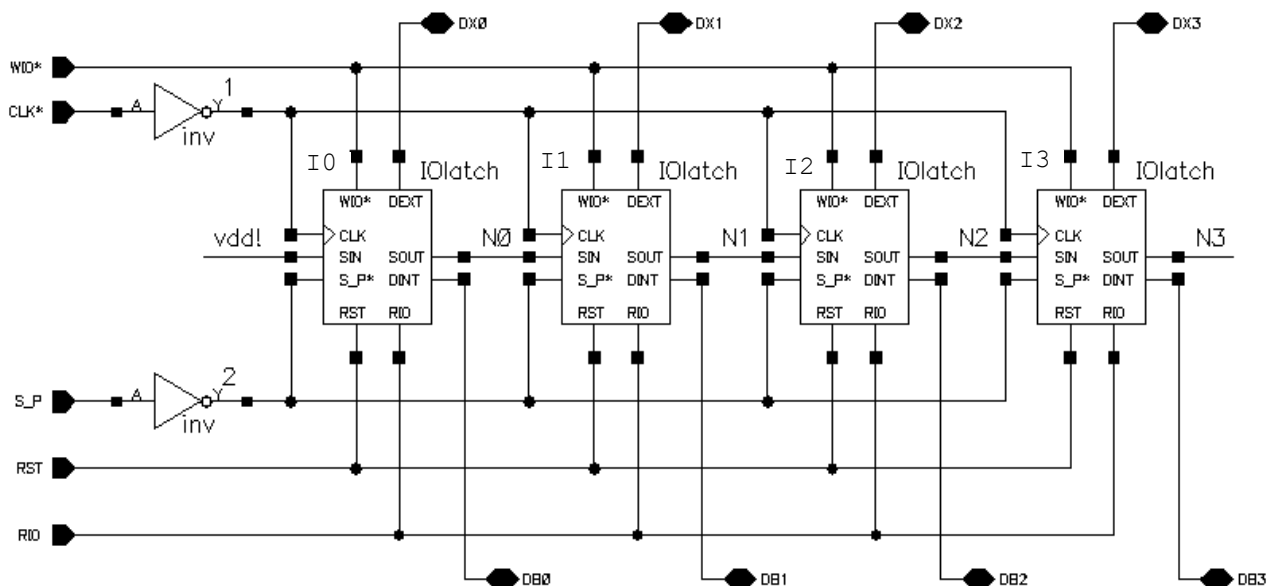
To add instances using an iterative expression:

1. From the schematic window, choose *Create – Instance*.

The Instance Form appears.

2. In the *Names* field, type a name using iterative expression format.

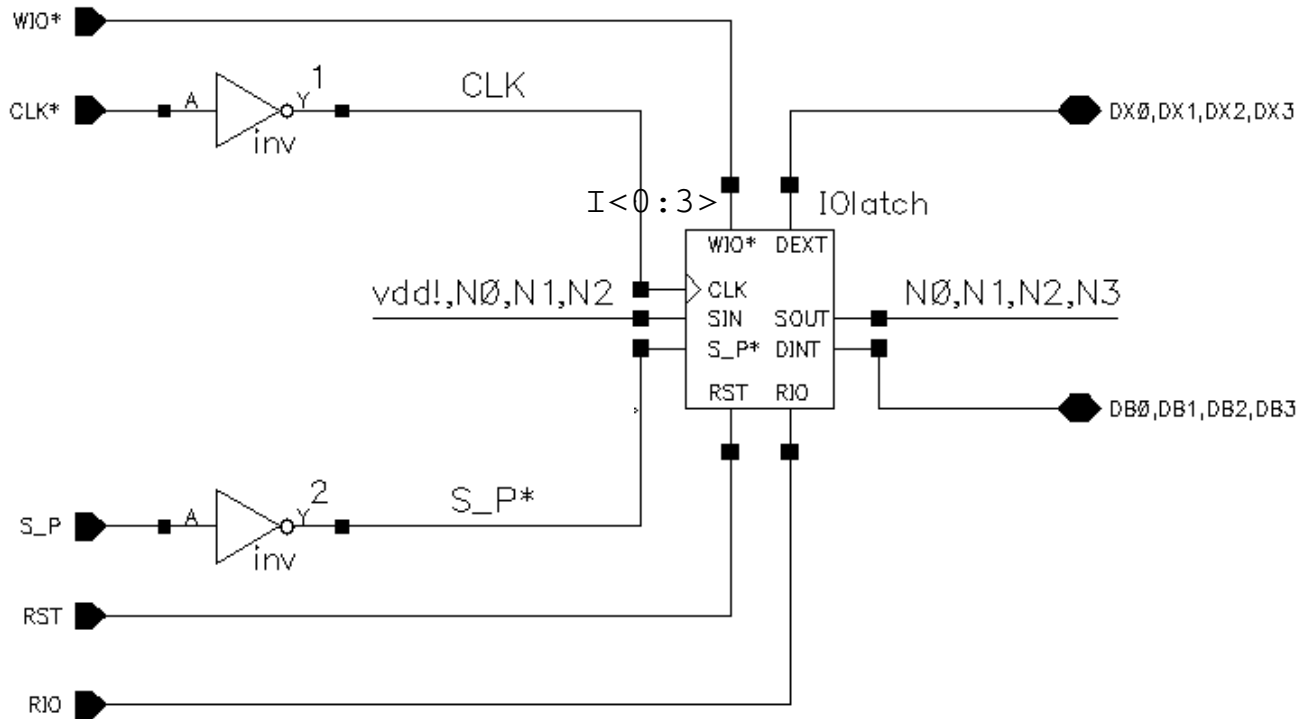
The following schematic shows four instances: I0, I1, I2, and I3. Compare this schematic with the following schematic showing iterated instances:



Virtuoso Schematic Editor User Guide

Creating Schematics

The following schematic shows one latch iterated four times: $I < 0 : 3 >$. Compare this schematic with the previous schematic showing four individual instances.



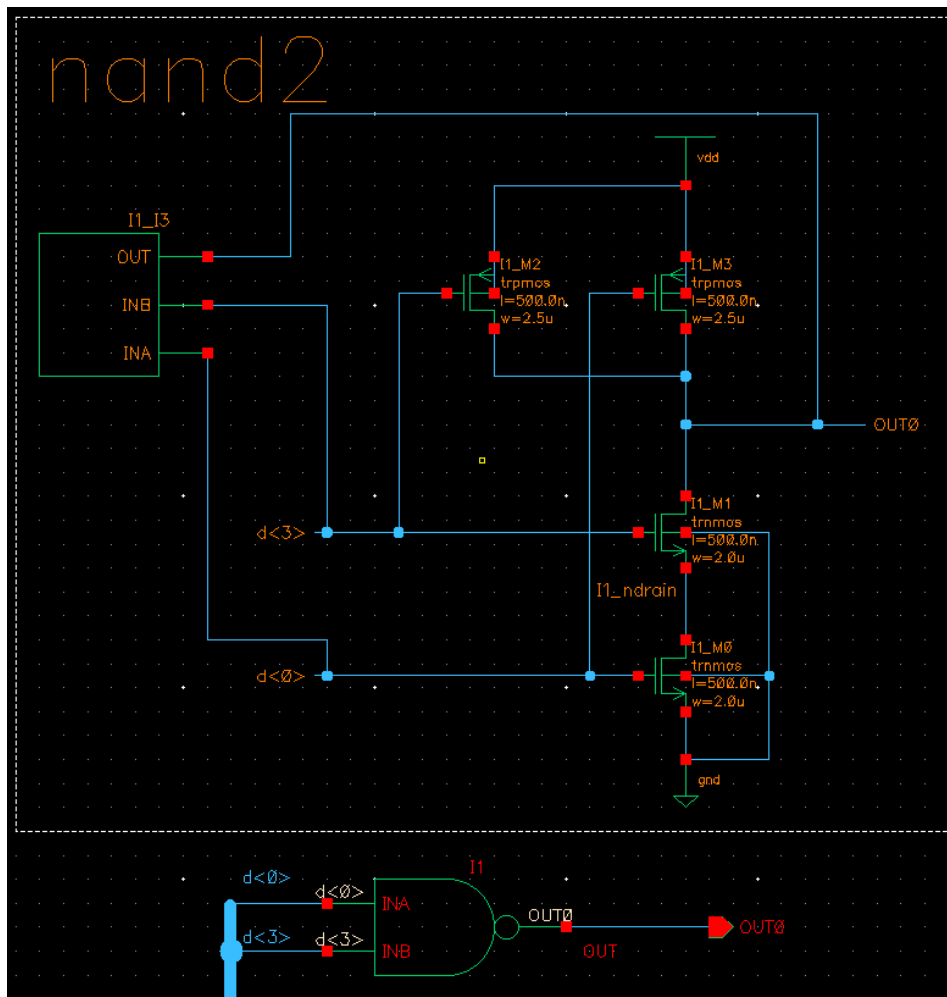
Flattening an Instance

Using the Flatten command, you can bring an instance up to the current hierarchy level, thereby enabling editing while preserving the connectivity of the flattened instance.

To flatten an instance:

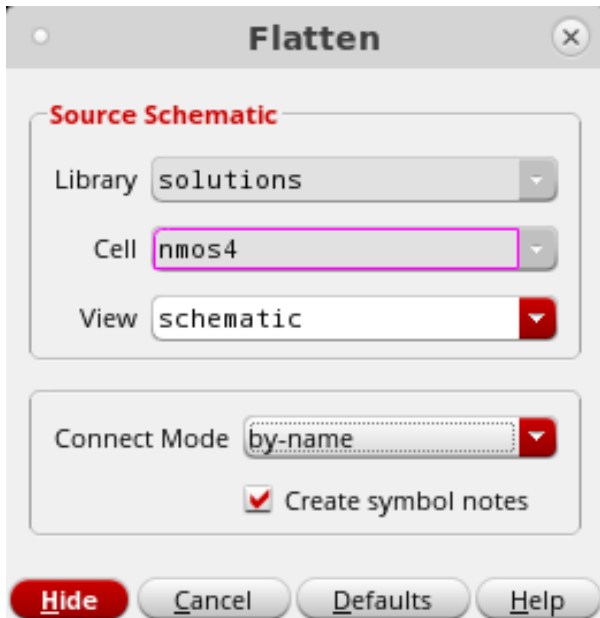
1. In Virtuoso Schematic Editor XL, choose *Edit – Hierarchy – Flatten*.
2. Click the instance to be flattened.

The outline of the flattened instance appears.



3. (Optional) Press the F3 key to display the Flatten form.

The *Flatten* form appears.



The screenshot shows the 'Flatten' dialog box with the following settings:

- Source Schematic:**
 - Library: solutions
 - Cell: nmos4
 - View: schematic
- Connect Mode:** by-name
- ☒ Create symbol notes

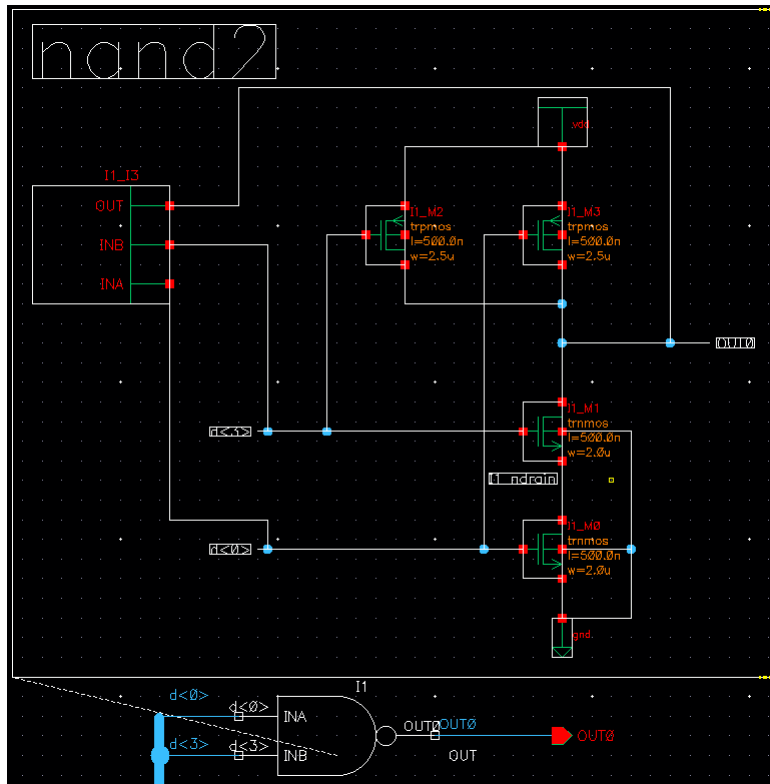
Buttons at the bottom: Hide, Cancel, Defaults, Help.

4. (Optional) Edit the source by changing the *View* field.
5. Specify the connection mode by using the *Connect Mode* drop-down list.

Virtuoso Schematic Editor User Guide

Creating Schematics

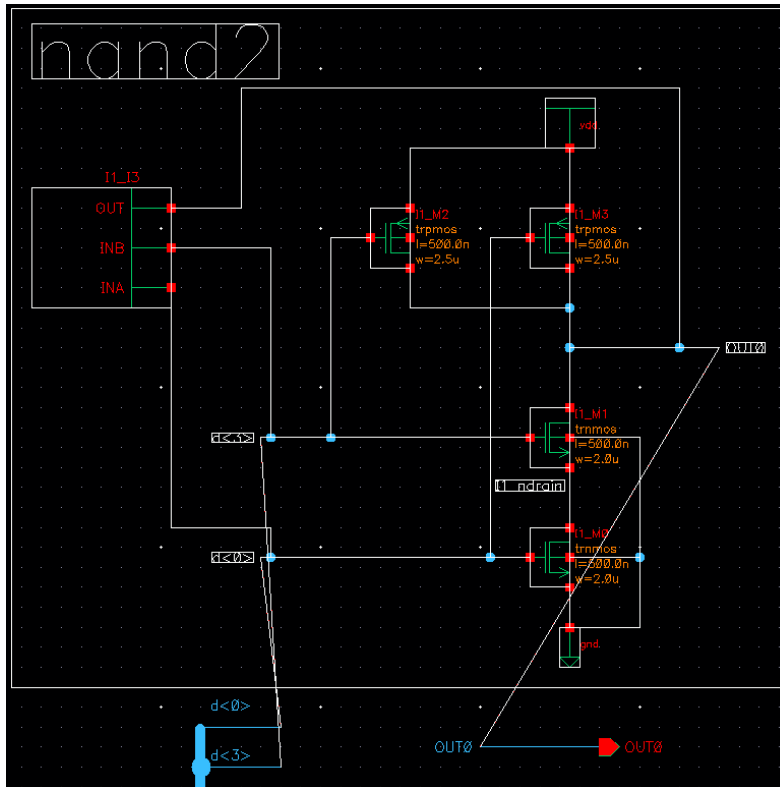
- ❑ *by-name* connects the flattened content to the existing schematic using connect-by-name:



Virtuoso Schematic Editor User Guide

Creating Schematics

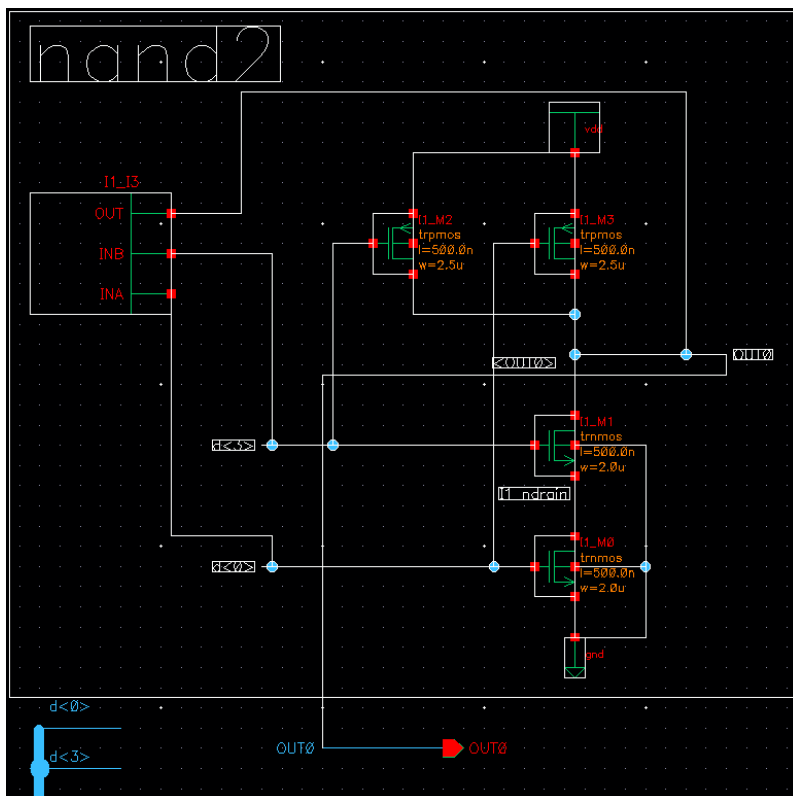
- ❑ *flight* connects the flattened content to the existing schematic using flight lines:



Virtuoso Schematic Editor User Guide

Creating Schematics

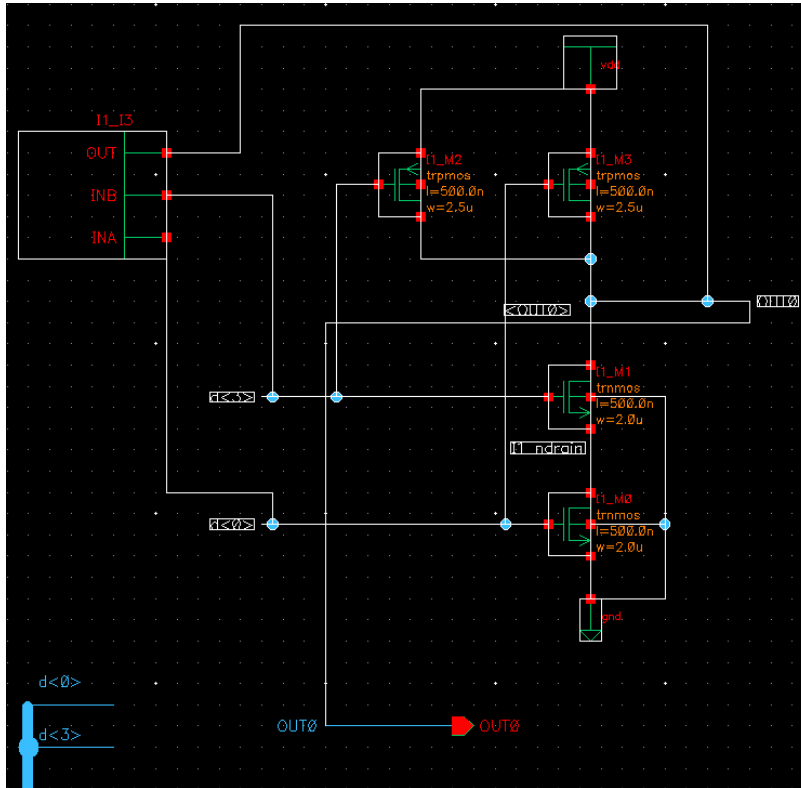
- ❑ *wire* connects the flattened content to the existing schematic using routed wires:



Virtuoso Schematic Editor User Guide

Creating Schematics

6. Select or deselect the *Create symbol notes* check box which creates a symbol note and a note line. Below is an example when the check box is deselected:



7. Click *Hide*.
8. Click on the canvas to place the flattened instance.

The flattened instance is placed on the canvas. When a flattened instance is placed on the canvas it can be interacted with as normal, that is, all its connectivity is intact, cross-probing is working, and so on.

Changing the Default Instance Name Prefix

Each instance requires a unique name. If you do not specify instance names on the Add Instance form, the schematic editor generates unique names automatically. The prefix is a symbol cellview property, not an instance object property.

To change the default instance name prefix from Γ to a prefix that you designate:

1. From the CIW, open a symbol cellview.
2. Choose *Edit – Properties – Cellview*.
The Edit Cellview Properties form appears.
3. Show the system properties on the form by choosing *system*.
4. Change the *instNamePrefix* property.
You can use more than one character.
5. Click *OK*.

When you place new instances of this same symbol in the same cellview, the new prefix name appears.

Editing Instance Masters

To edit the master of an instance and automatically change all occurrences of the instance,

1. From the CIW, open a symbol cellview.
2. Select the part of the symbol you want to edit.

You might want to select the shape, the pin, the label, or the selection box.

3. Choose *Edit – Properties – Objects* to edit the symbol.

The Edit Objects Properties form appears.

To edit a single instance, you edit object properties, not cellview properties. The schematic editor changes all cellviews containing instances of this master.

Specifying the View to Use While Placing the Instance

To change instance views (for example, from a `nand2 symbol` view to a `nand2 symbolNeg` view) as you add an instance,

1. From the schematic window, choose *Create – Instance*.
2. Before placing the instance, press the `Shift` key and click to toggle through the symbol views.

The following diagram shows an instance changing views from `nand2 symbol` to `nand2 symbolNeg`.



3. Place the instance.

Changing the View after an Instance Is Already Placed

To change instance views (for example, from a `nand2 symbol` view to a `nand2 symbolNeg` view) after you have added an instance,

1. From the schematic window, select the instance you want to change.
2. Choose *Edit – Alternate View*.

Renumbering Instances

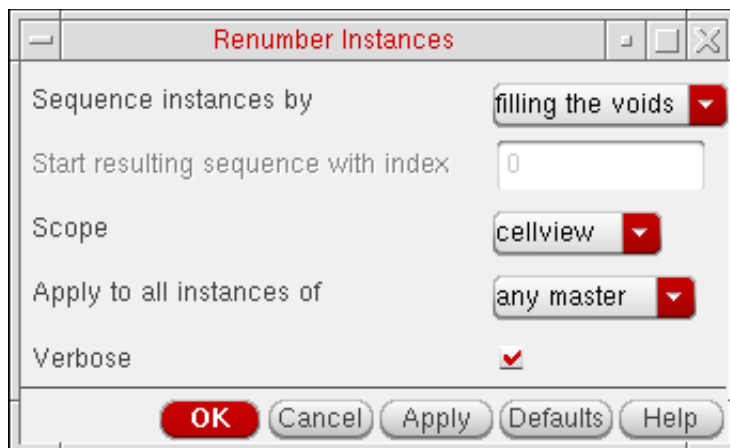
If you deleted several instances in your cellview, you might want to renumber the instances to make the numbers consecutive.

Note: Renumbering instances also works with iterated instances.

To renumber instances,

1. From the schematic window, choose *Edit – Renumber Instances*.

The Renumber Instances form appears.



Only the instance names that match with the *instNamePrefix* property are renumbered.

Mapping of the former and new instance names appears in the (CIW).

2. In the *Sequence instances by* cyclic field, choose how you want the instances renumbered.
3. In the *Start resulting sequence with index* field, type the number you want the renumbering to start with.

Note: The field entry limit here is $(2^{31} - 1)$.

4. In the *Scope* cyclic field, do one of the following:
 - ☐ Choose *cellview* to renumber the instances only in the current cellview.
 - ☐ Choose *hierarchy* to renumber all instances within the hierarchy.
 - ☐ Turn on *top* to renumber instances from the top-level cellview.

- ☐ Turn on *current cellview* to renumber instances in all cellviews below and including the current cellview.
 - ☐ Choose *library* to renumber all instances within the library and type the name of the library containing the instances you want to renumber.
- 5. In the *Apply to all instances of* cyclic field, do one of the following:
 - ☐ Choose *any master* to renumber all the instances of the schematic.
 - ☐ Choose *same master* to renumber all the instances of the particular schematic specified by the following fields:
 - ☐ In the *Library Name* field, type the library containing the instances to be renumbered.
 - ☐ In the *Cell Name* field, type the cell containing the instances to be renumbered.
 - ☐ In the *View Name* field, type the view containing the instances to be renumbered.
- 6. Click *OK*.

How Filling the Void Works When Renumbering Instances

When you invoke the renumber instances command (*Edit – Renumber Instances*), the schematic editing tool processes all the scalar instances. For each instance in the list, starting with the instance that has the highest index, each instance is renamed to use the lowest available index. This is known as *filling the voids*, and is an option in *Sequence instances by* field in the Renumber Instances form.

When performing this task, the following rules are taken into account:

- All scalar instances are considered.
- All user named instances, whose names are not in the same format as the system generated instance names, are ignored.

The following examples explain how the *filling the voids* algorithm works:

Example 1

Consider a schematic with instances I0, I3, and I5, where the master of the instantiated symbols have the prefix “I”, you would get the following output:

Renumber Output: I0, I2, I1

Details: The instance I5 is renamed as I1, and instance I3 is renamed as I2.

Example 2

Consider a schematic, with instances from different master symbols, which have the prefix names as “A”, “B”, “C” and “I”. With this, there are instances with names A1, A10, B2, B5, B9, C20, C50, I13, and I100.

Renumber Output: A1, A0, B2, B1, B0, C1, C0, I1, I0

Details: The instance A10 is renamed A0, the instances B9 and B5 are renamed B0 and B1 respectively, the instances C50 and C20 are renamed C0 and C1 respectively, and I100 and I13 are renamed as I0 and I1 respectively.

Example 3

Consider a symbol with the prefix “A”. Also suppose that you have a schematic with instances A20, A30, A40, and A50, which are instantiated from that symbol. After the creation of the schematic, assume now that you update the prefix for the “A” symbol to be “Y”.

Renumber Output: A20, A30, A40, and A50.

Details: Even with the symbol update, there is no change to the instances.

Magnifying and Rotating Instances Using the symbol_xform View

To magnify or rotate instances on the schematic using the `symbol_xform` view, do the following:

1. Place the `symbol_xform` view of the instance on the schematic.
2. Select the instances that you want to magnify or rotate.

- ☐ To magnify the selected instances, type the following in the CIW and press `Enter`:

```
car (geGetSelSet ()) ~>mag=2.0
```

Where `mag=2.0` indicates that the selected instances will be displayed at twice their default size on the schematic.

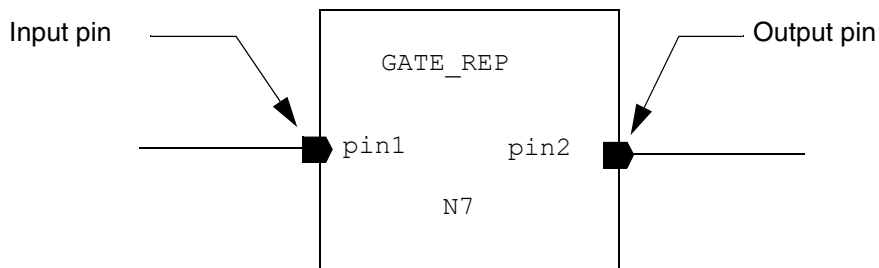
- ☐ To rotate the selected instances, type the following in the CIW and press `Enter`:

```
car (geGetSelSet ()) ~>angle=45
```

Where `angle=45` indicates that the selected instances will be displayed at an angle of 45 degrees on the schematic.

Adding Blocks

You can use blocks when you design with a top-down methodology. Blocks let you create a symbol from the context of a higher-level schematic.



You create a block to represent a lower-level cell. As you develop the logic of your higher-level schematic, you can add wires to the block, which automatically creates block pins. After you name the wires, you can rename the block pins so they inherit the connected signal names. You can automatically generate other views (for example, a schematic, VHDL, or Verilog[®] representation) for the lower-level block cell.



When you are creating a block (or the components of a block) that will be shared, distributed, or used elsewhere, Cadence recommends defining all inherited connections as explicit pins rather than as implicit terminals.

- [Adding a Lower-Level Cell Block](#)
- [Adding Wires and Pins to Blocks](#)
- [Changing the Block Pin Direction Rule](#)
- [Changing the Direction or Name of a Single Block Pin](#)
- [Changing the Names and Directions of Multiple Block Pins](#)
- [Deleting Block Pins](#)
- [Disabling Automatic Block Pin Creation](#)

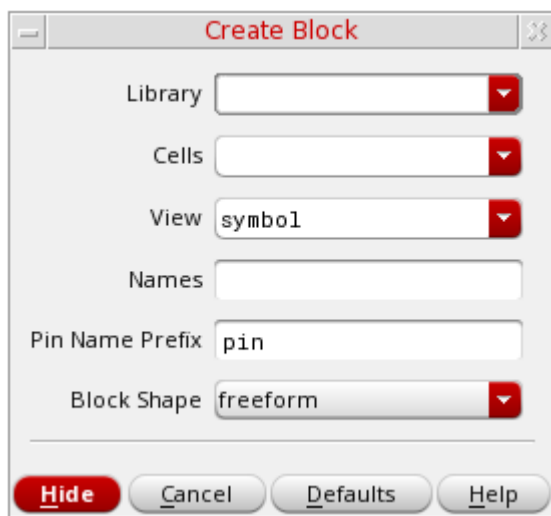
Adding a Lower-Level Cell Block

Unlike adding multiple copies of a single instance in a schematic based on a single *lib/cell/view* name, the *Create – Block* command lets you create multiple blocks, each with a different *lib/cell/view* name.

To create a block:

1. From the schematic window, choose *Create – Block*.

The Create Block form appears.



2. In the Library, Cells, and View fields, specify the path to the block cellview.

The specified cellview cannot already exist in the library. Only one name (instance) can be entered for each cell. There is a one-to-one correspondence between the entries in the *Cells* and the *Names* fields. You can specify multiple cell names by using a space to separate each name. The *Cells* and *Names* fields must match in number. If there are more entries in *Cells* than in *Names*, the schematic editor assigns a name for each. If there are more entries in *Names* than in *Cells*, the extra names are ignored.

3. In the *Names* field, type instance names using a space as a separator.

The schematic editor generates unique names for you if you leave the field blank.

4. In the *Pin Name Prefix* field, type a new prefix name for the pin or leave the default as *pin*.

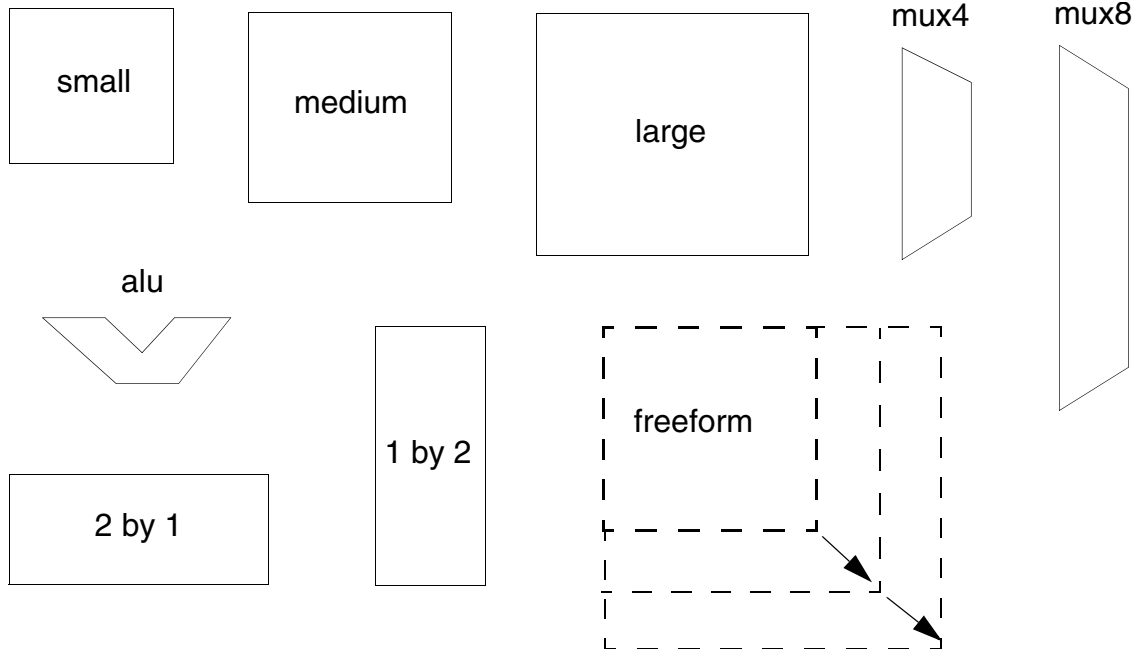
Later, when you add block pins, the system adds numbers sequentially to the prefix name; for example, *pin1*, *pin2*, *pin3*, and so forth.

5. Choose a block from the *Block Shape* pull-down.

Virtuoso Schematic Editor User Guide

Creating Schematics

The *freeform* option lets you create a rectangular block of any size by entering opposite corners of the rectangle.



6. Move the pointer into your design.

The block symbol is attached to the pointer unless you chose *freeform*. You can toggle the block shape before you place it by pressing the right mouse button until you reach the shape you want.

7. Click to place the block in your design.

As you place a block, the system creates the specified block cellview and an instance of that block appears in your schematic.

Adding Wires and Pins to Blocks



Caution

Do not create a block pin by using the Create – Pin command.

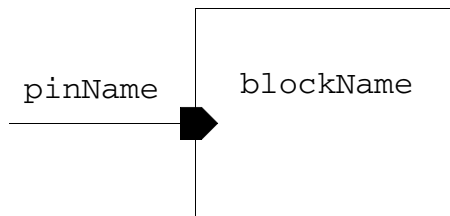
For general information, see [Adding Narrow or Wide Wires](#).

When you add a wire that touches the edge of a block, the schematic editor automatically creates a block pin attached to the wire.

To add wires to a block,

1. From the schematic window, choose *Create – Wire*.
2. Click to indicate wire segment endpoints.

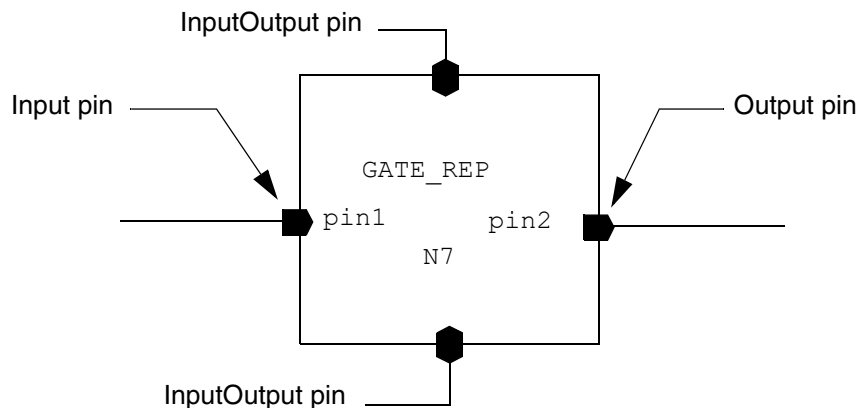
You can either start a wire or end a wire along the edge of a block. The system creates a pin where the wire touches the block edge. The pin name is constructed from the pin name prefix and a unique sequence number. The pin direction depends on the block pin direction rule.



Changing the Block Pin Direction Rule

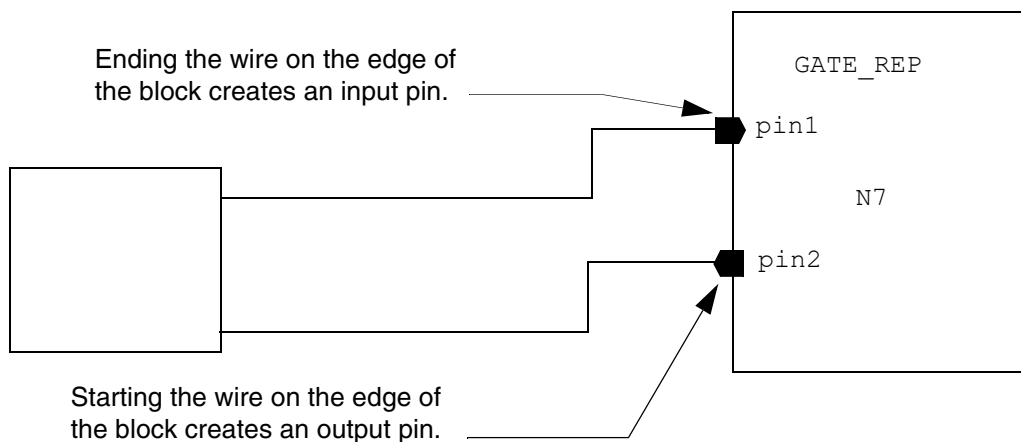
When the block pin direction rule is set to *edge*,

- Any pin created on the left side of the block becomes an input pin
- Any pin created on the right side of the block becomes an output pin
- Any pin created on the top or bottom side of the block becomes an inputOutput pin



When the block pin direction rule is set to *order of entry*, one of the following applies:

- Ending the wire on the edge of the block creates an input pin
- Starting the wire on the edge of the block creates an output pin



To set the block pin direction rule for new block pins to *edge* or *order of entry*,

1. From the view, choose *Options – Editor*.

Virtuoso Schematic Editor User Guide

Creating Schematics

The Editor Options form appears.

The screenshot shows the 'Editor Options' dialog box with the 'Commands' tab selected. The dialog is divided into two main sections: 'Command Controls' and 'Constraint Notes Controls'. The 'Command Controls' section contains various settings with radio buttons and text input fields. The 'Constraint Notes Controls' section contains settings with radio buttons. At the bottom, there are buttons for 'OK', 'Cancel', 'Defaults', 'Apply', and 'Help'.

Option	Value
Add Instance Browser Type	library
Show Schematic Views in Browser	off
Block Pin Direction Rule	edge
Wire Snapping	on
Automatic Solder Dots	on
Solder Dot Wire Overlap	0.03125
Symbol Generator Pin Sorting	geometric
Add Symbol Pins as Instances	off
Repeat Commands	on
Cross Selection	off
Gravity	on
Minimum Marker Size	0.05
View Name List	schematic symbol
Auto Wire Stub for Bus Terminals	wide

Option	Value
Auto Constraint Notes	disable
Sync on Design Open	disable
Sync on Design Check	disable

2. Select the *Block Pin Direction Rule* option you want.
3. Click *OK*.

Changing the Direction or Name of a Single Block Pin

To change the direction or name property of a pin,

1. From the view, select the block pin only.
2. Choose *Edit – Properties – Objects*.

The Edit Object Properties – Instance and Block form appears.

Edit Object Properties

Apply To: only current instance pin

Show: ☒ user

Name: PLUS Direction: off

Direction: inputOutput

Net Name: net030

Add Delete Modify

OK Cancel Apply Defaults Previous Next Help

3. Change the pin's *Name* or *Direction* properties.
4. Click *OK*.

Changing the Names and Directions of Multiple Block Pins

To change the name and direction of multiple block pins,

1. From the view, select the block.
2. Choose *Edit – Properties – Objects*.

The Edit Object Properties – Instance and Block form appears.

3. In the *Show* field, turn on the *system* button. The form expands.

Apply To: only current instance

Show: ☒ system ☒ user ☐ CDF

Browse Reset Instance Labels Display

Property	Value	Display
Library Name	amsPLL	off
Cell Name	ICforce	value
View Name	symbol	off
Instance Name	I5	value
Origin	(-3.125 -2.125)	
FreezePinCreate	☒	off

Pin Direction	Pin Name	Net Name	Inherit
output	p	setIC0!	☐
			All ☐

Add Delete Modify

OK Cancel Apply Defaults Previous Next Help

4. Do one of the following:

- In the *Pin Direction* cyclic fields, change the pin directions.

- ☐ Turn on the *Inherit* button to use the associated net name as the pin name.

5. Click *OK*.

Deleting Block Pins

To delete a block pin after you create it,

1. From the schematic window, select the block.
2. Choose either *Edit – Hierarchy – Descend Edit* or *Edit – Hierarchy – Edit In Place*.

The symbol editor opens.

3. Select the pin to be removed.
4. Choose *Edit – Delete*.
5. Save your design.

Disabling Automatic Block Pin Creation

To disable automatic block pin creation,

1. From the schematic window, select a block.
2. Choose *Edit – Properties – Objects*.

The Edit Object Properties – Instance and Block form appears.

3. In the *Show* field, turn on the *system* button.

The form expands.

4. Turn on the *FreezePinCreate* button.
5. Click *OK*.

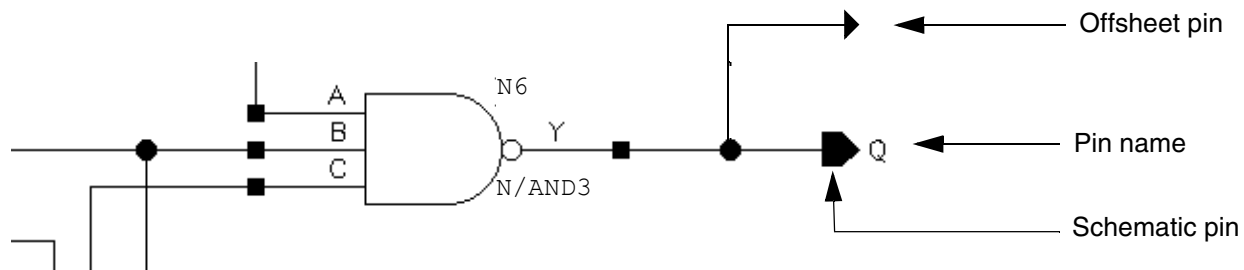
Adding Pins

This section describes the following procedures:

- [Adding Schematic and Offsheet Pins](#)
- [Showing Pin Connections](#)

Pins represent the schematic input, output, and inputOutput terminals to which a net connects in a schematic.

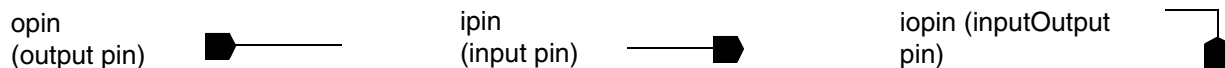
You add pins to establish connectivity between different levels of hierarchy and to connect different sheets of a multisheet schematic. For more information, see [Connectivity and Naming Conventions](#).



Hierarchical pins (schematic pins) connect signals that span different levels of a hierarchical design. Offsheet pins (offsheet connectors) connect signals that span multisheet schematics.

Cadence has a large set of pin definitions in the cell category PINS of the *basic* reference library. These are the pins that the system uses when you choose *Create – Pin*.

The following pins are examples of pins in the *basic* reference library.



Do not use the *Create – Instance* command to place these pins.

Adding Schematic and Offsheet Pins

To add a schematic pin or an offsheet pin,

1. From the schematic window, choose *Create – Pin*.

The Create Pin – Schematic form appears.

The screenshot shows the 'Create Pin' dialog box with the following fields and options:

- Names**: Text input field.
- Direction**: Dropdown menu set to 'input'.
- Usage**: Dropdown menu set to 'schematic'.
- Signal Type**: Dropdown menu set to 'signal'.
- ☐ Expand buses
- ☐ Place multiple pins
- Net Expression** (expanded):
 - ☐ Attach to pin
 - ☐ Derive from pin name
- Property Name**: Text input field.
- Default Net Name**: Text input field.
- Font** section:
 - Height**: Spin box set to 0.06250.
 - Style**: Dropdown menu set to 'stick'.
 - Justification**: Dropdown menu set to 'lowerCenter'.
- Supply Sensitivity** (expanded):
 - Attach To**: Dropdown menu set to 'Both'.
 - Power Terminal**: Dropdown menu set to 'None'.
 - Ground Terminal**: Dropdown menu set to 'None'.
- Rotation**: Four icons for different rotation angles.
- Buttons**: Hide (red), Cancel, Defaults, Help.

2. In the *Pin Names* field, type a pin name or multiple pin names separated by spaces.

Use the same naming conventions for pins as for wires. Use a vector expression to name bus pins.

Note: To create a pin with pwr or gnd sensitivity, the pin name should be a single-bit terminal or single-bit terminal with vector elements. If the name is not a single-bit, a signal type can only be added if the Expand Busses option is also selected which will apply to the array of pins. For more details, see Supply and Ground Sensitivity.

3. (Optional) To add a net expression to a schematic pin,

Note: For more information, see The Syntax of an Inherited Net Expression.

- a. Select the *Attach to pin* check box in the *Net Expression* section.
 - b. In the *Property Name* field, type a property name.
 - c. In the *Default Net Name* field, type the name of a global signal.
 - d. In the *Font* section, specify the height for the label in the *Height* field (see Entering Font Height Settings).
 - e. Specify the font style for the label using the *Style* drop-down list.
4. Select the options from the *Justification* drop-down list to describe the type of pin.
 5. Select or deselect the *Expand busses* check box.
 6. Select *schematic* for a schematic pin or *offsheets* to connect the pin to another sheet in a multisheet schematic from the *Usage* drop-down list.
 7. Select or deselect *Place multiple pins* check box.
 8. Move the cursor into the schematic.

The net expression label is attached to the cursor.
 9. Click to place the label.

The default net expression displays an asterisk (*) after the net expression label to distinguish it from a regular wire name.

Entering Font Height Settings

The *Height* label field is set in user units (UU), with a default setting of 0.0625.

The lowest value that a label height can be set at is the inverse of the `DBUPerUU` (database units per user units) value, as defined in a library's properties. As the `DBUPerUU` for a schematic is set at 160, 0.00625 (the inverse of 160) is the lowest value that can be set.

Values entered as label heights are also rounded to the nearest resolution value ($1/\text{DBUPerUU}$). For example, 0.003125 (which is half of 0.00625) will be rounded up to 0.00625, but 0.003124 would be rounded down to 0 (zero).

Defining Signal Types

The signal type (`sigType`) attribute helps in identifying the particular type of net in the design flow. It is important in the areas of design verification, low power, consistency checking for mixed-signal designs, and setting routing constraints. When you define signal type for a pin or wire, the `Signal Type` property is set for that specific pin or net. The default signal type of a net is `signal`. For example, each net has a `Signal Type` property and the property is set to `power` or `ground` to identify the net as a power net or a ground net, respectively. Therefore, while generating the logical Verilog netlist, these nets are treated as supply nets.

You need to define signal types in the [Create Pin – Schematic](#), [Edit Object Properties – Pin](#), [Edit Object Properties – Wire Segment](#), and [Schematic Rules Checks Setup](#).

The following table describes various types of signals:

Signal	The default signal type associated with a net.
Power	The signal type associated with the power nets.
Ground	The signal type associated with the ground nets.
Reset	The signal types used to reset the given conditions or settings in a design.
Clock	The signal type associated with the clock nets in a block.
Analog	The signal type associated with the analog nets in a block.
Scan	The signal type associated with the scan nets in a block.
tieHi	The signal type associated with the tieHi (resolve to 1'b1) nets in a block.
tieLo	The signal type associated with the tieLo (resolve to 1'b0) nets in a block.
tieOff	The signal type associated with either the tieHi or tieLo signals.

Virtuoso Schematic Editor User Guide

Creating Schematics

<code>optical</code>	The signal type associated with a general optical connection. For more details, see Virtuoso Photonics in VSE, ADE, and VLS XL .
<code>singleModeOptical</code>	The signal type associated with a single mode optical connection.
<code>multiModeOptical</code>	The signal type associated with a multi mode optical connection.

Note: Signal type is an attribute of an explicit net. An explicit net is one that is specifically named within a design. For example, net bundles `a, b, c` and `x, z, b` are explicit nets. Net `b` is an implicit net as it is not named directly within the design; it is implied by the existence of the net bundles `a, b, c` and `x, z, b`. The explicit net determines the signal type.

When multiple explicit nets that share an implicit net have conflicting signal types, the implicit net might be associated with a randomly selected signal type. In such a situation, to ensure consistent signal results from connectivity updates, create the implicit net as a tap. For details on creating a tap, see [Tapping Wire Intersections by Name](#).

Related topics

[Supply and Ground Sensitivity](#)

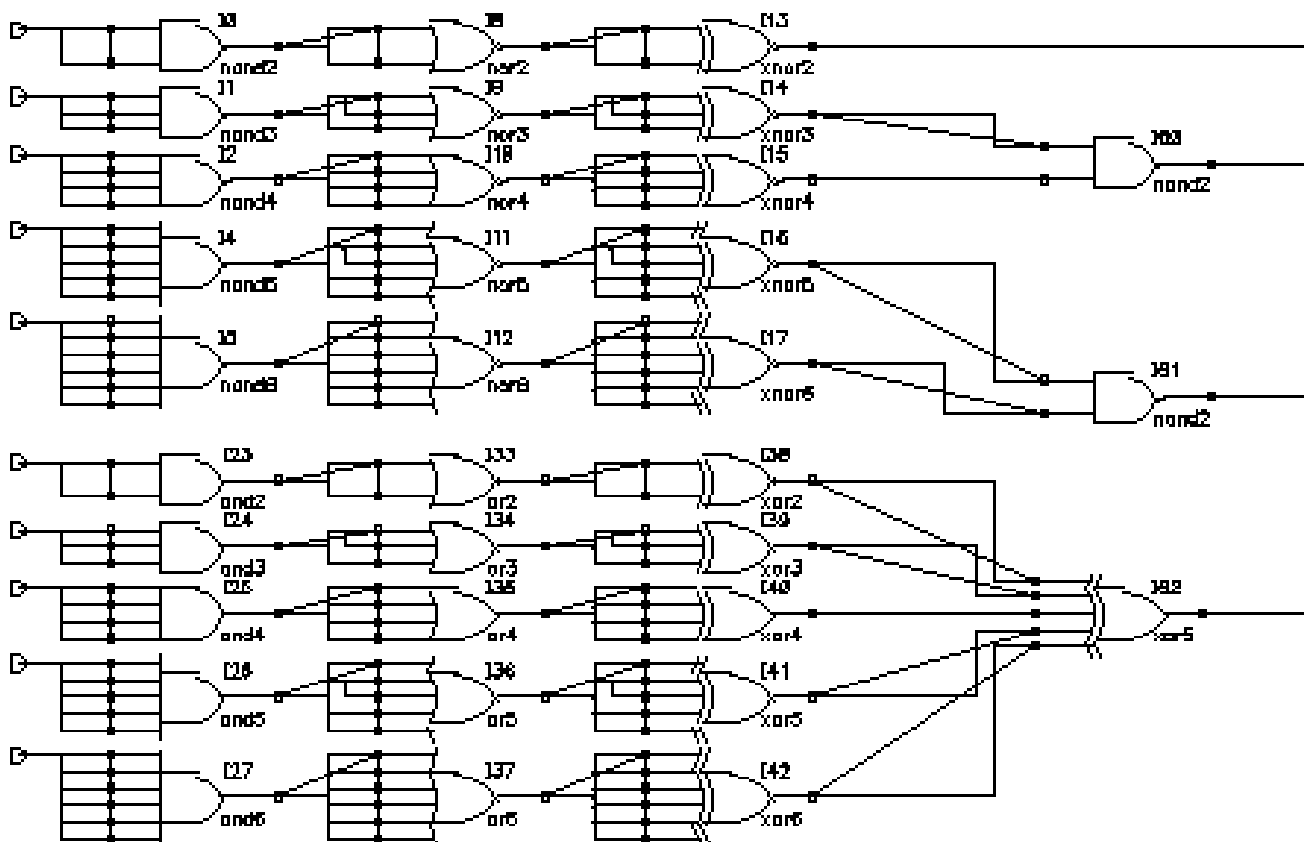
Showing Pin Connections

To enable the display option for highlighting the connection between pins,

1. From the view, choose *Options – Display*.

The Display Options form appears.

2. Turn the *Show Direct Connections* option on.
3. Click *OK*.



Adding Wires

A wire is a line that connects a pin and an instance or a line that intersects another wire. When you draw a wire between a pin and an instance, you create a net.

- [Adding Narrow or Wide Wires](#)
- [Using Wire Snap and Gravity Options](#)
- [Using Wire Draw, Width, and Route Options](#)
- [Using Wire Flight Lines Option](#)
- [Routing Flight Lines](#)
- [Manipulating Wires with Only the Mouse](#)
- [Adding Wires to Schematic Instance Pins Using SKILL](#)

Adding Narrow or Wide Wires

To add wires,

1. From the schematic window, choose *Create– Wire (narrow)* or *Create – Wire (wide)*.

You create buses by naming the wire using a specific notation.

2. Click the wire starting point (for example, at a pin).
3. Click again for each wire segment.

As you move the pointer, a small diamond shape appears over the connection object closest to the pointer. The diamond jumps from one connection object to another as you move the pointer around the design window.

4. To end the wire, do one of the following:

- ☐ Click a schematic pin, an instance pin, or another wire or double-click a new wire endpoint.
- ☐ Press **s** to snap to the nearest object that shows a diamond shape.

Note: Switching the *Wire Snapping* option on from the Editor Options form lets you use the **s** key to quickly complete a wire drawing action. When wire snapping is enabled, the system highlights an unconnected pin in the design when you move the cursor near the pin.



Tip

To create a wire from an instance pin, use the pop-up menu or object direct manipulation.

Note: To terminate wires to pins that have been placed within a selection box, you must be in draw mode (not route mode).

Using Wire Snap and Gravity Options

To turn the wire snap or gravity option on or off,

1. From the schematic window, choose *Options – Editor*.

The Editor Options form appears.

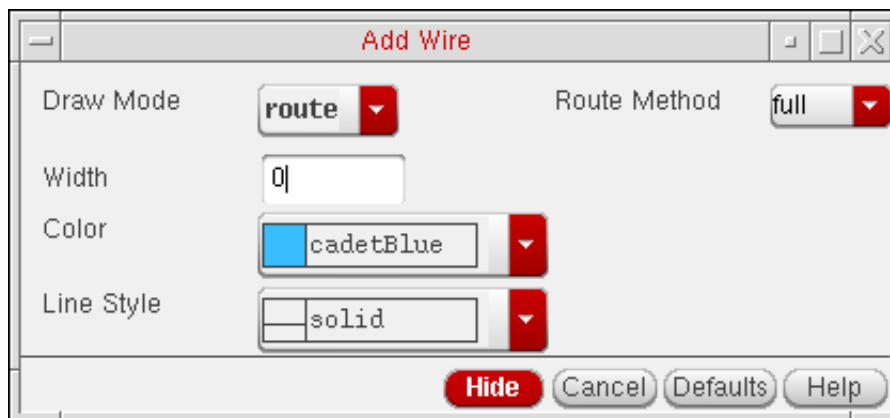
2. Turn the *Wire Snapping* or *Gravity* field on or off.

Using Wire Draw, Width, and Route Options

To change wire draw, width, and route options,

1. From the schematic window, choose *Create – Wire (narrow or wide)*.

The Add Wire form appears. If the form does not appear, press F3.



2. Change the *Draw Mode*, *Width*, *Color*, *Line Style*, and *Route Method* options as follows:

- ☐ The *full* or *direct* routing method cannot complete the connection wire. In this case, use a draw mode other than *route* to manually enter the wire.
- ☐ The selection area boxes around two instances overlap each other and the *full* or *direct* wire connection cannot be made. In this case, move the instances apart to make room for the wire to pass through.

Using Wire Flight Lines Option

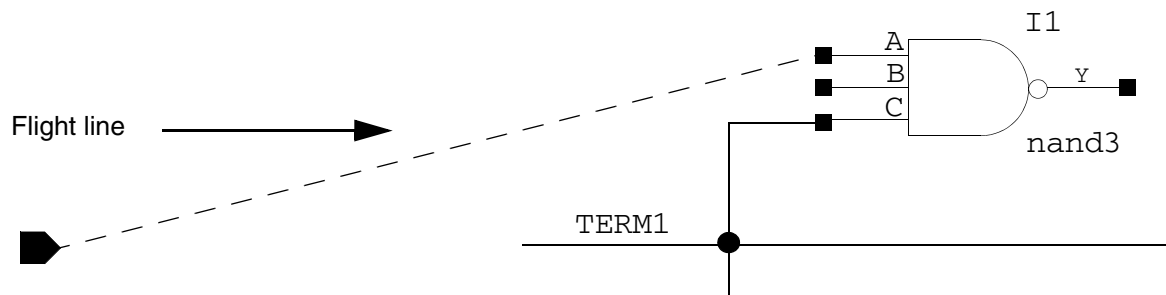
You might find it easier to use flight lines (dotted lines) while you are initially creating a schematic so that you will have more space to add instances. You use a flight line to show an intended logic connection. After all the instances are in place, you can let the schematic editor route the wires for you automatically to establish a physical connection represented by solid wires.

To draw flight lines to indicate an intended connection,

1. From the schematic window, choose *Create – Wire*.

The Add Wire form appears. If the form does not appear, press F3.

2. Set *Draw Mode* to *route*.
3. Set *Route Method* to *flight*.
4. Draw a direct line between two objects.



Routing Flight Lines

To convert wires to normal wires after adding them using the *flight* route method on the Add Wire form,

1. From the schematic window, choose *Edit – Route Flight*.
2. Click a flight line to route.

The schematic editor routes the logical connections.

To change the route method,

1. From the schematic window, choose *Edit – Route Flight*.

The Route Flight Line form appears. If the form does not appear, press F3.

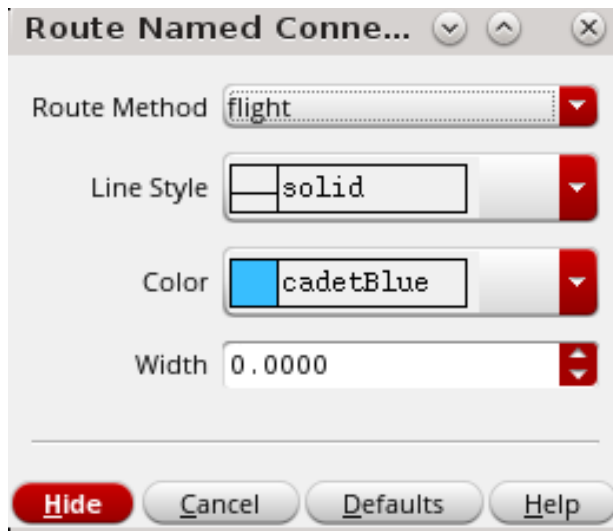


2. Select a route method.
 - ☐ To draw complex lines to complete the connection, choose *full*.
 - ☐ To draw a direct line between two given points, choose *direct*.

Routing Named Connections

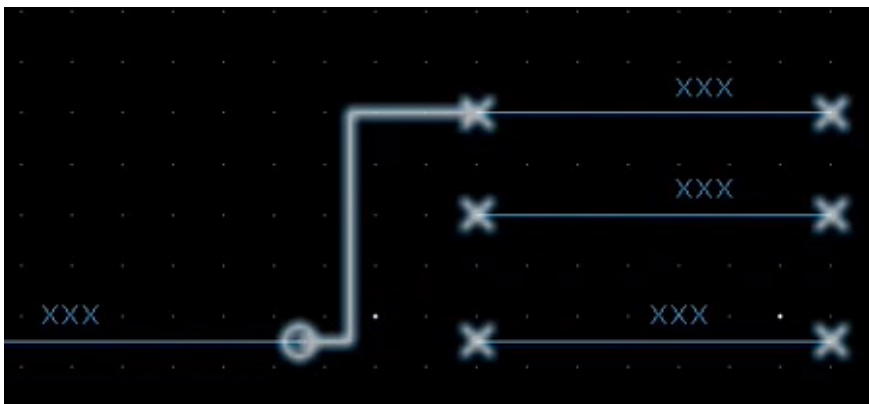
To route physically disjoint wires or groups of wires that use wire-by-name connection,

1. From the schematic window, choose *Edit – Route Named Connection*.
2. Press F3 to display the Route Named Connection form.



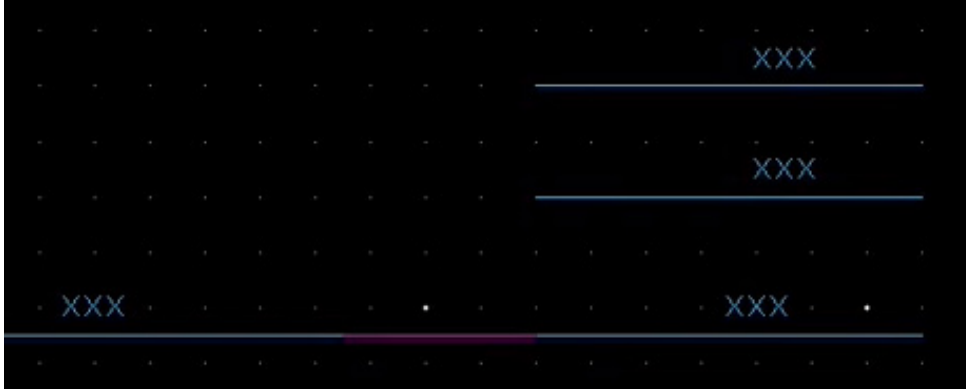
3. Select *flight* as the route method.
4. Hover the mouse over a source point (shown as a circle) to preview the proposed route.

Note: Press TAB to cycle between destination points (shown as an X).



5. Left-click to commit the route.

The schematic editor routes the logical connections.



Manipulating Wires with Only the Mouse

Direct manipulation is the quickest way to create a wire.

To create a wire from a schematic pin, instance pin, or block pin,

1. Press the `ESC` key.
2. Click the pin and drag the cursor away from the pin.

Adding Wires to Schematic Instance Pins Using SKILL

To find the coordinates of the `instTerms` of each instance, query the following `dbObject`:

- With the schematic view open, type the following commands in the CIW:

```
list_of_points=nil
cv=geGetEditCellView()
foreach( inst cv~>instances
    when( inst~>purpose == "cell"
        printf( "      %L\n" inst~>name )
        foreach( instTerm inst~>instTerms
            printf( "          %L\n" instTerm~>name )
            foreach( pin instTerm~>term~>pins
                printf( "              %L" pin~>name )
                if( pin~>fig then
                    bBox = dbTransformBBox( pin~>fig~>bBox inst~>transform )
                    a=xCoord(car(bBox)) + (xCoord(cadr(bBox)) -
                                            xCoord(car(bBox)))/2
                    b=yCoord(car(bBox)) + (yCoord(cadr(bBox)) -
                                            yCoord(car(bBox)))/2
                    list_of_points=cons(a:b list_of_points)
                );if
            );foreach
        );foreach
    );when
);foreach
```

The above code is a sample. Modify it to suit your requirements.

Adding Wire Names

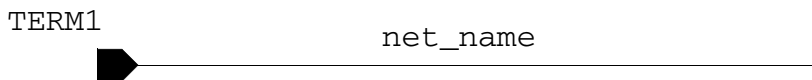
Naming a wire affects the connectivity of your design. You must follow specific naming rules.

- [Naming a Wire](#)
- [Adding Wire Names](#)
- [Adding Multiple Wire Names](#)
- [Registering Default Net Names](#)
- [Verifying Invisible Wire Names](#)
- [Using Net Name Prefixes](#)

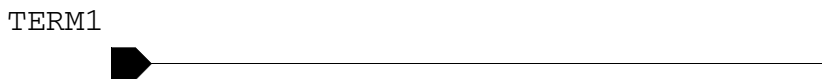
Naming a Wire

The following examples show ways you can name a wire.

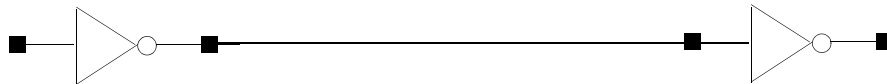
- Name a wire explicitly. Assign a wire name with the *Create – Wire Name* command. The wire name becomes the net name. For VHDL designs, the net name must match the pin name; if not, the schematic rules checker (SRC) flags it as a VHDL violation.



- Let the wire inherit a wire name from the system. When you do not assign a wire name, the net takes the name of the pin; for example, `TERM1`.



- Let the system automatically assign a name to a wire not connected to a schematic pin. You can view the wire name by choosing *Edit – Properties – Objects*.



- Name a wire that consists of multiple signals (a bus) that have the same base name and subscripts.



- Name a wire that consists of a collection of signals with different names. A bundle is represented by names separated by a comma.



Note: The maximum wire name length that is supported by the schematic editor is 32768 characters. Wire names will however be truncated at 100 characters when reported as errors and warnings.

Adding Wire Names

To add wire names:

1. From the schematic window, choose *Create – Wire Name*.

The Create Wire Name form appears.

The screenshot shows the 'Create Wire Name' dialog box. It has a title bar with the text 'Create Wire Name'. Inside, there is a text field labeled 'Names'. Below it are four unchecked checkboxes: 'Expand bus names', 'Attach to multiple wires', 'Create net alias labels', and 'Display bundles vertically'. A section titled 'Font' contains three dropdown menus: 'Height' set to '0.06250', 'Style' set to 'stick', and 'Justification' set to 'lowerCenter'. Another section titled 'Entry Style' has a dropdown menu set to 'fixed offset'. Below this is a table with three columns: 'Position', 'Offset', and 'Rotation'. The 'Horizontal Wires' row has 'above' for Position, '0.04500' for Offset, and 'R0' for Rotation. The 'Vertical Wires' row has 'left' for Position, '0.04500' for Offset, and 'R90' for Rotation. At the bottom left is a 'Rotation' section with two icons. At the bottom right are four buttons: 'Hide' (highlighted in red), 'Cancel', 'Defaults', and 'Help'.

2. In the *Names* field, type the names of the wires.

To establish connectivity, use specific wire naming conventions. To name a bus, use vector expressions.

If you specify multiple names for several wires, separate each name with a space.



Tip

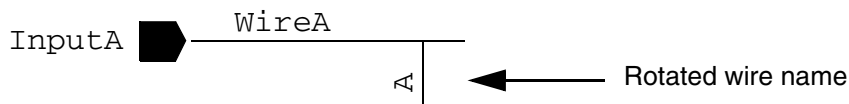
If you enter, for example, `<*12>wireName`, for the net *Name* and select the *Expand bus names* option, you can then place the same label 12 times in different places in the schematic without retyping that label name.

3. (Optional) Change any of the wire name attributes.
4. (Optional) To add a net expression to the wire:
 - a. Open the Create Net Expression.
 - b. In the *Property Name* field, type a property name.
 - c. In the *Default Net Name* field, type the name of a global signal.
5. Move the pointer into your design.

The first wire name is attached to the pointer.

6. Place the name in your design by clicking near or on the wire.

If you name a vertical wire, the name rotates.



The text justification option displays a plus sign (+) in your cellview at the location of the justification.

When a wire name is associated with a wire, any action you perform with the wire affects the wire name. For example, if you stretch the wire, the wire name remains with the wire. If you delete the wire, the wire name is deleted.

If you select the entry style of *fixed offset*, the schematic editor uses predefined default settings to automatically calculate the position of the wire name.

To change the *fixed offset* defaults:

1. In the Create Wire Name form, expand *Entry Style*.

Virtuoso Schematic Editor User Guide

Creating Schematics

The Create Wire Name expands.

The 'Create Wire Name' dialog box is shown with the following settings:

- Names:** (Empty text field)
- ☐ Expand bus names
- ☐ Attach to multiple wires
- ☐ Create net alias labels
- ☐ Display bundles vertically

Font

- Height: 0.06250
- Style: stick
- Justification: lowerCenter

Entry Style

fixed offset

	Position	Offset	Rotation
Horizontal Wires	above	0.04500	R0
Vertical Wires	left	0.04500	R90

Rotation

(Two rotation icons: one showing a wire with a rotation arrow, the other showing a wire with a rotation arrow and a label)

Buttons: Hide, Cancel, Defaults, Help

2. Edit the *Offset* fields.

Adding Multiple Wire Names

To add multiple wire names,

1. From the schematic window, choose *Create – Wire Name*.

The Create Wire Name form appears.

2. In the *Names* field, type the names you want to create; for example, `addr<0:3>` or `addr1 addr2 addr3`.

3. Set the following options as needed:

- ☐ If you want bus expansion of multiple wire names, select the *Expand bus names* option.

This option places multiple-bit bus names as multiple names, one for each bit of the bus range, instead of one name. For example, for the name `A<0:1>`, the label `A<0>` is placed first, then `A<1>`.

- ☐ If you want to place multiple wire names, select the *Attach to multiple wires* option.

This option lets you place wire names as if in single mode but prompts you to bring the rubberbanding line across remaining wires to be named.

4. Move the pointer to the desired location and click once.

If you place the wire name at a greater distance from the wire than the default, click the wire that the label names.

5. Place the remaining names by moving the pointer above each wire to be named and clicking once to place the remaining wire names.

Registering Default Net Names

Note: See also [Adding Wire Stubs](#) and [Registering Terminals for use with Add Wire Stubs](#).

To register default net names, or net expressions, on a terminal basis:

1. Select (for any instance symbol) all of those instance terminals that are to have a common default net name for connection by name, or that are to have a net expression set.

Note: You can select instances/instance terminals from the design canvas or the *Navigator* assistant.

2. Select *Options – Register Net Name per Terminal*.

The [Register Net Name per Terminal](#) form is displayed.

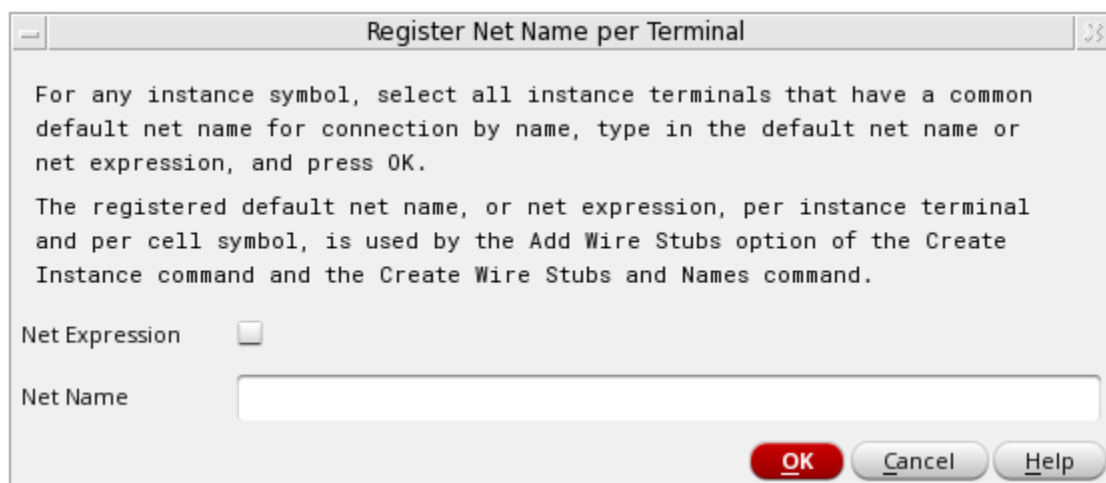


Figure 3-5 Register Net Name per Terminal Form

3. Optionally, check the *Net Expression* checkbox to allow you specify a net expression to define the default global signal name for a connection, and the name of the property that can be used to override that global signal name.

Note: Checking the *Net Expression* checkbox will remove the *Net Name* field and replace it with two additional options: *Property Name* and *Global Net Name*.

- ☐ *Property Name* specifies the name of the property that can be used to override the default global signal.

- ❑ *Global Net Name* specifies the name of the global net to be used to make connections across the design hierarchy (without the need for pins and wires to be used to explicitly pass a signal through the hierarchy).

4. Enter the *Net Name* to set the default net name to be registered to the selected instance terminals.

Note: The *Net Name* option is removed if the *Net Expression* option has been selected.

5. Click *OK*.

The default *Net Name* or *Net Expression* connection settings will now be registered to the selected terminals/instance terminals.

The registered settings can then be used by the *Add Wire Stubs* option in the Add Instance form, and the *Create – Wire Stubs and Names* option.

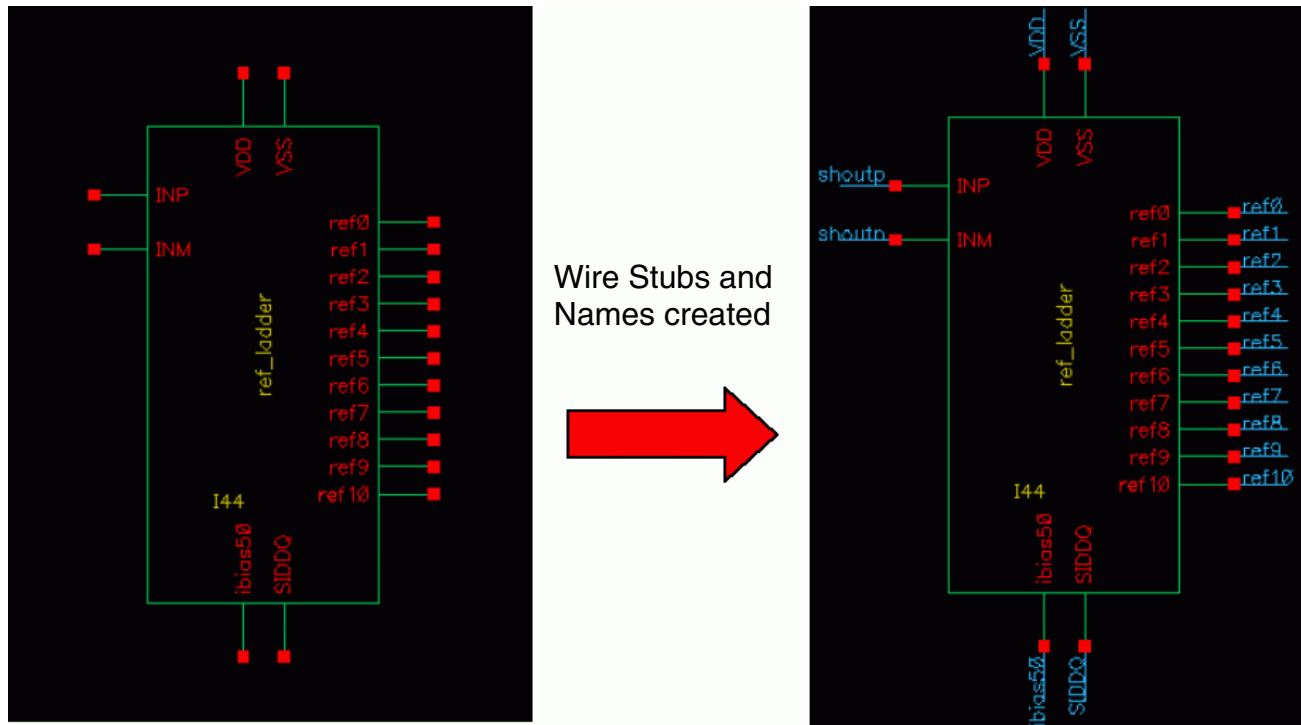
Creating Wire Stubs and Names

You can create wire stubs and names for selected instance terminals or instances using *Create – Wire Stubs and Names*. Here, wire stubs and names are created according to each terminal's existing connectivity.

Note: The usefulness of this option is maximized where instances have multiple terminals.

You can use the following environment variables to customize wire stubs and name labels:

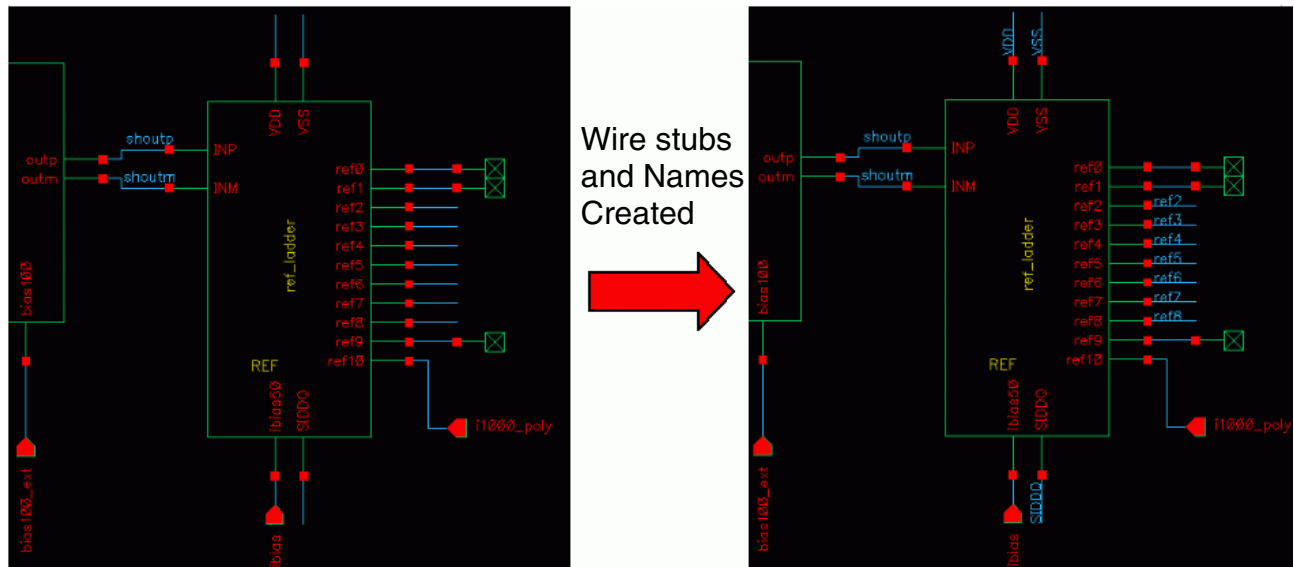
- ☐ createLabelAutoOffset
- ☐ createLabelFontHeight
- ☐ stubLength



- If no wire exists, a wire stub and a wire name are created for the selected instance when *Create – Wire Stubs and Names* is selected.
- If only one wire exists, and it is not connected, a wire name is added.
- Otherwise, no wire name is added.

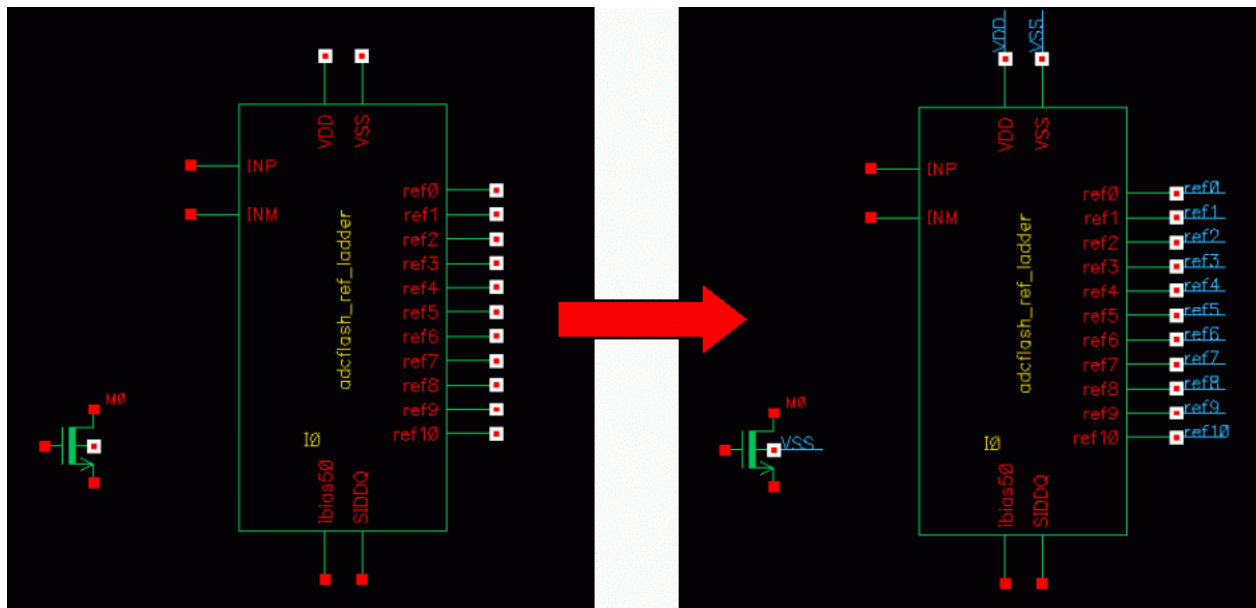
Virtuoso Schematic Editor User Guide

Creating Schematics



- When one or more instance terminals are selected when using *Create – Wire Stubs and Names*, the wire stubs and wire names are only created for those selected terminals.

Note: If a net name is registered for the selected terminal of the master cell, the registered wire name is used.



Verifying Invisible Wire Names

To see a system-generated wire name or net name,

1. From the schematic window, choose *Check – Current Cellview*.

The schematic editor assigns a name to all unlabeled wires.

2. Select the wire.
3. Choose *Edit – Properties – Objects*.

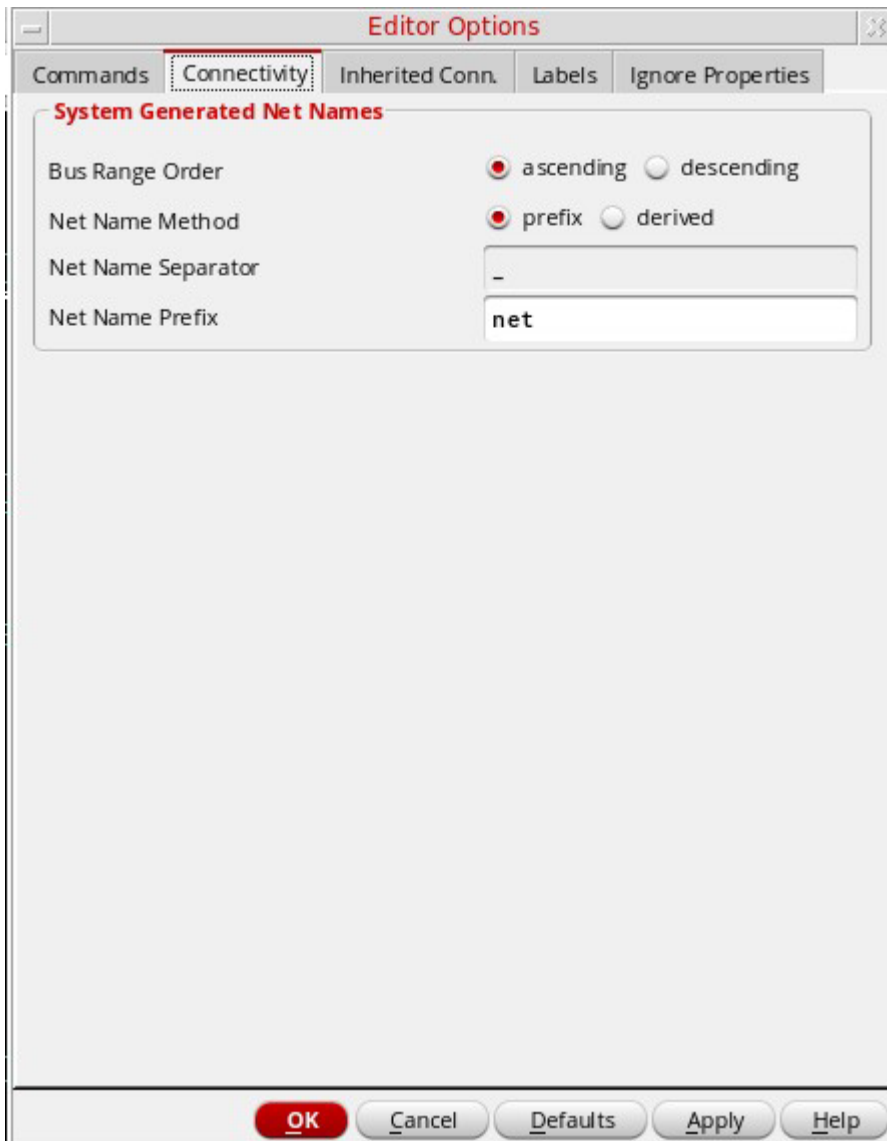
The Edit Object Properties form appears and displays the wire name.

Using Net Name Prefixes

To change the default net name prefix,

1. From the schematic window, choose *Options – Editor*.

The Editor Options form appears.



2. In the *Net Name Prefix* field, type a name of your choice.
3. Click *OK*.

The change affects all new wires you create.

Automatic Justification of Wire Label Names

Automatic label justification is used to instruct the schematic editor to calculate the text justification, and exact position, required for smart placement of labels relative to the parent wires.

Note: The *automatic* label justification option (see below) is not a valid label justification term. Rather, if set, *automatic* chooses an option (*lowerCenter*, *upperLeft*, and so on) that “best-fits” the current placement circumstances. This is reflected when viewing a label’s properties.

Activating Label Justification

Automatic (smart) label justification functionality can be activated using one of the following methods:

1. In the Create Wire Name form (*Create – Wire Name*) ensure that *automatic* is selected in the *Justification* drop-down list before placing the wire name.

If automatic label justification is set to *on*, clicking the mouse button to place a label will result in the label being “snapped” into its auto-position, rather than it being placed directly under the cursor location.

However, auto-justification of labels is not dynamic. Therefore, if you move a wire the attached labels are not repositioned.

Note: The *automatic* option is available in the *Justification* drop-down list in both the Create Wire Name and Create Net Expression forms.

Or

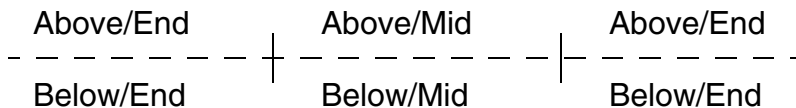
2. Highlight the wire label text and then access the Edit Object Properties form (*Edit – Properties – Objects*). From here, again ensure that the *automatic* option is selected in the *Justification* field.

Adding an *automatic* option to the Edit Object Properties form allows you to automatically justify wire labels that have already been created and placed. For example, you would select the wire label first, access the Edit Object Properties form, and then select the *automatic* option before applying the setting (this can be applied to *current / all selected / all wires* as normal).

Label Positioning

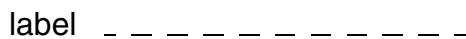
The schematic editor uses the original label placement as a guide as to where it was originally intended to be placed.

For example, consider the following label placements:

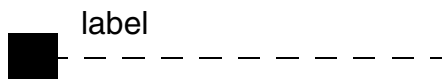


Any label that is initially placed in one of the above regions will automatically be snapped to an appropriate location for that region, for example:

1. A label that is placed near the end of a wire is moved so that it sits on the tip of the wire:

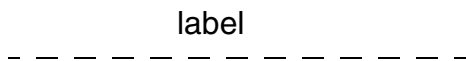


2. If there is a pin, at the chosen wire end, the label will be adjusted to sit on the wire as close to the pin as possible:



Note: Automatic label justification only attempts to avoid pins, it is unable to avoid other obstacles that can exist at the point of label placement.

3. If a label is placed near the middle of a wire, it will be adjusted to ensure that is centralized:

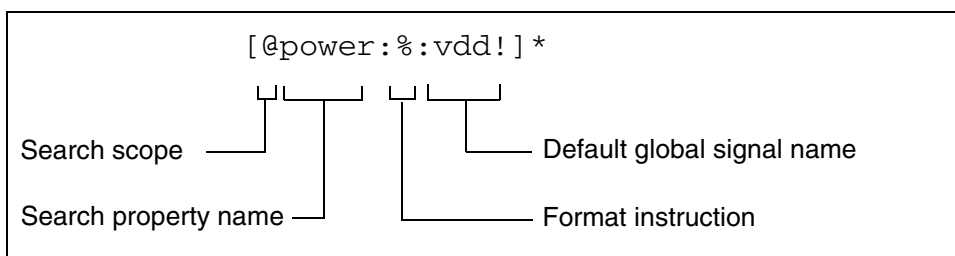


All of the above examples also work in a similar manner for vertical wires. In all cases, apart from (1.), the schematic editor will maintain the label on the same side of the wire (that is, above or below) that it started on. However, labels that are attached to non-horizontal/vertical wires will not be adjusted, and a warning message will be displayed informing you of this.

Adding Net Expression Labels to Create an Inherited Connection

A net expression defines the default global signal name for the connection and the name of the property that can be used to override that global signal name. To create an inherited connection:

- Add a net expression label to a power supply (or wire) in a lower-level schematic, which is described in [Adding a Net Expression Label to a Wire](#).
- Add a `netSet` property to an instance in a higher-level schematic, which is described in [Adding netSet Properties to Create an Inherited Connection](#).



Caution

When you are creating a block (or the components of a block) that will be shared, distributed, or used elsewhere, Cadence recommends defining all inherited connections as explicit pins rather than as implicit terminals.

You place the net expression on a wire segment that represents the signal. The net expression names the signal in the same way that a wire name names a signal. The default name in the net expression serves to name the underlying signal.

Refer to [Inherited Connections](#) for a description of inherited connections.

This section describes the following:

- [Adding a Net Expression Label to a Wire](#)
- [Adding an Inherited Supply Symbol to a Wire](#)
- [Editing a Net Expression Label](#)
- [Changing the Net Expression Label Display](#)
- [Disabling the Automatic Usage of the Default Signal Name](#)

■ Deleting a Net Expression Label

Adding a Net Expression Label to a Wire

To add a net expression label to a wire in a lower-level schematic.

1. From the view, choose *Create – Net Expression*.

The Create Net Expression form appears.

2. In the *Property Name* field, type a property name.
3. In the *Default Net Name* field, type the name of a global signal.
4. Select the appropriate Font options for the net expression.
5. If required, specify the mechanism for placing the net expression label on a wire or pin.
6. Move the cursor into your design.

The net expression label is attached to the cursor.

7. Click to place the label.

8. If the label does not automatically attach to the wire, click the wire you want to add the net expression to.

The default net expression displays an asterisk (*) after the net expression label to distinguish it from a regular wire name.

The net expression takes effect the next time connectivity is updated.

When a net expression label is attached to a wire, the default net name in the expression names the net.

Note: The schematic editor processes expressions associated with symbol pins only when an instance of the inherited symbol is encountered in a schematic.

You can change the net expression label display using the *Options – Display* command. When initially becoming familiar with net expressions, the *value [expression]* option can be very useful.

Related Topics

[Updating Connectivity](#)

Adding an Inherited Supply Symbol to a Wire

Add an inherited supply symbol by placing an instance of an inherited power supply on a wire in a schematic (which is done only once).

Creating a library with inherited power and ground supplies is done only once. In a similar way, regular power supplies can be used instead of wire names to give a global signal name to a net. Inherited power supplies can be used, instead of net expression labels, to assign a net expression to a net.

To place an instance of an inherited power supply on a wire in a schematic,

1. Choose *Create – Instance*.
2. Place an instance of an inherited power supply symbol on a wire.



3.

The inherited supply takes effect the next time you update connectivity.

Related Topics

[Updating Connectivity](#)

Editing a Net Expression Label

To edit a net expression label,

1. In the lower-level schematic, select the net expression label.
2. From the view, choose *Edit – Properties – Objects*.

The Edit Object Properties – Net Expression form appears.

The screenshot shows the 'Edit Object Properties' dialog box. At the top, the title bar reads 'Edit Object Properties'. Below the title bar, there are two dropdown menus: 'Apply To' set to 'only current' and 'net expression'. Below these is a button labeled 'Attachment'. The main section is titled 'Net Expression' and contains four text input fields: 'Property Name' with the value 'Prop1', 'Default Net' with the value 'NetExpr1!', 'Evaluated Name' with the value 'NetExpr1!', and 'Height' with the value '0.0625'. Below these fields are two dropdown menus: 'Font' set to 'stick' and 'Justification' set to 'lowerCenter'. At the bottom of the dialog are several buttons: 'OK' (highlighted in red), 'Cancel', 'Apply', 'Defaults', 'Previous', 'Next', and 'Help'.

Note: The *Evaluated Name* field contains the result of evaluating the net expression for the current descend path of the window.

3. Change the *Property Name*, *Default Net*, *Height*, *Font*, or *Justification* value of the net expression label.
4. Click *Attachment* to reattach the label to the new wire or pin, if required.
5. Click *OK*.

Changing the Net Expression Label Display

To change the net expression label display so that you can see the value, property, or expression,

1. From the view, choose *Options – Display*.

The Display Options form appears.

2. From the *Net Expression Display* cyclic field, choose one of the following:

Option	Display
<i>value only</i>	vdd! *
<i>value (default)</i>	vdd! (vdd!) *
<i>value [property]</i>	vdd! [power] *

Virtuoso Schematic Editor User Guide

Creating Schematics

Option	Display
<i>value [expression]</i>	vdd! [@power:%:vdd!]*
<i>expression only</i>	[@power:%:vdd!]*

The value option implies the resulting signal name from the evaluated net expression for the current hierarchical path. The hierarchical path is the instance path that was descended to get to this level.

3. Click *OK*.

To display the hierarchical path, choose *Edit – Hierarchy – Show Scope*.

Disabling the Automatic Usage of the Default Signal Name

You can disable the automatic usage of the default signal name by setting the `CDS_No_Defaults_For_Inhconn` environment variable. When the environmental variable is set to 1, the software gives an error message whenever an inherited net expression with no explicit override is found. When the environment variable is set to 0, the default signal is used to make the connection.

Deleting a Net Expression Label

To delete a net expression label from a wire or pin,

1. In the lower-level schematic, choose *Edit – Delete*.

2. Select the net expression label.

The net expression label is deleted.

3. Press the `ESC` key to cancel the *Delete* command.

The impact of the deletion takes effect the next time connectivity is updated.

Related Topics

[Updating Connectivity](#)

Adding netSet Properties to Create an Inherited Connection



When you are creating a block (or the components of a block) that will be shared, distributed, or used elsewhere, Cadence recommends defining all inherited connections as explicit pins rather than as implicit terminals.

Adding a `netSet` property to an instance in a higher-level schematic lets you override the default signal name specified in a net expression of a lower-level schematic.

- Adding a netSet Property to an Instance
- Modifying a netSet Property
- Deleting a netSet Property

Refer to Inherited Connections for a description of `netSet` properties used as inherited connections.

Adding a netSet Property to an Instance

To add a netSet property to an instance,

1. At the desired level of hierarchy, select an instance that represents the branch of the design hierarchy to be affected.
2. Choose *Edit – Properties – Objects*.

The Edit Object Properties – Basic form appears.

Edit Object Properties

Apply To: only current instance

Show: ☐ system ☒ user ☒ CDF

Browse Reset Instance Labels Display

Property	Value	Display
Library Name	Two_Stage_Opamp	off
Cell Name	opamp_dc_feedback	off
View Name	symbol	off
Instance Name	I8	off

Add Delete Modify

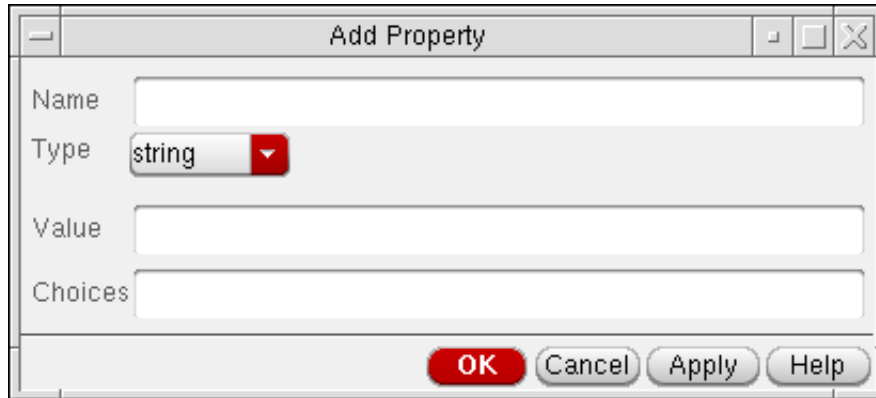
User Property	Master Value	Local Value	Display
interfaceLastCh..	22 04:12:10 2000		off

CDF Parameter	Value	Display
MAX	0.35	off

OK Cancel Apply Defaults Previous Next Help

3. Click *Add*.

The Add Property form appears.



4. In the *Name* field, type the name of the property used in the net expression label.
5. Set the *Type* cyclic field to *netSet*.
6. In the *Value* field, type the name of the signal that is to override the lower-level default signal name.

This will be the new net name. It may be a global or a local signal name, or it may be a net expression. The value must be one of the following:

- ☐ A scalar (one-bit) name that is the name of a net in this schematic
- ☐ Another net expression whose default name is a net in this schematic

7. Click *OK*.

The Edit Object Properties form shows the added `netSet` property.

8. (Optional) For clarity, you could change the *Display* cyclic field from the default *off* to *both* to display both the name and value of the `netSet` property on the instance.
9. Click *OK*.

The new `netSet` property takes effect the next time connectivity is updated.

If the current cellview does not have a signal with the signal name used within the `netSet` property value, a small wire with this name is created automatically when the connectivity is updated.

Related Topics

Updating Connectivity

Virtuoso Schematic Editor User Guide

Creating Schematics

Modifying a netSet Property

To modify a `netSet` property,

1. From the schematic window, choose *Edit – Properties – Objects*.

The Edit Object Properties form appears.

2. Select an instance in the schematic window you want to modify.

You can also change the *Display* cyclic field from the default *off* to *both* to display both the value and name of the `netSet` property on the instance.

3. Click *Modify*.

The Modify Property – Edit Object Properties form appears.

4. Edit the *Type*, *Name*, and *Value* fields for the property.

The name should match the property name in a net expression label within the design hierarchy below this instance.

5. Click *OK*.

For more details, see Modifying Object Properties.

Deleting a netSet Property

To delete a `netSet` property,

1. From the schematic window, choose *Edit – Properties – Objects*.

The Edit Object Property form appears.

2. Select an instance in the schematic window to delete.

3. Click *Delete* in the *User Property* section of the Edit Object Properties form.

For more details, see Deleting Object Properties.

Analyzing Net Expression Property Names and Evaluated Names

A designer who uses inherited connections during the many phases of the design process can take advantage of the analyzing commands. You can find the net expression property names that are available for setting on the selected instances as well as investigate the evaluated names produced by all net expressions below the selected instances.

The analysis commands might be useful for the following tasks:

- Block authoring
- Module instantiation
- Chip assembly
- Debugging

The most helpful task in analyzing net expression property names and evaluated names is searching for net expressions and related names and properties. The Virtuoso® Schematic Editor has various ways to search for a net expression. You can also find net expression and `netSet` overrides in the instance hierarchy and find the actual evaluated results, which can be the default signal when there is no `netSet` override.

- [Finding a Net Expression](#)
- [Finding and Replacing a Net Expression](#)
- [Manipulating Inherited Connections](#)
- [Viewing the Evaluated Net Names](#)
- [Viewing the Full List of all Hierarchical Paths](#)
- [Checking the Net Expression](#)

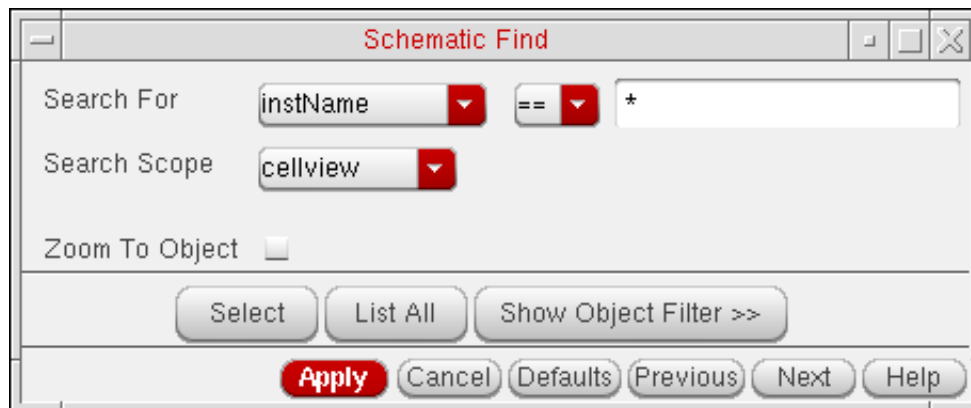
Finding a Net Expression

You can search for a net expression, a `netSet` name, or a `netSet` value. Wildcard searches are also permitted. For more than one match, you can cycle through the match list.

To search for a net expression,

1. From the view, choose *Edit – Find*.

The Schematic Find form (or the Symbol Find form) appears. The fields are the same on both forms.



2. In the *Search For* cyclic field, choose *net expression*, *netSet name*, or *netSet value*.
3. Specify the *Search Scope* from *selected set*, *cellview*, *hierarchy*, or *library*.
4. Optionally, choose to *Zoom To Object*.

Note: If this option is checked you will be asked to then specify the *Zoom Scale* (1-15).

5. Optionally, click the *Select*, *List All*, or *Show Object Filter* options to specify your search requirements.

For more information on these options see the Schematic Find.

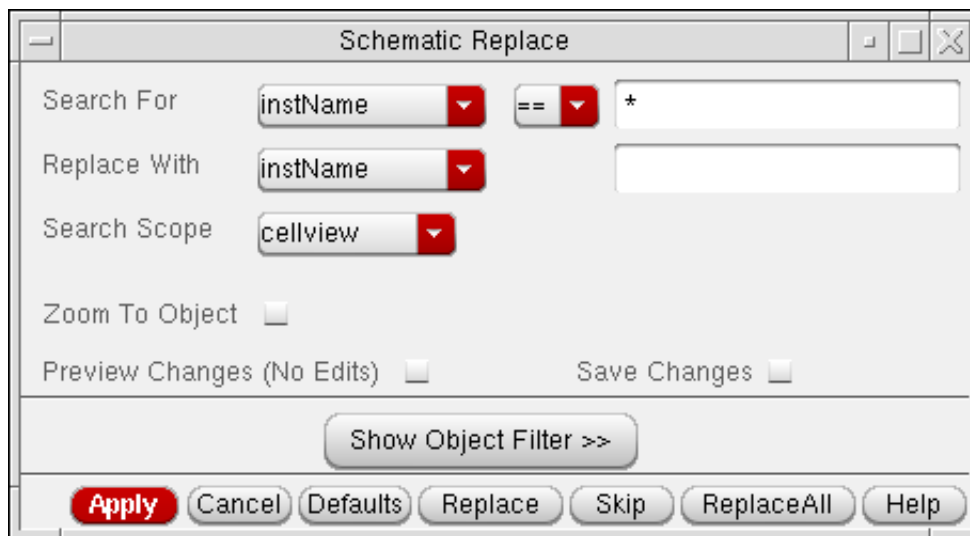
6. Click *Apply*.

Finding and Replacing a Net Expression

You can also use the *Edit – Replace* command to preview the changes and save the changes if you are satisfied with the operation. As with the *Find* command, you can search for a net expression, a `netSet` name, or a `netSet` value. Wildcard searches are also permitted. For more than one match, you can cycle through the match list.

1. From the view, choose *Edit – Replace*.

The Schematic Replace form (or the Symbol Replace form) appears. The fields are the same on both forms.



The screenshot shows the 'Schematic Replace' dialog box. It has a title bar with standard window controls. The main area contains several fields and checkboxes: 'Search For' with a dropdown menu set to 'instName', a comparison operator dropdown set to '==', and a text input field containing '*'; 'Replace With' with a dropdown menu set to 'instName' and an empty text input field; 'Search Scope' with a dropdown menu set to 'cellview'; 'Zoom To Object' with an unchecked checkbox; 'Preview Changes (No Edits)' with an unchecked checkbox; and 'Save Changes' with an unchecked checkbox. Below these fields is a button labeled 'Show Object Filter >>'. At the bottom of the dialog is a row of buttons: 'Apply' (highlighted in red), 'Cancel', 'Defaults', 'Replace', 'Skip', 'ReplaceAll', and 'Help'.

2. In the *Search For* cyclic field, choose *net expression*, *netSet name*, or *netSet value*.
3. Use any of the remaining field options depending on your need (for information on all the replace options that are available see the Schematic Replace), then click *Apply*.

Manipulating Inherited Connections

The Net Expression Available Property Names form allows you to manipulate inherited connections. The upper section of the form displays the instance underneath which an inherited connection was located, the default value for that connection, and a preview of the resolution *Action* currently selected, if any.

1. Select one or more instances from the current design canvas.

Note: If you have not selected an instance the form will prompt you to select one or more.

2. From the schematic window, choose *Edit – Net Expression – Available Properties*.

The Net Expression Available Property Names form appears.

Instance	Connection	Value	Default	Action

Action: Leave as is

Override net name: Freq

Pin name: Direction: Input

Connection name:

Default net name:

OK Cancel Apply Help

3. Select an inherited connection (belonging to an *Instance*) to propagate and enable/disable the appropriate editing fields in the lower section of the form.

4. Select an *Action* to apply to the selected inherited connection.

Particular option fields below the *Action* field will become editable dependent upon the *Action* selected.

Note: For details on the *Actions* that can be selected (and the impact of such selection on the other fields) see the [Net Expression Available Property Names](#) form.

Selecting an action to perform on an inherited connection will only record that action as “pending”. The *Action* column in the table will display a summary of which actions are batched.

5. Where required, enter or select the appropriate values from the remaining form fields.
6. Click *OK* or *Apply* to apply the selected inherited connection action.

Viewing the Evaluated Net Names

To view the net names that result from the evaluation of all net expressions that are found in the hierarchy below the selected instances,

1. Select one or more instances from the current design canvas.

Note: If you have not selected an instance the form will prompt you to select one or more.

2. From the schematic window, choose *Edit – Net Expression – Evaluated Names*.

The Net Expression Evaluated Names form appears.

The form displays all the different evaluated names that result from evaluations of all net expressions in the hierarchy below the selected instance; in this case, the I1 instance.

Evaluated Name	Number of Cellviews	Number of Occurrences

Note: You can alphabetically sort on the column *Evaluated Name* column header.

3. Choose one of the evaluated names.
4. Click *List Cellviews with Selected Evaluated Name* to obtain a comprehensive picture of the inherited connection usage in the design.

The Cellviews with Evaluated Net Expression Name form appears.

This form contains the full list of cellviews that contain net expressions that evaluate to the selected evaluated name. It shows the lookup property name and the default signal for each of these net expressions.

Viewing the Full List of all Hierarchical Paths

To view the full list of all hierarchical paths,

1. From the schematic window, choose *Edit – Net Expression – Evaluated Names*.

The Net Expression Evaluated Names form appears.

2. Choose an evaluated name.
3. Click *List Occurrences for Selected Evaluated Name*.

The Occurrence Paths to Evaluated Name form appears.

This form contains the full list of all hierarchical paths (for example, all lineages) that can be followed down to all net expressions that evaluate to the selected evaluated name.

4. Set the *Display occurrence paths with cyclic* field to *instance names*, *cell names*, or *instance and cell names*, depending on the occurrence path display you want.
5. Review the results for gathering information to debug your design, then click *OK* to close the form.

Checking the Net Expression

To check the net expression for accuracy,

1. Add or change a net expression label or `netSet` property.

The impact of the change takes effect the next time connectivity is updated. The Command Interpreter Window (CIW) shows the results of the check, and a dialog box appears.

Flashing highlighted markers in your schematic indicate the objects that have errors or warnings.

2. On the dialog box, click *Close*.

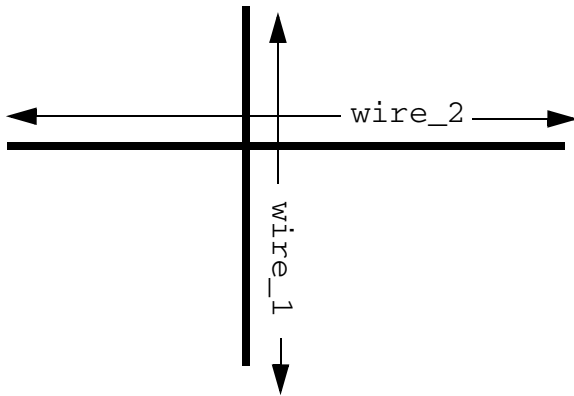
Related Topics

[Checking a Design](#)

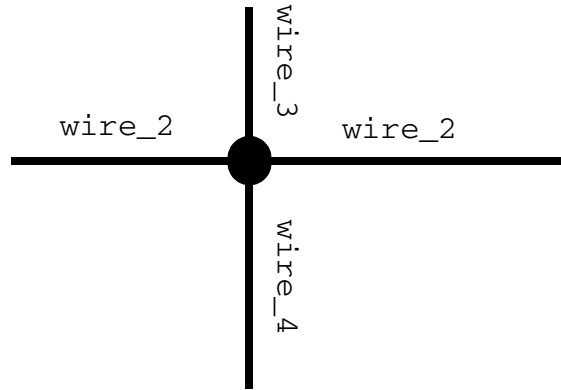
[Updating Connectivity](#)

Adding Solder Dots

Place a solder dot where more than two wire segments intersect. Do not confuse four-wire intersections, which are always connected logically, with two-wire overpasses.



Two overpass wires



Four connected wires

- [Adding Solder Dots](#)
- [Deleting Solder Dots](#)
- [Disabling Automatic Solder Dot Creation](#)
- [Changing Solder Dot Size](#)

Adding Solder Dots

To add a solder dot,

1. From the schematic window, choose *Create – Solder Dot*.
2. Click where you want to create a four-way orthogonal wire connection.

Note: Solder dots are placed automatically when two wire segments form a T-wire intersection.

Deleting Solder Dots

To delete a solder dot from your design,

1. Click the solder dot you want to delete.
2. Press the `Delete` key or choose *Edit – Delete*.

Note: You can delete the solder dot only at a four-way orthogonal wire connection. The connection is changed into a two-wire overpass.

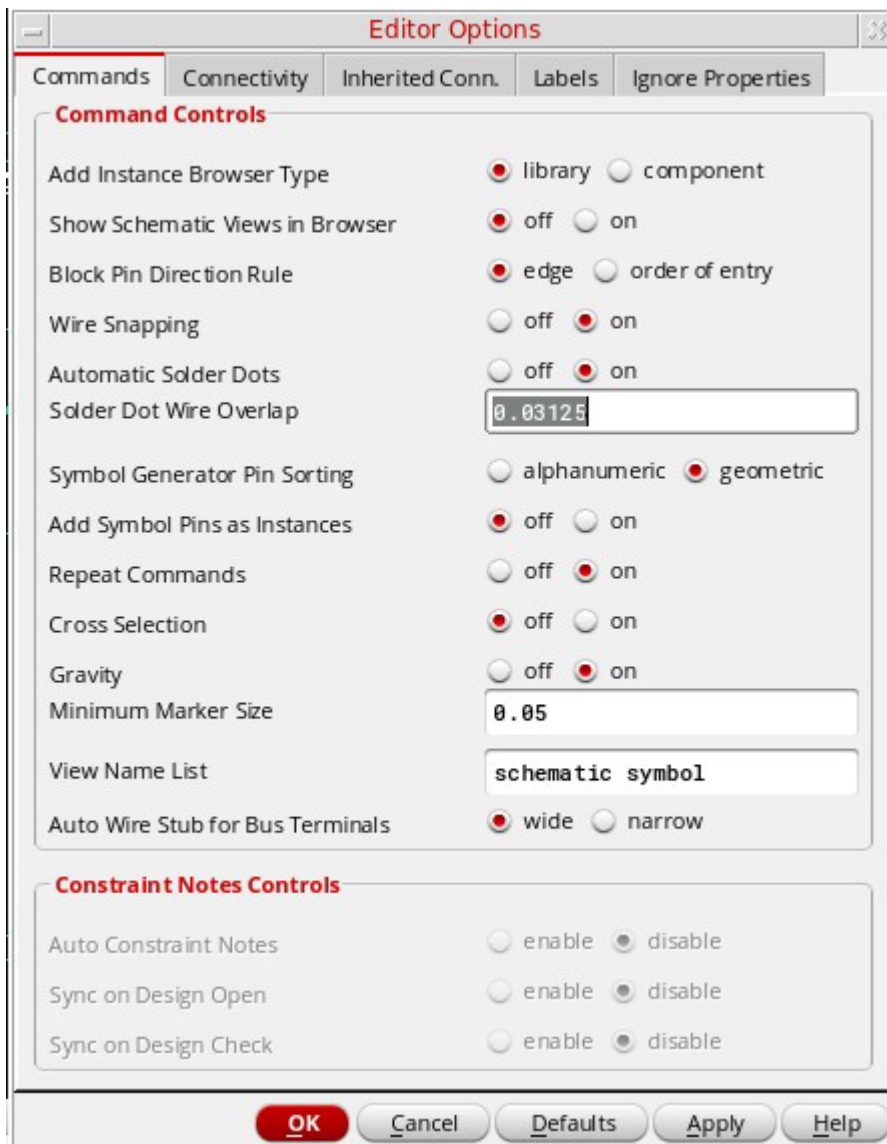
Disabling Automatic Solder Dot Creation

By default, a solder dot is automatically created when you intersect two wire segments.

To stop solder dots being added automatically,

1. From the schematic window, choose *Options – Editor*.

The Editor Options form appears.



2. Set *Automatic Solder Dots* to *off*.

3. Click *OK*.

Changing Solder Dot Size

To change the size of the solder dot,

1. From the schematic window, choose *Options – Editor*.

The Editor Options form appears.

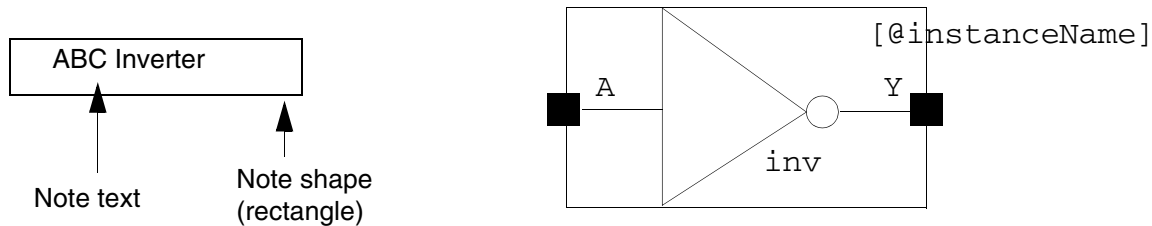
2. In the *Solder Dot Wire Overlap* field, specify a value in user units to define the distance that solder dots overlap wires.
3. Click *OK*.

Adding Notes

To document a schematic, you can use

- Note shapes
- Note text

The following diagram shows an inverter with documentation notes to the left of the inverter.



- Adding Note Shapes
- Adding Note Text
- Importing an ASCII File for Note Text

Adding Note Shapes

To add an annotation within a defined shape on your design:

1. From the view, choose *Create – Note – Shape*.

The Create Note Shape form is displayed.

The screenshot shows the 'Create Note Shape' dialog box with the following settings:

- Shape:** A dropdown menu set to 'line'. Below it, the 'Dashed line' checkbox is checked, and the 'Wide line' checkbox is unchecked. A 'Width' field shows the value '0.06250'.
- Draw Mode:** A dropdown menu showing an arrow icon. Below it, the 'Lock to 45 degrees' checkbox is unchecked.
- Text:** A large empty text area with a vertical scrollbar on the right.
- File:** A text field for the file name, followed by a browse button (three dots) and a save button (floppy disk icon).
- Font:** A section containing 'Height' (2.00000), 'Style' (stick), and 'Justification' (lowerCenter), each with a dropdown menu. Below these is an unchecked 'Overbar' checkbox.
- Buttons:** At the bottom are four buttons: 'Hide' (highlighted in red), 'Cancel', 'Defaults', and 'Help'.

Virtuoso Schematic Editor User Guide

Creating Schematics

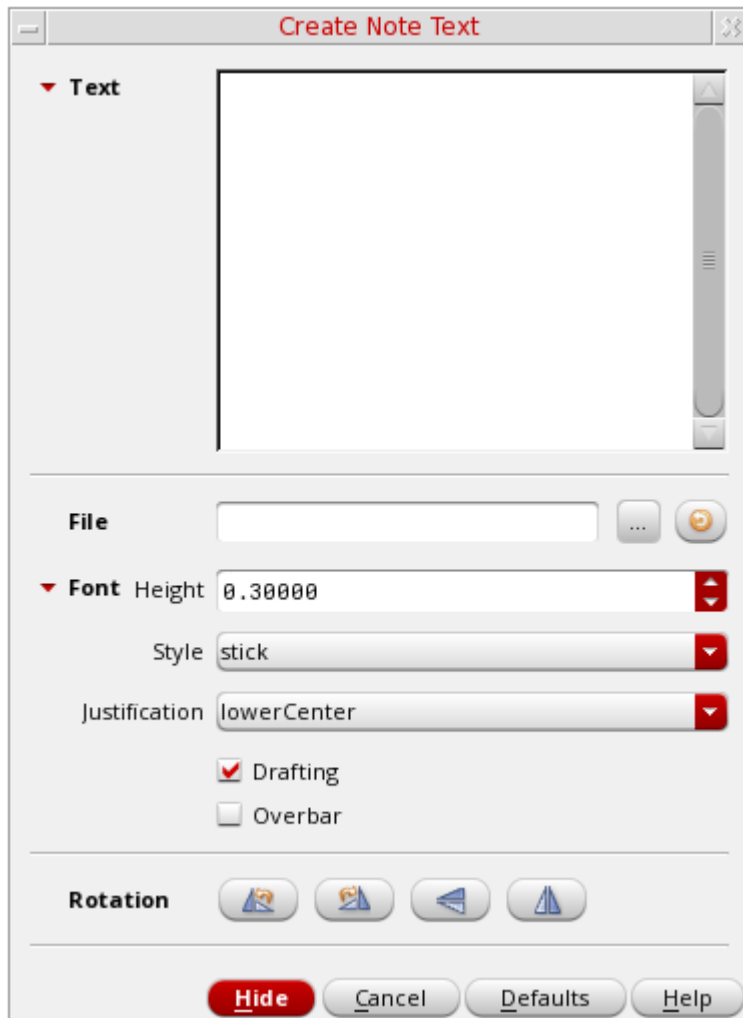
2. Select a shape (line, rectangle, polygon, circle, ellipse, arc).
3. Select the appropriate options for the shape.
4. (Optional) In the *Text* field, type any note text or use the *File Browse (...)* button to display the *Choose a File* form, where you can select the file required.
5. Select the appropriate *Font* options for the shape.
6. Read the prompts at the bottom of your design window for further steps, depending on the shape you are drawing.

Adding Note Text

To add text to annotate your design for documentation purposes:

1. From the view, choose *Create – Note – Text*.

The Create Note Text form is displayed.



2. In the *Text* field, type any note text or use the *File Browse (...)* button to display the *Choose a File* form, where you can select the file required.

Keystrokes are available for editing the text on the form.

Note: See Default Keybindings for Text Fields in the *Virtuoso Design Environment User Guide*

3. Select the appropriate *Font* and *Rotation* options for the note.
4. Place the text in your cellview by moving the pointer into your design and clicking.

Importing an ASCII File for Note Text

To import an ASCII file to document a design,

1. From the view, choose *Create – Note – Text*.

The Create Note Text form appears.

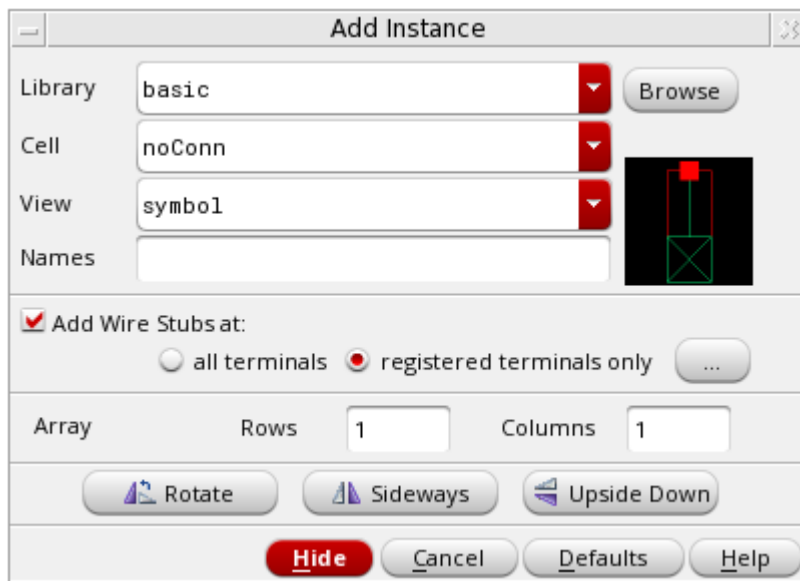
2. In the *Read File* field, type the path to the file.
3. Click the *Read File* button.

Adding Patch Cords Using the Add Instances Form

To add a patch cord to a schematic

1. From the schematic window, choose *Create – Instance*.

The Add Instance form appears.



2. In the *Library* field, type `basic`.

3. In the *Cell* field, type `patch`.

The default connection expression is `0=0`, which indicates that the nets are equal to or greater than 1 bit wide.

If the nets are more than 1 bit wide, you can update the connection expression value on the Edit Object Properties form.

4. Move the cursor into your design and click to add the patch cord.

To add properties to a patch cord symbol:

1. From the schematic window, choose *Edit – Properties – Objects*.

The Edit Object Properties form appears.

2. Click the *Add* button and add the following properties on the form that appears.

```
nlAction = ignore  
pcbAction = ignore
```

Virtuoso Schematic Editor User Guide

Creating Schematics

```
schType = patchCord
```

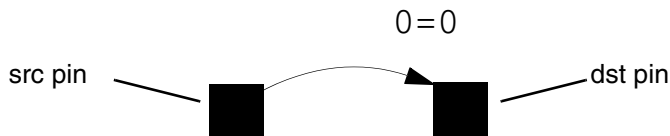
These properties must exist on the patch cord symbol to ensure that the netlister ignores patch cord instances.

3. Make sure that the symbol view of the patch cord has two pins:

- ☐ The source pin is named *src*.
- ☐ The destination pin is named *dst*.

The schematic editor puts $0=0$ in the *Connection Expression* field, which indicates that the nets are equal to or greater than 1 bit wide.

If the nets are more than 1 bit wide, you can update the connection expression value on the Add Instance form.



4. Click *OK*.

Note: The *basic* library contains the component `cds_thru`, which creates a pseudo-short for the situations where a patch does not suffice; for example, if you need a short between two terminals or a terminal and a global signal. The component is arranged to netlist in sensible ways for the following tools:

- `cdl/auCdl`: a small value resistor which can be ignored with `*.RESI`
- `spectre`: an iprobe (effectively a zero-voltage source)
- `verilog`: a module describing the short (terminal-terminal assigns are legal in verilog)

Related Topics

[Patch Cord Connections and Patch Cord Naming Conventions](#)

[Adding Patch Cords Using the Create Patchcord Form](#)

Adding Patch Cords Using the Create Patchcord Form

You can add patch cords using the Create Patchcord form by selecting *Create – Patchcord*.

The *Create – Patchcord* option provides you with a simple method of adding a patch cord symbol to a design, with two net stubs attached. This will allow you to create an alias for existing design nets conveniently.

Note: No checks are performed by the Create Patchcord form to check that the design nets do exist. This process is still performed whenever connectivity is updated.

To add a patch cord:

1. Select *Create – Patchcord* to display the Create Patchcord form.



Note: Your cursor is now ready to place the patch cord dependent upon the form selections you will now make.

2. Complete the Create Patchcord form as required:

- ☐ If enabled, the *Add net stubs* option automatically adds two net stubs when adding a patch cord symbol. When disabled, the Create Patchcord form will only display the *Style* and *style Preview* options, and a simple patch cord instance will be created when placed.
- ☐ *Indexing*
 - ☐ When you only have the *ascending* index option enabled, this allows for a single prefix to be entered, and the alias net will be set up as a bus which depends upon the number of nets entered in the *Design Nets* field. The *preview* option to the right of the *Alias Nets/Prefix* option provides an example of what this bus will appear as.
 - ☐ When the *auto* and *ascending* index options are both enabled, this instructs that the alias bus will display as `alias<0:3>`, assuming that there are four design nets, otherwise it will toggle to display as `alias<3:0>`.

Note: When *auto* is disabled (see image below), you have to fill out both the *Alias Nets/Prefix* and *Design Nets* options to match each other. There will also be no *ascending* or *auto preview* options available.

- ☐ The *Alias Nets/Prefix* option specifies a single string (if *auto* indexing is enabled) that will provide the name to be used for the alias net or bus. If *auto* indexing is disabled, then this can be one or more, comma separated, net names. The number of net names must however match the number of nets entered in the *Design Nets* field.
- ☐ The *Design Nets* option displays a list of comma separated net or bus names, for example `"in, x, cntrl, data<0:3>"`.
- ☐ The *Style* option specifies the format of the patch cord to be placed. You can select between *straight*, which displays a horizontal layout, with both nets and patch cords in a straight line, or *curved*, where the layout is a horseshoe shape, with a vertical patch cord and horizontal nets. The *Preview* field, below the *Style* field, graphically illustrates your current selection.
- ☐ The *Bundle Display* option lets you select the bundle display mode, *horizontal* or *vertical*, for the *Design Nets* if the *Add net stubs* option is enabled and the design net being placed is a bundle.

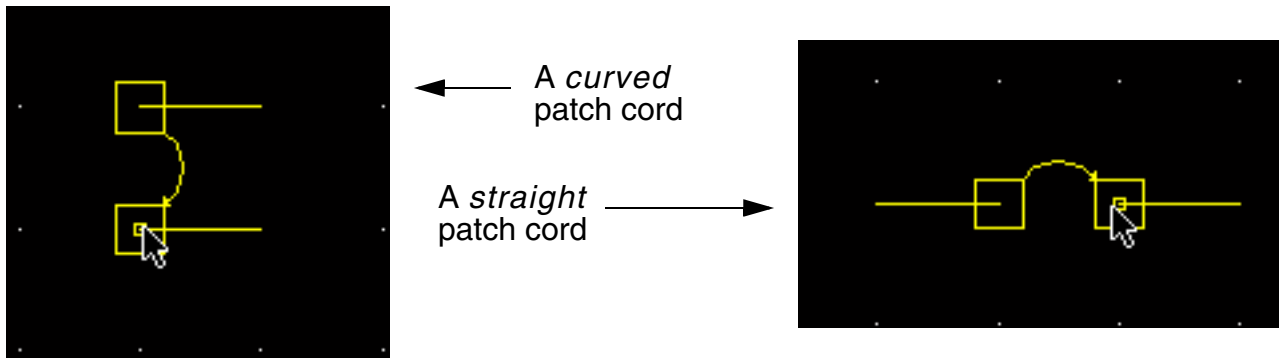
The *Bundle Display* option will be disabled if the *Design Nets* option is empty or if the design net is not a bundle.

Note: A bundle is a wire or pin name consisting of simple names separated by a

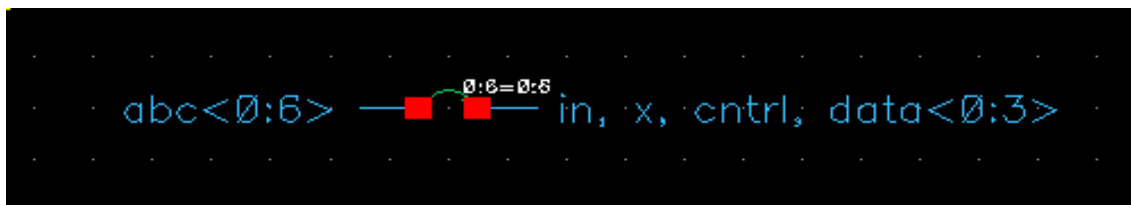
Virtuoso Schematic Editor User Guide

Creating Schematics

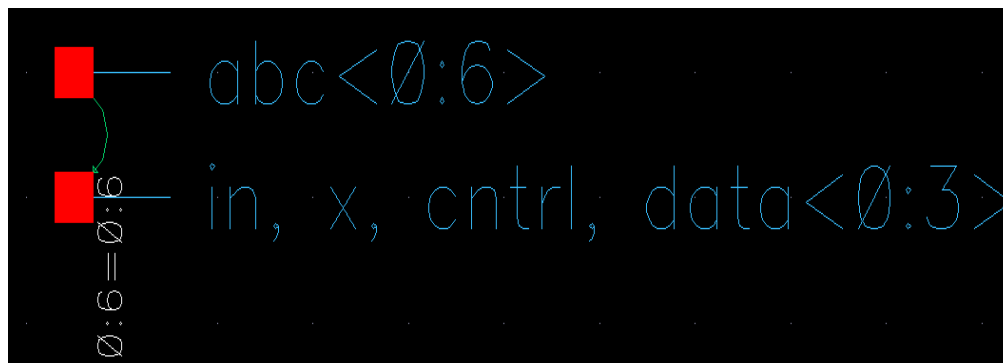
comma; for example, S, R.



3. Once you have completed the Create Patchcord form, point with your cursor at the location on the schematic window where you want to display the patch cord.
4. Click your cursor to place the patch cord.



A straight patch cord has been added to the schematic.



A curved patch cord has been added to the schematic.

Virtuoso Schematic Editor User Guide

Creating Schematics

Related Topics

[Patch Cord Connections and Patch Cord Naming Conventions](#)

Adding Probes

The *Create – Probe – Add* command lets you select the data you want to display in another cellview and select the data for the net for which you want to show a cross-reference.

- [Adding Probes to a Net, Pin, or Instance](#)
- [Showing Probe Information](#)
- [Removing Probes from a Net, Pin, or Instance](#)
- [Removing All Probes](#)
- [Saving Probes](#)
- [Loading Probes](#)
- [Probing a Feedthrough Net](#)
- [Tracing the Hierarchy of a Probed Net Using the Probes Assistant](#)

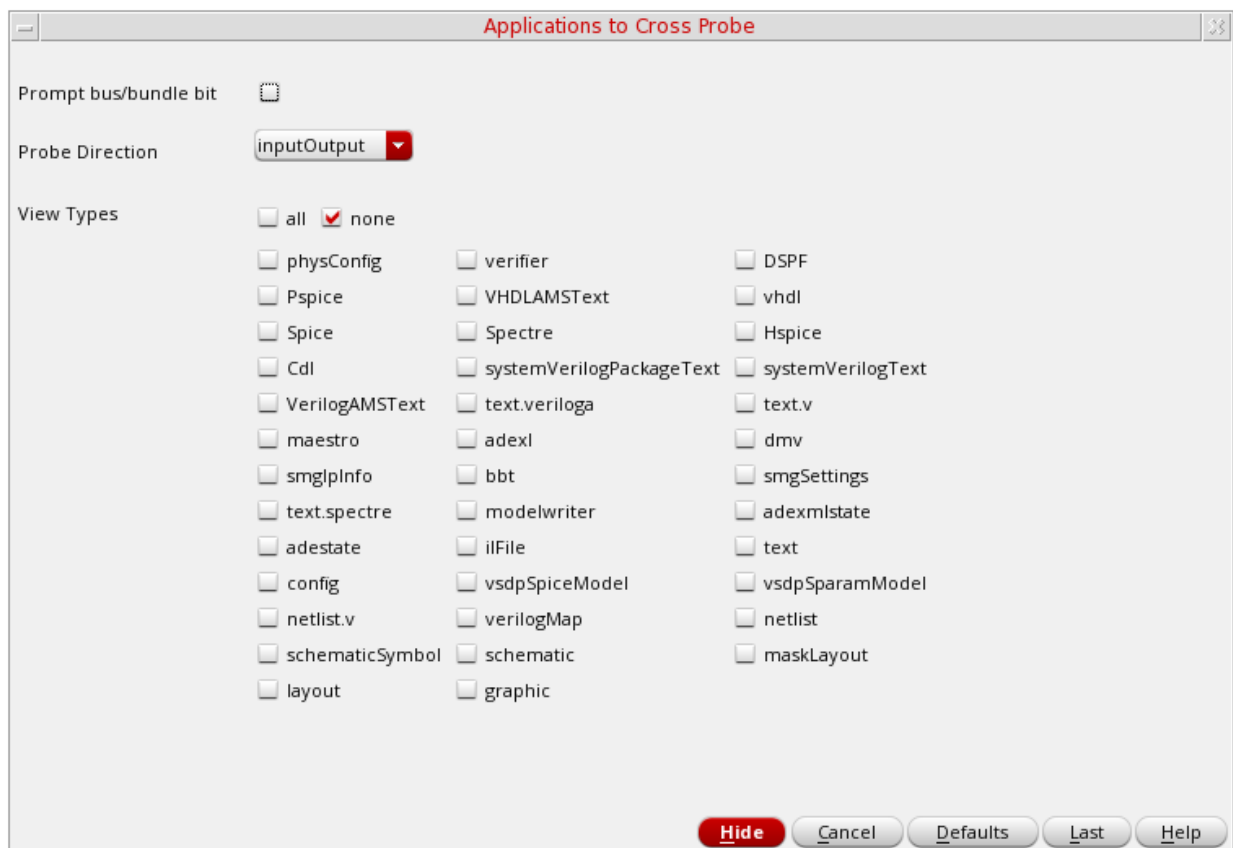
Adding Probes to a Net, Pin, or Instance

To add probes to a net, pin, or instance,

1. From the view, choose *Create – Probe – Add Net*, *Create – Probe – Add Pin*, or *Create – Probe – Add Instance*.
2. Click a net, pin, or instance or type its name.
3. For iterated instance, the Set Instance Number form appears. Enter the instance number to which you want to add the probe.

The Applications to Cross Probe form appears.

Note: To display the Applications to Cross Probe form at any time, when already active, press the F3 key.



The color of the net, pin, or instance changes to show that your probes are added.

Note: For iterated instances, the Set Instance Number form appears, which enables you to enter the instance number to which you want to add instance.

4. To display a prompt that lets you select the specific bit of a bus or bundle to be probed, select *Prompt bus/bundle bit*.
5. Specify the probe direction by selecting the required option in the *Probe Direction* drop-down list.

Note: There are two other ways to specify the probe direction.

- ❑ You can specify the probe direction using `geGetAllProbe`, which returns the ID list of all the existing probes in a window. Therefore, you can modify the probe using these IDs in the `geSetProbeDirection` function.
- ❑ The third method is to use the `hiGetCurrentWindow()` SKILL function and the `probeDirection` environment variable. You can specify the direction as shown below:

```
hiGetCurrentWindow()->probeDirection = "input"
```

Note: If you are working with a large design containing multiple probed nets, it may become difficult to trace the probed net you are working on in the design canvas. To view the probed nets across the design hierarchy, select *Window – Assistants – Probes* to open the Probes assistant.

You can also add probes by typing the net, pin, or instance name as follows:

1. From the view, choose *Create – Probe – Add Net*, *Create – Probe – Add Pin*, or *Create – Probe – Add Instance*.
2. While the cursor is in the schematic window, press `Control-v`.

The Command Interpreter Window (CIW) appears.

3. Type the name of the net, pin, or instance you want to probe.

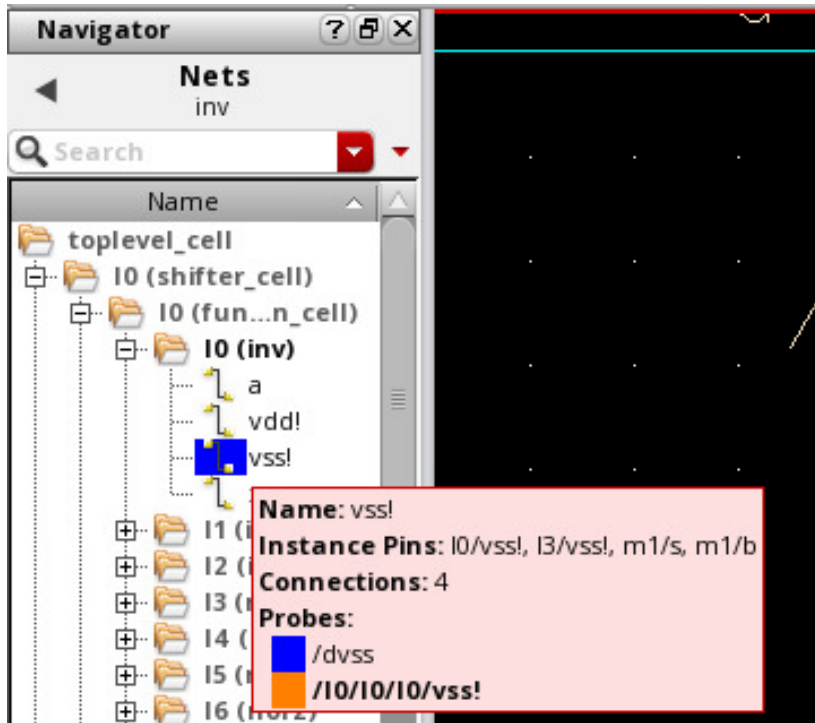
You must enclose the name in quotation marks. For example, to add a probe to a net named `net_20`, type `"net_20"`.

4. Press the `Return` key.

The color of the net, pin, or instance changes to show that your probes are added.

Showing Probe Information

To show an existing probe type and object name, hover over the probe in the Navigator to display a tool tip similar to the one below.



Alternatively, you can display probe information as follows:

1. From the view, choose *Create – Probe – Explain*.
2. Click near an object in your design that has a probe.

A list window is displayed showing the probe type and name of the object. If you clicked near two or more objects with probes, the window lists each object and its probe type.

In more complex, larger designs with multiple probed nets, you can view the schematic hierarchy or the lib/cell/view of a specific probed net using the [Probes assistant](#).

Removing Probes from a Net, Pin, or Instance

The *Create – Probe – Remove ...* commands remove the data you displayed in another cellview or removes the data that cross-referenced a net, pin, or instance.

To remove probes from a selected net, pin, or instance,

1. From the view, choose *Create – Probe – Remove Net*, *Create – Probe – Remove Pin*, or *Create – Probe – Remove Instance*.

The Delete Cross Probes form appears. If the form does not appear, press F3.

If you do not click *Cross Probe*, the cross-probes are left in place.

2. Click a net, pin, or instance in your design that contains a probe.

If you assigned more than one probe to this net, pin, or instance, only one is deleted. The color of the object changes to show that the probe was deleted.

You can also remove probes by typing the object name, as follows:

1. From the view, choose *Create – Probe – Remove Net*, *Create – Probe – Remove Pin*, or *Create – Probe – Remove Instance*.
2. While the cursor is in the schematic window, press `Control-v`.

The CIW appears.

3. Type the name of the net, pin, or instance you want to remove.

You must enclose the name in quotation marks. For example, to remove a probe from a net named `net_20`, type `"net_20"`.

4. Press the `Return` key.

If you assigned more than one probe to this net, pin, or instance, only one is deleted. The color of the object changes to show that the probe was deleted.

Removing All Probes from Nets

A single or all the probed nets in a design can be removed using the Probes assistant. For more details see [Removing a Probe or All Probes](#).

Removing All Probes

To remove all probes, including cross-probes attached to objects in the top-level design of the current window,

- From the view, choose *Create – Probe – Remove All*.

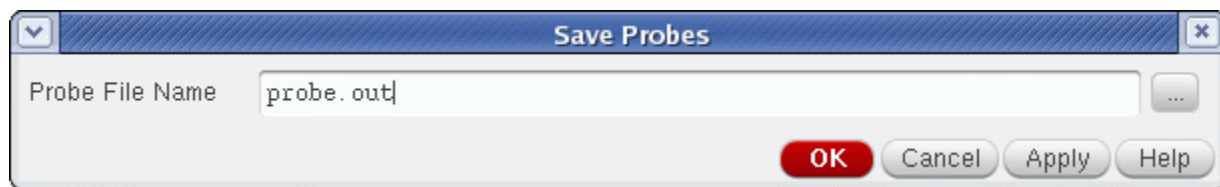
Saving Probes

Probes are not saved with your design when you choose *File – Save* or *File – Save a Copy*. If you want to reattach the probes, you need to save them to a file, then choose *Create – Probe – Load* to reapply them the next time you open your design.

To save the list of currently attached probes to a file,

1. From the view, choose *Create – Probe – Save*.

The Save Probes form appears.



2. In the *Probe File Name* field, type the name of the file in which you want to save the probes.

Note: Alternatively, you can use the *Browse (...)* option to display the Choose a File form, where you can then locate the name of the file you want to save the probes to.

3. Click *OK*.

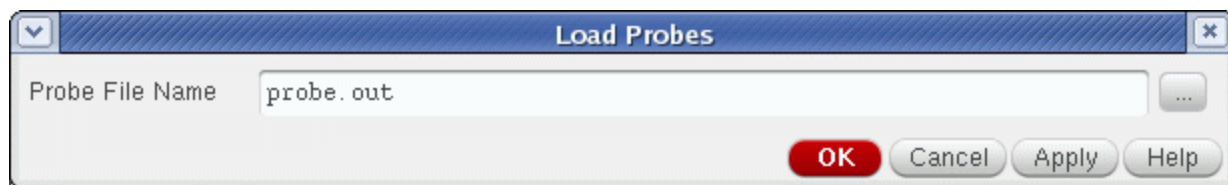
Loading Probes

If you want to reattach the probes, you need to save them to a file, then choose *Create – Probe – Load* to reapply them the next time you open your design.

To load and attach probes from a saved file,

1. From the view, choose *Create – Probe – Load*.

The Load Probes form appears.



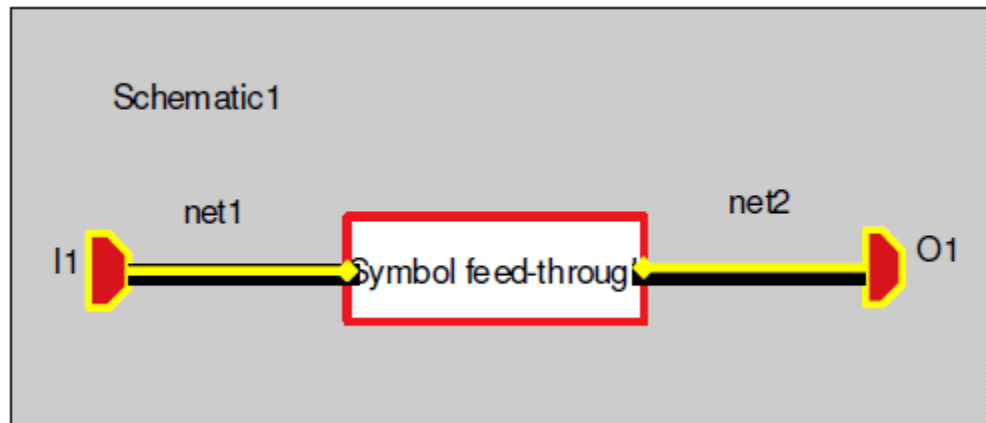
2. In the *Probe File Name* field, type the name of the file containing the probes you want to load.

Note: Alternatively, you can use the *Browse (...)* option to display the Choose a File form, where you can then locate the name of the file that contains the probes that you want to load.

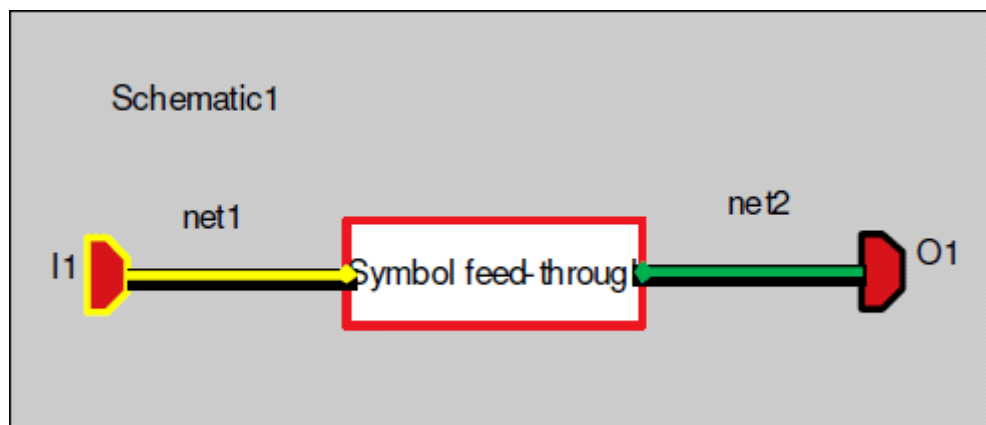
3. Click *OK*.

Probing a Feedthrough Net

A net that passes through a block, which does not contain any logic devices, is a feedthrough net. When you probe a feedthrough net, the probe signal will traverse the path and highlight the pins connected to the net as shown in the figure below. This probing capability helps you in tracing nets in a convenient manner.



Earlier, as soon as a symbol cellview was encountered, the probing would stop as shown in the figure below.



In order to probe a feedthrough net in a design, you need to enable the probeTraverseFeedThrough environment variable:

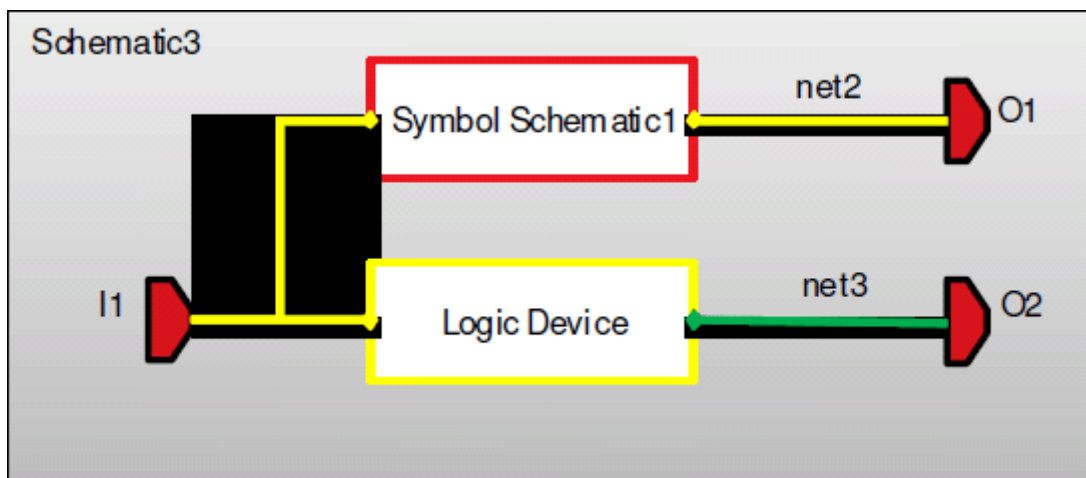
```
envSetVal("graphic" "probeTraverseFeedThrough" 'boolean t)
```

Additionally, some other highlights of this probing capability are as follows:

- Probing Traverses Design Hierarchy
- Conditional Probing

Probing Traverses Design Hierarchy

When you add probes to the design you can also specify the levels of hierarchy up/down for the probing to traverse through by setting the `probeTraverseHierarchy` and `probeNetDisplayHierarchyLevel` environment variables. In the design hierarchy path, the probing will traverse an instance which is a feedthrough or has a feedthrough inside it (refer to the figure shown below). On the other hand, if the instance is neither a feedthrough nor has a feedthrough inside it, the probing will not traverse through and the instance itself will be highlighted (refer to the figure shown below).



In order to probe a feedthrough net across hierarchies, you need to enable the following environment variables in addition to `probeTraverseFeedThrough`:

1. Enable the `probeTraverseHierarchy` environment variable:

```
envSetVal("graphic" "probeTraverseHierarchy" 'boolean t)
```

2. Enable the `probeNetDisplayHierarchyLevel` environment variable which lets you specify the levels of hierarchy in which the probing should be followed. For example, if you want the probing to traverse through 5 levels of hierarchy, you need to set it as shown below:

```
envSetVal("graphic" "probeNetDisplayHierarchyLevel" 'int 5)
```

You can view all the probed objects across the design hierarchy using the [Probes assistant](#). Additional flexibility is also available when tracing a net using the SKILL `schTraceNet` which can be customized to answer more complex user-defined questions across a schematic hierarchical net.

Virtuoso Schematic Editor User Guide

Creating Schematics

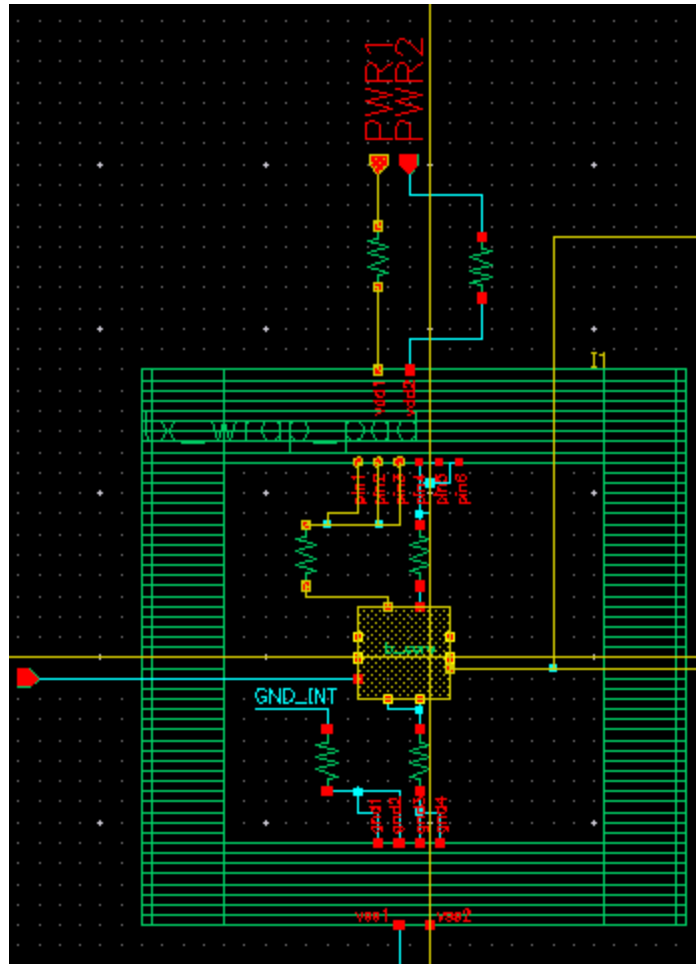
Conditional Probing

You can perform conditional probing by including `shortTermExpr` expression in the `ciRegisterdevice` function. This is a SKILL expression and is evaluated at the time of probing. Only if the expression is evaluated to `t`, the device or its occurrences in the design are considered as a feedthrough.

To register the feedthrough using the `ciRegisterDevice` function, for example, if you want to short `PLUS` and `MINUS` terminals of `res`, you need to enter the following:

```
ciRegisterDevice("probeShort" '( (nil "res" nil (nil
shortedTerminalMap ( ("PLUS" "MINUS") ) ) ) ) )
```

Now on probing, the design will be highlighted as shown in the figure below.



Notice in the figure that all the nets with `res` are highlighted.

Virtuoso Schematic Editor User Guide

Creating Schematics

Note: In the example shown above, `shortedTerminalMap` has been used to specify the list of terminals to be probed. You need to specify at least two terminals in the list. Additionally, you can also use `shortTermExpr` for conditional probing.

In order to do this, you need to use the `schvProbeFeedThrough` global variable in the expression. This variable has the following properties:

- `schvProbeFeedThrough.device`

This refers to the instance ID (in case of first level and below) or the cellview ID (in case of top level). For example, if you want to probe all the `res` in the design with a value less than 10, you need to enter the following:

```
ciRegisterDevice("probeShort" '( (nil "resistor" nil (nil
shortedTerminalMap ( ("PLUS" "MINUS") ) shortTermExpr
"_schvProbeFeedThrough.device->res < 10"))))
```

- `schvProbeFeedThrough.topcv`

This refers to the top cellview. For example, if you want to probe all the devices in `ampTestcellview`, you need to enter the following:

```
ciRegisterDevice("probeShort" '( (nil "resistor" nil (nil
shortedTerminalMap ( ("PLUS" "MINUS") ) shortTermExpr
"_schvProbeFeedThrough.topcv->cellName == \"ampTest\""))))
```

- `schvProbeFeedThrough.hierpath`

This refers to the hierarchical path to the device. For example, if you want to probe all the devices in `ampTestcellview` and hierarchy path is `/I0/I1`, you need to enter the following:

```
ciRegisterDevice("probeShort" '( (nil "resistor" nil (nil
shortedTerminalMap ( ("PLUS" "MINUS") ) shortTermExpr
"_schvProbeFeedThrough.topcv->cellName == \"ampTest\"    &&
index(_schvProbeFeedThrough.hierPath \"/I0/I1\") " ) ) ) )
```

- `schvProbeFeedThrough.map`

This contains `shortedTerminalMap` for the device. It can be modified, filtered, or generated dynamically based on instance parameters and other device information. For example, if you want to register a device in which the terminal map is generated.

```
ciRegisterDevice("probeShort"
' (
  ("library" "switch" nil
```

```
(nil shortedTerminalMap ( ("CONNECT" "out"))

      shortTermExpr "_schvProbeFeedThrough.map =
mySwitchTerminalMap(_schvProbeFeedThrough.device) ")))

procedure( mySwitchTerminalMap(device)

  let(((cdfData cdfGetInstCDF(device))

        connectValue (outputName "out"))

    connectValue = cdfData->connect->value

    when( connectValue && connectValue != "0"

      list(list(connectValue outputName))))))
```

Tracing the Hierarchy of a Probed Net Using the Probes Assistant

If you are working with a large design containing multiple probed nets, it may become difficult to trace the probed net you are working on in the design canvas. Instead, you can view the probed nets across the design hierarchy using the [Probes assistant](#).

1. Select *Window – Assistants – Probes* to open the Probes assistant.
2. In the Navigator, right-click on the required net and select *Probe*. Select a color to highlight the probe.

The probed net is highlighted in the selected color in the Navigator, in the canvas, and in the Probes assistant.

3. Select an object on the Probes assistant to highlight it on the canvas. If a selected object is on a different level of hierarchy it will not be visible at the same time. To highlight such objects in the Navigator and in the canvas, right-click on the object in the Probes assistant and select *Show In Instance*. Use the *Go* toolbar option to navigate as required.
4. Use the filter buttons on the Probes assistant to filter and manage the probes information displayed.

Additional flexibility is also available when tracing a net using the SKILL [schTraceNet](#) which can be customized to answer more complex user-defined questions across a schematic hierarchical net.

Creating a Mapping Schematic

Schematic pin maps enable you to switch between design tasks, such as new cell netlisting, layout creation, or any other design task, without the need to modify existings cells or the design text file.

For example, this facility would be beneficial if you wanted to, import read-only IP (as described in a text cellview), and integrate this information into a design to replace an existing cellview. However, if, for example, the port names in the text file differed from port names in the symbol, there could be a requirement to change the symbol and all of the schematics where that cell is instantiated (or perhaps even be enforced to edit the IP file). The ability to create schematic pin maps can however alleviate these potential issues.

The following procedure explains how you can use the Create a Mapping Schematic forms (x2 steps) to successfully add schematic pin maps. By following this procedure you will be able to automatically map relevant attributes, and create a reliable and quick flow to map one symbol with another symbol, concluding with each view's nets adopting the same functionality.

To create a mapping schematic:

1. Select *Create – Mapping Schematic*.

The Create a Mapping Schematic Step 1 of 2 form is displayed.

Create a Mapping Schematic Step 1 of 2

Lower Level Cellview	Upper Level Cellview
Referenced Symbol:	Referenced Symbol:
Library: analogLib	Library: zambezi45
Cell: TTL_a2d	Cell: LP_pll_dig_combo
View: symbol	View: symbol
Mapping Schematic:	
View Name: schematic_TTL_a2d	
Next Close Help	

Figure 3-6 Create a Mapping Schematic Step 1 of 2 Form

2. In the *Lower Level Cellview* and the *Upper Level Cellview* group boxes, specify the library, cell, and view settings for the cellviews that contain the referenced symbol views.

3. Specify the mapping schematic *View Name*.

4. Click the *Next* button.

The Create a Mapping Schematic Step 2 of 2 form is displayed.

In step 2, you map the unmapped nets in the *Lower Level Cellview* (specified in step 1) to the unmapped nets in the *Upper Level Cellview* (also specified in step 1).

5. Under *Unmapped Nets*, select from the `lowerCell` and `upperCell` lists the nets that you want to map.

Specific options in the Other Setup group box become available when a pin in either of the two symbol views is selected. Which options are available is based on the type of pins that are currently selected.

6. Click the *Map* button to map the selected nets. You can map only one pair of nets at a time. Or you can automatically map all the nets listed under *Unmapped Nets*, by selecting *Auto-Map*.

These nets are displayed in the *Mapped Nets* section.

7. Repeat the net mapping process as required.

8. If required, select *Save To* to save the mapped net settings to the text file listed in the *File* field.

9. Click *OK* to create the mapping schematic.

Important

All the pins listed under *Unmapped Nets* must be mapped or defined before you can click *OK* to complete the creation of the mapping schematic.

Virtuoso Schematic Editor User Guide

Creating Schematics

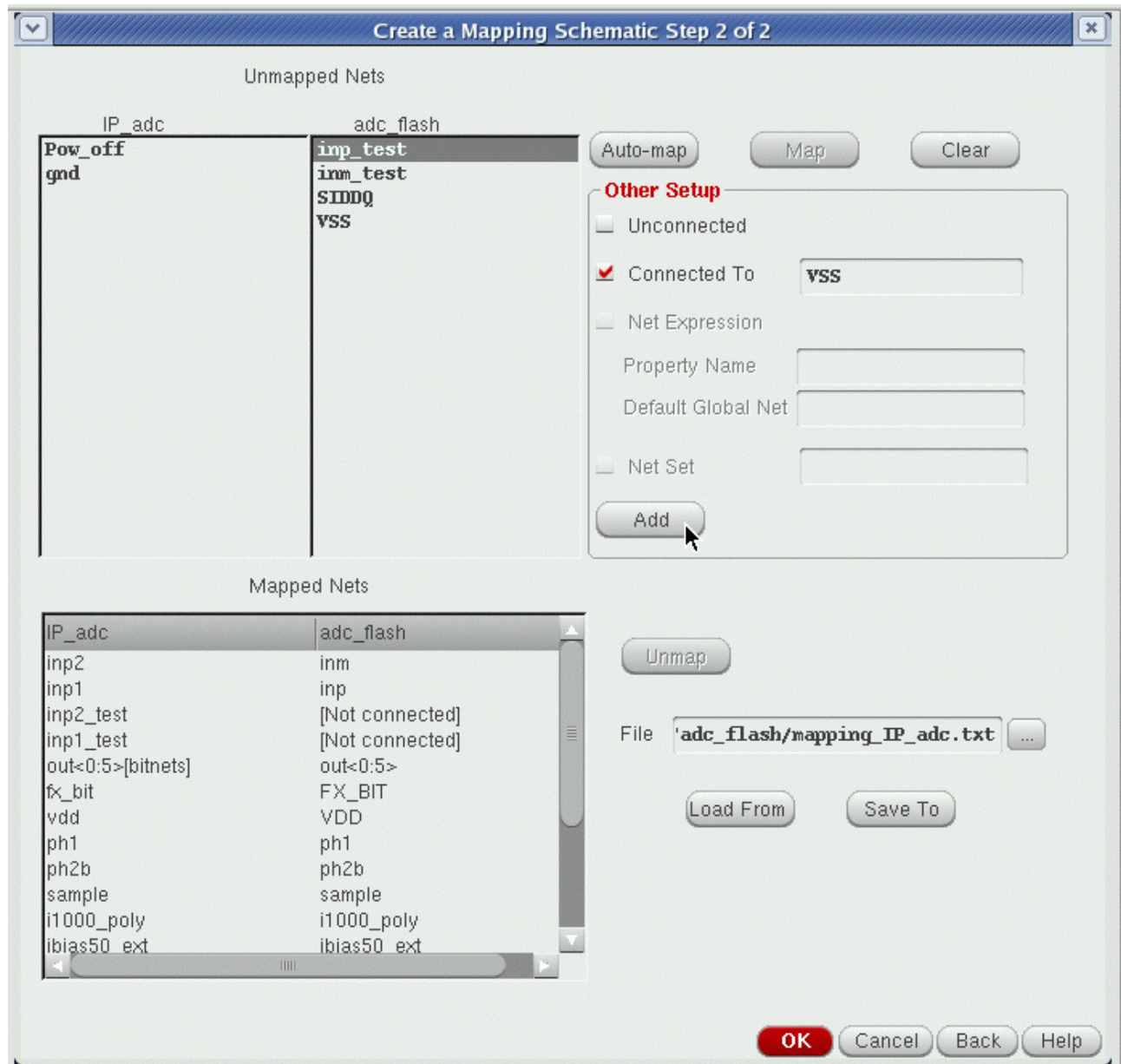


Figure 3-7 Creating Mapping Schematic Step 2 of 2 Form - All Nets Mapped

Note: If the upper or lower level symbol has bit pins like $a<0>$, $a<1>$, and $a<2>$, then these bit pins will be represented in the form of a bus, represented as $a<0:2>$ [bitnets]. This will enable mapping of a bus (vector) pin with individual pins combined. Additionally, if we have $b<0:2>$ pin in one symbol, and $a<0>$, $a<1>$, and $a<2>$ in the other symbol, this will enable mapping of $b<0:2>$ with individual bit pins ($a<0>$ $a<2>$).

Virtuoso Schematic Editor User Guide

Creating Schematics

After the mapping schematic is complete, the net mappings are reflected in the lowerCell view (specified in step 1), as shown.

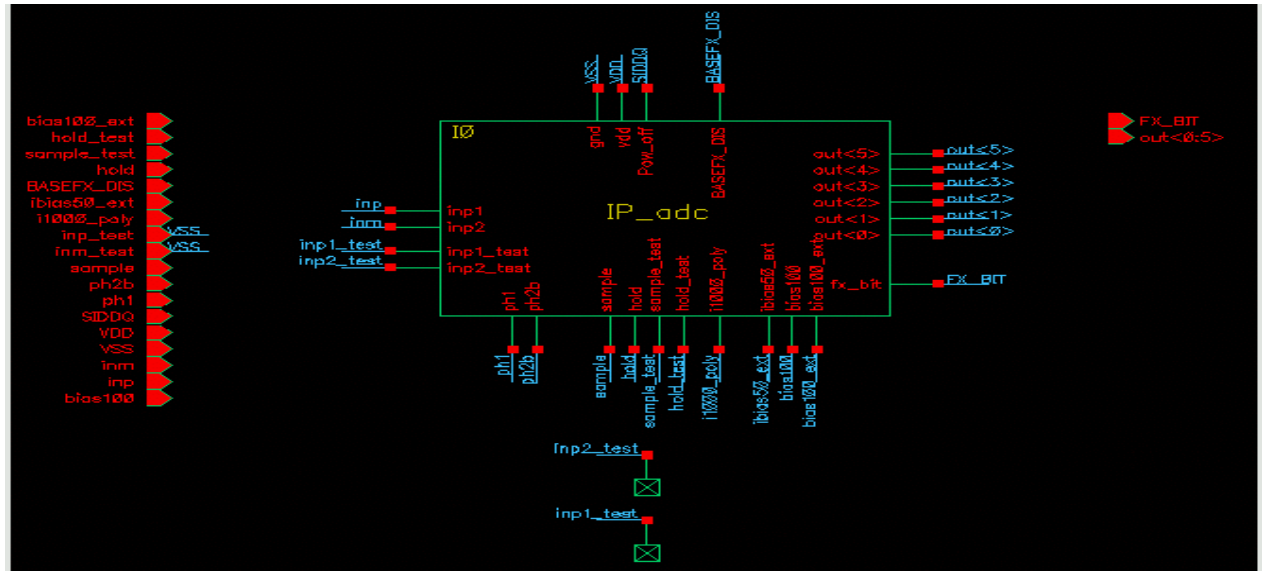


Figure 3-8 Nets Mapped between Lower- and Upper-Level Cellviews

Virtuoso Schematic Editor User Guide

Creating Schematics

Creating a Multisheet Schematic

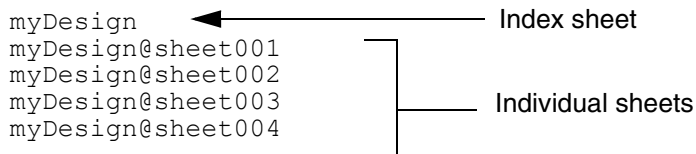
You can use multisheet schematics to partition your design among several sheets, thus creating a hierarchy.

This chapter provides information about the following topics:

- [Browsing a List of Multisheets](#)
- [Converting a Single-Sheet Schematic to a Multisheet Schematic](#)
- [Adding New Sheets](#)
- [Editing the Title Block](#)
- [Editing the Sheet Border Size and Type](#)
- [Understanding Offsheet Pins and Their Naming Conventions](#)
- [Adding Offsheet Pins to Multisheet Schematics](#)
- [Understanding the Index Schematic](#)
- [Going to and from the Index Schematic](#)
- [Repositioning a Schematic within the Sheet Borders](#)
- [Deleting Sheets](#)
- [Reordering Sheets](#)
- [Renumbering Sheets](#)
- [Working with Cross-References](#)
- [Customizing Multisheets](#)
- [Customizing Pin Cross-References](#)
- [Presetting the Title Block Values](#)

Browsing a List of Multisheets

The system uses a base name to group sheets of a multisheet schematic in a library. By default, the base name is the cell name of the original single-sheet schematic. Each sheet of a multisheet schematic has a unique sheet number.



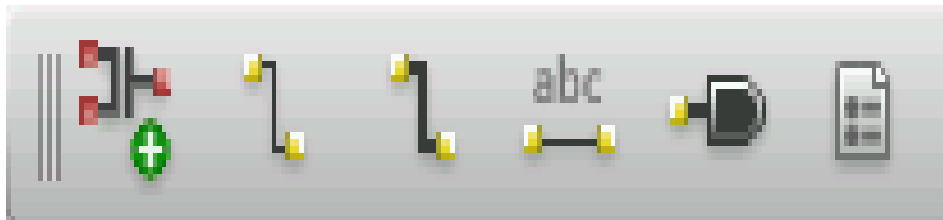
Converting a Single-Sheet Schematic to a Multisheet Schematic

When you extend your design over several sheets, you improve readability, maintain drafting standards, and produce better plot output.

To convert a single-sheet schematic to a multisheet schematic,

1. Open a single-sheet schematic.
2. Choose *Create – Multisheet*.

The Create Schematic Sheet form appears.



3. Change the options to specify a multisheet schematic of the size and type you want.
4. Click *OK*.

A sheet border and title block appear in your design window as a new tab. You might need to change the size of the border, reposition your schematic within the border, or edit the sheet title.

Important

A new *Sheet* menu is added to the menu bar.

Adding New Sheets

To add a new sheet to a multisheet schematic,

1. From the schematic window, choose *Create – MultiSheet*.

The Create Schematic Sheet form appears.

2. In the *Number* field, type the number of the sheet you want to add.

If you want to determine the total number of pages, look at the title block in the lower right-hand corner of your schematic.

The size and type automatically default to the current values of the present set of sheets.

3. Click *OK*.

If a sheet with the number you chose already exists, you are prompted to confirm the creation of the new sheet. The schematic editor creates the new sheet with the number you chose and renumbers the remaining sheets. The schematic editor also updates all properties in the index schematic.

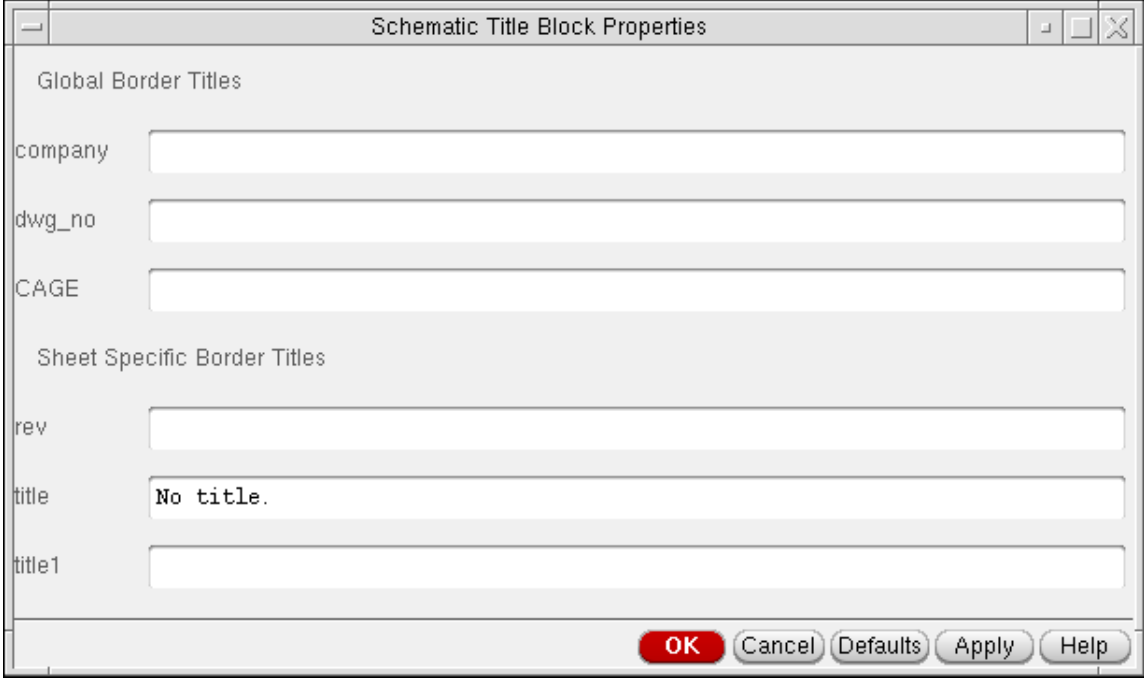
Editing the Title Block

Title blocks help you keep track of the titles and revisions.

To add or modify the title block of a multisheet schematic,

1. From the schematic window, choose *Edit – Sheet Title*.

The Schematic Title Block Properties form appears.



The screenshot shows a dialog box titled "Schematic Title Block Properties". It is divided into two sections: "Global Border Titles" and "Sheet Specific Border Titles".

Global Border Titles

- company: [text input field]
- dwg_no: [text input field]
- CAGE: [text input field]

Sheet Specific Border Titles

- rev: [text input field]
- title: [text input field containing "No title."]
- title1: [text input field]

At the bottom right, there are five buttons: **OK** (highlighted in red), **Cancel**, **Defaults**, **Apply**, and **Help**.

2. Change the options to edit the title block.

From an index schematic, only the *Global Border Titles* fields are editable.

All fields are visible when you edit a sheet.

3. Click *OK*.

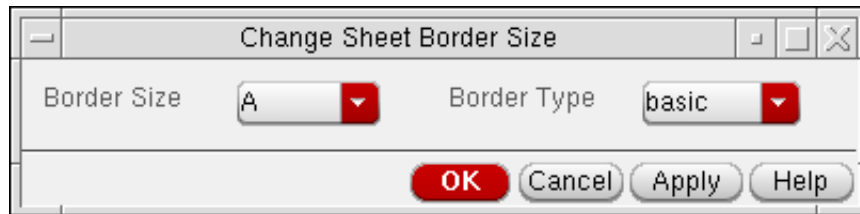
The following sheet title block shows the information from the form on the previous page.

Editing the Sheet Border Size and Type

To change the existing size and type of a schematic border,

1. From the schematic window, choose *Edit – Sheet Size*.

The Change Sheet Border Size form appears.



2. Change the *Border Size* and *Border Type* cyclic fields.

To delete the sheet border of a schematic sheet, set the *Border Size* to *none*.

3. Click *OK*.

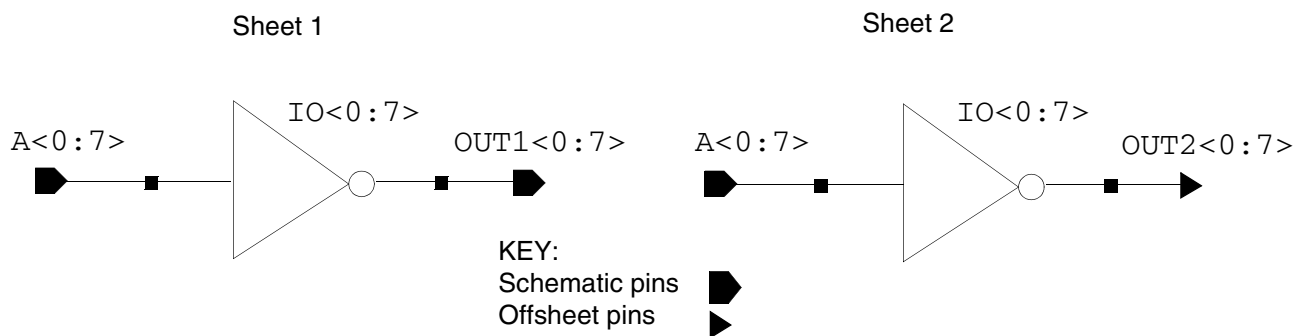
Understanding Offsheet Pins and Their Naming Conventions

Offsheet pins are the pins you use to connect nets between sheets of a multisheet schematic. Hierarchical pins (schematic pins) connect signals that span different levels of a hierarchical design. When you place a hierarchical pin or offsheet pin in a sheet schematic, an equivalent pin is created in the index schematic.

The following examples show how to use hierarchical and offsheet pins in a multisheet schematic. See the [pin naming conventions](#) used in these four examples.

Example 1

Two hierarchical pins, $A<0:7>$ and $OUT1<0:7>$, are placed in sheet 1. Bus signal $A<0:7>$ appears in both sheets. To connect these sheets, sheet 1 must also contain a matching hierarchical pin named $out2<0:7>$ or an offsheet pin.



Virtuoso Schematic Editor User Guide

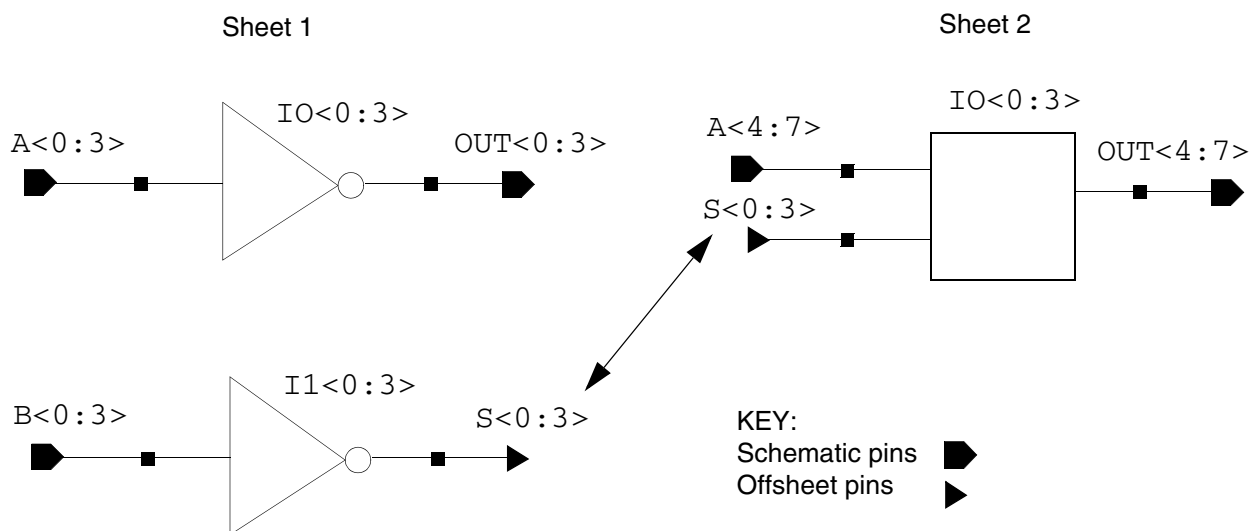
Creating a Multisheet Schematic

Example 2

In this example, the following hierarchical pins represent the partitioned bit width of a bus signal, $A<0:7>$ and $OUT<0:7>$:

$A<0:3>$, $B<0:3>$, $OUT<0:3>$ (in sheet 1)
 $A<4:7>$ and $OUT<4:7>$ (in sheet 2)

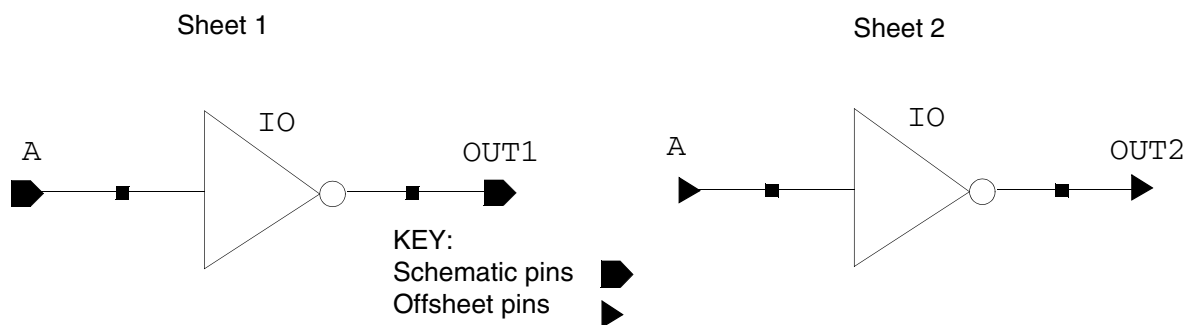
The example shows the usage of offsheet pin $S<0:3>$ connecting sheet 1 to sheet 2 and sheet 2 to sheet 1.



Example 3

Pin A is hierarchical in sheet 1.

To access pin A from sheet 2, you use an offsheet pin.

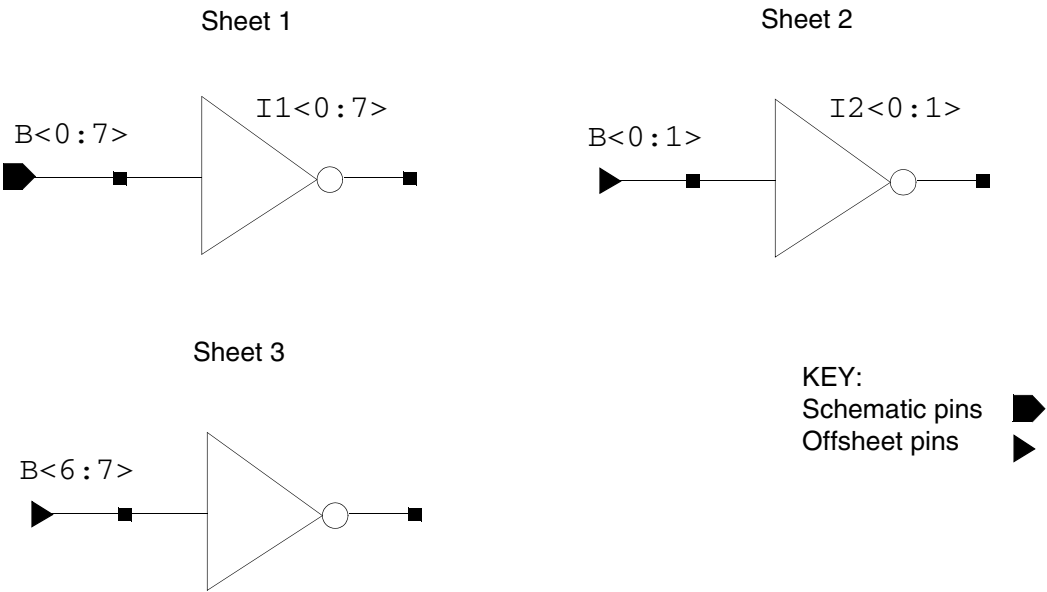


Example 4

A hierarchical pin for the bus $B<0:7>$ is placed in sheet 1.

Sheet 2 uses only two bits of the bus, which you access with offsheet pin $B<0:1>$.

Sheet 3 uses another two bits, which you access with offsheet pin $B<6:7>$.



Adding Offsheet Pins to Multisheet Schematics

To add offsheet pins to connect signals that span several sheets,

1. From the schematic window, choose *Create – Pin*.

The Create Pin – Schematic form appears.

The 'Create Pin' dialog box is shown with the following settings:

- Names:** (empty text field)
- Direction:** input
- Usage:** schematic
- Signal Type:** signal
- ☐ Expand buses
- ☐ Place multiple pins
- Net Expression:**
 - ☐ Attach to pin
 - ☐ Derive from pin name
- Property Name:** (empty text field)
- Default Net Name:** (empty text field)
- Font:**
 - Height:** 0.06250
 - Style:** stick
 - Justification:** lowerCenter
- Supply Sensitivity:**
 - Attach To:** Both
 - Power Terminal:** None
 - Ground Terminal:** None
- Rotation:** (four rotation icons: 0°, 45°, 90°, 135°)

Buttons at the bottom: **Hide**, **Cancel**, **Defaults**, **Help**.

Virtuoso Schematic Editor User Guide

Creating a Multisheet Schematic

2. Change the *Usage* cyclic field to *offSheet*.

For more details, see the single-sheet schematic *Create– Pin* command.

3. Move the pointer into your design and click to place the pin.

Understanding the Index Schematic

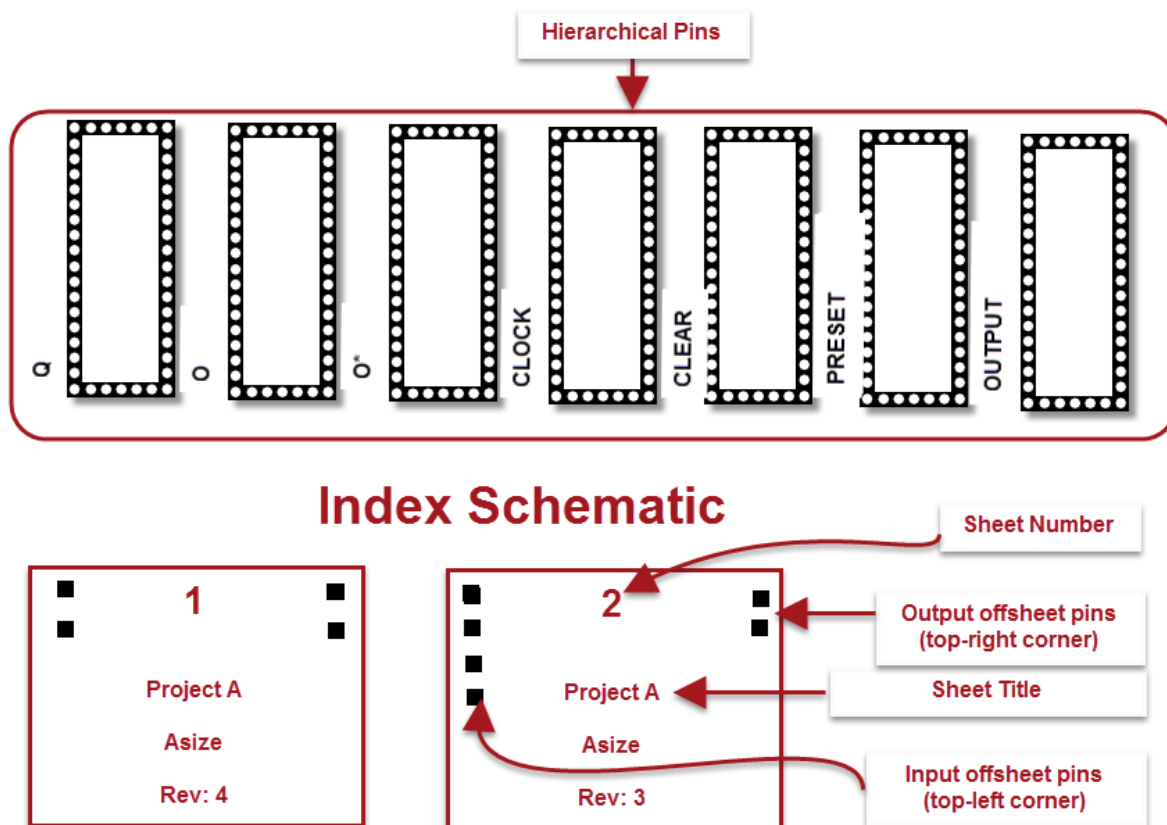
Important

To create a multisheet schematic select *Create – MultiSheet*. This adds a *Sheet* menu to the menu bar.

When you create a multisheet schematic, an index schematic to organize the sheets is automatically created. The index schematic provides a graphical overview of all the sheets in the schematic.

When you run a check on your design, all hierarchical pins and their names are displayed at the top of the index schematic.

The following diagram is an example of an index schematic.



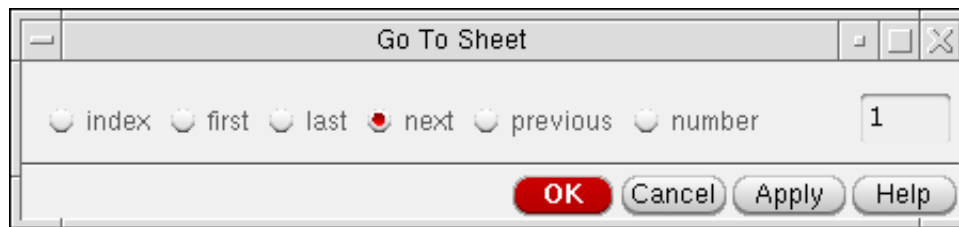
Note: When renaming the cell name of a multisheet schematic's index cell with the library manager *Edit – Rename* command, only the index cell is renamed. The cell names for all the sheet cells are not renamed as expected. To correct this situation, use the *Edit – Rename* command to rename all the cells of a multisheet schematic individually.

Going to and from the Index Schematic

To go to the index schematic,

1. From the schematic window, choose *Sheet – Go To*.

The Go To Sheet form appears.



2. Choose *index*.
3. Click *OK*.

To go to a sheet from the index schematic,

1. Press and hold down the middle mouse button while the pointer is over your destination sheet.
2. On the object-sensitive menu (OSM) that pops up, choose *Go To*.

You can set the OSMs on or off at any time. For details, refer to *Virtuoso Design Environment User Guide*.

Repositioning a Schematic within the Sheet Borders

If part of your schematic extends beyond the sheet border, you must reposition it.

To reposition a schematic within the sheet borders,

1. In the schematic window, draw a selection box around all the objects in your schematic.
2. Choose *Edit – Move*.
3. Select a reference point by clicking in the design.

The design is now attached to the pointer.

4. Move the pointer within the sheet border and click to place the reference point of the highlighted schematic.
5. Click in an open area to deselect the highlighted schematic.

The schematic now appears within the sheet border.

Deleting Sheets



This operation cannot be undone.

You can use this operation only when editing the index schematic of a multisheet design. You can delete a sheet or range of sheets from a multisheet schematic.

To delete a sheet,

1. From an index schematic, choose *Sheet – Delete*.

The Delete Schematic Sheet form appears.

2. Type the page range you want to delete.

You are prompted to confirm or cancel the deletion.

Reordering Sheets



Caution

This operation cannot be undone.

To change the order in which sheets are numbered,

1. From an index schematic, choose *Sheet – Renumber*.

The Edit Schematic Sheet Number form appears.

2. Specify the sheet to be renumbered and the new sheet number.
3. Click *OK*.

If there is already a sheet with the new sheet number, the sheet whose number you want to change is inserted before the existing sheet and gets the specified sheet number. The sheets following it are automatically renumbered. On the example form above, sheet 1 becomes sheet 10. sheets 2 through 9 stay the same, and sheets 10 and up, if they exist, are increased by 1. The values displayed in the title block change to reflect the new sheet numbers.

Renumbering Sheets



This operation cannot be undone.

To sequentially renumber all sheets in a multisheet schematic beginning with 1,

- From an index schematic, choose *Sheet – Resequence*.

All sheets are sequentially renumbered beginning with sheet 1. For example, if you previously deleted sheets 4 and 5 so that your sheets are numbered 1 2 3 6 7, this command reorders the pages in the correct sequence, 1 2 3 4 5.

Working with Cross-References

You can display cross-reference annotations to schematic and offsheet pins to indicate other locations in the design where the same pins are used on different sheets.

You can cross-reference multisheet pins to

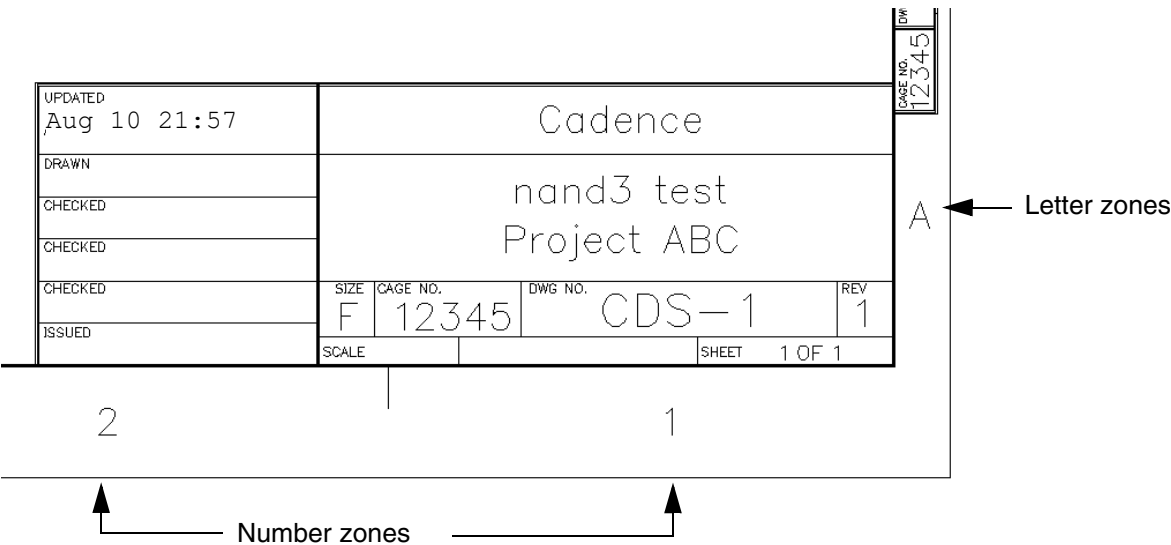
- Reference all pins on all sheets of a multisheet schematic
- Zoom in automatically from one pin to the cross-referenced pin
- Use the index schematic to link directly to specific hierarchical pins in your design
- Produce a cross-reference report of all locations in your schematic where pins are cross-referenced

This section discusses the following topics:

- [Understanding Cross-Reference Zones](#)
- [Understanding Pin Cross-References](#)
- [Understanding Cross-Reference Reports](#)
- [Preparing to Cross-Reference](#)
- [Displaying Cross-References](#)
- [Zooming In to View All Cross-Referenced Pins One by One](#)
- [Going to a Specific Hierarchical Pin](#)
- [Viewing Cross-Reference Reports](#)
- [Editing a Cross-Reference Display](#)
- [Controlling the Format String Syntax](#)
- [Customizing the Format String Syntax](#)

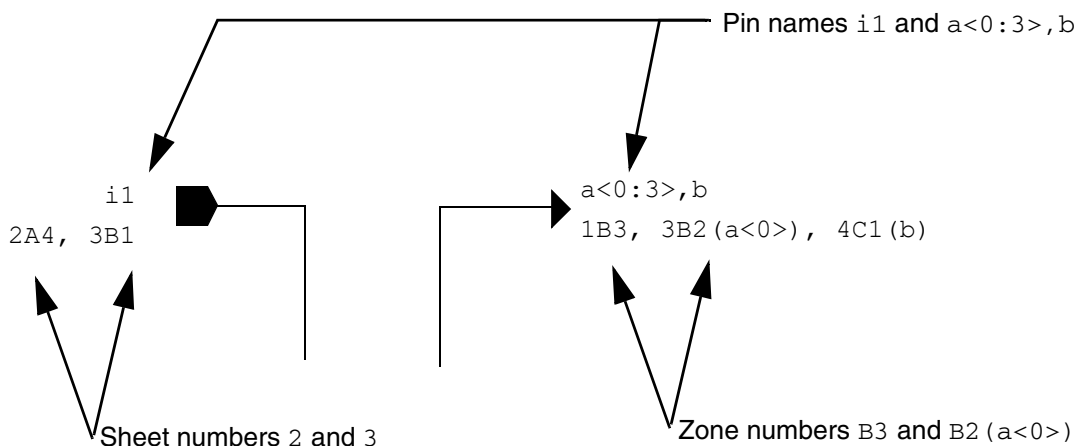
Understanding Cross-Reference Zones

Multisheet schematics with borders show numbers and letters that reference different zones, similar to the squares on a map.



Understanding Pin Cross-References

Pin cross-references are displayed on the schematic next to the pin name.



Example 1

Two pin names reference each other if any of the individual bits of the pin names are shared. For example, the bundle `a<0:3>,b` references the following pins:

`a<0:3>`
`b`

Example 2

Pin `PP<7:0>` is referenced at sheet 1, zone A2 by pin name `PP<0:3>`; at sheet 2, zone A1 by pin name `PP<7:0>`; and at sheet 3, zone B4 by pin name `PP<7:0>`. The reference names are not listed for the last two locations because the reference name matches the pin name.

`PP<7:0>` `1A2 (PP<0:3>), 2A1, 3B4`

Understanding Cross-Reference Reports

The following cross-reference report shows pin names and lists locations where the pins are referenced.

Schematics Cross Reference Report
Library name: myLib
Module Name: test2

Sheet Number: 1 Cellname: test2/
sheet001

Pin Name	References
B<0:7>	2A1 (B<0:1>), 3A1 (B<6:7>)
CLEAR	1C3
CLEAR	1C3
CLEAR	1B3
PRESET	1A1
PRESET	1C3

Sheet Number: 2 Cellname: test2/
sheet002

Pin Name	References
B<0:1>	1C4 (B<0:7>)

Sheet Number: 3 Cellname: test2/
sheet003

Pin Name	References
B<6:7>	1C4 (B<0:7>)

A comma separates location references

Duplicate zones are shown if you turn the option on.

Sheet 1 shows where the signal is referenced. Zone C3 specifies the area where the pin is referenced, using alphanumeric grids on the sheet borders.

The reference name lists the name of the pin that made the reference, B<0:7>.

November 2023
© 2023

355

Product Version IC23.1
All Rights Reserved.

Preparing to Cross-Reference

You must make sure your schematic is error free before you cross-reference pins, but you do not have to correct warnings.

To check a multisheet schematic for errors before displaying cross-references,

1. From the index schematic, choose *File – Check and Save*.

A dialog box appears, showing you the number of errors and warnings in your schematic. The CIW shows details about the errors and warnings and how to fix them.

2. Correct all errors, if applicable.

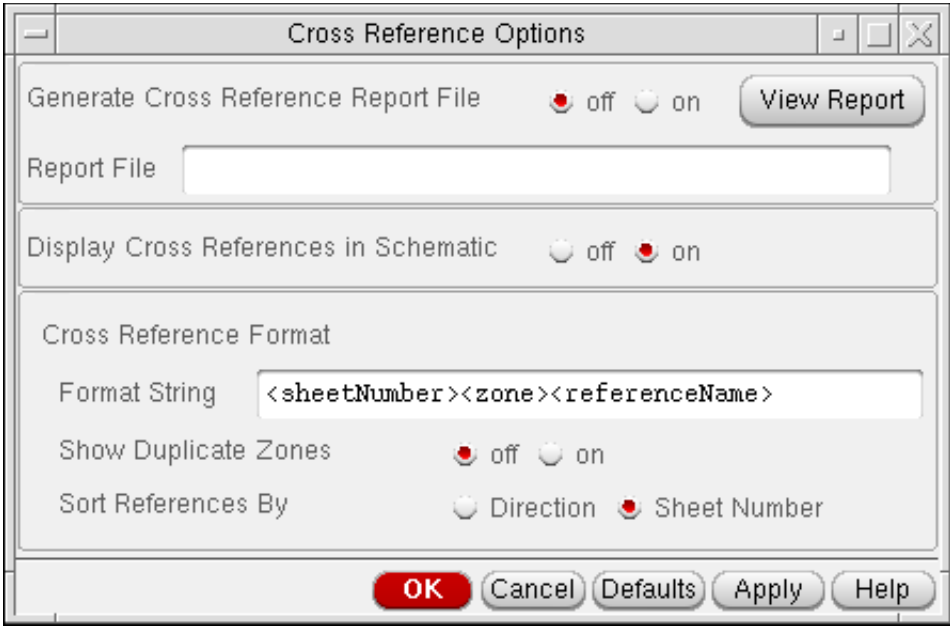
Displaying Cross-References

You can display cross-reference annotations to schematic and offsheet pins to indicate other locations in the design where the same pins are used on different sheets.

To display cross-reference annotations to schematic or offsheet pins,

1. From the schematic window, choose *Sheet – Cross Reference*.

The Cross Reference Options form appears.



The screenshot shows the 'Cross Reference Options' dialog box. It has a title bar with standard window controls. The main area contains several sections: 'Generate Cross Reference Report File' with radio buttons for 'off' (selected) and 'on', and a 'View Report' button; a 'Report File' text field; 'Display Cross References in Schematic' with radio buttons for 'off' and 'on' (selected); a 'Cross Reference Format' section with a 'Format String' text field containing '<sheetNumber><zone><referenceName>'; 'Show Duplicate Zones' with radio buttons for 'off' (selected) and 'on'; and 'Sort References By' with radio buttons for 'Direction' and 'Sheet Number' (selected). At the bottom are buttons for 'OK' (highlighted in red), 'Cancel', 'Defaults', 'Apply', and 'Help'.

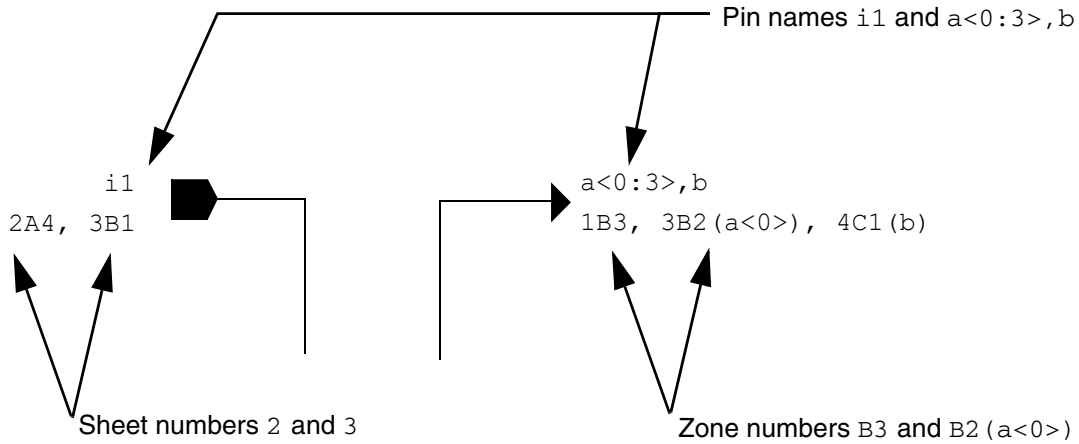
2. (Optional) To produce a report that you can view in a window,
 - a. Turn on *Generate Cross Reference Report File*.
 - b. In the *Report File* field, type the name of the file in which to save the report.
 - c. Edit the *Format String* field, if necessary.
3. Click *OK*.

The system searches for each pin on each sheet and creates a list of the locations where you use each pin or a subset of a bus or bundle pin. The reference pin must have at least one shared member name with the pin name.

Virtuoso Schematic Editor User Guide

Creating a Multisheet Schematic

Pin cross-references are displayed on the schematic next to the pin name.



You can now zoom in from one pin to another by choosing *Sheet – Go To Pin*.

Zooming In to View All Cross-Referenced Pins One by One

To see each occurrence of a cross-referenced pin on all sheets of a multisheet schematic,

1. From the index schematic, choose *Check – Current Cellview* to make sure the cellview is error free.
2. Choose *Sheet – Cross Reference* to generate the pin cross-references.
3. Choose *Sheet – Go To Pin*.

The Go To Pin form appears.



4. Select the pin or offsheet connector in your design to follow throughout the multisheets.

The schematic editor zooms in to the area that contains the specified pin or offsheet connector. For example, if you selected *first*, the schematic editor shows the cellview that contains the first reference of the pin or offsheet connector you selected.

You can continue to view all related pins using the Go To Pin form.

Going to a Specific Hierarchical Pin

To go to a specific hierarchical pin,

1. From the index schematic, choose *Check – Current Cellview* to make sure the cellview is error free.
2. Choose *Sheet – Cross Reference* to generate the pin cross-references.
3. Go to the index schematic.
4. Press and hold down the middle mouse button while the pointer is over one of the hierarchical pins.
5. On the object-sensitive menu (OSM) that pops up, choose *Go To*.

You can set the OSMs on or off at any time. For details, refer to [Virtuoso Design Environment User Guide](#).

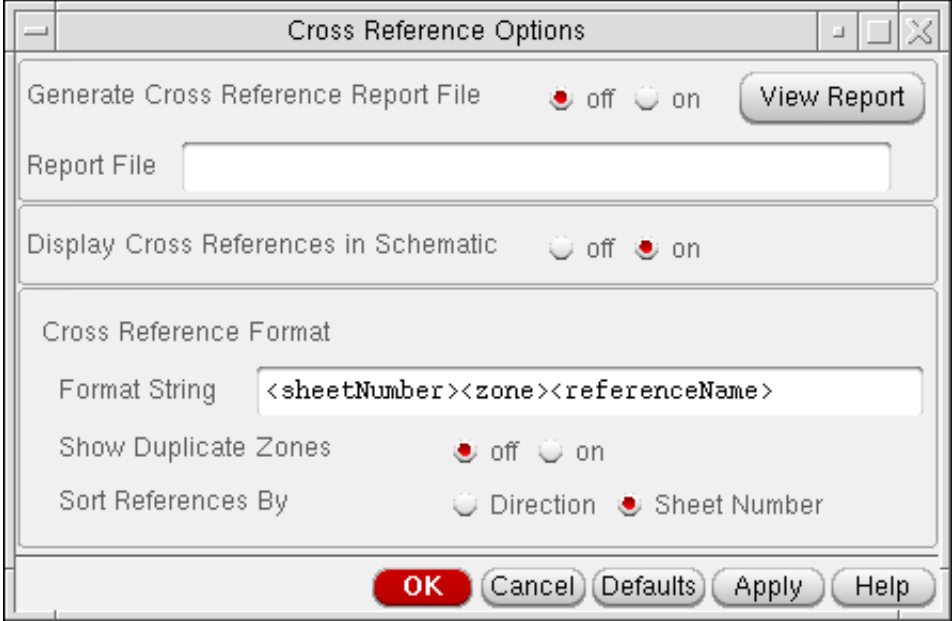
Viewing Cross-Reference Reports

The cross-reference report file contains a list of pin names followed by all of the locations where each pin is referenced in a multisheet schematic. The report lists the cross-references in the same format as the strings that are displayed in the schematic.

To view a cross-reference report,

1. From the schematic window, choose *Sheet – Cross Reference*.

The Cross Reference Options form appears.



The image shows a dialog box titled "Cross Reference Options". It contains several sections: "Generate Cross Reference Report File" with radio buttons for "off" (selected) and "on", and a "View Report" button; "Report File" with a text input field; "Display Cross References in Schematic" with radio buttons for "off" and "on" (selected); "Cross Reference Format" with a "Format String" input field containing "<sheetNumber><zone><referenceName>"; "Show Duplicate Zones" with radio buttons for "off" (selected) and "on"; and "Sort References By" with radio buttons for "Direction" and "Sheet Number" (selected). At the bottom are buttons for "OK", "Cancel", "Defaults", "Apply", and "Help".

2. Click *View Report*.

A window listing the pin references appears, as follows:

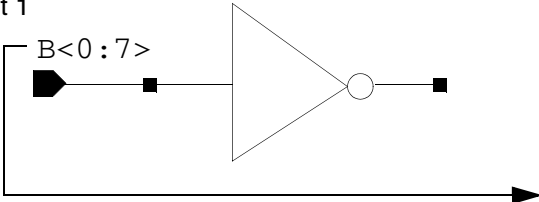
Virtuoso Schematic Editor User Guide

Creating a Multisheet Schematic

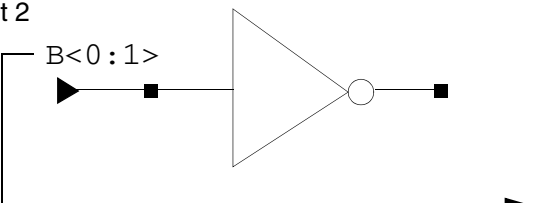
Example

In this sample design, a hierarchical pin for the bus B<0 : 7> is declared in sheet 1. Sheets 2 and 3 use two bits of the bus declared in sheet 1. These two bits are represented by offsheet pins.

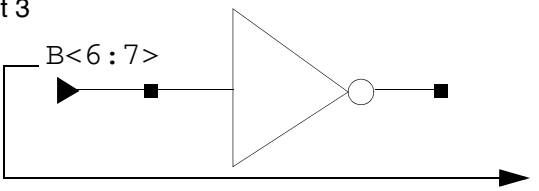
Sheet 1



Sheet 2



Sheet 3



Schematics Cross Reference Report

Library name: myLib
Module Name: test2

Sheet Number: 1 Cellname: test2/sheet001

Pin Name	References
B<0:7>	2A1 (B<0:1>), 3A1 (B<6:7>)
CLEAR	1C3
CLEAR	1C3
CLEAR	1B3
PRESET	1C3
PRESET	1C3

Sheet Number: 2 Cellname: test2/sheet002

Pin Name	References
B<0:1>	1C4 (B<0:7>)

Sheet Number: 3 Cellname: test2/sheet003

Pin Name	References
B<6:7>	1C4 (B<0:7>)

Editing a Cross-Reference Display

You can edit the cross-reference display using standard editing commands and control the visibility of the strings. You can

- Make text invisible
- Change fonts and sizes to make text easier to read
- Move the text independent of the pin or pin label (when you move the pin, cross-reference text also moves)

To edit cross-reference text,

1. From the view, choose *Edit – Properties – Objects*.

The Edit Object Properties form appears.

2. Select the cross-reference data in your design.
3. Edit the data on the Edit Object Properties form.
4. Click *OK*.
5. Choose *Edit – Move* to move text, if applicable.

Controlling the Format String Syntax

You can control the format of cross-references (written to the report file or displayed next to pins) by using your own cross-reference format string. You construct a string using the following parameters in any order:

[sheetNumber]	→	Sheet number that contains the pin
[zone]	→	The alphanumeric grids that contain the pin
[direction]	→	The direction of the pin
[referenceName]	→	The name of the pin. The referenced pin name is only printed when it does not match the original pin name. For example, the following string lets you see which bits of a bus are used on which sheets:

`"1A1 (bus<0:7>), 2B2 (bus<0:3>), 3C3 (bus<4:7>)"`

Syntax

pinName location, location, ...

location := [sheetNumber] [zone] [direction] [referenceName]

direction := 'i' | 'o' | 'io'

for input, output, or input/output respectively

referenceName := B<0:7>

pin name that has at least one shared bus component

Example

Using the default format of *sheetNumber zone referenceName*

Pin Name	References
PP<7:0>	1A2 (PP<0:3>), 2A1, 3B4

The reference name (PP<0:3>) is enclosed in parentheses only if it is different from the pin name. PP<7:0> is referenced at

- Sheet 1, zone A2, by pin name PP<0:3>
- Sheet 2, zone A1, by pin name PP<7:0>
- Sheet 3, zone B4, by pin name PP<7:0>

Customizing the Format String Syntax

You can customize the following:

- Pin reference format

The `pinRefFormat` environment variable specifies the format for displaying the pin reference information that appears next to each pin when you choose *Sheet – Cross Reference*.

- Pin reference duplicates

The `pinRefDuplicates` environment variable specifies whether duplicate pin references within the same zone of a multisheet schematic are reported when you choose *Sheet – Cross Reference*.

- Pin reference sorting

The `pinRefSorting` environment variable specifies how pin references are sorted when you choose *Sheet – Cross Reference*.

Customizing Multisheets

The following sections describe in detail the process of customizing multisheets:

- [Using Customized Sheets](#)
- [Customizing the Types and Sizes of Sheet Borders](#)
- [Customizing the Cross Reference Options Form](#)
- [Drawing a Sheet Border](#)
- [Drawing Sheet Zones](#)
- [Drawing Zone Labels](#)
- [Adding the Sheet Zone Property](#)
- [Adding a Title Block](#)
- [Adding Title Block Values with Normal Labels](#)
- [Adding Title Block Values with NLP and IL Labels](#)
- [Verifying the Label Type](#)
- [Adding Title Block Properties](#)

Using Customized Sheets

In general, to use customized sheets in your schematic,

1. Create your own sheet as a symbol or use one from Cadence-supplied *US_10ths* library.
2. Open the schematic view in Virtuoso® Schematic Editor.
3. Instantiate the desired sheet symbol in your schematic.
4. Fit the sheet symbol into the page.
5. Move, align, and resize the schematic within the sheet border.
6. Check and save your schematic.

Customizing the Types and Sizes of Sheet Borders

You can use your own customized sheet border symbols when you create new sheet schematics. You can customize the display of pin cross-references.

The fields on the Create Schematic Sheet appear as follows:



To customize the default settings on the Create Schematic Sheet form,

1. In the configuration file, search for `schSheetMasters`.
2. Define or delete the types and sizes of sheet borders.

For example, you can modify the C-size sheet as follows:

```
list("myLib" "myCsize" "symbol")  
list("myLib" "myCsize.c" "symbol"))
```

where

`myLib` is the name of your design library.

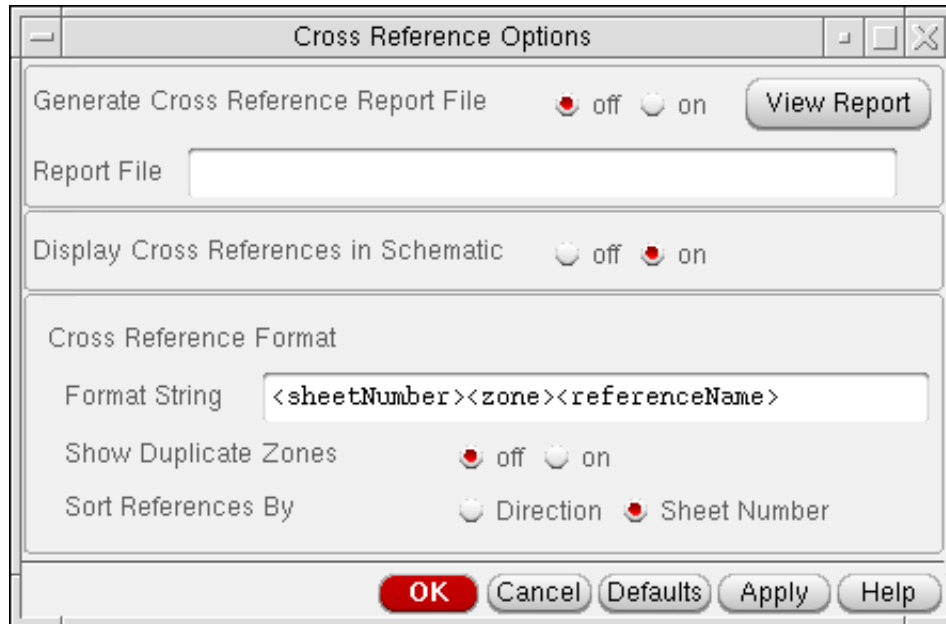
`myCsize` is your name for the size of your customized sheet.

`myCsize.c` is the continuation sheet.

`symbol` specifies all multisheet borders are symbols.

Customizing the Cross Reference Options Form

The fields on the Cross Reference Options appear as follows:



The screenshot shows the 'Cross Reference Options' dialog box. It has a title bar with standard window controls. The main area contains several sections: 1. 'Generate Cross Reference Report File' with radio buttons for 'off' (selected) and 'on', and a 'View Report' button. 2. 'Report File' with an empty text input field. 3. 'Display Cross References in Schematic' with radio buttons for 'off' and 'on' (selected). 4. 'Cross Reference Format' section containing: a 'Format String' text field with the value '<sheetNumber><zone><referenceName>', 'Show Duplicate Zones' with radio buttons for 'off' (selected) and 'on', and 'Sort References By' with radio buttons for 'Direction' and 'Sheet Number' (selected). At the bottom are five buttons: 'OK' (highlighted in red), 'Cancel', 'Defaults', 'Apply', and 'Help'.

You can customize default settings on the Cross Reference Options form.

The cross-reference format string variable, PinRefFormat, can contain any text and the following keywords: `zone`, `sheetNumber`, `direction`, `referenceName`.

For example:

```
PinRefFormat = "Zone:zone, Sheet:sheetNumber, Dir:direction"
```

yields reference strings such as

```
"Zone:A5, Sheet:3, Dir:input"
```

Drawing a Sheet Border

A sheet border symbol contains several cellview properties that you can customize. The border symbol needs to contain two other cellview properties, both with the property type `string`.

- `"nlAction" = "ignore"`

Choose *Edit – Properties – Cellview* to add this property.

- `geQuerySelSet()`

Type this function in the CIW and use the *Add* button to add a type `string` property:

`"schType" = "border"`

To draw a customized outer sheet border,

1. From the symbol window, choose *Create – Shape – Rectangle*.

The Add Symbol Shape form appears. If the form does not appear, press F3.

2. Draw a rectangle that represents the edge of your customized sheet.
3. Press `ESC` to cancel the *Create – Shape – Rectangle* command, and continue with the following section.

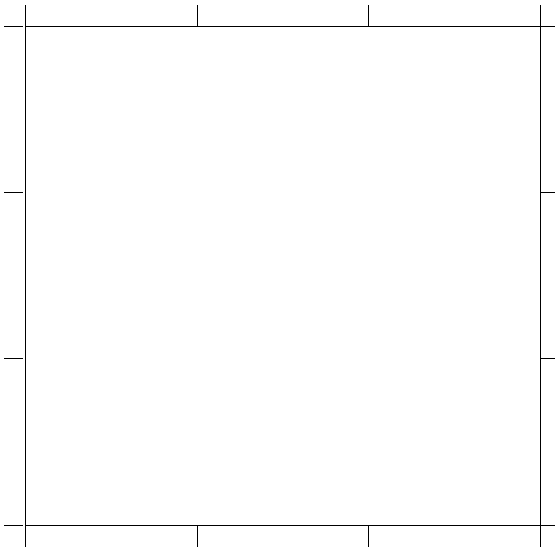
Drawing Sheet Zones

To add sheet zones,

1. From the symbol window, choose *Create – Shape – Line*.
2. Press **F3**.

The Add Symbol Shape form appears.

3. Draw equally spaced tick marks by drawing the first segment and then using the *Edit – Copy* command array option.



Drawing Zone Labels

To add customized sheet zone labels to the sheet border,

1. From the symbol window, choose *Create – Label*.

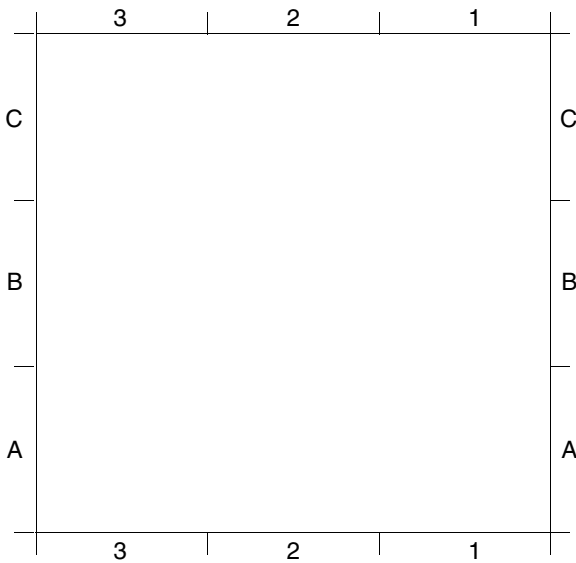
The Add Symbol Label form appears.

2. Type a label name in the *Label* field.

You must type and place each label separately.

3. Select the *normalLabel* option.

Sheet zones describe a coordinate system that helps you identify where an object is located in a cellview. You can divide your sheet into a 3 x 3 matrix with sheet zones 1, 2, 3, A, B, and C.



Adding the Sheet Zone Property

To add the `sheetHasZones` sheet zone property,

1. From the symbol window, choose *Edit – Properties – Cellview*.

The Edit Cellview Properties form appears.

2. Click *Add*.

The Add Property form appears.

3. In the *Name* field, type `schHasZones`.

4. In the *Value* field, type `(3 2 1) (C B A)`.

The value consists of two lists. The first describes the X coordinates `(3, 2, 1)` and the second describes the Y coordinates `(C, B, A)`. The order of the coordinates determines how they are displayed in the sheet border zones. The X coordinate list is displayed left to right, and the Y coordinate list is displayed top to bottom.

5. Click *OK*.
6. Click *OK* on the Edit Cellview Properties form.

Adding a Title Block

To add sheet titles and properties to a customized sheet border, you can

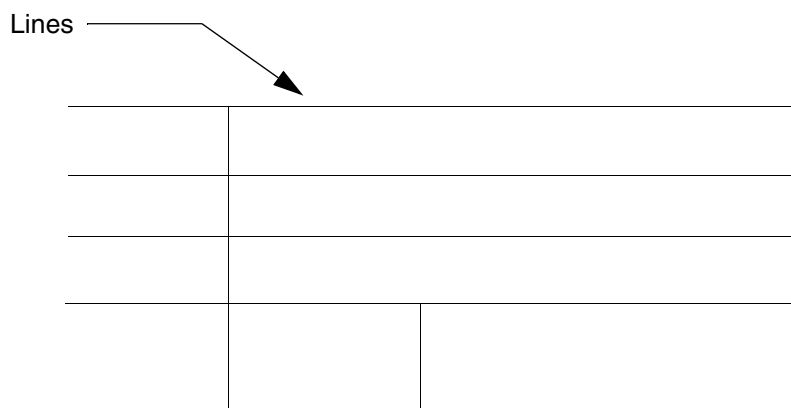
- Use the standard title symbol in the *US_8ths* reference library
- Create your own title block

To create your own title block (add an instance of a title block symbol),

1. Using the schematic editor, add an instance of the title block symbol to a schematic design.
2. Open the window that contains your customized border.
This window uses the symbol editor.
3. In the schematic window, select the title block instance.
4. Choose *Edit – Copy* to copy the instance from the schematic window to the symbol window that contains your customized border.

To draw a sheet title block for a customized sheet border,

1. From the symbol window, zoom in on the lower right corner of the sheet border.
2. Choose *Create – Shape – Line* and draw vertical and horizontal lines that define the block and the block fields.



Adding Title Block Values with Normal Labels

To name the sheet title block fields for a normal label,

1. From the symbol window, choose *Create – Label*.

The Add Symbol Label form appears.

Note: Do not use the *Create – Note – Text* command to add block titles.

2. Type a label name and choose *normalLabel* as the label type.
3. Click in your design to place the label.

Company		
Title		
Date		
		Sheet No.

4. Repeat steps 2 and 3 to place all the labels you plan to use.

Adding Title Block Values with NLP and IL Labels

To add the values of the title block fields with NLP and IL labels,

1. From the symbol window, choose *Create – Label*. The Add Symbol Label form appears.

You must type and place each label separately. An example follows:

Company	[@COMPANY]	← Label 1
Title	[@TITLE]	← Label 2
Date	ilInst~>cellView~>fileTimeStamp	← Label 3
	Sheet No. [@SHEET]	← Label 4

2. On the first line, type `[@COMPANY]`.
3. In the *Label Type* cyclic field, choose *NLPLabel*.
4. In your design, click the label to place it in the sheet title block.
5. On the second line, type `[@TITLE]`.
6. In the *Label Type* cyclic field, choose *NLPLabel*.
7. In your design, click the label to place it in the sheet title block.
8. On the third line, type the following Cadence® SKILL language code:

```
ilInst~>cellView~>fileTimeStamp
```

ilInst is a SKILL variable used to provide self-reference to an instance during graphic display of the cellview containing the instance.
9. In the *Label Type* cyclic field, choose *ILLabel*.
10. In your design, click the label to place it in the sheet title block. The SKILL code displays a time stamp; for example,

```
Jun 5 15:07:33 2000
```
11. On the forth line, type `[@SHEET]`.
12. In the *Label Type* cyclic field, choose *NLPLabel*.
13. In your design, click the label to place it in the sheet title block.

Verifying the Label Type

To verify the label type,

1. From the symbol window, select the label of the symbol.
2. In the CIW, type

```
geGetSelSet () ~>??
```

You can now see all the attributes of the label.

Adding Title Block Properties

The schematic editor recognizes a sheet title property called `schSheetTitles`. The `schSheetTitles` property is added to the `titleblock` cellview, which is a symbol. The `schSheetTitles` property contains the list of properties that can be edited with the *Edit – Sheet Title* command.

To add the `schSheetTitles` property to your sheet title block,

1. From the view, choose *Edit – Properties – Cellview*.

The Edit Cellview Properties form appears.

2. Click *Add*.

The Add Property form appears.

3. In the *Name* field, type, `schSheetTitles`.

4. Change the *Type* cyclic field to *ILList*.

5. Fill in the *Value* field.

The *Value* field contains two lists. The first list contains global titles that apply to all sheets. The second list contains local titles that apply to a specific sheet, all in list form.

```
Value := ( ( global-titles ) ( local-titles ) )
global-titles := nil |
    ( title-pair ) |
    ( title-pair ) global-titles
local-titles := nil |
    ( title-pair ) |
    ( title-pair ) local-titles
title-pair := "propertyName" "propertyType"
```

Note: The *Sheet – Edit Title* command creates the `schSheetTitles` properties within the index schematic (on the instances of the `msymbols`). Properties for global titles are placed on all `msymbol` instances. Properties for local titles are placed only on the instance of the corresponding `msymbol`. When you display a sheet schematic, the `NLPLabels` in the title block are evaluated; these labels use the property values stored in the index schematic.

6. Verify the creation of the `schSheetTitles` property by selecting an instance of `msymbol`.

7. In the CIW, type

```
geQuerySelSet ( )
```

8. Click *OK*.

9. Click *OK* to add the property to the cellview.

Customizing Pin Cross-References

You can customize environment variables for pin cross-references as follows:

- Pin reference format

The `pinRefFormat` environment variable specifies the format for displaying the pin reference information that appears next to each pin when you choose *Sheet – Cross Reference*.

- Pin reference duplicates

The `pinRefDuplicates` environment variable specifies whether duplicate pin references are reported within the same zone of a multisheet schematic when you choose *Sheet – Cross Reference*.

- Pin reference sorting

The `pinRefSorting` environment variable specifies how pin references are sorted when you choose *Sheet – Cross Reference*.

Presetting the Title Block Values

The Schematic Title Block Properties form, available when you choose *Edit – Sheet Title* from a schematic design having a title block, can be preset with values if the properties already exist on the title instance. The title form is dynamic and created every time you want to see the title information. The values in the form are taken from the properties of the title instance.

You can preset the values by adding CDF properties to the title cell and saving the defaults. The properties exist on the instance and will be presented on the form.

Note: See also CDF Commands in the Component Description Format User Guide.

To add CDF properties,

1. From the CIW, choose *Tools – CDF – Edit*.

The Edit Component CDF form appears.

2. Set *CDF Scope* to *Cell*.
3. Set *CDF Layer* to *Base*.
4. Set *Library Name* to *US_8ths*.

Note: You need to have write permission to edit this library.

5. Set *Cell Name* to *Title*.
6. In the *Component Parameter* section set:
 - a. In the *Name* field, type `company`.
 - b. Set *Type* to *string*.
 - c. In the *Prompt* field, type `company`.
 - d. In the *Default Value* field, type the company name you want to see.
 - e. Click *Apply*.
 - f. In the *Name* field, type `title`.
 - g. Set *Type* to *string*.
 - h. In the *Prompt* field, type `title`.
 - i. In the *Default Value* field, type the title name you want to see.

Virtuoso Schematic Editor User Guide

Creating a Multisheet Schematic

- j. Click *Apply*.
 - k. In the *Name* field, type `title1`.
 - l. Set *Type* to *string*.
 - m. In the *Prompt* field, type `title1`.
 - n. In the *Default Value* field, type another title name you want to see.
7. Click *OK* in the Edit Component CDF form to save the added CDF properties.

You now see your default values when you instantiate the title cell.

If you have write permission to the *US_8ths* library, you can also modify the NLPlabels for *company* and *title* from the symbol view of the title cell in the *US_8ths* library to customize the title blocks. You can have default values by adding the variables on the symbol view.

Virtuoso Schematic Editor User Guide

Creating a Multisheet Schematic

Creating Symbols

You can quickly create a new symbol by importing (copying) an existing symbol into the symbol editor window and then modifying the imported symbol. If you cannot use an existing symbol—for example, if your schematic requires an electrical or logical function that does not have a corresponding symbol in a Cadence[®] design library—you either need to create your own symbol using the symbol editor or generate one.

This chapter describes the following topics:

- [Setting Grid Options](#)
- [Importing Symbols](#)
- [Creating New Symbols](#)
- [Creating Symbol Cellviews Using Generators](#)

Setting Grid Options

Before you create a symbol, you might want to set the grids in the symbol window to assist you in drawing. If you want to create shapes that are smaller than what the current snap spacing allows, choose a smaller snap spacing.

To set grid options,

1. From the view, choose *Options – Display*.

The Display Options form appears.

The screenshot shows the 'Display Options' dialog box with the following settings:

- Display Controls:**
 - Background Color: default
 - Net Expression Display: value only
 - Drag Limit: 100
 - Axes: off
 - Cross Cursor: off
 - Object-Sensitive Pop-Up Menus: on
 - Dynamic Highlighting: on
 - Dynamic Net Highlighting: off
 - Connect By Name Highlighting: on
 - Show Direct Connections: off
 - Show Text Justification: off
 - Show Text Indicator: off
 - Ignored Instances: on
 - Display Status Banner: on
 - Coordinates on Status Banner: off
- Grid Controls:**
 - Type: dotted
 - Spacing: 0.125
 - Multiple: 8
 - Snap Spacing: 0.0625
- Zoom / Pan Controls:**
 - Dynamic Zooming: on
 - Mode: Pan To Selected
 - Zoom Scale (%): 90
- Dimming Controls:**
 - Display Dimming: off
 - Mode: Automatic
 - Intensity: 50
 - Grayscale: off

Buttons at the bottom: OK, Cancel, Defaults, Apply, Help.

2. Set the grid controls according to your specifications.
3. Click *OK*.

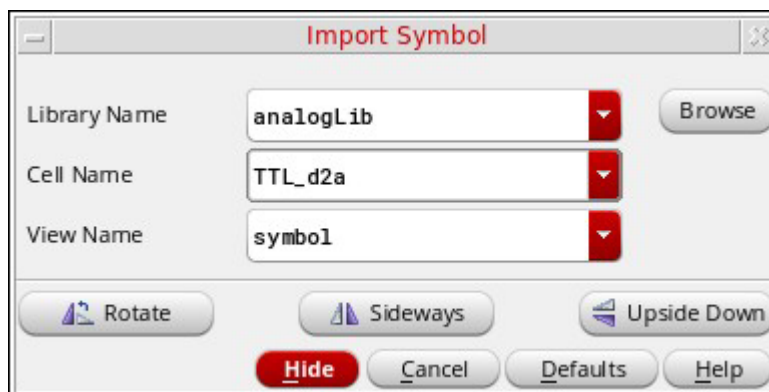
Importing Symbols

You can import (copy) an existing symbol into the symbol window. You can then modify the imported symbol and create a new symbol cellview.

To import an existing symbol,

1. From the symbol window, choose *Create – Import Symbol*.

The Import Symbol form appears.



2. Specify the symbol (in the Library Name, Cell Name, and View Name fields) you want to copy into the symbol design window.
3. Move the pointer into your design and click to place the symbol.

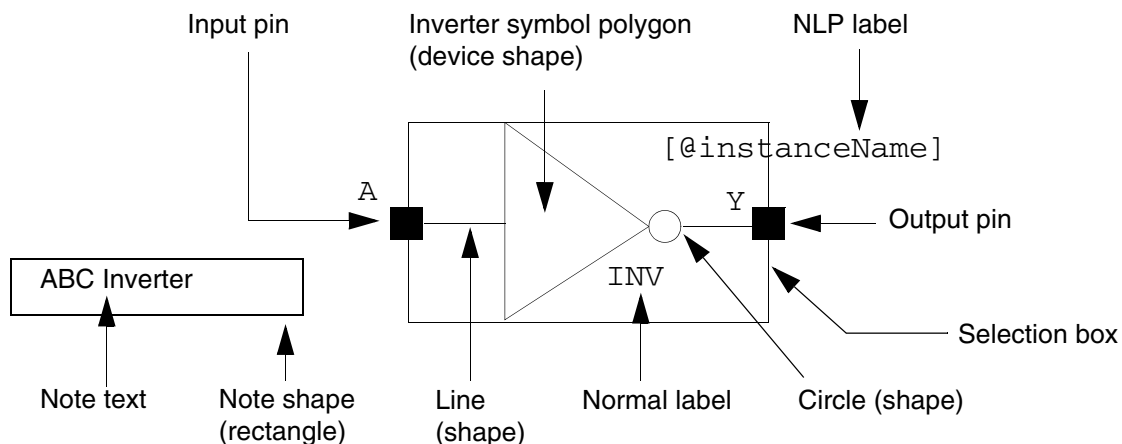
You can now edit the symbol to create a new symbol.

Creating New Symbols

On occasion, your design might require an electrical function that does not have a corresponding symbol in an existing library. In such situations, you need to create your own symbol. This section describes how to create your own symbol using the symbol editor.

You create a symbol by assembling elements in your design such as

- Lines, rectangles, polygons, circles, ellipses, and arcs
- Pins used to connect the symbol to the rest of a schematic
- Labels to identify the symbol and pins
- A selection box to surround a symbol that lets you select an instance of the symbol in the schematic window
- Notes that consist of text and shapes to annotate the symbol.



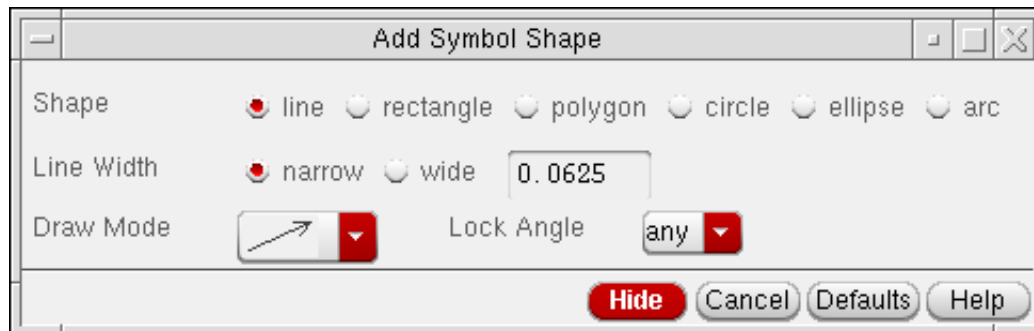
Note: For information on the *Create* toolbar, used as a shortcut to create lines, pins, notes, and so on, see [Quick Reference - Toolbars](#).

Drawing Lines

To draw lines for a symbol,

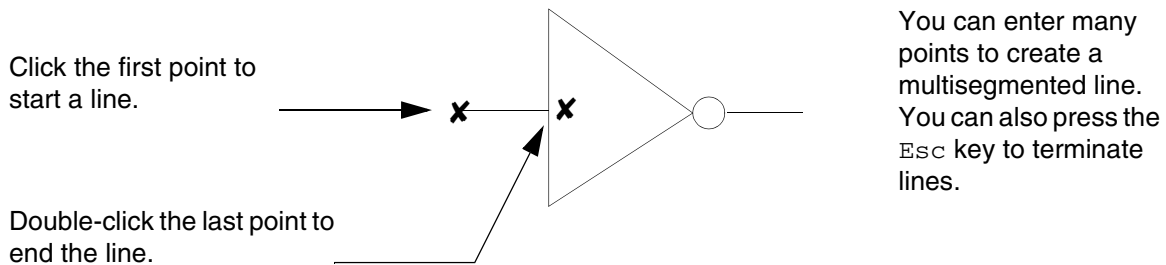
1. From the symbol window, choose *Create – Shape – Line* (or select *Create Line* from the *Create* toolbar).

The Add Symbol Shape form is displayed. If the form does not appear, press F3.



2. Select `line`.
3. Set the options to your specifications.
4. Move the pointer into your design window.

The following example shows a line being added to a triangle shape.



Drawing Rectangles

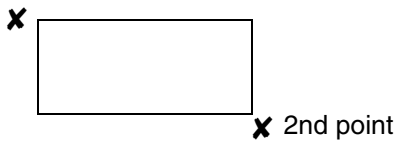
To draw a rectangle,

1. From the symbol window, choose *Create – Shape – Rectangle* (or select *Create Rectangle* from the *Create* toolbar).

The Add Symbol Shape form is displayed. If the form does not appear, press F3.

2. Select `rectangle`.
3. Set the options to your specifications.
4. Move the pointer into your design window and click to enter the first corner of the rectangle.
5. Click the opposite corner.

1st point



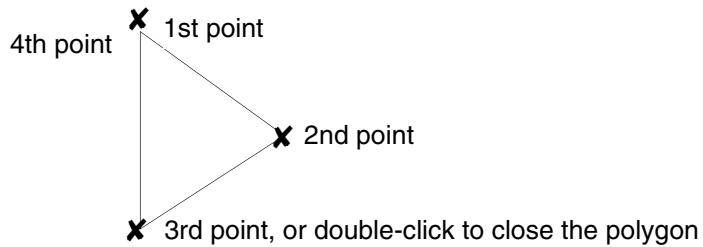
Drawing Polygons

To draw a polygon,

1. From the symbol window, choose *Create – Shape – Polygon* (or select *Create Polygon* from the *Create* toolbar).

The Add Symbol Shape form is displayed. If the form does not appear, press F3.

2. Select `Polygon`.
3. Set the options to your specifications.
4. Move the pointer into your design window and draw the segments that form a polygon symbol by clicking on each point.



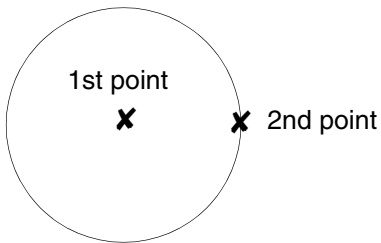
Drawing Circles

To draw a circle,

1. From the symbol window, choose *Create – Shape – Circle* (or select *Create Circle* from the *Create* toolbar).

The Add Symbol Shape form is displayed. If the form does not appear, press F3.

2. Select `circle`.
3. Set the options to your specifications.
4. Move the pointer into your design window, and click to specify the center of the circle.
5. Move the cursor away from the center point, and click to specify the distance of the radius and complete the circle.



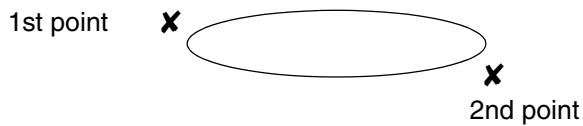
Drawing Ellipses

To draw an ellipse,

1. From the symbol window, choose *Create – Shape – Ellipse* (or select *Create Ellipse* from the *Create* toolbar).

The Add Symbol Shape form is displayed. If the form does not appear, press F3.

2. Select `ellipse`.
3. Set the options to your specifications.
4. Move the pointer into your design window and click one point to begin creation of the ellipse.
5. Click the second point to complete the ellipse.

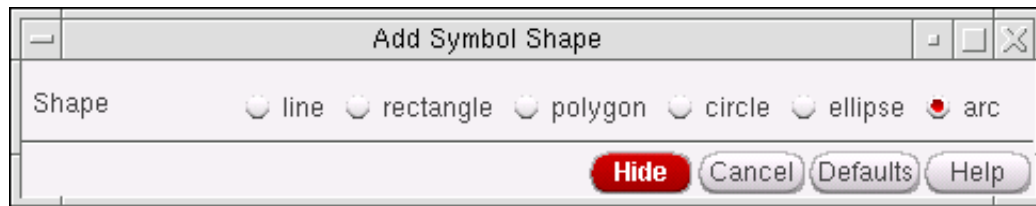


Drawing Arcs

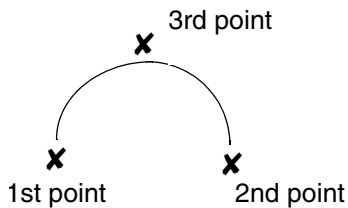
To draw an arc,

1. From the symbol window, choose *Create – Shape – Arc* (or select *Create Arc* from the *Create* toolbar).

The Add Symbol Shape form is displayed. If the form does not appear, press F3.



2. Select `arc`.
3. Move the pointer into your design window and specify three points to create an arc.



Adding Pins as Graphic Images

By default, the symbol pin that you create using the *Create – Pin* command in the symbol editor is a graphic image of a symbol master.

To add a symbol pin,

1. From the symbol window, choose *Create – Pin*.

The Create Pin – Symbol form appears.

Create Pin

Names

Direction

Usage

Signal Type

☐ Expand buses

☐ Place multiple pins

▼ **Net Expression** ☐ Attach to pin

☐ Derive from pin name

Property Name

Default Net Name

Font Height

Style

Justification

▼ **Supply Sensitivity**

Attach To

Power Terminal

Ground Terminal

Rotation

2. In the *Pin Names* field, type the pin names.

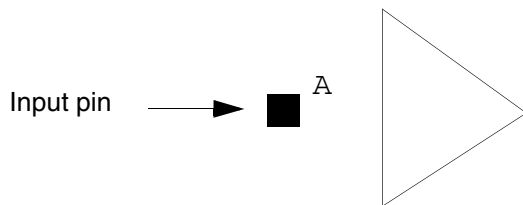
Use a space to separate each name.

3. In the *Direction* cyclic field, choose the pin direction.
4. In the *Type* cyclic field, choose the pin type.
5. Specify the *Label Offset* value (in user units) and the *Label Location*.
6. Optionally, click the *Show Sensitivity >>* button at the bottom of the form to expand the form to reveal three additional field options: *Attach Sensitivity*, *Sensitive To Power Terminal*, and *Sensitive To Ground Terminal*.

Note: For more information on sensitivity options see [Create Pin – Symbol](#).

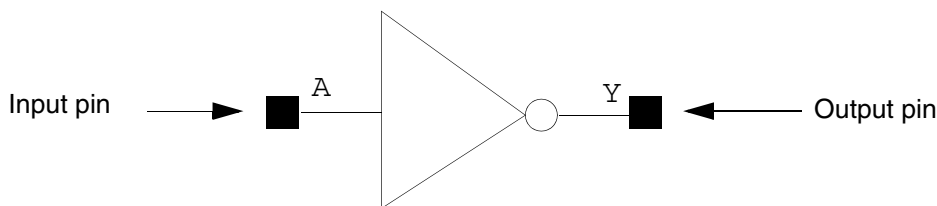
7. Move the pointer into the cellview and click to place the first pin.

The following example shows how an input pin with a pin name of **A** appears after you click your mouse in the design.



8. Change the *Direction* cyclic field, if necessary, before placing a subsequent pin.
9. Move the pointer into the design window and click to place pin **Y**.

The following example assumes you changed the pin direction.



Adding Pins as Instances

When you add symbol pins as instances, any changes you make to the graphics of the symbol pin master are reflected in all the pin instances that were previously added from the symbol pin master.

To turn on the option for adding symbol pins as instances,

1. From the view, choose *Options – Editor*.

The Editor Options form appears.

Editor Options

Commands | Connectivity | Inherited Conn. | Labels | Ignore Properties

Command Controls

Add Instance Browser Type: ☒ library ☐ component

Show Schematic Views in Browser: ☒ off ☐ on

Block Pin Direction Rule: ☒ edge ☐ order of entry

Wire Snapping: ☐ off ☒ on

Automatic Solder Dots: ☐ off ☒ on

Solder Dot Wire Overlap:

Symbol Generator Pin Sorting: ☐ alphanumeric ☒ geometric

Add Symbol Pins as Instances: ☒ off ☐ on

Repeat Commands: ☐ off ☒ on

Cross Selection: ☒ off ☐ on

Gravity: ☐ off ☒ on

Minimum Marker Size:

View Name List:

Auto Wire Stub for Bus Terminals: ☒ wide ☐ narrow

Constraint Notes Controls

Auto Constraint Notes: ☐ enable ☒ disable

Sync on Design Open: ☐ enable ☒ disable

Sync on Design Check: ☐ enable ☒ disable

Buttons:

2. In the *Commands* tab, turn on the *Add Symbol Pins as Instances* option.
3. Click *OK*.
4. Refer to [Adding Pins as Graphic Images](#) to create a symbol pin.

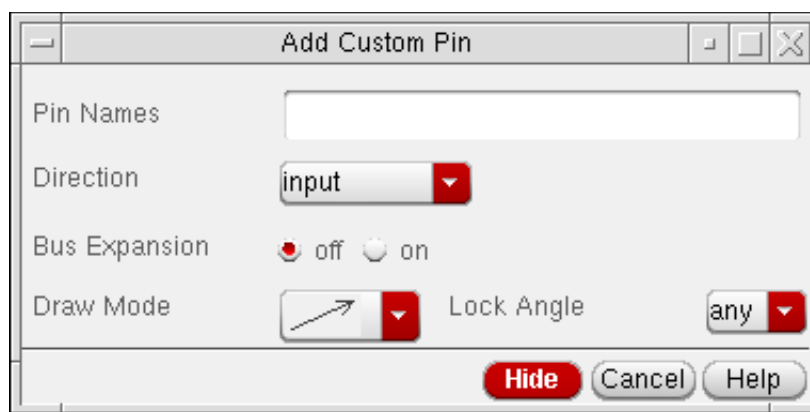
Creating Custom Pins as Graphic Images

You can use the *Create – Custom Pin* command to create your own graphic images (polygons) for the pins you want to place in your design.

To create and place a new custom pin,

1. From the symbol window, choose *Create – Custom Pin*.

The Add Custom Pin form appears.



2. In the *Pin Names* field, type the name of the custom pin.
3. In the *Direction* cyclic field, choose the pin direction.
4. In your design window, click the first point and each subsequent point of the polygon shape that you want to create.
5. Double-click the last point to close the polygon.

The pin color is filled in and you are prompted to use the Edit Object Properties form to display the pin name. By default, the pin names are not visible.

6. Press the `ESC` key.

Making the Custom Pin Name Visible

To make the custom pin name visible or to edit other custom pin properties,

1. From the symbol window, choose *Edit – Properties – Objects*.

The Edit Object Properties form appears.

2. In the design window, click the custom pin.

The *Value* field displays the name you gave to your custom pin.

3. Set the *Display* cyclic field to *value*.

4. Click *OK*.

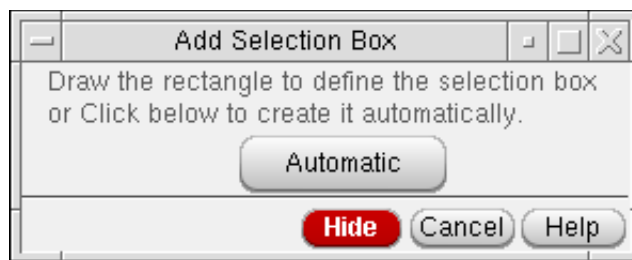
Adding a Selection Box

A symbol selection box defines the area of the symbol that you can use when selecting an instance of that symbol from the schematic window (it is also used in pre-selection).

To create a selection box,

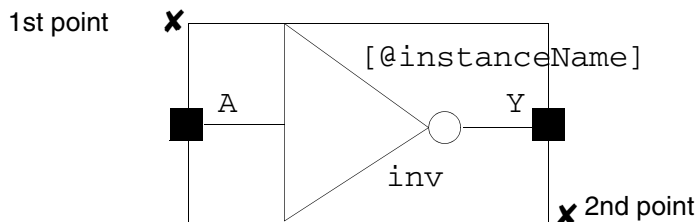
1. From the symbol window, choose *Create – Selection Box*.

The Add Selection Box form appears.



2. Draw a selection box manually, or click *Automatic*.

- ❑ To draw the selection box manually, click the first corner of the rectangle, then move the cursor to the opposite corner and click.



- ❑ To draw the box automatically, click *Automatic*. The software generates a selection box that encloses all symbol pins and symbol shapes.

Note: You can use the Stretch command to adjust the box shape.

Adding Labels

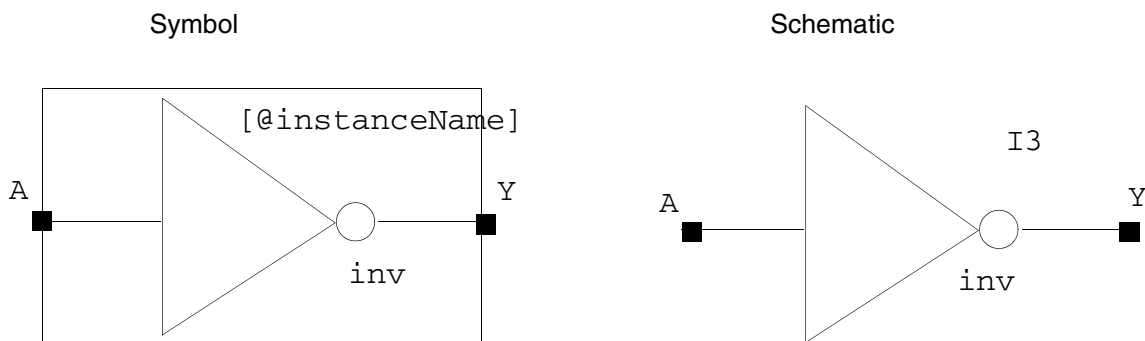
Symbol labels display assorted textual information. You can add one of three types of labels to your symbol:

1. Normal labels, which have fixed text

This type of label is used to identify the device and pins.

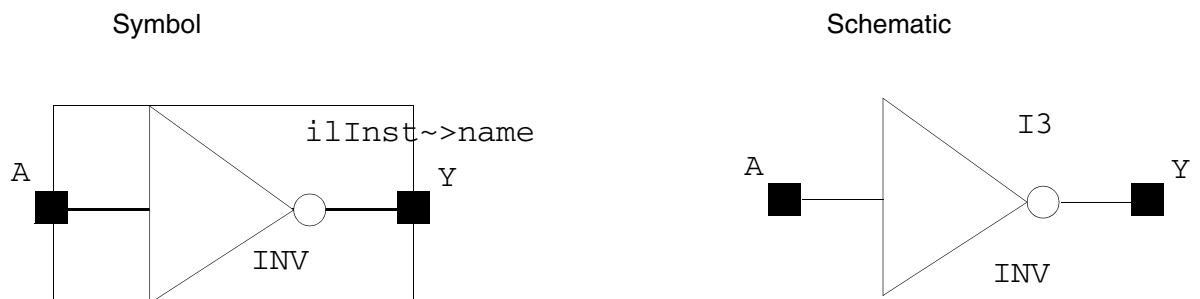
2. NLP labels, which have expressions that the Virtuoso® Schematic Editor evaluates to determine the value

When you place an instance in your schematic, the system evaluates the NLP expression and displays the resulting value. For example, the system displays the NLP label `[@instanceName]` in the symbol window but displays a value—for example, `I3`—in the schematic window.



3. IL labels, which have expressions that are evaluated by the SKILL language to determine the value

When you place an instance in your schematic, the system evaluates the SKILL expression and displays the resulting value. For example, the system displays the IL label `ilInst~>name` in the symbol window, but displays a value—for example, `I3`—in the schematic window.

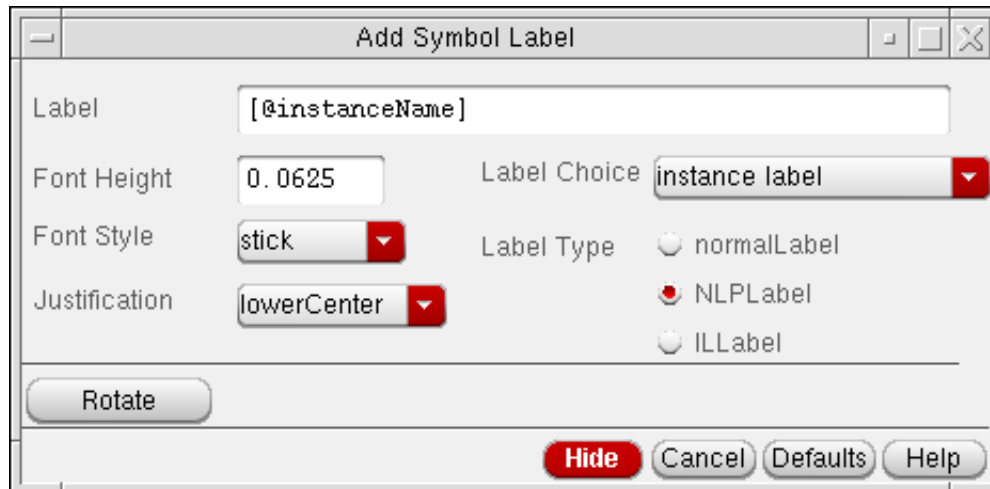


Adding a Normal Label

To add a normal label,

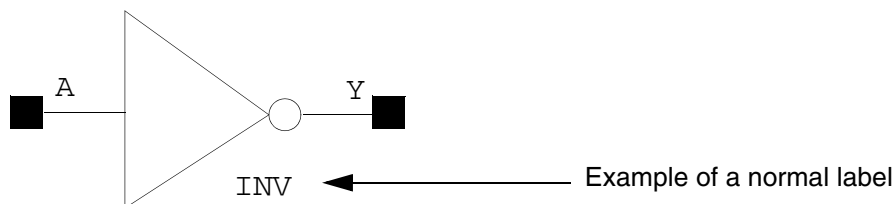
1. From the symbol window, choose *Create – Label*.

The Add Symbol Label form appears.



2. In the *Label* field, type the name of your symbol.
3. In the *Label Choice* cyclic field, choose *logical label*.
4. In the *Label Type* field, click *normalLabel*.
5. Move the pointer into your design window.

The *normalLabel* expression is attached to the pointer.



6. Click to place the label.

When you place an instance of the symbol in your design, the schematic editor displays the label text. To force an evaluation of the SKILL expression or function and display the results, use the *View – Redraw* command.

Adding an NLP Label

To add an NLP label,

1. From the symbol window, choose *Create – Label*.

The Add Symbol Label form appears.

2. In the *Label* field, do one of the following:

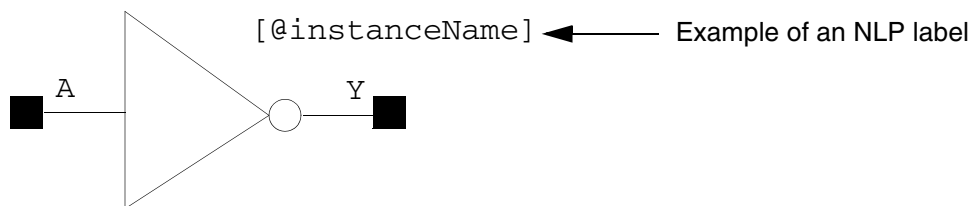
- ☐ Accept the default, `[@instanceName]`
- ☐ Enter your own expression.

3. In the *Label Type* field, click *NLPLabel*.

The label choice affects the color of the label.

4. Move the pointer into your design window.

The *NLPLabel* expression is attached to the pointer.



5. Click to place the label next to the symbol.

When you place an instance of a symbol in a schematic cellview, the schematic editor interprets the `[@instanceName]` label and displays the current value of the property. To force an evaluation of the SKILL expression or function and display the results, use the *View – Redraw* command.

The schematic editor evaluates NLP labels using netlisting rules.

Example – Syntax for an NLP Label Expression Containing One Property

```
[@propertyName : displayFormat1 [: displayFormat2 ] ]
```

In the example, *displayFormat1* is used when a specified property exists, and *displayFormat2* (optional) is used when a specified property cannot be located.

If *displayFormat2* is not defined and the property does not exist, the NLP label is not displayed.

Virtuoso Schematic Editor User Guide

Creating Symbols

The rules table below and the examples that follow the table show how NLP label expressions are interpreted.

Syntax Expression	Definition
[]	Delimits the property substitution expression. Text outside square brackets remains unevaluated.
@	Specifies the beginning of a property substitution expression, as [<i>@instanceName</i>].
<i>propertyName</i>	Specifies the property name.
:	Separates the property name from the display format string. For example, in [<i>@w:w=%</i>], the colon separates the property name (<i>w</i>) from the expression (<i>w=%</i>).
%	A placeholder in the format string for the current value of <i>propertyName</i> .
\ plus a reserved character	Lets you include one of the above reserved characters as a text character in your label.
\n	Generates a carriage return.
\t	Generates a tab.
\nnn	Substitutes an ASCII character defined by octal <i>nnn</i> .

Examples – Evaluating NLP Label Expressions

If the current value of a named property is defined, the schematic editor substitutes the defined value for the percent character (%) in the *NLPLabel* expression when you place the instance in the design.

If the schematic editor does not find a value for a property when you place an instance of a symbol, *displayFormat2* is used instead.

A label expression represents the current value of a property.

■ Example 1 – Basic expressions

```
[@w: %]
```

or

```
[@w]
```

If you reference only one property in the NLP label expression, you specify only the name of the property. When evaluated, the schematic editor returns the value of the property. When *w* has a value of 8, the editor generates the following label:

```
8
```

■ Example 2 – Simple expressions

```
[@w: w= %]
```

The schematic editor considers any nonreserved characters after the separator (:) to be a text string. When evaluated, the schematic editor returns the string. If there is a special character in the expression, it is also evaluated. When *w* has a value of 8, the following label is generated:

```
w=8
```

■ Example 3 – Compound expressions

```
[@w: w= %; ] [@l: l= %; ]
```

If *w* is not defined and *l* has a defined value of 8, the following label is generated:

```
l=8;
```

If *w* has a value of 4 and *l* has a value of 8, the following label is generated:

```
w=4; l=8;
```

If you specify a value for *w* and *l* after you place the instance, or if you change the value after you place the instance, the label is updated to display the current value.

■ Example 4 – Combined label expressions

```
[@w: w= %: w is not defined; ]
```


Virtuoso Schematic Editor User Guide

Creating Symbols

If `w` is not defined, the schematic editor goes on to the next label expression, `w is not defined`, which is a text string. The string is evaluated as a value and the result is displayed as a label. The semicolon is part of the text string.

`w is not defined;`

■ Example 5 – Nested label expressions

`[@w:w=%;[@l:l=%]]`

The schematic editor returns the values for both properties if they are both defined. For example, if the value of `w` is 4 and the value of `l` is 8, the following label is generated:

`w=4;l=8`

Adding an IL Label

To display the value of an instance property, you can add an IL label using a Cadence® SKILL language expression that can be evaluated by the SKILL language.

To add an IL label,

1. From the symbol window, choose *Create – Label*.

The default `[@instanceName]` label is attached to the cursor.

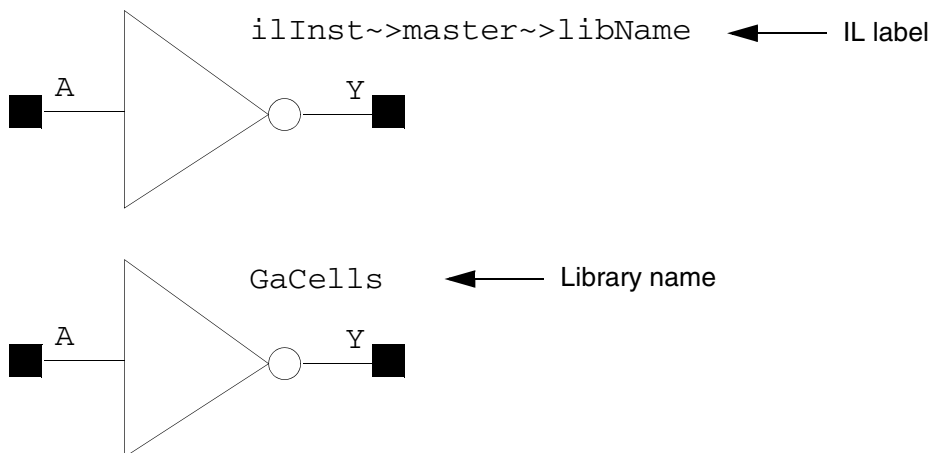
The Add Symbol Label form appears.

2. In the *Label* field, type a SKILL expression or function, with the form

```
ilInst~>master~>libName
```

ilInst is a SKILL variable used to provide self-reference to an instance during graphic display of the cellview containing the instance. Note also the SKILL variable ilInstPath, which works like ilInst, but is evaluated for the instances at all display levels. You should use ilInstPath, rather than ilInst, when instantiating schematics in schematics.

3. Set the *Label Choice* cyclic field to *instance label*.
4. In the *Label Type* field, click *ILLabel*.
5. Move the pointer into your cellview and click to place the IL label.



□ Example 1 – SKILL code

```
ilInst~>cellView~>fileTimeStamp
```

The SKILL code displays a time stamp; for example,

```
Jun 5 15:07:33 2000
```

Store Default of CDF Parameter

When *Store Default* is "Yes" for a CDF parameter of a cell, the parameter value is stored on instance of the cell when instance is created. Therefore, in this case the CDF parameter does have a default value but stored on instance. When *Store Default* is "No" for a CDF parameter of a cell, the parameter value is stored in Cell CDF, and not on instances of the cell in higher level schematics.

However, irrespective of the *Store Default* setting, if you modify parameter value on an instance, the parameter value is then stored on instance, even if the modified value is the default value.

Effect of Netlisting Mode on netSets

When the CDS_Netlisting_Mode is *Analog*, the property lookup includes lookup in database(OA) and CDF. For netSet properties set via CDF, the CDF property settings are used rather than those in the database.

Even if the CDF *use* callback's expression evaluates to *nil* to specify that the parameter for a netSet is not available for display, the schematic editor intentionally ignores this setting and uses the CDF property setting specified.

IL v. NLP labels

NLP labels are used to lookup properties using database properties (note that sometimes database lookup does include CDF lookup, for more information see [Effect of Netlisting Mode On Labels](#)). IL labels are used to lookup properties using CDF properties. For example, "cdsParam(1)" ILLabel will display the value of first CDF parameter setup in *Interpreted labels* section of CDF.

Effect of Netlisting Mode On Labels

The effect of netlisting mode on labels is as follows:

When CDS_Netlisting_Mode is "Analog"

Property lookup includes lookup in database(OA) as well as in CDF. Therefore, if your NLPLabel/ILLabel refers to a CDF parameter, it is displayed on the instance whether it has a default value or an overriding value on the instance.

When CDS_Netlisting_Mode is "Digital"

Property lookup does not include lookup in CDF. Therefore, if your NLPLabel/ILLabel refers to a CDF parameter, and the value of the CDF parameter is not stored on an instance, the labels are not displayed on the instance.

In your CIW you can change the netlist mode:

```
setShellEnvVar("CDS_Netlisting_Mode=Digital")
cdsSetNetlistMode()
```

or

```
setShellEnvVar("CDS_Netlisting_Mode=Analog")
cdsSetNetlistMode()
```

NLPLabels that Display Non-default Value of CDF Parameters

NLPLabels use only database properties for lookup, however, when netlisting mode is "Analog", database lookup includes CDF lookup, and hence NLPLabel can display the value of CDF parameters. But when netlisting mode is "Digital", any NLPLabels referring to CDF parameters will not display CDF parameter value. However, when a CDF parameter value is modified on an instance, the modified value is stored on the instance where NLPLabel lookup can find it. Similarly, if a CDF parameter has *Store Default* value as "YES", the parameters are always stored on the instance where NLPLabel lookup can find it.

So you can create an NLPLabel which refers to a CDF parameter, such that when netlisting mode is digital, the label will display only non-default values of the parameter on instances.

To set this:

1. From the CIW, select *Tools – CDF – Edit* to display the Edit Component CDF form.

Note: For more information on this form see the [CDF Commands](#) chapter in the *Component Description Format User Guide*.

2. Enter the *Library Name*.
3. Enter the *Cell Name*.
4. Set *CDF Layer* to *Base*.
5. In the *Component Parameters* section of the form scroll to the parameter of interest.
6. Once selected, set *Store Default* to *no*.
7. Click *Apply*.
8. Repeat steps 6-8 for all parameters that have defaults that you do not want displayed on the schematic.

Virtuoso Schematic Editor User Guide

Creating Symbols

Note: You may also have to delete the local properties on the instance:

9. Select the instance in your schematic.
10. In the CIW enter `geQuerySelSet()`
11. Delete the local properties on your instance.

Additionally you will need to correctly set up your NLPLabels on your symbols. For example, the label `p1: [@p1: %: ]`, should be set to `[@p1: p1= %]`.

12. Set CDS_Netlisting_Mode to "Digital" as explained above.

Notice that NLPLabels on instances that have default values of CDF parameters disappear. If you modify CDF parameters on an instance now, the labels appear on that instance displaying the value of the parameter.

Adding Note Text and Note Shapes

You add note text and note shapes in the symbol window in the same way you add notes in the schematic window. Refer to [Adding Note Text](#).

Adding a Net Expression Label

Select *Create – Net Expression* to add a net expression label in the symbol window in the same way you add a net expression label in the schematic window. Refer to [Adding Net Expression Labels to Create an Inherited Connection](#).

Setting the Symbol Origin

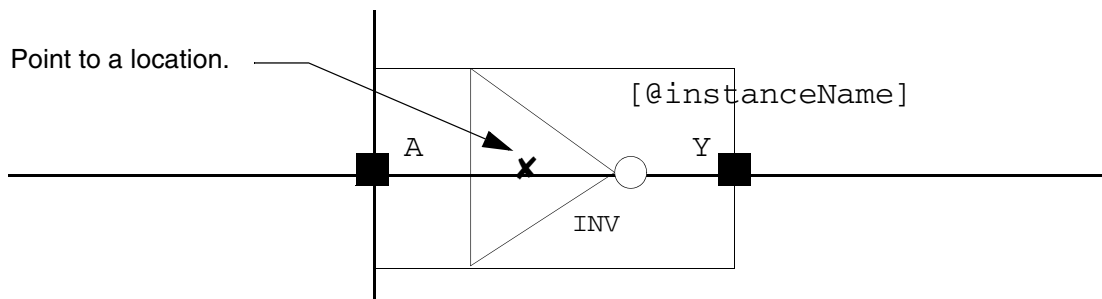
The origin of a symbol is the (0,0) location within a symbol. When you place instances in a schematic, the symbol origin affects the position of the symbol graphics and pins relative to the instance placement point.



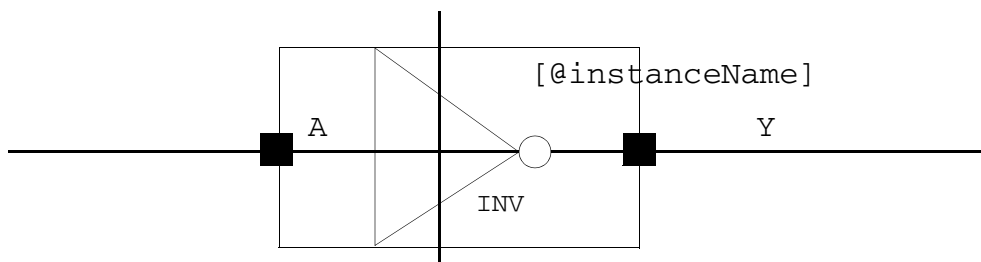
Changing the origin of a symbol moves instance pins within all schematics that use that symbol. This might affect the connectivity of the schematic because schematic wires do not automatically move to the new pin locations.

To set the origin of a symbol,

1. From the symbol window, choose *Edit – Origin*.
2. Point to a location for the new origin of the symbol.



Pointing to the middle of the symbol moves the symbol graphic, repositioning the $X=0$, $Y=0$ origin point of the symbol to the new location.



Note: When you use this command, it might be helpful to turn on the display of the following:

- ☐ X and Y axes
- ☐ Coordinates in the status banner

3. Choose *View – Fit* to view the enlarged symbol, if necessary.

Creating Symbol Cellviews Using Generators

A symbol cellview is an alternative view of the schematic cellview.

- Besides using the *File – Create Cellview* command mentioned in [Automatically Creating Cellviews](#) you can create symbol cellviews using generators.

- Text-to-Symbol Generator

TSG is a Cadence application program that automatically generates symbol cellviews for schematic capture and subsequent simulation processes. TSG provides a quick way to generate a symbol from a list of pins in a TSG file.

- Symbol and Simulation Library Generator

The Cadence SKILL-based symbol and simulation library generator (S/SLG) is a library management program that generates symbol and simulation views.

Virtuoso Schematic Editor User Guide

Creating Symbols

Automatically Creating Cellviews

This chapter discusses the following topics:

- [Features of Creating Cellviews](#)
- [Automatically Creating a Cellview from Another Cellview](#)
- [Automatically Creating a Cellview from a Pin List](#)
- [Automatically Creating a Cellview from an Instance](#)
- [Creating a Cellview from the Canvas \(Make Cell\)](#)
- [Editing Symbol Generation Options](#)
- [Specifying Symbol Generator Pin Sorting](#)

 Important

For more information on cellview creation, see also [VHDL In Forms in the VHDL In](#) for *Virtuoso Design Environment User Guide and Reference*.

Features of Creating Cellviews

You can automatically

- Create a new cellview from an existing cellview. See [Automatically Creating a Cellview from Another Cellview](#).
- Create a new cellview from a list of port/pin names. See [Automatically Creating a Cellview from a Pin List](#).
- Create a new cellview from an existing instance. See [Automatically Creating a Cellview from an Instance](#).
- Create a new cellview from the canvas by using Make Cell. See [Creating a Cellview from the Canvas \(Make Cell\)](#).
- Generate circuit symbols for schematic capture and subsequent simulation processes
You can automatically generate a symbol from a list of pins in a text-to-symbol generator (TSG) file. Refer to the in [Text-to-Symbol Generator](#).
- Generate symbol and simulation views. Refer to the Cadence® SKILL-based library management program in [Symbol and Simulation Library Generator](#).

The software uses the TSG program to generate a symbol when you choose either *Create – Cellview – From Cellview* or *Create – Cellview – From Pin List*.

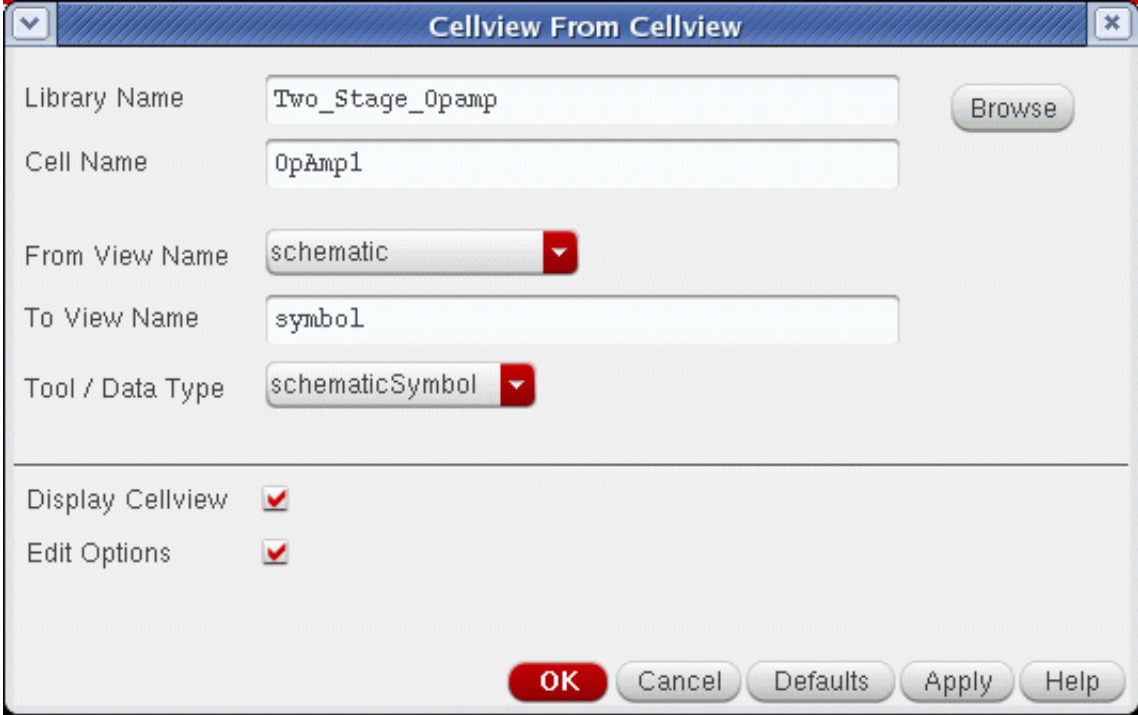
Automatically Creating a Cellview from Another Cellview

You can automatically create a new cellview from an existing cellview. For example, you can automatically create a symbol from a schematic. Automatic cellview creation can be useful for both top-down and bottom-up design approaches.

To create a cellview from another cellview,

1. From the view, choose *Create – Cellview – From Cellview*.

The Cellview From Cellview form appears.



By default, the software initializes the form with information about the cellview currently open (the default cellview).

2. Use a different cellview as the source of the port/pin information by changing the *Library Name*, *Cell Name*, and *From View Name* fields.
3. Change the *To View Name* or *Tool / Data Type* field to change the cellview to be created.

Note: If either the *From View Name* cyclic field or the *Tool / Data Type* cyclic field does not contain the desired view or conversion function, you must modify the `schViewToPinListReg` list or the `schPinListToViewReg` list, which are defined in

Virtuoso Schematic Editor User Guide

Automatically Creating Cellviews

your `schConfig.il` file, which is described in *Virtuoso Design Environment User Guide*.

4. Display the cellview in a new window by turning on the *Display Cellview* button.
5. Edit various cellview creation options by turning on the *Edit Options* button.
6. Click *OK*.

When generating a symbol, the Symbol Generation Options form appears if you turned on the *Edit Options* button and the cellview is being newly created or replaced.

If a cellview already exists with the same name, a dialog box prompts you to either replace the cellview, modify it (symbol cellview only), or cancel the command.

From this point, the exact behavior of the command depends on the type of data being created.

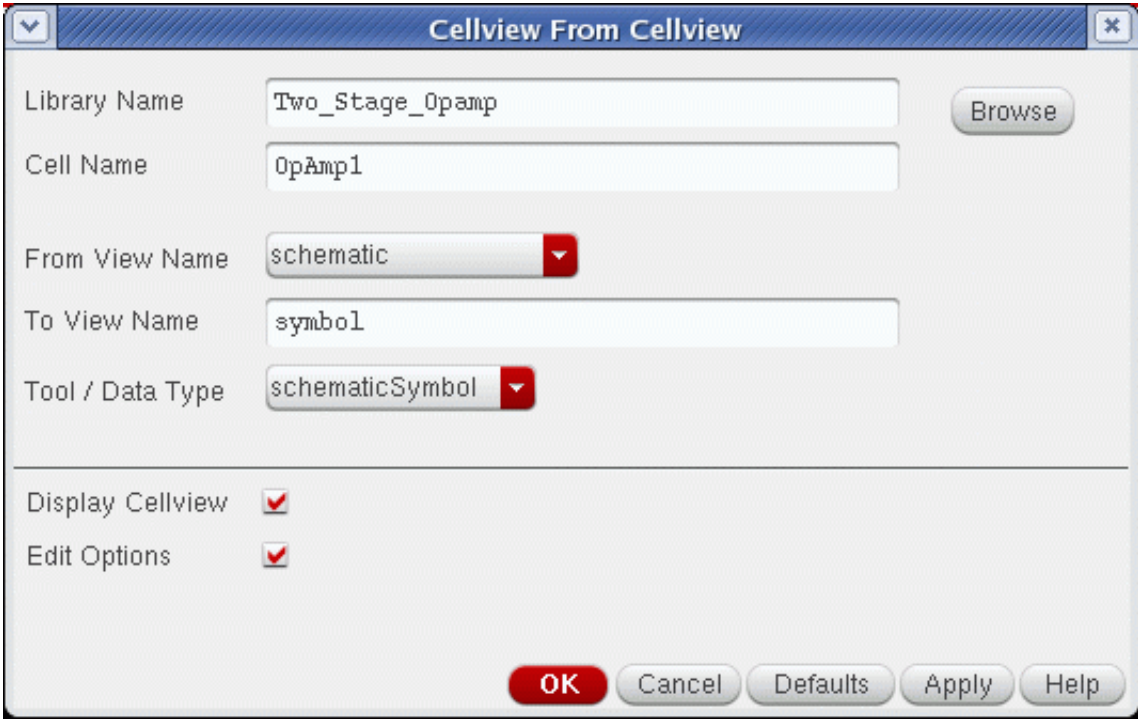
Note: If a symbol view has been created from a schematic view that contains nets that have a defined `sigType` property (by default “`signal`”) and pins, then each symbol pin will inherit a `sigType` property that has the same value of the corresponding net.

Replacing an Existing Symbol Cellview

You can replace or modify existing symbol cellviews. You can only replace (not modify) all other types of existing cellviews.

1. From the schematic window, choose *Create – Cellview – From Cellview*.

The Cellview From Cellview form appears.



2. In the *To View Name* field, type `symbol`.
3. Click *OK*.

If the symbol cellview to be created already exists, the Create Cellview dialog box appears.



Replacing the symbol cellview does the following:

- ☐ Discards any modifications you may have made to that cellview
- ☐ Creates only the pins, properties, selection box, labels, and other graphics; all schematic instances, schematic wires, and HDL body text are lost

Virtuoso Schematic Editor User Guide

Automatically Creating Cellviews

- ☐ May alter the position of pins in the new symbol cellview

4. Click *Replace*.

The Symbol Generation Options form appears.

The screenshot shows the 'Symbol Generation Options' dialog box. It has a title bar with standard window controls. The dialog is divided into several sections. At the top, there are three text boxes: 'Library Name' containing 'amsPLL', 'Cell Name' containing 'TopPllWithDC', and 'View Name' containing 'symbol'. Below these is a section titled 'Pin Specifications' with a sub-header 'Attributes'. It contains four rows of text boxes for 'Left Pins', 'Right Pins', 'Top Pins', and 'Bottom Pins'. The 'Left Pins' box contains 'ibias refclk reset', and the 'Right Pins' box contains 't2 p0 p45 p90 p135 p180 p225 p270 p315 vcoclk vcom vcop'. Each of these four rows has a 'List' button to its right. Below the pin specifications is a section titled 'Exclude Inherited Connection Pins:' with three radio buttons: 'None' (selected), 'All', and 'Only these:'. The 'Only these:' option has an empty text box next to it. At the bottom of the dialog, there are four checkboxes: 'Load/Save', 'Edit Attributes', 'Edit Labels', and 'Edit Properties', all of which are currently unchecked. At the very bottom right, there are four buttons: 'OK' (highlighted in red), 'Cancel', 'Apply', and 'Help'.

5. Click *OK*.

The Overwrite Base Cell CDF dialog box appears.

The existing cellview is deleted and a new one is created.

6. Click *Yes*.

Modifying an Existing Symbol Cellview

You can replace or modify existing symbol cellviews. You can only replace (not modify) all other types of existing cellviews.

1. From the view, do one of the following:

- ☐ Choose *Create – Cellview – From Cellview*.

The Cellview From Cellview form appears.

The screenshot shows the 'Cellview From Cellview' dialog box. The title bar is 'Cellview From Cellview'. The dialog contains the following fields and controls:

- Library Name:** A text field containing 'zambezi45' with a dropdown arrow and a 'Browse' button to its right.
- Cell Name:** A text field containing 'LP_pll_dig_combo' with a dropdown arrow.
- From View Name:** A text field containing 'schematic' with a dropdown arrow.
- To View Name:** A text field containing 'symbol'.
- Tool / Data Type:** A text field containing 'schematicSymbol' with a dropdown arrow.
- Display Cellview:** A checkbox that is checked.
- Edit Options:** A checkbox that is checked.
- Buttons:** At the bottom, there are five buttons: 'OK' (highlighted in red), 'Cancel', 'Defaults', 'Apply', and 'Help'.

- ☐ Choose *Create – Cellview – From Pin List*.

Virtuoso Schematic Editor User Guide

Automatically Creating Cellviews

The Cellview From Pin List form appears.

Cellview From Pin List

Input Pins

Output Pins

IO Pins

Switch Pins

Jumper Pins

Tristate Pins

Unused Pins

Library Name: zambezi45

Cell Name: LP_pll_dig_combo

View Name: schematic

Tool / Data Type: schematic

Display Cellview: ☒

Edit Options: ☒

Buttons: OK, Cancel, Defaults, Apply, Help

2. Click *OK*.

If the symbol cellview to be created already exists, the Create Cellview dialog box appears.

Note: Before clicking *Modify*, be aware that modifying a symbol cellview will:

- ☐ automatically edit the cellview with a minimal amount of changes to the current pin locations, graphics, and properties within the symbol cellview.
- ☐ delete pins that are no longer needed.
- ☐ add pins that do not yet exist in the new cellview.
- ☐ add pins near the bottom of the cellview. Use the symbol editor to reposition these new pins and to adjust any symbol graphics. The system determines attributes of

Virtuoso Schematic Editor User Guide

Automatically Creating Cellviews

the new pins by the currently active TSG template file (see [Loading and Saving a TSG Template File](#)).

- ☐ add or update cellview properties and terminal properties.
- ☐ annotate the changes that were made in note format at the bottom of the cellview and in the Command Interpreter Window (CIW).
- ☐ not delete cellview or terminal properties.

3. Click *Modify*.

The symbol cellview is automatically edited and saved.

The Overwrite Base Cell CDF dialog box appears.

4. Click *Yes* to update the base cell CDF with new simInfo port information (see Figure 6-3).

Examples of Replacing and Modifying Pins on a Symbol

The following three figures show the difference between choosing *Replace* and *Modify*.

- Figure 6-1—The symbol
- Figure 6-2—The symbol after choosing *Replace*
- Figure 6-3—The symbol after choosing *Modify*

Figure 6-1 illustrates an original symbol that was created using *Create – Cellview – From Cellview* from a schematic that had the pins { a bb ccc dddd eeeee f gg hhh iii k ll mmm }.

The following changes were then made to the schematic:

- Deleted input pin bb
- Deleted inputOutput pin iii
- Added input pins wwwwww and zz
- Added inputOutput pin ii

All three figures show pin a at the exact same location, $(X, Y) = (0, 0)$ which is the origin point of the symbol. By default, TSG puts the top-most left pin at the origin of the symbol.

Figure 6-1 The Symbol

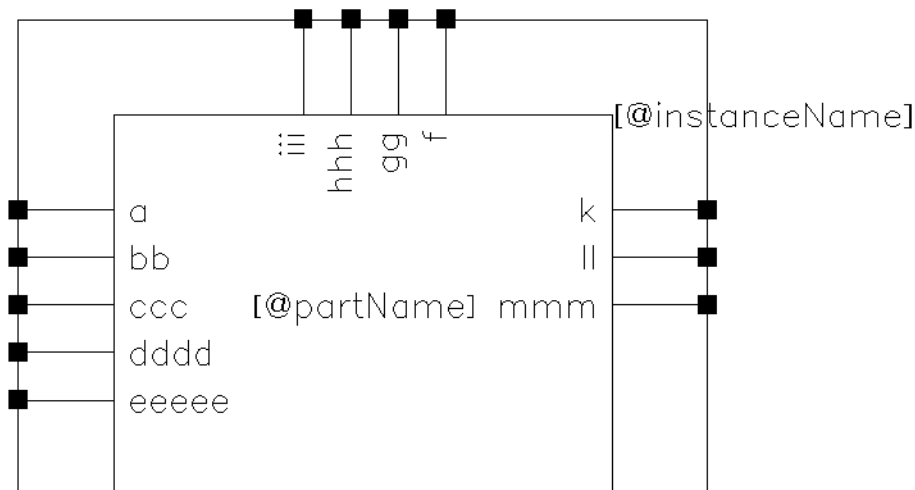
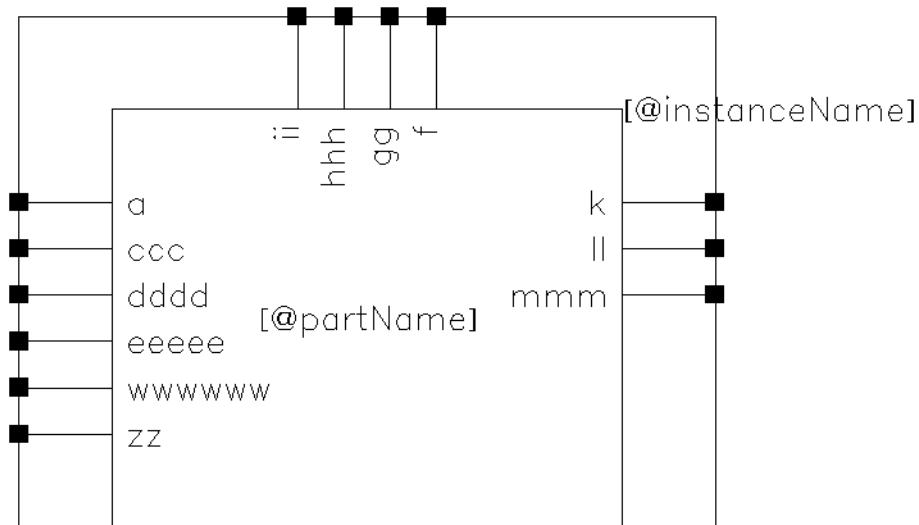


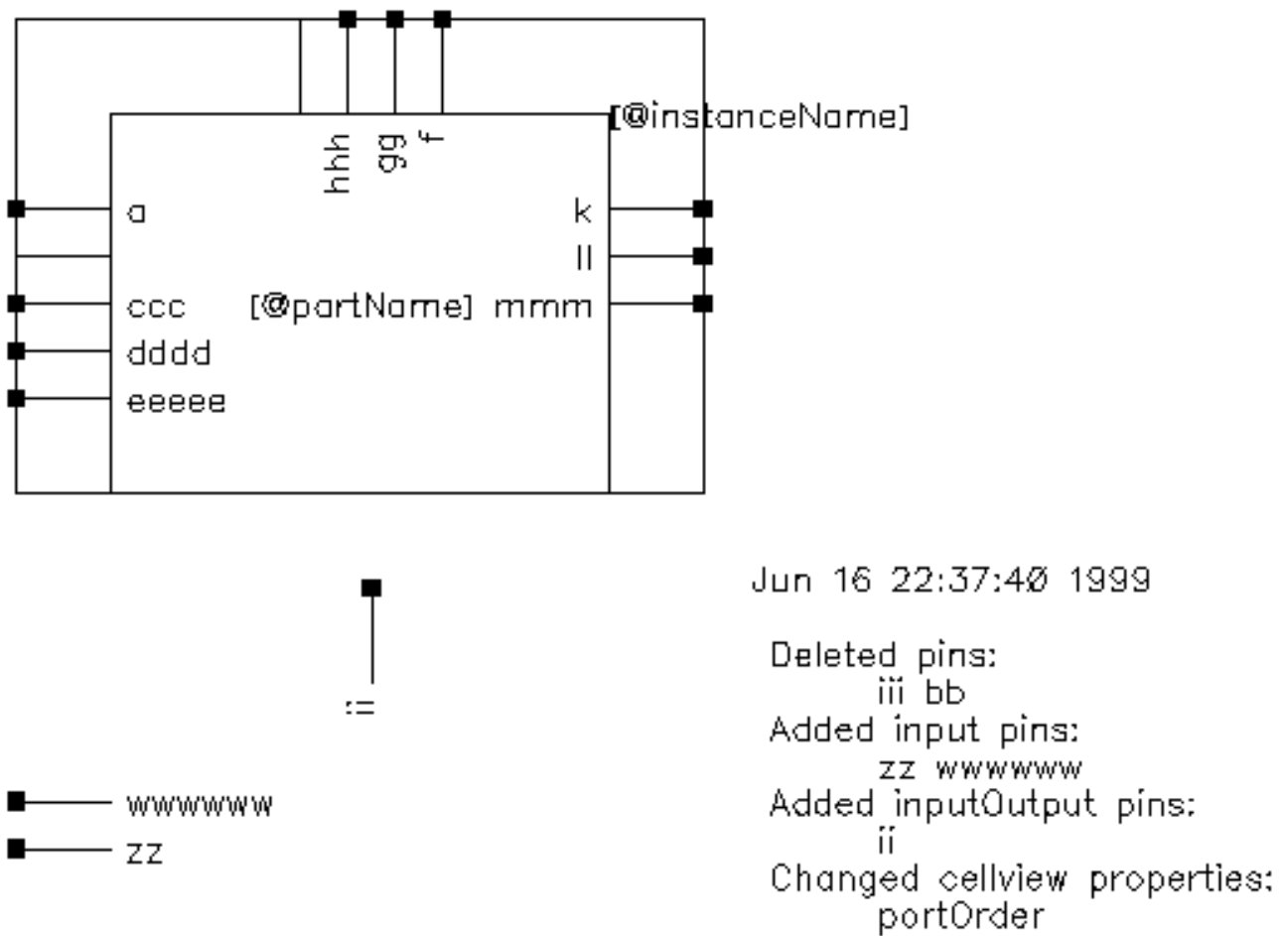
Figure 6-2 The Symbol after Choosing Replace



After a *Replace*, only pin *a* is in the exact same position as the original pins. All other unaffected pins are in new locations for the following reasons:

- Pins *ccc*, *dddd*, and *eeee* have moved up by one grid space, which was freed up by the deletion of pin *bb*.
- Although the ordering and number of pins along the top is similar to the original (with new pin *ii* replacing pin *iii*), all four top pins have moved down by one-half a grid (due to the shorter label).
- Due to the length of new or deleted pin names, the graphic has been recalculated in size accordingly.

Figure 6-3 The Symbol after Choosing Modify



The modified cellview has all of the unaffected pins {a ccc dddd eeeee f gg hhh k ll mmm} at their original locations. The new pins {www, zz, ii} have been placed at the bottom of the cellview, outside of the bounding box of the original symbol. The pin and pin names {bb, iii} have been deleted, but the graphic pin line (pin stub) remains to show the position of the pins that were deleted. If necessary, you can delete these graphic pin lines using the symbol editor. Also shown is the note used to describe the changes that were made to the symbol.

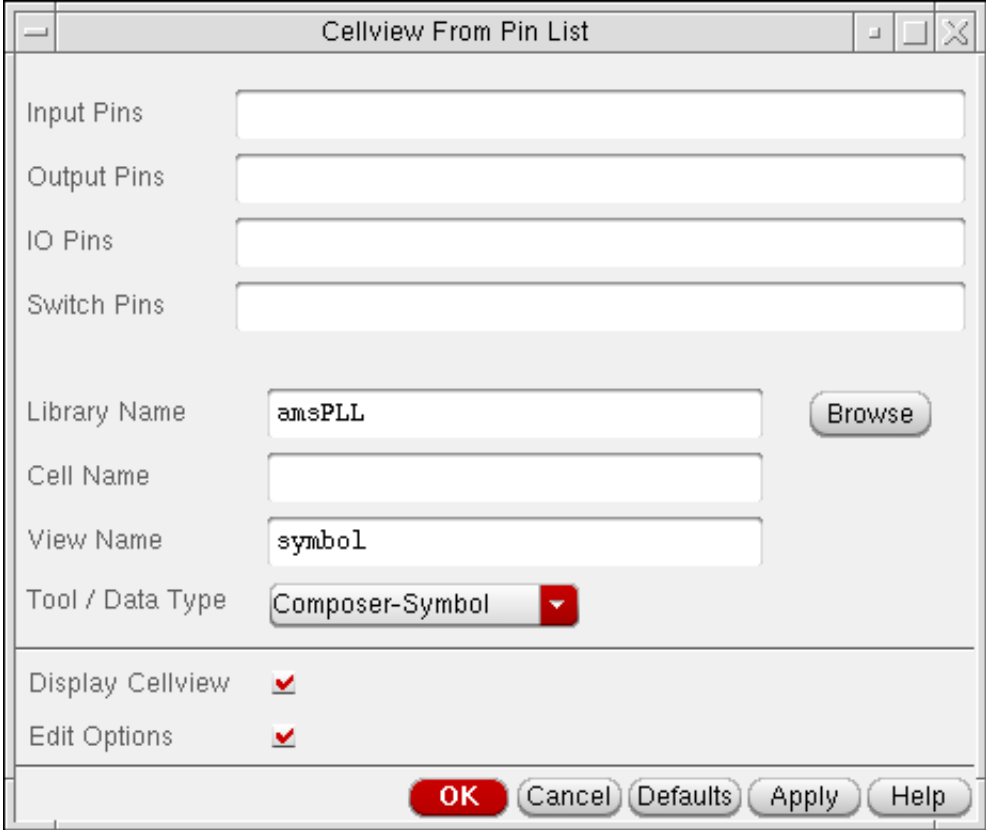
Automatically Creating a Cellview from a Pin List

A pin list can be any list of input, output, I/O, or switch pins. You can create a cellview from existing pins, such as those from the *basic* pin library, or you can create the cellview from new pin names.

To create a symbol cellview from a pin list,

1. From the view, choose *Create – Cellview – From Pin List*.

The Cellview From Pin List form appears.



2. Type the names of the pins that you want to appear in the automatically created cellview. Separate each pin name with a space.
3. (Optional) To use a different cellview as the source of the port/pin information, change the *Library Name*, *Cell Name*, and *View Name* fields.

Note: If either the *View Name* field or the *Tool / Data Type* cyclic field does not contain the desired view or conversion function, you must modify the `schViewToPinListReg`

Virtuoso Schematic Editor User Guide

Automatically Creating Cellviews

list or the `schPinListToViewReg` list, both of which are defined in your `schConfig.il` file, which is described in *Virtuoso Design Environment User Guide*.

4. (Optional) To display the cellview in a new window, turn on the *Display Cellview* button.
5. (Optional) To edit various cellview creation options, turn on *Edit Options*.
6. Click *OK*.

When generating a symbol, the Symbol Generation Options form appears if you turned on *Edit Options* and the cellview is being newly created or replaced.

If the cellview already exists, a dialog box prompts you to either replace the cellview, modify it (symbol cellview only), or cancel the command.

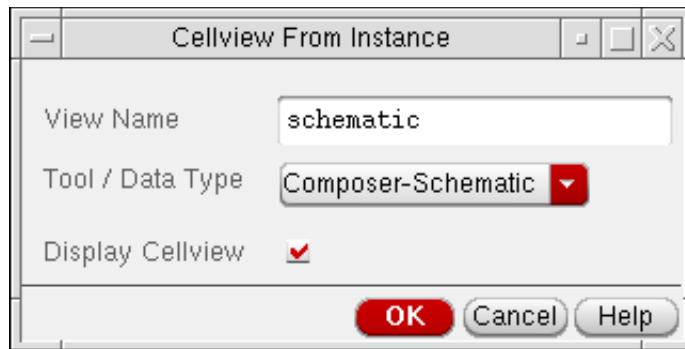
The system generates the specified cellview from the pin information you enter on the form.

Automatically Creating a Cellview from an Instance

To create a cellview from an instance,

1. From the view, choose *Create – Cellview – From Instance*.
2. Click an instance in your design window.

The Cellview From Instance form appears.



3. Change the options on the form, if needed.
4. Click *OK*.

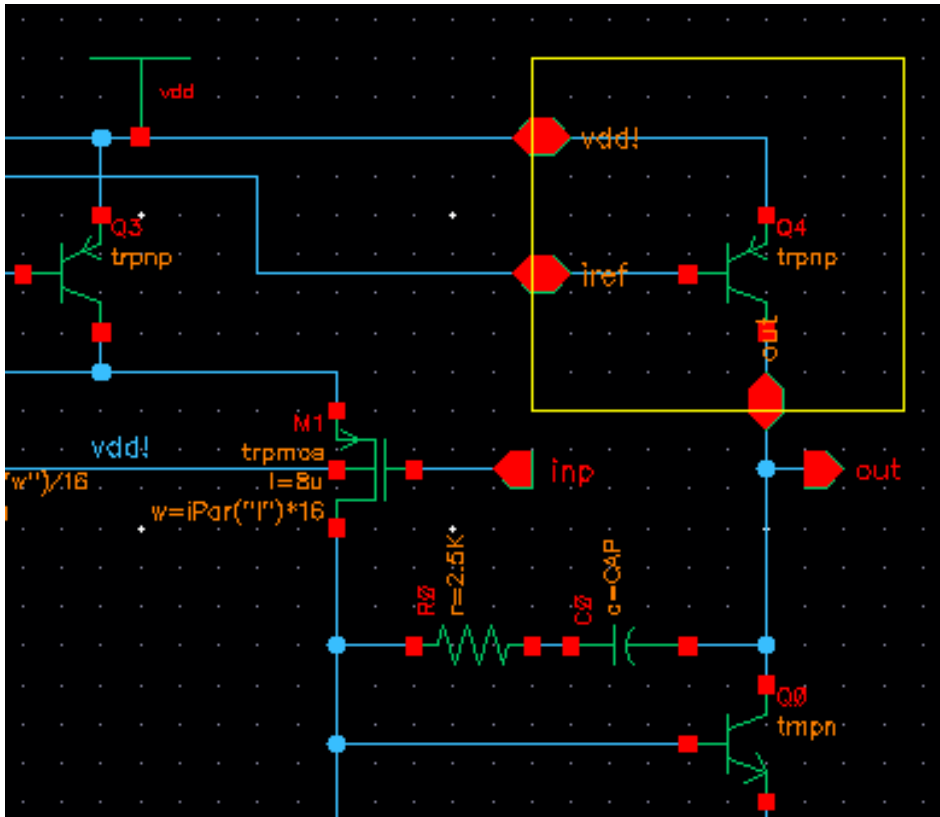
The system generates the specified cellview from the instance you selected. When you turn on *Display Cellview*, the system displays the cellview in a new window.

Creating a Cellview from the Canvas (Make Cell)

You can create a new cellview from objects within a specified canvas area by selecting that area, and using the Make Cell command. This process excludes the selection of pins.

To create a cellview from the current window:

1. In Virtuoso Schematic Editor XL, choose *Edit – Hierarchy – Make Cell*.
2. Draw a rectangle on the canvas over the objects to be included. The cursor turns into an  icon if an invalid object is selected.



Virtuoso Schematic Editor User Guide

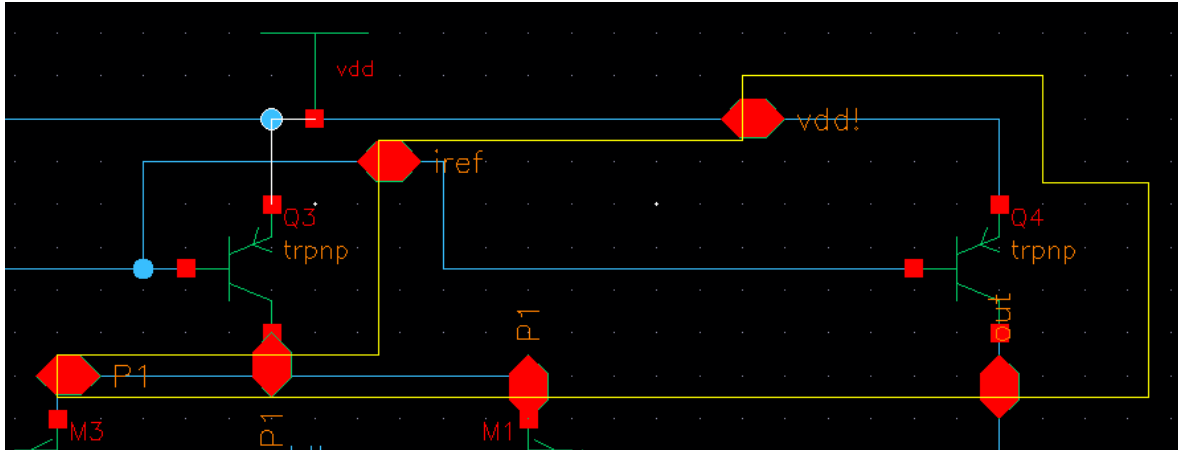
Automatically Creating Cellviews

The *Make Cell* form appears.

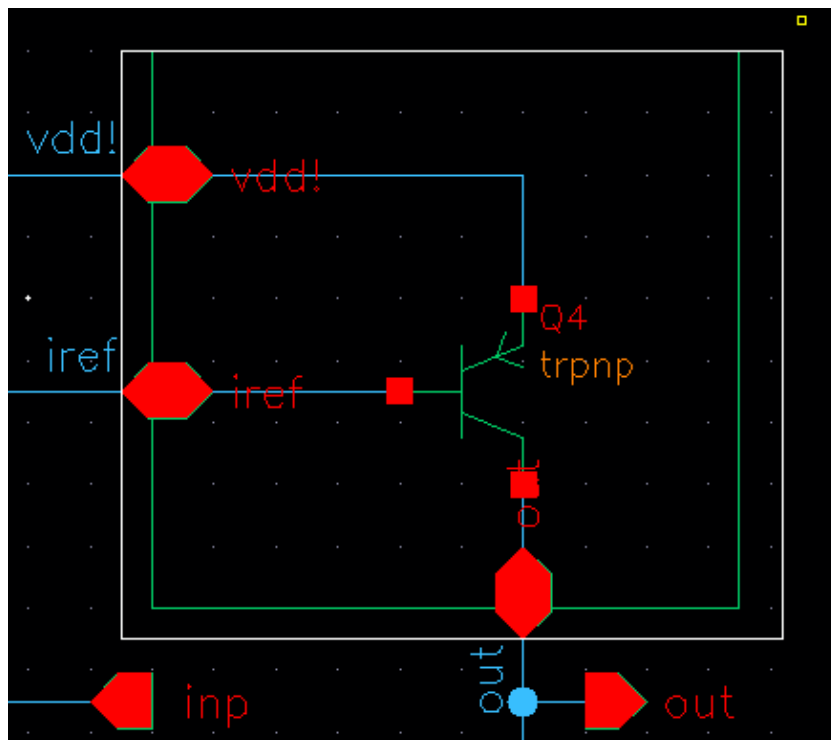
The **Make Cell** dialog box is shown with the following fields and options:

- Cut area:**
 - Top: 1.5
 - Bottom: 0.375
 - Left: 1.25
 - Right: 2.4375
 - Buttons: Add..., Subtract..., Redraw...
- Destination:**
 - Library: solutions
 - Cell: amplifier_cut2
 - View: schematic
 - ☐ Overwrite existing cellview
- Options:**
 - ☒ Simplify iterated names
 - ☒ Place symbol
 - ☐ Remove duplicate symbol pins
- Buttons:** OK, Cancel, Help

3. (Optional) Adjust the selection by using the options in the *Cut area* field.



4. (Optional) Edit the destination by changing the *Library*, *Cell*, and *View* fields.
5. Select or deselect the *Simplify iterated names* check box.
6. Select or deselect the *Place symbol* check box. Deselecting it places the new cell directly in the current schematic:

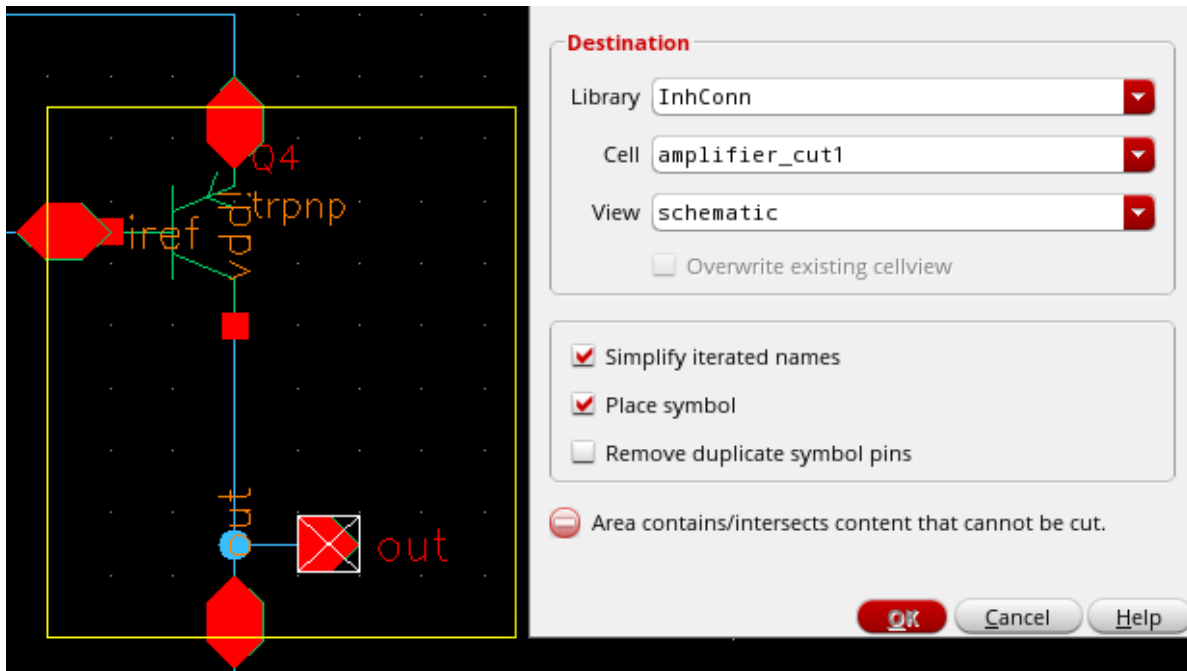


Virtuoso Schematic Editor User Guide

Automatically Creating Cellviews

7. Select or deselect the *Remove duplicate symbol pins* check box. This is only available when *Place symbol* is selected.

Note: If any incorrect selections are made, the form displays an error message and the *OK* button is unavailable. Resolve any issues, for example, remove any pins from the selection. When all issues are resolved, the *OK* button becomes available.

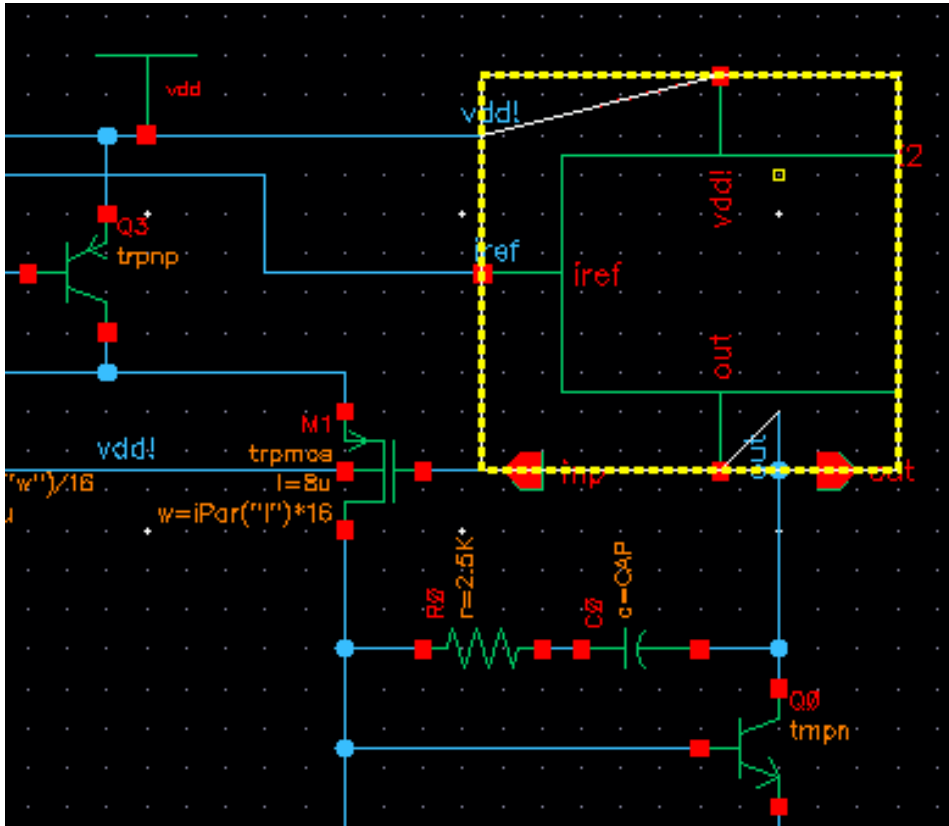


8. Click *OK*.

Virtuoso Schematic Editor User Guide

Automatically Creating Cellviews

The newly created cell is placed on the canvas.



9. Click and drag the mouse to move the symbol cell the canvas.
10. (Optional) Press the 5 key on the keyboard or select *Edit – Route Flight* to route the connections.

Editing Symbol Generation Options

The Symbol Generation Options form displays the property and attribute settings that the text-to-symbol generator (TSG) uses to generate symbols. The software loads the default form settings from a template file, which you specify using the tsgTemplateType environment variable.

This section discusses the following tasks:

- Adding, Changing, and Reordering Pin Names
- Editing Pin Attributes
- Excluding or Including Inherited Connection Pins
- Loading and Saving a TSG Template File
- Editing Symbol Attributes
- Editing Symbol Labels
- Creating a New Label
- Deleting Labels
- Modifying Label Attributes
- Editing Properties
- Creating a New Property
- Deleting Properties
- Modifying Property Attributes

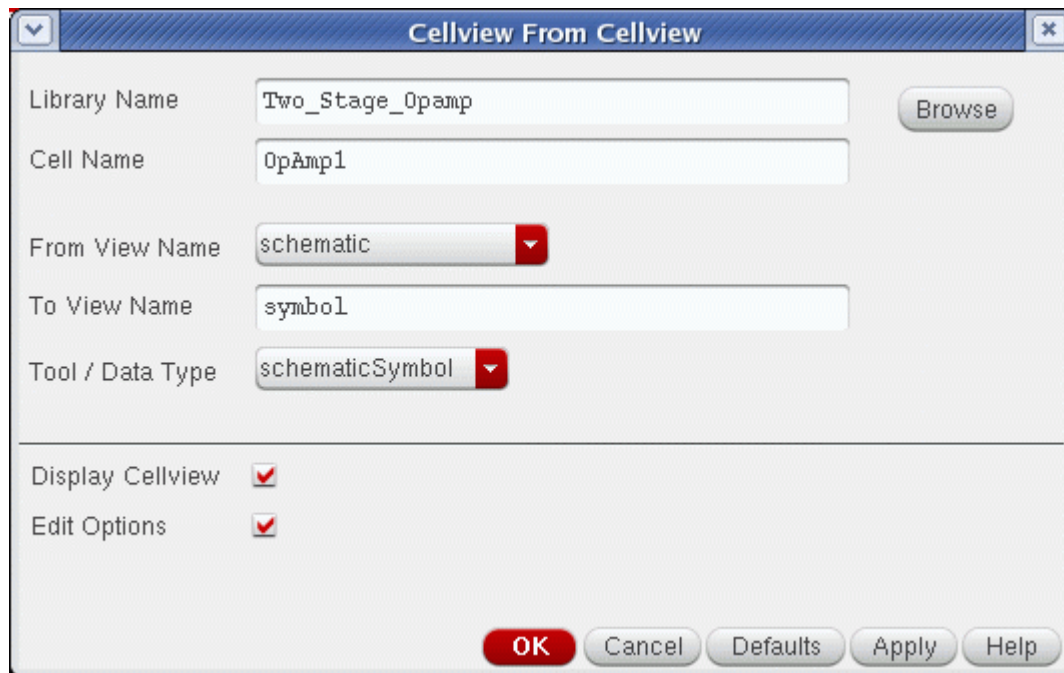
Adding, Changing, and Reordering Pin Names

To add, change, or reorder pin names,

1. From the view, do one of the following:

- ☐ Choose *Create – Cellview – From Cellview*.

The Cellview From Cellview form appears.



- ☐ Choose *Create – Cellview – From Pin List*.

Virtuoso Schematic Editor User Guide

Automatically Creating Cellviews

The Cellview From Pin List form appears.

Cellview From Pin List

Input Pins

Output Pins

IO Pins

Switch Pins

Library Name: amsPLL Browse

Cell Name

View Name: symbol

Tool / Data Type: Composer-Symbol

Display Cellview ☒

Edit Options ☒

OK Cancel Defaults Apply Help

2. Turn on *Edit Options*.

3. Click *OK*.

If the symbol cellview to be created already exists, the Create Cellview dialog box appears.

a. Click *Replace*.

The Symbol Generation Options form appears.

Virtuoso Schematic Editor User Guide

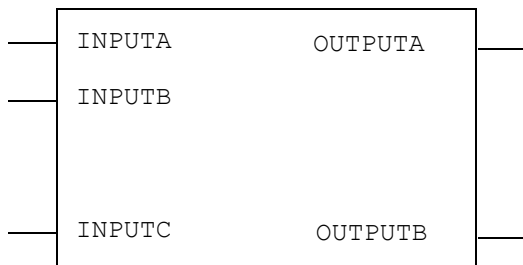
Automatically Creating Cellviews

By default, the software initializes the form with information from the current cellview and the Cellview From Cellview.

The screenshot shows the 'Symbol Generation Options' dialog box. It has three input fields at the top: 'Library Name' with 'amsPLL', 'Cell Name' with 'TopPllWithDC', and 'View Name' with 'symbol'. Below these are four 'Pin Specifications' fields: 'Left Pins' (ibias refclk reset), 'Right Pins' (it2 p0 p45 p90 p135 p180 p225 p270 p315 vcoclk vcom vcop), 'Top Pins' (empty), and 'Bottom Pins' (empty). Each pin specification field has a 'List' button to its right. Below the pin fields is a section 'Exclude Inherited Connection Pins:' with three radio buttons: 'None' (selected), 'All', and 'Only these:' (with an empty text field). At the bottom are four checkboxes: 'Load/Save', 'Edit Attributes', 'Edit Labels', and 'Edit Properties'. At the very bottom are four buttons: 'OK' (highlighted in red), 'Cancel', 'Apply', and 'Help'.

4. To space pins further apart, add one or more asterisks (*) between the pin names in the *Left Pins*, *Right Pins*, *Top Pins*, and *Bottom Pins* fields.
5. Click *OK*.

This setting results in the following:



Editing Pin Attributes

To edit the attributes of a pin,

1. From the view, do one of the following:

- ☐ Choose *Create – Cellview – From Cellview*.

The Cellview From Cellview form appears.

- ☐ Choose *Create – Cellview – From Pin List*.

The Cellview From Pin List form appears.

The screenshot shows the 'Cellview From Pin List' dialog box. It features a title bar with the text 'Cellview From Pin List'. The main area contains several input fields for pin types: 'Input Pins', 'Output Pins', 'IO Pins', 'Switch Pins', 'Jumper Pins', 'Tristate Pins', and 'Unused Pins'. Below these are four dropdown menus: 'Library Name' (currently set to 'zambezi45'), 'Cell Name' (set to 'LP_pll_dig_combo'), 'View Name' (set to 'schematic'), and 'Tool / Data Type' (set to 'schematic'). A 'Browse' button is located to the right of the 'Library Name' dropdown. At the bottom of the dialog, there are two checked checkboxes: 'Display Cellview' and 'Edit Options'. The bottom right corner contains five buttons: 'OK', 'Cancel', 'Defaults', 'Apply', and 'Help'.

2. Turn on *Edit Options*.

3. Click *OK*.

Virtuoso Schematic Editor User Guide

Automatically Creating Cellviews

If the symbol cellview to be created already exists, the Create Cellview dialog box appears.

- a. Click *Replace*.

The Symbol Generation Options form appears.

By default, the software initializes the form with information from the current cellview and the Cellview From Cellview form.

4. Select *List* from one of the *Attributes* cyclic fields.

The Left, Right, Top, Bottom Pin Attributes form appears.

Name	Direction	Pin Graphic
ibias	input	actHi
refclk	input	actHi
reset	input	actHi

The banner title on the form indicates left, right, top, or bottom pin attributes.

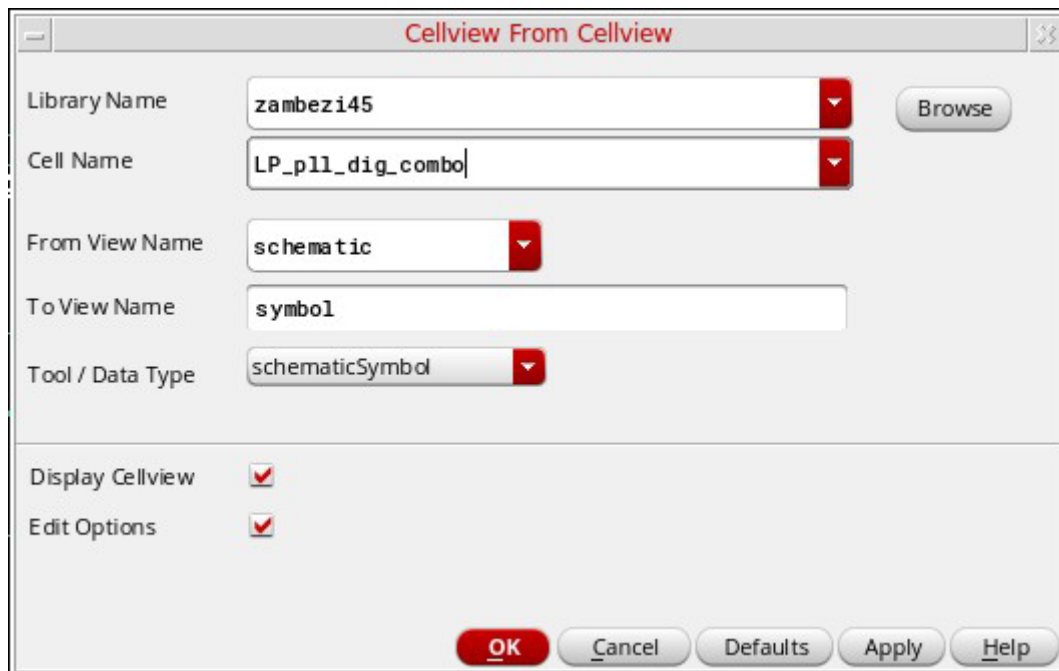
5. Use the *Direction* cyclic fields to change the direction of the pins.
6. Use the *Pin Graphic* cyclic fields to change the graphic of the pin.
7. Click *OK*.

Excluding or Including Inherited Connection Pins

To exclude or include inherited connection pins in a symbol view:

1. Choose *Create – Cellview – From Cellview*.

The Cellview From Cellview form appears.



By default, the software initializes the form with information from the current cellview (in the Library Name, Cell Name, and View Name fields) and the Cellview From Cellview form.

2. In the *To View Name* field enter name of the symbol view to be created.
3. Click the *OK* button.

Virtuoso Schematic Editor User Guide

Automatically Creating Cellviews

The Symbol Generation Options form is displayed.

The screenshot shows the 'Symbol Generation Options' dialog box. It has a title bar with standard window controls. Inside, there are three input fields at the top: 'Library Name' with 'amsPLL', 'Cell Name' with 'TopPllWithDC', and 'View Name' with 'symbol'. Below these is a section titled 'Pin Specifications' with a sub-header 'Attributes'. It contains four rows: 'Left Pins' with 'ibias refclk reset', 'Right Pins' with 'vt2 p0 p45 p90 p135 p180 p225 p270 p315 vcoclk vcom vcop', 'Top Pins' (empty), and 'Bottom Pins' (empty). Each row has a 'List' button to its right. Below the pin specifications is a section 'Exclude Inherited Connection Pins:' with three radio buttons: 'None' (selected), 'All', and 'Only these:'. The 'Only these:' option has an associated empty text field. At the bottom, there are four checkboxes: 'Load/Save', 'Edit Attributes', 'Edit Labels', and 'Edit Properties', all of which are unchecked. At the very bottom are four buttons: 'OK' (highlighted in red), 'Cancel', 'Apply', and 'Help'.

4. Choose whether or not to *Exclude Inherited Connection Pins*, or name specific inherited connection pins to be excluded.

Some design flows will require pins with inherited connections to be explicitly shown on the symbol view, whereas other flows will require pins with inherited connection attributes to not be shown. To facilitate both cases the *Exclude Inherited Connection Pins* section allows you to specify whether all pins with inherited connections are to be excluded during symbol generation, or only a named set. The (Exclude) *Only these* option will be pre-seeded with the list of all the pins that show inherited connection attributes, and you can manually edit this list.

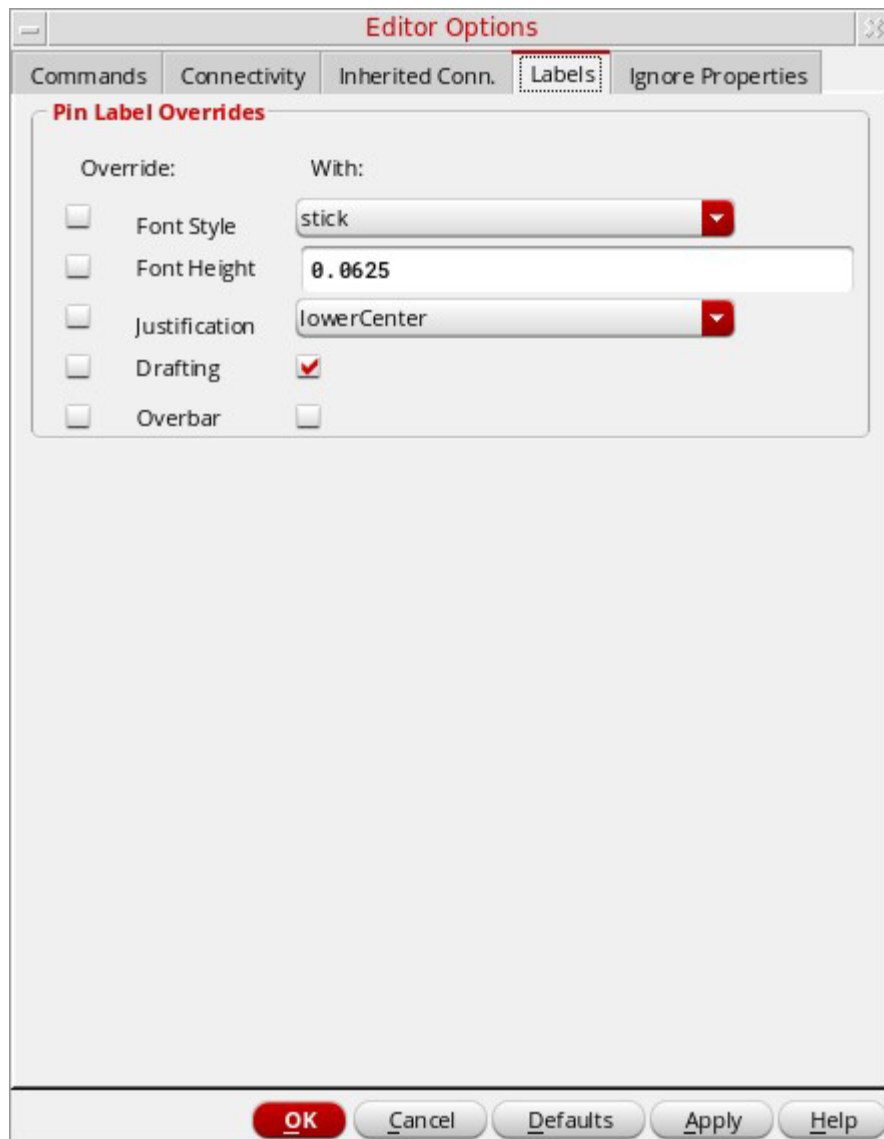
See also the [`tsgExcludeInhConnPins`](#) and [`tsgExcludeInhConnPinsNames`](#) environment variables.

5. Click **OK** to generate the new symbol view.

Editing Pin Label Defaults

You can use the *Labels* tab in the Editor Options form to control and override pin label properties such as font and justification settings.

1. Select *Options – Editor* to display the Editor Options.
2. Click the *Labels* tab.



3. Amend the options in the *Pin Label Overrides* section as required.

Note: You have to first of all check the appropriate *Override* option, then you need to

Virtuoso Schematic Editor User Guide

Automatically Creating Cellviews

choose what you want to override this option *With*. For example, to apply an overbar, you will need to check both the checkboxes associated with this option, one to choose to override the *Overbar* option, and the other to apply the overbar.

For more information see the Labels tab in the Editor Options form.

Click the *OK* button to apply the pin label overrides.

The above pin label options store their values in schematic environment variables, with two variables applicable for each option. One variable controls whether or not to override the associated symbol value, while the other provides the overriding value.

For more information see:

- pinLabelSetFontStyle
- pinLabelFontStyle
- pinLabelSetFontHeight
- pinLabelFontHeight
- pinLabelSetJustify
- pinLabelJustify
- pinLabelSetDrafting
- pinLabelSetDrafting
- pinLabelDrafting
- pinLabelSetOverbar
- pinLabelOverbar

Loading and Saving a TSG Template File

The TSG template file contains the default property and attribute settings that appear on the Symbol Generation Options form. The software loads the TSG template file the first time you use one of the *Create Cellview* commands to automatically generate a symbol. You use the tsgTemplateType environment variable to specify the TSG template file.

To load or save a TSG template file,

1. From the view, do one of the following:

- ☐ Choose *Create – Cellview – From Cellview*.

The Cellview From Cellview form appears.

- ☐ Choose *Create – Cellview – From Pin List*.

The Cellview From Pin List form appears.

2. Turn on *Edit Options*.

3. Click *OK*.

If the symbol cellview to be created already exists, the Create Cellview dialog box appears.

- a. Click *Replace*.

The Symbol Generation Options form appears.

By default, the software initializes the form with information from the current cellview and the Cellview From Cellview form.

4. In the Symbol Generation Options form, turn on *Load / Save*.

Virtuoso Schematic Editor User Guide

Automatically Creating Cellviews

The Symbol Generation Options expands to display the template data fields.

The screenshot shows the 'Symbol Generation Options' dialog box. It has a title bar with standard window controls. The main area contains several sections: 'Library Name' (amsPLL), 'Cell Name' (TopPllWithDC), and 'View Name' (symbol). Below these are 'Pin Specifications' with fields for 'Left Pins' (ibias refclk reset), 'Right Pins' (vt2 p0 p45 p90 p135 p180 p225 p270 p315 vcoclk vcom vcop), 'Top Pins', and 'Bottom Pins'. Each pin field has a 'List' button. There is an 'Exclude Inherited Connection Pins' section with radio buttons for 'None' (selected), 'All', and 'Only these:'. Below this are checkboxes for 'Load/Save' (checked), 'Edit Attributes', 'Edit Labels', and 'Edit Properties'. A 'Load/Save Symbol Template Configuration' section includes a 'Load' button, a dropdown menu set to 'digital', a text field with the path '60727/tools.sun4v/dfII/samples/symbolGen/default.tsg', a 'Save' button, a 'TSG File' label, and a 'Load Template Data From Symbol' button. At the bottom of this section are three more input fields for 'Library Name', 'Cell Name', and 'View Name' (which already contains 'symbol'). The dialog ends with 'OK', 'Cancel', 'Apply', and 'Help' buttons.

5. In the *Load / Save Symbol Template Configuration* section, type the path to the TSG template file.
6. Click *Load* or *Save*.

The system initializes the *Load / Save Symbol Template Configuration* section of the form with the values from the specified TSG file.

The software builds all subsequent symbols using the new values.

Editing Symbol Attributes

To edit symbol attributes,

1. From the view, do one of the following:

- ☐ Choose *Create – Cellview – From Cellview*.

The Cellview From Cellview form appears.

- ☐ Choose *Create – Cellview – From Pin List*.

The Cellview From Pin List form appears.

2. Turn on *Edit Options*.

3. Click *OK*.

If the symbol cellview to be created already exists, the Create Cellview dialog box appears.

- a. Click *Replace*.

The Symbol Generation Options form appears.

By default, the software initializes the form with information from the current cellview and the Cellview From Cellview form.

4. In the Symbol Generation Options form, turn on *Edit Attributes*.

Virtuoso Schematic Editor User Guide

Automatically Creating Cellviews

The Symbol Generation Options form expands to display the symbol attributes options.

Symbol Generation Options

Library Name: Cell Name: View Name:

Pin Specifications

Left Pins:

Right Pins:

Top Pins:

Bottom Pins:

Exclude Inherited Connection Pins:

☒ None ☐ All ☐ Only these:

Load/Save ☐ Edit Attributes ☒ Edit Labels ☐ Edit Properties ☐

Symbol Attributes

Pin Spacing: Height: Pin Connector:

Stub Length: Width: Origin:

Reverse Pin Order ☐

5. Edit the attributes.

Note: The symbol origin can be specified as `topLeftPin`, `bottomLeftPin`, or `centerLeft` in the Symbol Generation Options form or using the text-to-symbol generator (TSG) origin setting.

- ☐ If the symbol origin is `topLeftPin`, the top-most, left-most pin is used.
- ☐ If the symbol origin is `bottomLeftPin`, the bottom-most, left-most pin is used.
- ☐ If the symbol origin is `centerLeft`, the origin is the middle of the left-hand side of the symbol box.

6. Click *OK*.

Editing Symbol Labels

To edit symbol labels:

1. From the view, do one of the following:

- ☐ Choose *Create – Cellview – From Cellview*.

The Cellview From Cellview form appears.

- ☐ Choose *Create – Cellview – From Pin List*.

The Cellview From Pin List form appears.

2. Turn on *Edit Options*.

3. Click *OK*.

If the symbol cellview to be created already exists, the Create Cellview dialog box appears.

- a. Click *Replace*.

The Symbol Generation Options form appears.

By default, the software initializes the form with information from the current cellview and the Cellview From Cellview form.

4. In the Symbol Generation Options form, turn on *Edit Labels*.

Virtuoso Schematic Editor User Guide

Automatically Creating Cellviews

The Symbol Generation Options form expands to display the label options.

Symbol Generation Options

Library Name: Cell Name: View Name:

Pin Specifications

Left Pins:

Right Pins:

Top Pins:

Bottom Pins:

Exclude Inherited Connection Pins:

☒ None ☐ All ☐ Only these:

Load/Save ☐ Edit Attributes ☐ Edit Labels ☒ Edit Properties ☐

Label: 1. {pinName} (bottom)

Name: Label Choice:

Apply To: Label Type:

Orientation: Font Height:

Justification: Font Style: isOverbar ☐

Location:

5. Edit the symbol labels.

6. Click **OK**.

Creating a New Label

To create a new label,

1. From the view, do one of the following:

- ☐ Choose *Create – Cellview – From Cellview*.

The Cellview From Cellview form appears.

- ☐ Choose *Create – Cellview – From Pin List*.

The Cellview From Pin List form appears.

2. Turn on *Edit Options*.

3. Click *OK*.

If the symbol cellview to be created already exists, the Create Cellview dialog box appears.

- a. Click *Replace*.

The Symbol Generation Options form appears.

By default, the software initializes the form with information from the current cellview and the Cellview From Cellview form.

4. In the Symbol Generation Options form, turn on *Edit Labels*.

Virtuoso Schematic Editor User Guide

Automatically Creating Cellviews

The Symbol Generation Options form expands to display the data that can be edited.

The screenshot shows the 'Symbol Generation Options' dialog box. At the top, there are three text fields: 'Library Name' (amsPLL), 'Cell Name' (TopPllWithDC), and 'View Name' (symbol). Below these are four 'Pin Specifications' sections: 'Left Pins' (ibias refclk reset), 'Right Pins' (vt2 p0 p45 p90 p135 p180 p225 p270 p315 vcoclk vcom vcop), 'Top Pins' (empty), and 'Bottom Pins' (empty). Each has a 'List' button. Below the pins is an 'Exclude Inherited Connection Pins' section with radio buttons for 'None' (selected), 'All', and 'Only these:' (with an empty text field). At the bottom of the main section are four checkboxes: 'Load/Save' (unchecked), 'Edit Attributes' (unchecked), 'Edit Labels' (checked), and 'Edit Properties' (unchecked). The bottom section contains a 'Label' list with one entry: '1. {pinName} (bottom)'. To the right are 'Add', 'Delete', and 'Modify' buttons. Below the list are fields for 'Name' ({pinName}), 'Label Choice' (pin name), 'Apply To' (bottom pins), 'Label Type' (normalLabel), 'Orientation' (R270), 'Font Height' (0.0625), 'Justification' (centerRight), 'Font Style' (stick), 'isOverbar' (unchecked), and 'Location' (upperCenter). A text field next to 'Location' contains the formula '0:1.15*stubLength'. At the very bottom are 'OK', 'Cancel', 'Apply', and 'Help' buttons.

5. In the *Label* cyclic field, choose *new*.
6. Choose the desired *Label Choice* and other options.
7. In the *Name* field, type the name of the new label.
8. In the *Apply To* field, specify which objects the label attaches to.
9. Click *Add*.

Virtuoso Schematic Editor User Guide

Automatically Creating Cellviews

The new label name is added to the list of labels that will automatically be created during symbol generation.

Deleting Labels

To delete a symbol label from the *Label* cyclic field,

1. From the view, do one of the following:

- ☐ Choose *Create – Cellview – From Cellview*.

The Cellview From Cellview form appears.

- ☐ Choose *Create – Cellview – From Pin List*.

The Cellview From Pin List form appears.

2. Turn on *Edit Options*.

3. Click *OK*.

If the symbol cellview to be created already exists, the Create Cellview dialog box appears.

- a. Click *Replace*.

The Symbol Generation Options form appears.

By default, the software initializes the form with information from the current cellview and the Cellview From Cellview form.

4. In the Symbol Generation Options form, turn on *Edit Labels*.

The Symbol Generation Options form expands to display the label options.

5. In the *Label* cyclic field, choose the name of the label you want to delete.

6. Click *Delete*.

The label name no longer appears in the *Label* cyclic field.

Modifying Label Attributes

To modify label attributes,

1. From the view, do one of the following:

- ☐ Choose *Create – Cellview – From Cellview*.

The Cellview From Cellview form appears.

- ☐ Choose *Create – Cellview – From Pin List*.

The Cellview From Pin List form appears.

2. Turn on *Edit Options*.

3. Click *OK*.

If the symbol cellview to be created already exists, the Create Cellview dialog box appears.

- a. Click *Replace*.

The Symbol Generation Options form appears.

By default, the software initializes the form with information from the current cellview and the Cellview From Cellview form.

4. In the Symbol Generation Options form, turn on *Edit Labels*.

The Symbol Generation Options form expands to display the label options.

5. In the *Label* cyclic field, choose the name of the label you want to modify.

6. Make any desired changes to the various label display attributes or label location fields.

7. Click *Modify*.

The label is modified.

Editing Properties

To modify edit properties,

1. From the view, do one of the following:

- ☐ Choose *Create – Cellview – From Cellview*.

The Cellview From Cellview form appears.

- ☐ Choose *Create – Cellview – From Pin List*.

The Cellview From Pin List form appears.

2. Turn on *Edit Options*.

3. Click *OK*.

If the symbol cellview to be created already exists, the Create Cellview dialog box appears.

- a. Click *Replace*.

The Symbol Generation Options form appears.

By default, the software initializes the form with information from the current cellview and the Cellview From Cellview form.

4. In the Symbol Generation Options form, turn on *Edit Properties*.

Virtuoso Schematic Editor User Guide

Automatically Creating Cellviews

The Symbol Generation Options form expands.

The screenshot shows the 'Symbol Generation Options' dialog box. At the top, there are three input fields: 'Library Name' (amsPLL), 'Cell Name' (TopPllWithDC), and 'View Name' (symbol). Below these are four 'Pin Specifications' sections: 'Left Pins' (ibias refclk reset), 'Right Pins' (vt2 p0 p45 p90 p135 p180 p225 p270 p315 vcoclk vcom vcop), 'Top Pins' (empty), and 'Bottom Pins' (empty). Each has a 'List' button. Below the pins is the 'Exclude Inherited Connection Pins' section with radio buttons for 'None' (selected), 'All', and 'Only these:' (with an empty text field). At the bottom of the main section are four checkboxes: 'Load/Save' (unchecked), 'Edit Attributes' (unchecked), 'Edit Labels' (unchecked), and 'Edit Properties' (checked). The 'Edit Properties' section is expanded, showing a list of properties. The first property is 'partName'. It has buttons for 'Add', 'Delete', and 'Modify'. Below this, the 'partName' property is detailed: 'Name' is 'partName', 'Value' is '{cellName}', 'Apply To' is 'cellview', 'Property Type' is 'string', 'Orientation' is 'R0', 'Font Height' is '0.0625', 'Visibility' is 'off', 'Justification' is 'lowerCenter', 'Font Style' is 'stick', 'isOverbar' is unchecked, and 'Location' is 'upperRight' with a text field containing 'xright: ytop'. At the bottom right are buttons for 'OK', 'Cancel', 'Apply', and 'Help'.

5. Edit any property.

6. Click **OK**.

Creating a New Property

To create a new property,

1. From the view, do one of the following:

- ☐ Choose *Create – Cellview – From Cellview*.

The Cellview From Cellview form appears.

- ☐ Choose *Create – Cellview – From Pin List*.

The Cellview From Pin List form appears.

2. Turn on *Edit Options*.

3. Click *OK*.

If the symbol cellview to be created already exists, the Create Cellview dialog box appears.

- a. Click *Replace*.

The Symbol Generation Options form appears.

By default, the software initializes the form with information from the current cellview and the Cellview From Cellview form.

4. In the Symbol Generation Options form, turn on *Edit Properties*.

The Symbol Generation Options form expands.

5. In the *Property* cyclic field, choose *new*.

6. In the *Name* field, type the name of the new label.

7. In the *Apply To* field, specify which objects the label attaches to.

8. Click *Add*.

The new property name is added to the list of properties that are automatically created during symbol generation.

Deleting Properties

To delete properties,

1. From the view, do one of the following:

- ☐ Choose *Create – Cellview – From Cellview*.

The Cellview From Cellview form appears.

- ☐ Choose *Create – Cellview – From Pin List*.

The Cellview From Pin List form appears.

2. Turn on *Edit Options*.

3. Click *OK*.

If the symbol cellview to be created already exists, the Create Cellview dialog box appears.

- a. Click *Replace*.

The Symbol Generation Options form appears.

By default, the software initializes the form with information from the current cellview and the Cellview From Cellview form.

4. In the Symbol Generation Options form, turn on *Edit Properties*.

The Symbol Generation Options form expands.

5. In the *Properties* cyclic field, choose the name of the property you want to delete.

6. Click *Delete*.

The property no longer appears in the *Properties* cyclic field.

Modifying Property Attributes

To modify property attributes,

1. From the view, do one of the following:

- ☐ Choose *Create – Cellview – From Cellview*.

The Cellview From Cellview form appears.

- ☐ Choose *Create – Cellview – From Pin List*.

The Cellview From Pin List form appears.

2. Turn on *Edit Options*.

3. Click *OK*.

If the symbol cellview to be created already exists, the Create Cellview dialog box appears.

- a. Click *Replace*.

The Symbol Generation Options form appears.

By default, the software initializes the form with information from the current cellview and the Cellview From Cellview form.

4. In the Symbol Generation Options form, turn on *Edit Properties*.

The Symbol Generation Options form expands.

5. In the *Properties* cyclic field, choose the property name you want to modify.

6. Make any desired changes to the various property display attributes or property location fields.

7. Click *Modify*.

Specifying Symbol Generator Pin Sorting

Note: Pin sorting is valid only when you create a symbol from a schematic.

When symbols are automatically created, TSG sorts the pins either alphanumerically by pin name (default) or geometrically according to the pin's relative position in the source schematic view.

To specify how TSG sorts the pins,

1. From the view, choose *Options – Editor*.

The Editor Options form appears.

2. In the *Command Controls* section of the form, set the *Symbol Generator Pin Sorting* option to *alphanumeric* or *geometric*.
3. Click *OK*.

Automatic symbol generation now uses the specified pin sorting method.

You can change the default (*alphanumeric* to *geometric*) by editing the [ssgSortPins](#) environment variable. The system uses this variable when you choose *Options – Save Defaults*. It is also available as arguments to the [schGetEnv](#) and [schSetEnv](#) procedural interface SKILL functions.

Editing Objects

This chapter discusses commands such as *Copy*, *Move*, *Stretch*, and *Delete*. The next chapter discusses *Edit Properties* commands.

- [Using Direct Manipulation](#)
- [Undoing and Redoing an Edit](#)
- [Stretching](#)
- [Copying](#)
- [Moving](#)
- [Snapping to Grid](#)
- [Ignoring Instances](#)
- [Deleting](#)
- [Rotating](#)
- [Aligning](#)
- [Distributing](#)
- [Discarding Edits](#)
- [Alternating Symbol Views](#)
- [Toggling Objects](#)
- [Updating Pins Across Cellviews](#)
- [Annotating Data on the Schematic Window](#)

Using Direct Manipulation

Direction manipulation is one of the quickest ways to edit an object. You use the mouse, rather than commands, to edit.

To directly manipulate an object,

1. Place the pointer over an object or shape.
2. Hold down the mouse button and the appropriate key to copy, move, stretch, or create an object.
3. Drag the mouse to complete the desired action.

You can use direct manipulation to do the following tasks:

- Copy an object or shape
- Move an object or shape
- Stretch an object or shape
- Create a wire

Virtuoso Schematic Editor User Guide

Editing Objects

Direct Manipulation Operations

The following table lists operations you can perform using direct manipulation.

Object Type	Drag	Shift-Drag	Control-Drag
wires	stretch	copy	move
wire vertex	partial stretch	add wire	
instances	stretch	copy	move
schematic pins	stretch	add wire	move
schematic labels	move	copy	move
note figures	move	copy	move
edge of note figure	stretch edge	stretch edge	
vertex of note figure	stretch vertex	stretch vertex	
instance pins	add wire	add wire	
instance labels	move	move	
symbol pins	move	add line	
symbol/note shapes	move	copy	
symbol shape edge	stretch edge	stretch edge	
symbol shape vertex	stretch vertex	stretch vertex	
symbol/note labels	move	copy	

Turning Off Direct Manipulation

To turn off direct manipulation,

1. Open the `schBindKeys.il` file located in the `your_install_dir/tools/dfII/samples/local` directory.
2. Comment out (1), (2), and (3).

These three bindkeys are keyed to the three mouse buttons for direct manipulation.

3. Load the current session or add the *Load* command to the `.cdsinit` file for future sessions.

Undoing and Redoing an Edit

To undo an edit,

- Choose *Edit – Undo*.

This command reverses the action of the previous edit command. You can undo, one at a time, the last 10 levels of edit commands. Commands that cause data to be saved to disk cannot be undone. You can set the undo limit using the *Options – User Preferences* command from the Command Interpreter Window (CIW).

To cancel the previous undo commands,

- Choose *Edit – Redo*.

Undo/Redo using DFII on OpenAccess

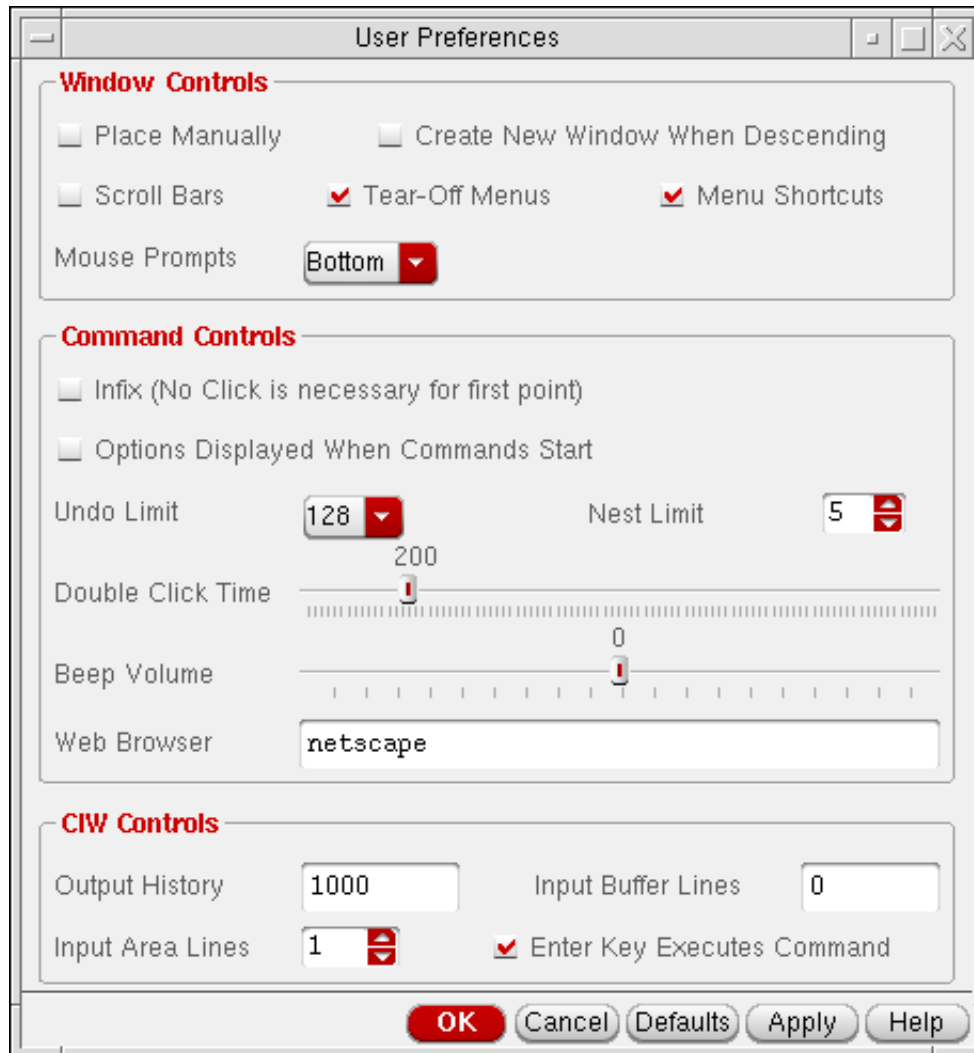
- DFII on OpenAccess supports the undo/redo of changes to parameterized cell parameters on instances, including stretchable Pcells.
- DFII on OpenAccess allows undoing changes made to the design only before saving it.
- DFII on OpenAccess supports 128 undo levels, where zero turns off the undo mechanism, and 128 turns on the undo mechanism for up to 128 levels of undo.
- DFII on OpenAccess prompts you to save only if the cellview has been changed.
- In DFII on OpenAccess, when all the changes to a cellview are undoable and have been undone, the `dbIsCellViewModified` function returns `FALSE` and the cellview attribute `modifiedButNotSaved` returns `nil` when queried with `cv~>modifiedButNotSaved`.
- DFII on OpenAccess does not add checkpoints for failed edits to the undo stack.
- Undo/redo for DFII on OpenAccess is unavailable after running either of the Save commands, *File – Save* or *File – Check and Save*, unless the environment variable `dbUndoAcrossSave` is set to `t`.

Changing the Undo Limit

To change the undo limit,

1. From the CIW, choose *Options – User Preferences*.

The User Preferences form appears.



2. Move the *Undo Limit* slider to the limit you want.

Note: In OpenAccess, the *Undo Limit* option is a drop-down box with two values, 0 and 128. You can undo the last 128 levels of edit commands. However, if `dbUndoAcrossSave` is set to the default `nil`, the undo stack is reset each time the Save commands, *File – Check and Save* or *File – Save* are run.

3. Click *OK*.

Stretching

Use the *Stretch* command to reposition the placement of objects in your design without losing connectivity between wires and pins.

- Stretching works most effectively when infix mode is turned on.
- Direct manipulation is the easiest and fastest way to stretch objects or shapes.
- The *Stretch* command does not stretch objects from one cellview to another.
- Use extended selection to select an object (such as an instance, pin, or block) and associated wires so you can stretch them all at once.

The *Stretch* command operates differently in the schematic editor and the symbol editor.

In the schematic editor, *Stretch*

- Stretches selected objects and maintains connectivity by rerouting wires when necessary
- The *Stretch* command supports partially selected wire vertexes. The remaining nonselected vertexes are anchored in place.
- Moves wires while maintaining connectivity with the object attached to the wire
 - Moves a pin while maintaining connectivity with the wire that is attached to the pin

In the symbol editor, *Stretch*

- Stretches an entire shape
- Shapes move independently of one another. They do not need to maintain connectivity.
- Stretches a partial shape (an edge or a vertex of a shape or an endpoint of a line)

Stretching with Bindkeys

To stretch an object using bindkeys,

1. From the symbol window, select the object to stretch.
2. Press **m**.
3. Click a destination for the stretch.

Stretching in the Schematic Window

To stretch an object while in the schematic window,

1. Choose *Edit – Stretch*.
2. Select the object you want to stretch.

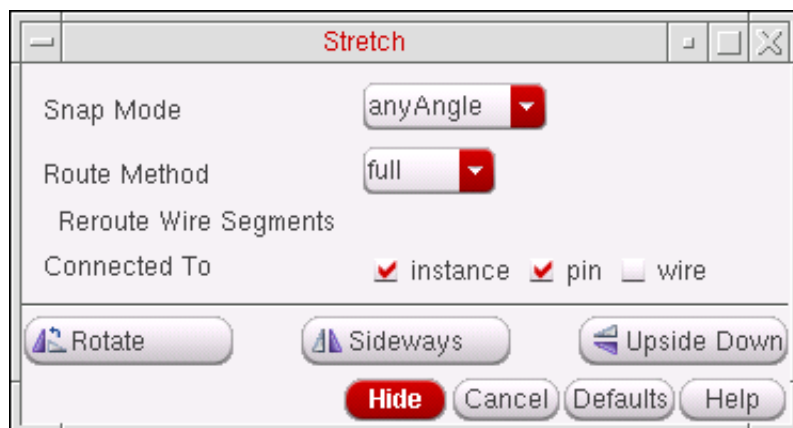
The point used to select the object becomes the reference point.

3. Point at a reference point.
4. Move the mouse, then click a destination for the stretch.

To change *Stretch* command options,

- Press **F3**.

The Stretch – Schematic View form appears.



Virtuoso Schematic Editor User Guide

Editing Objects

The selected objects are moved rather than stretched in the following cases:

- *Snap Mode* restricts the direction of any move that a selected object is to make.
- *Route Method* specifies how the wires are to rerouted.
- *Reroute Wire Segments Connected To* specifies which wire segments will be rerouted with regards to the type of selected object that they are currently connected to (for more information see [Using the Reroute Wire Segments Connected To Option](#)).

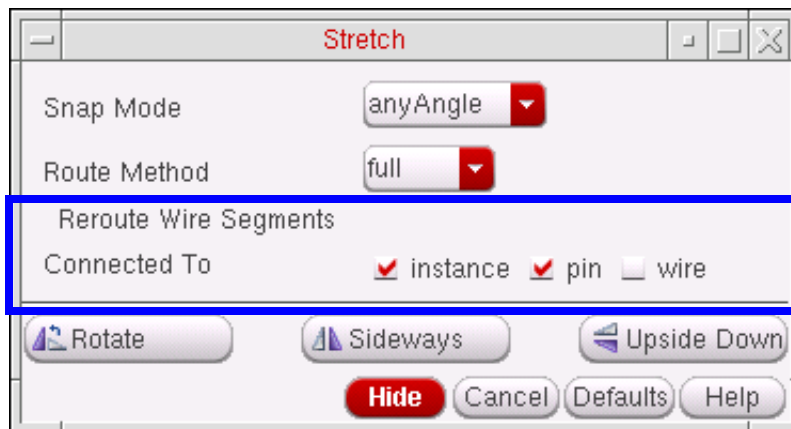
Note: In the schematic view you can use the *Text Select Filter* option on the *Select* toolbar to reposition overlapping symbol labels.

Using the Reroute Wire Segments Connected To Option

The *Reroute Wire Segments Connected To* option, in the Stretch form, can be used with instances, pins, and/or wires.

When an instance, pin, or wire is selected in the design canvas, the *Reroute Wire Segments Connected To* option will determine (dependent upon the appropriate options being checked or unchecked) which wire segments are to be rerouted.

Note: The wire segments to be stretched will be previewed by one or more dashed, yellow, flightlines.



Rerouting Instances

- When the *instance* option **is not checked**, *only* those wire segments that are directly connected to the selected instance will be rerouted (as shown in [Figure 7-1](#) on page 473).

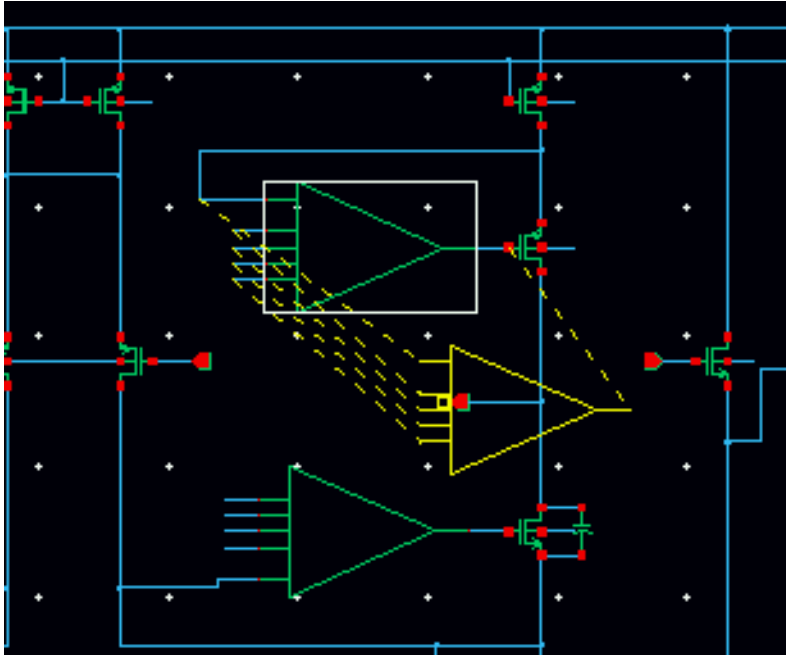


Figure 7-1 Instance option unchecked so only directly connected wire segments will be rerouted

- When the *instance* option **is checked**, *all* wire segments between the selected instance and the next solder dot, instance, or pin, will be rerouted (as shown in [Figure 7-2](#) on page 474).

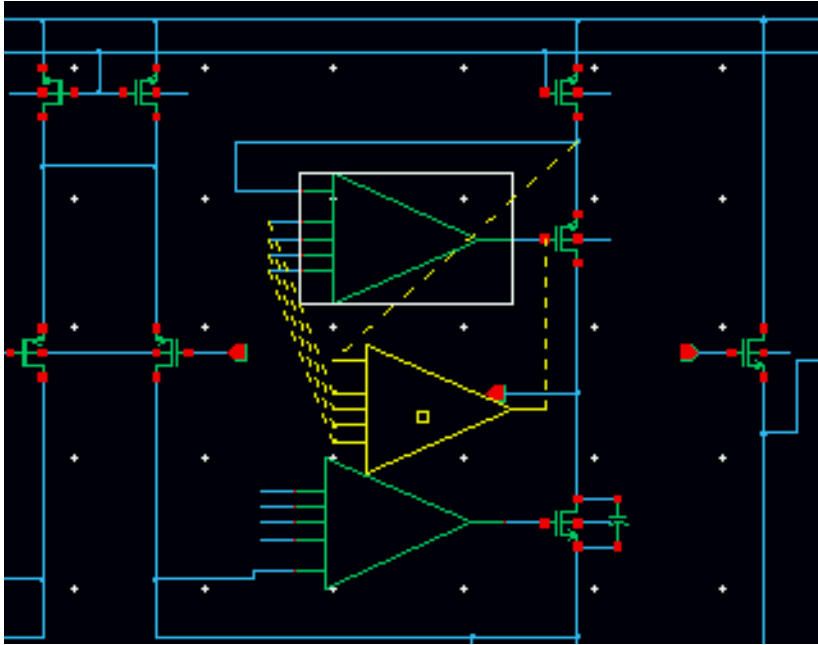


Figure 7-2 Instance option checked so all connected wire segments will be rerouted

Rerouting Pins

- When the *pin* option is **not checked**, *only* the wire segment that is directly connected to the selected pin will be rerouted (as shown in [Figure 7-3](#) on page 475).

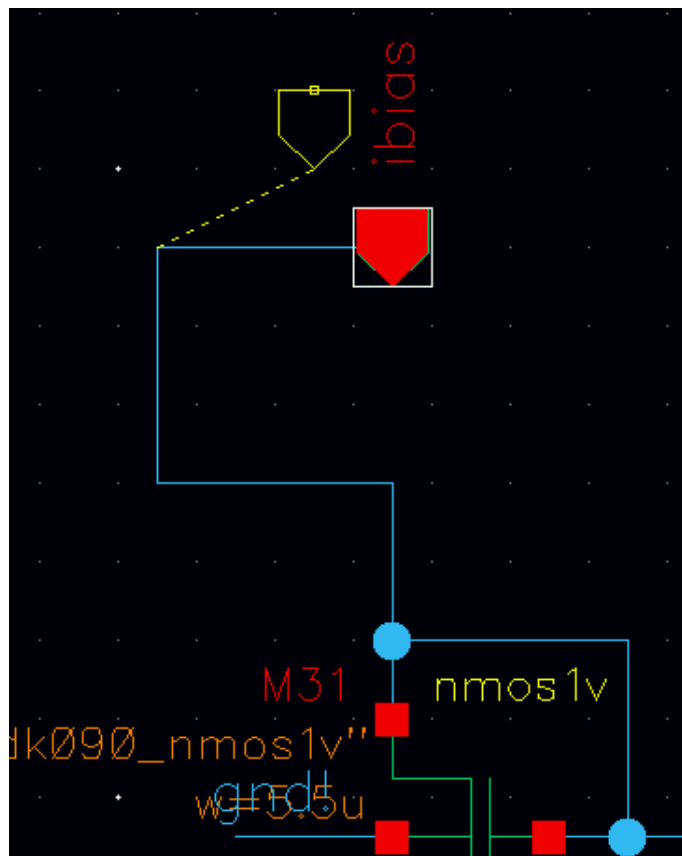


Figure 7-3 Pin option unchecked so only the wire segment directly connected to the pin will be rerouted

- When the *pin* option is **checked**, *all* wire segments between the selected pin and the next solder dot, instance, or pin, will be rerouted (as shown in [Figure 7-4](#) on page 476).

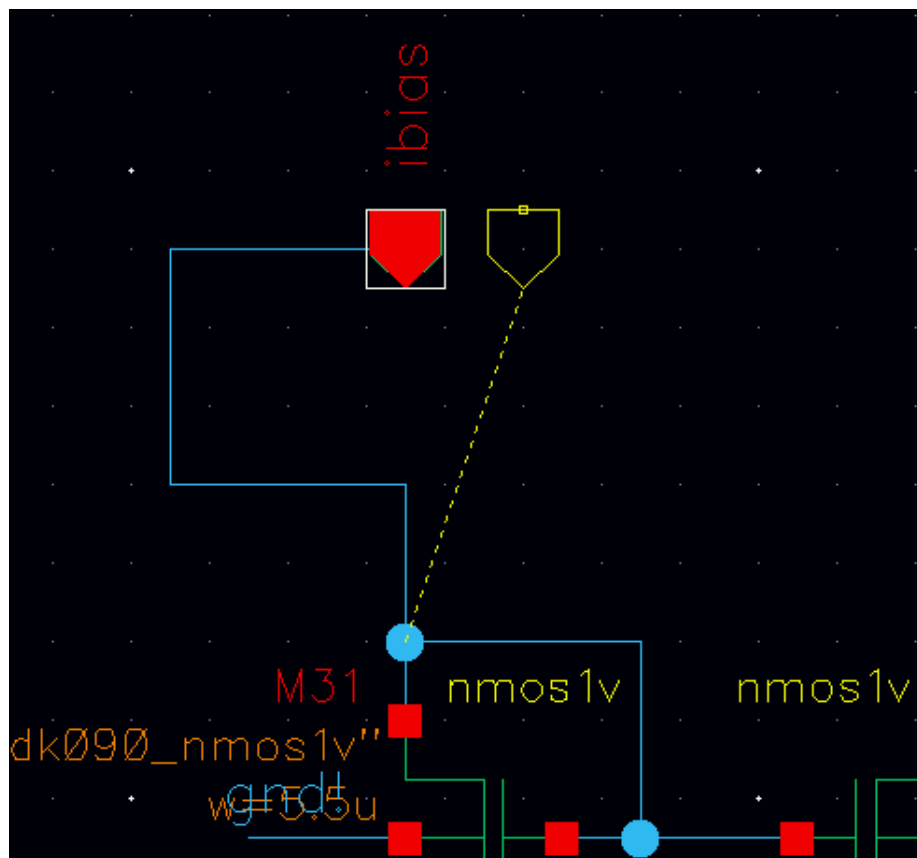


Figure 7-4 Pin option checked so all connected wire segments will be rerouted

Rerouting Wires

- When the *wire* option is **not checked**, *only* the wire segments that are directly connected to the selected wire will be rerouted (as shown in [Figure 7-5](#) on page 477).

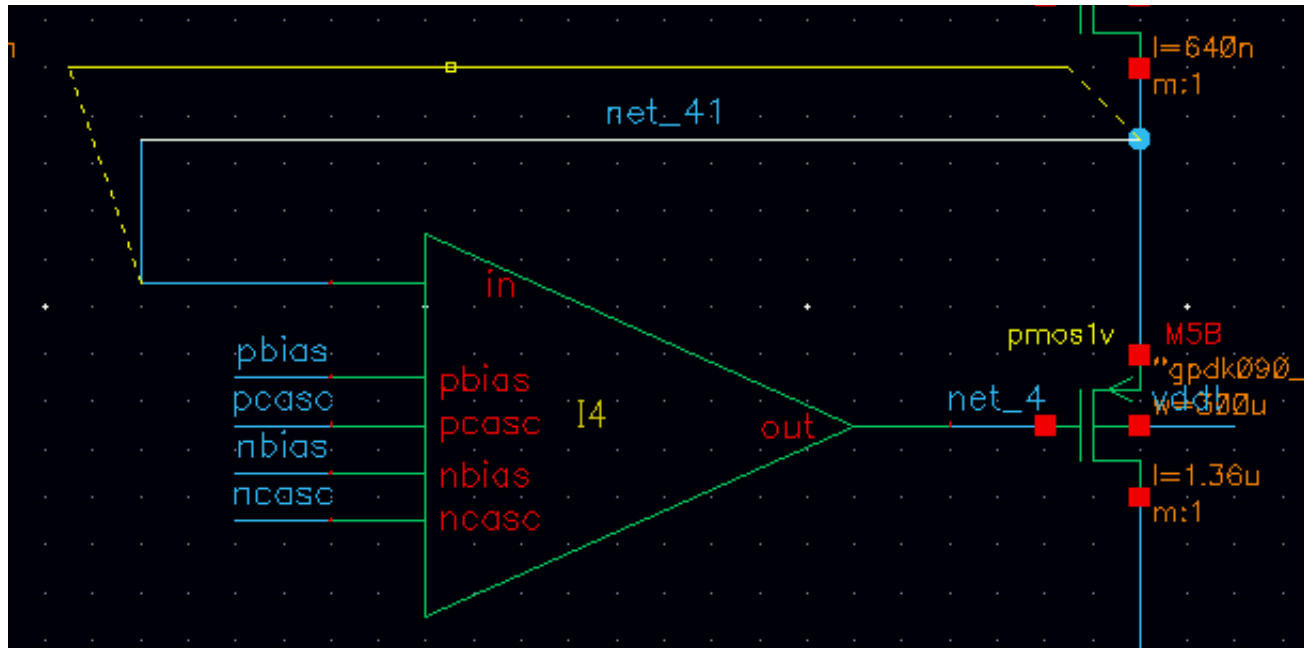


Figure 7-5 Wire option unchecked so only wire segment directly connected will be rerouted

- When the *wire* option is **checked**, *all* wire segments connected between the selected wire and the next solder dot, instance, or pin, will be rerouted (as shown in [Figure 7-4](#) on page 476).

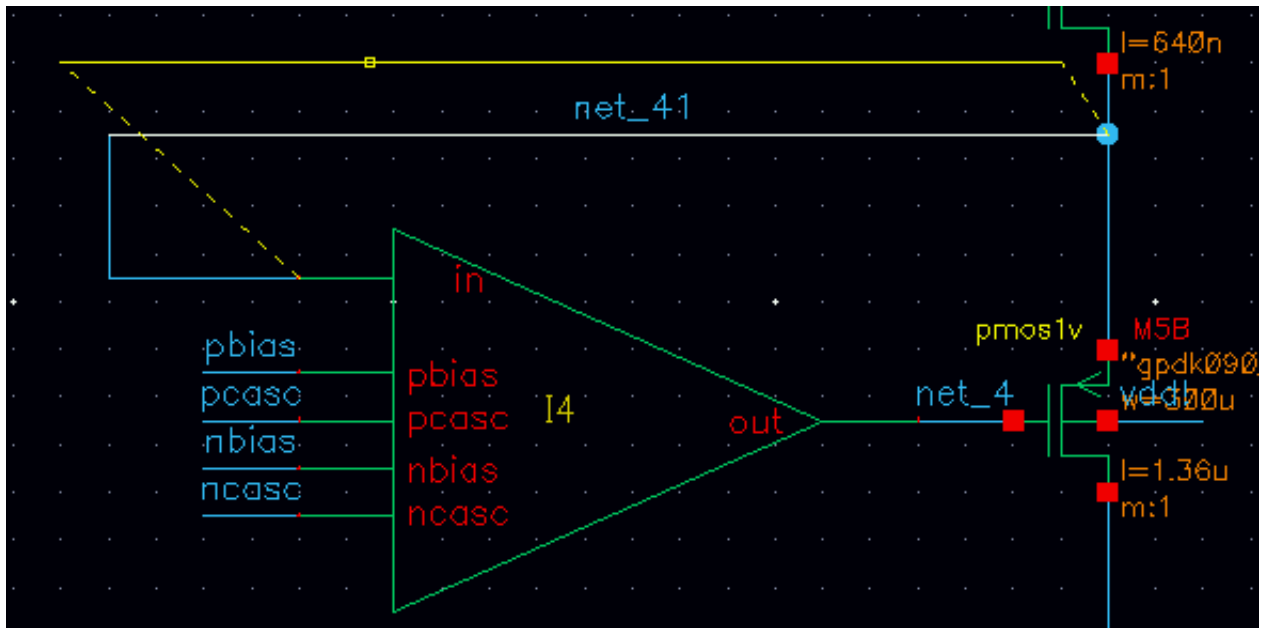
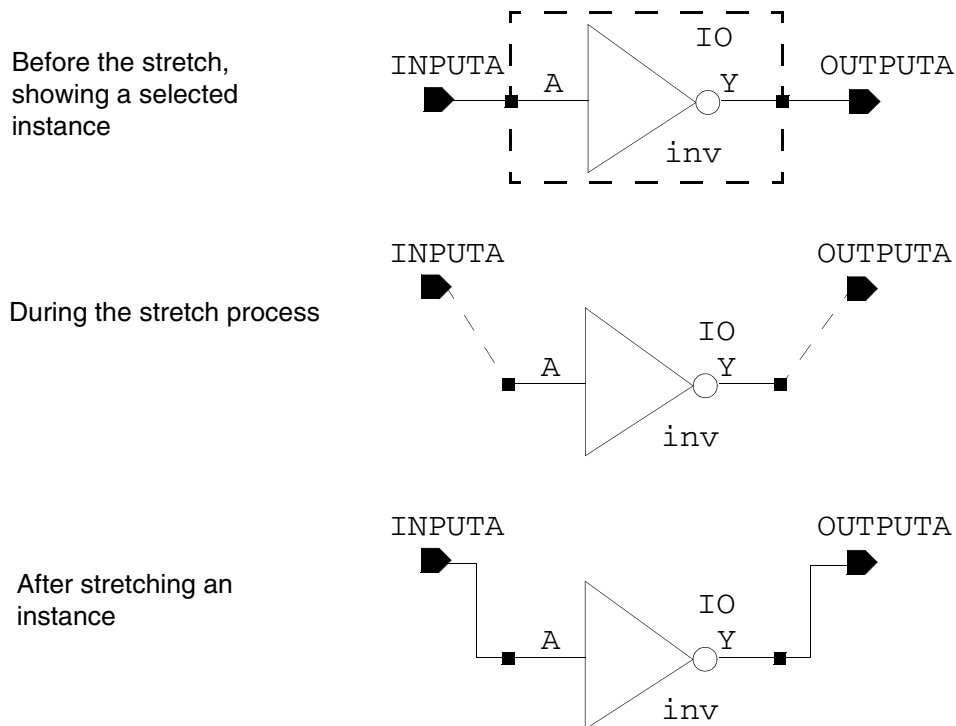


Figure 7-6 Wire option checked so all wire segments will be rerouted

Stretching an Instance

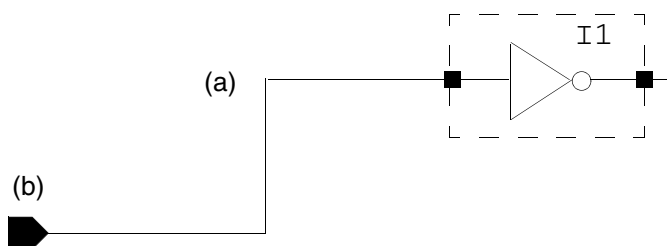
The following diagram shows how to stretch an instance using the default settings (*Route Method: full, Snap Mode: anyAngle*).



To reroute wire segments,

- On the Stretch form, turn on *Reroute Wire Segments Connected To Instance*.

The following diagram illustrates stretching an instance named I1.



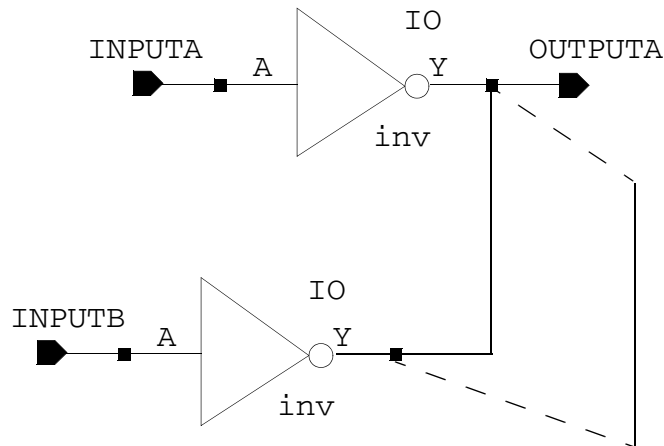
When you set *Reroute Wire Segments Connected To Instance* to *off*, the rubberband lines begin at the wire (a).

When you set *Reroute Wire Segments Connected To Instance* to *on*, the rubberband lines begin at the pin (b).

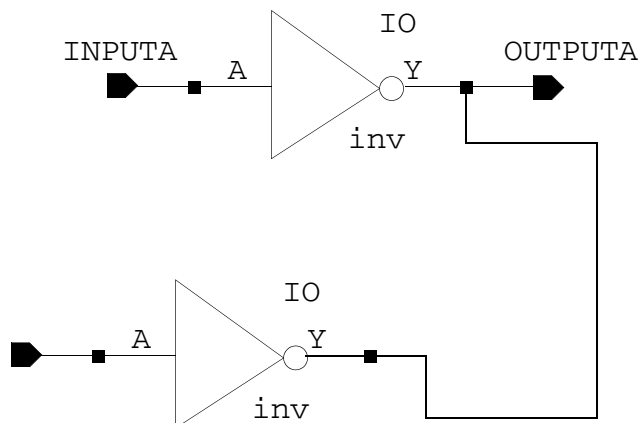
Stretching a Wire

The following diagram shows how to stretch a wire using the default settings.

During the stretch process



After the stretch process

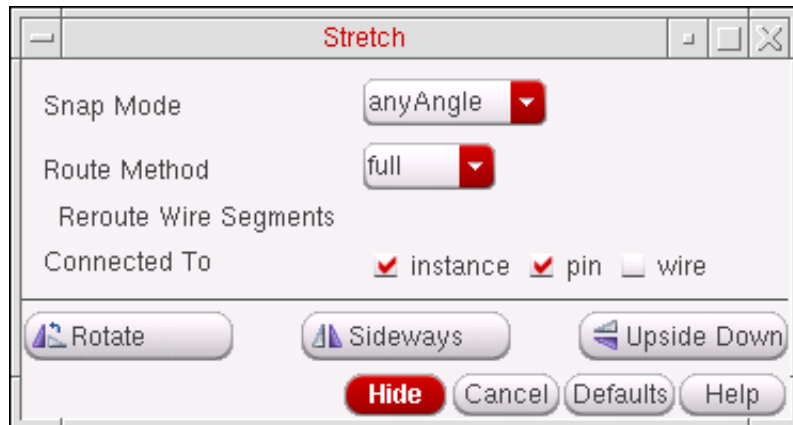


You can stretch a wire in any direction. The wire is rerouted and connectivity is maintained. When you stretch a wire, all wire segments that are connected to another object (such as a pin) stretch.

Stretching Using Route Method Simple

Route method *simple* is one of three routing methods available from the Stretch form by selecting *Edit – Stretch*.

For more information on the options in this form see [Stretch – Schematic View](#).



Setting *Route Method* to *simple* means that only those wires that are connected to instances, and that are also aligned with the stretch direction, will get re-routed when the instance is stretched. Routing will only be attempted along a single horizontal or vertical line.

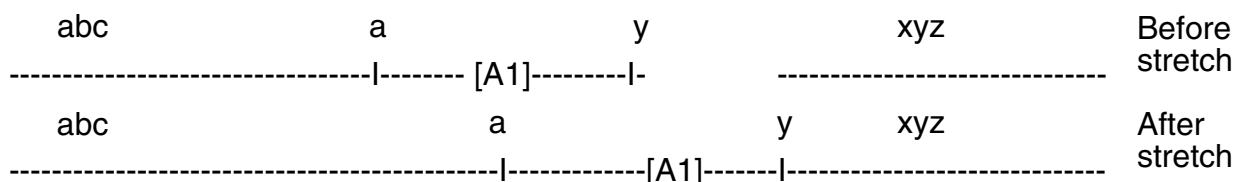
Note: Only when *Route Method* is set to *full* will the router attempt to perform point-to-point routing by considering all of the available manhattan paths. In this case, the router will attempt to complete the routing by creating equal width segments that are connected to each other.

Stretch Mode Simple Examples

When using the *simple* routing method, you should be aware of how the *simple* algorithm can operate and what potential “side effects” could occur.

Note: Route method *simple* does not guarantee to maintain connections in all circumstances.

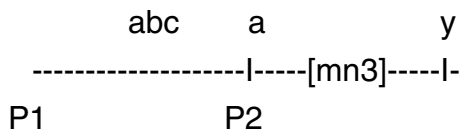
■ *Stretch example 1:*



If you stretch `A1` to the right so that pin `y` just touches the left end of wire `xyz`, this will result in the instance pin `y` being auto-connected to wire `xyz`.

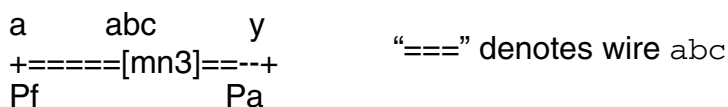
■ *Stretch example 2:*

The examples below attempt to illustrate what would occur if you tried to stretch instance `mn3` to the left of wire `abc`.



The *right* end of the wire is known as the *anchor point* (this will be referred to as `Pa`) since it would remain in its original position.

When you select `mn3`, stretch it to its left, and then place it such that its instance pin `y` is still to the right of `Pa`, the wire `abc` will be re-routed between `Pa` and instance terminal `a`. This is illustrated below:



Note: If you made an attempt to stretch instance `mn3` to the left so that instance pin `y` is also to the left of `Pa`, then the router, while re-routing wire `abc` between instance pin `a` and point `Pa`, will uncover that the yet to be created wire would be intercepted by instance terminal `y`. In this situation, a failure will occur and a flight wire would be created instead. If you performed the same stretch using the *full* route method, the router would route the wire from pin `a` to the point `Pa` by creating a wire loop around the instance. However, when attempting this in *simple* mode, the router will not do this.



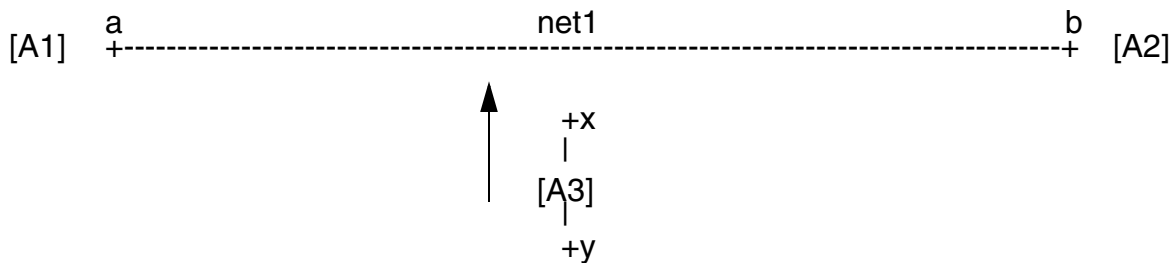
Potential Impact on Circuit Topology

When using the Stretch command care should be taken so that you do not inadvertently cause the circuit topology to be altered. For example, a previously unconnected pin could be moved on top of a wire causing a connection to be made.

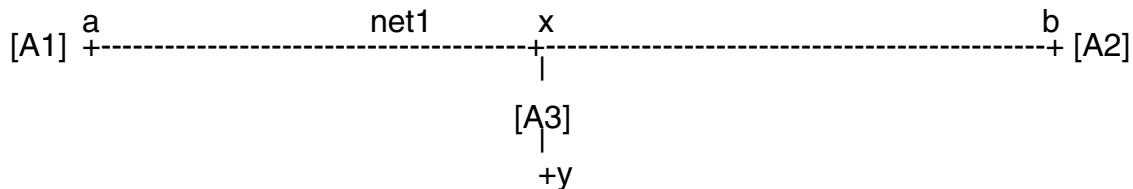
□ Topology impact example:

The example below describes an instance being moved in such a manner. The unconnected pin is dropped onto a net which is already connecting instance pins.

This behavior can cause shorts in your design.



If [A3] is moved so that x coincides with net1, a connection will be made.



Copying

Use the *Copy* command to clone an object in your design.

The *Copy* command

- Copies objects in the same cellview or another cellview
- Copies arrays of objects
- Rotates an object or turns it upside down or sideways
- Works best when infix mode is turned on
- Works differently when pre- and postselecting an object before copying it
- Works easiest and fastest using direct manipulation.

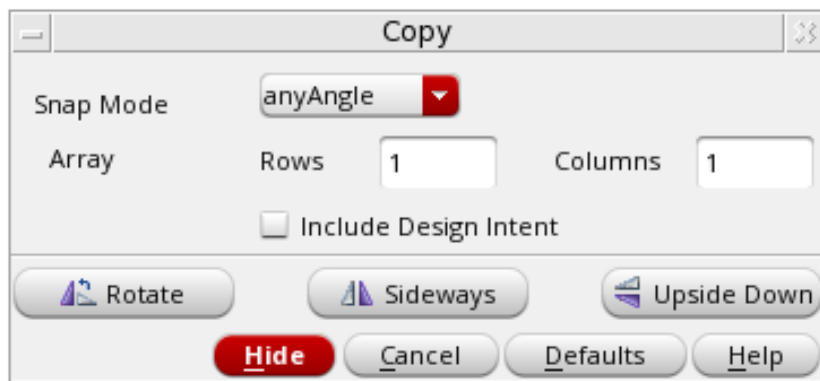
Note: When you select an object or drag your cursor through the design to select several objects for copying, all the components associated to an object are selected. For example, if you select a wire for copying, the associated wire name will also be selected with the wire. You can filter the selection by selecting the appropriate option from the Schematic Selection Filter form.

Copying Single Objects

To copy an object,

1. From the view, choose *Edit – Copy*.
2. Select the object you want to copy.
3. Click a destination for the copy.
4. To change *Copy* command options, press F3.

The Copy form appears.



Copying Arrays of Objects

To copy an array,

1. From the view, choose *Edit – Copy*.
2. Select the object you want to copy as an array.

The Copy form appears. If the form does not appear, press F3.



Virtuoso Schematic Editor User Guide

Editing Objects

3. Fill in the number of rows and columns for your array.
4. Click multiple points to create the array.
5. Select *Include Design Intent* if the design intent associated with the object is also to be included in the copy.

Copying Across Cellviews

You can copy objects from one cellview to another. Copied objects remain attached to the cursor when you move the cursor into another window containing a schematic or symbol cellview. You can copy data from a read-only schematic to a schematic opened for edit.

To copy an object from one cellview to another,

1. From the view, select the object to copy.
If *Infix* is off, you are prompted to first click a reference point for the copy.
2. Choose *Edit – Copy*.
3. Move the cursor to the destination cellview.
4. Click a location in the cellview for the copy.

Copying with Direct Manipulation

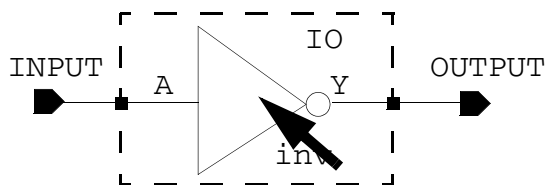
Direct manipulation is one of the quickest ways to copy an object.

To copy an object using direct manipulation,

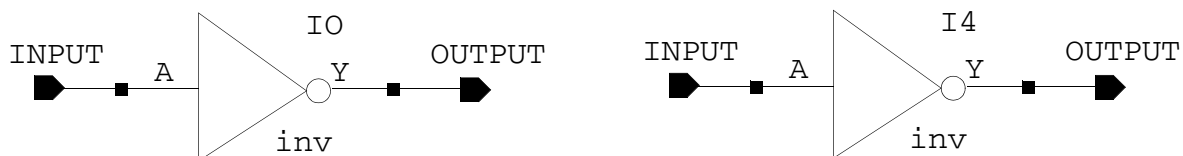
1. From the view, place the pointer over an object.
2. Press the `Shift` key, and click and drag the mouse.

When you release the mouse button, the original object is copied and placed.

Before copying



After copying



Copying with Bindkeys

To copy an object using bindkeys,

1. From the view, select the object to copy.
2. Press `c`.
3. Click a destination for the copy.

Copying Wires and Wire Names

When you copy a wire, any attached wire names are also copied with the wire. To disable this, deselect the *wire name* option in the Schematic Selection Filter form.

Moving

Use the *Move* command to reposition objects in your design. Unlike the *Stretch* command, the *Move* command does not maintain connectivity of the selected objects. The *Move* command:

- Moves an object without maintaining connectivity
- Rotates an object or turns it upside down or sideways
- Works best when infix mode is turned on
- Works differently when pre- and postselecting an object before moving it
- Works easiest and fastest using direct manipulation
- Moves selected objects between different schematic cellviews

You cannot move partially selected objects to a different cellview.

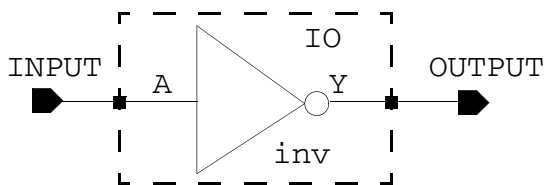
- When you work with wires, note shapes, or device shapes (rectangles, polygons, circles, ellipses, and arcs), you can select a vertex, set of vertexes, or edges; when you work with partially selected objects, only the selected vertexes move while the other vertexes or edges are anchored in place.

Moving Objects

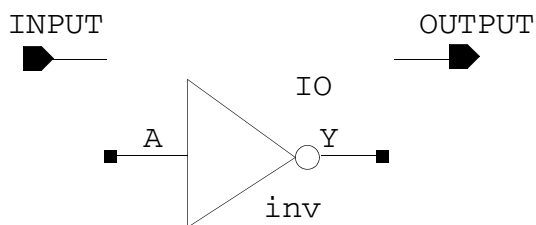
To move an object,

1. From the view, choose *Edit – Move*.
2. Select an object you want to move.
3. Click a destination for the object.

Before moving



After moving



To change *Move* command options,

- Press F3.

The Move form appears.



After a move, all connectivity breaks between the selected object and the objects to which it was connected.

If you want to maintain connectivity, use the Stretch command.

Moving with Direct Manipulation

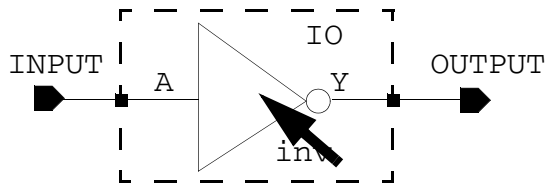
Direct manipulation is one of the quickest ways to move an object.

To move an object using direct manipulation,

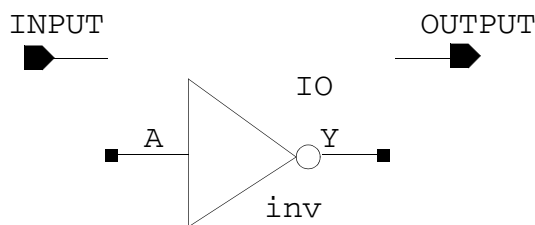
1. From the view, place the pointer over an object.
2. Press the `Control` key, and click and drag the mouse.

You are now in the *Move* command. When you release the mouse button, the object is placed.

Before moving



After moving



Moving with Bindkeys

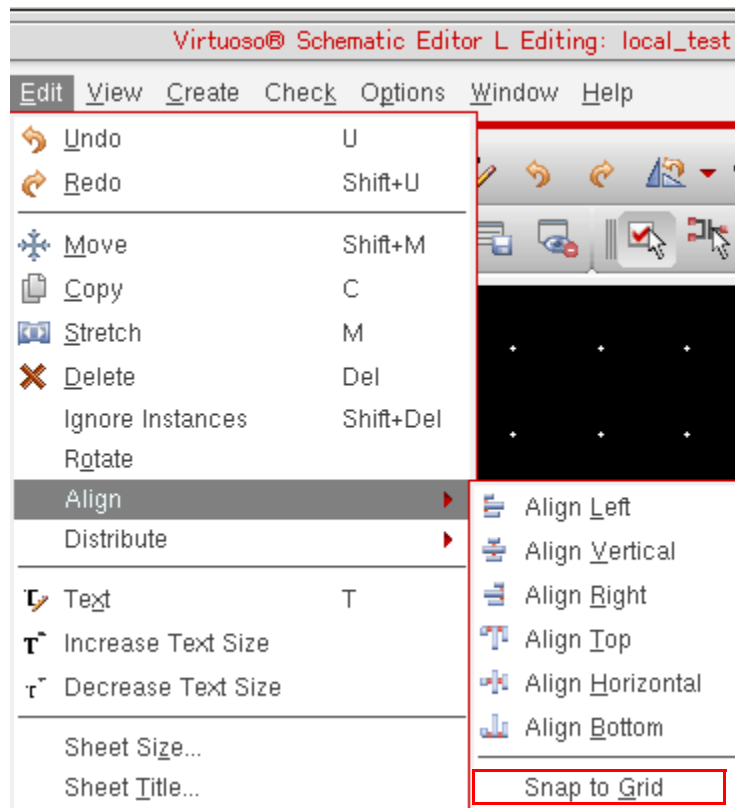
To move an object using bindkeys,

1. From the view, select the object to move.
2. Press `Shift-m`.
3. Click a destination for the move.

Snapping to Grid

While dragging the mouse on the schematic canvas to select various objects, such as nets, labels, instances, instTerms, pins, and so on, the mouse action fails to select those objects. This happens because the objects are not on the grid as per the current snap spacing setting.

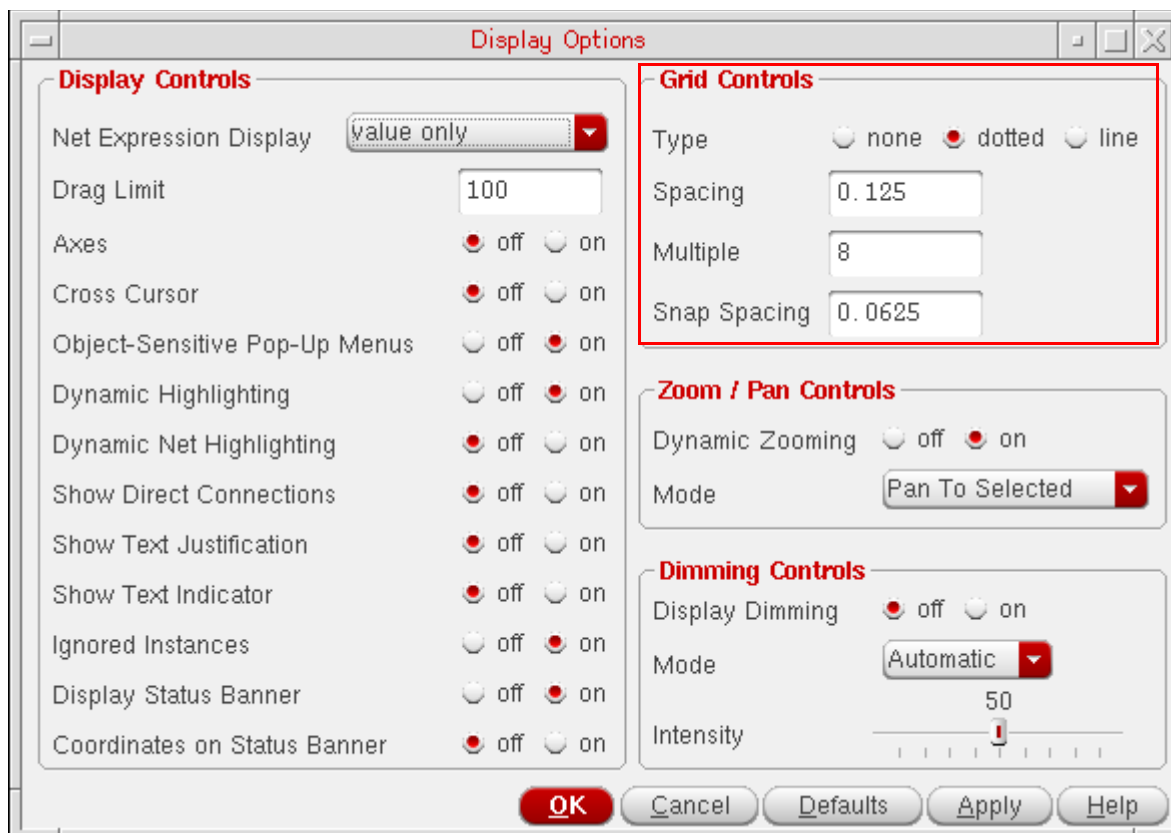
To place the off-grid objects on the given grid, select those objects on the schematic or symbol and click the *Edit – Align – Snap to Grid* command.



Virtuoso Schematic Editor User Guide

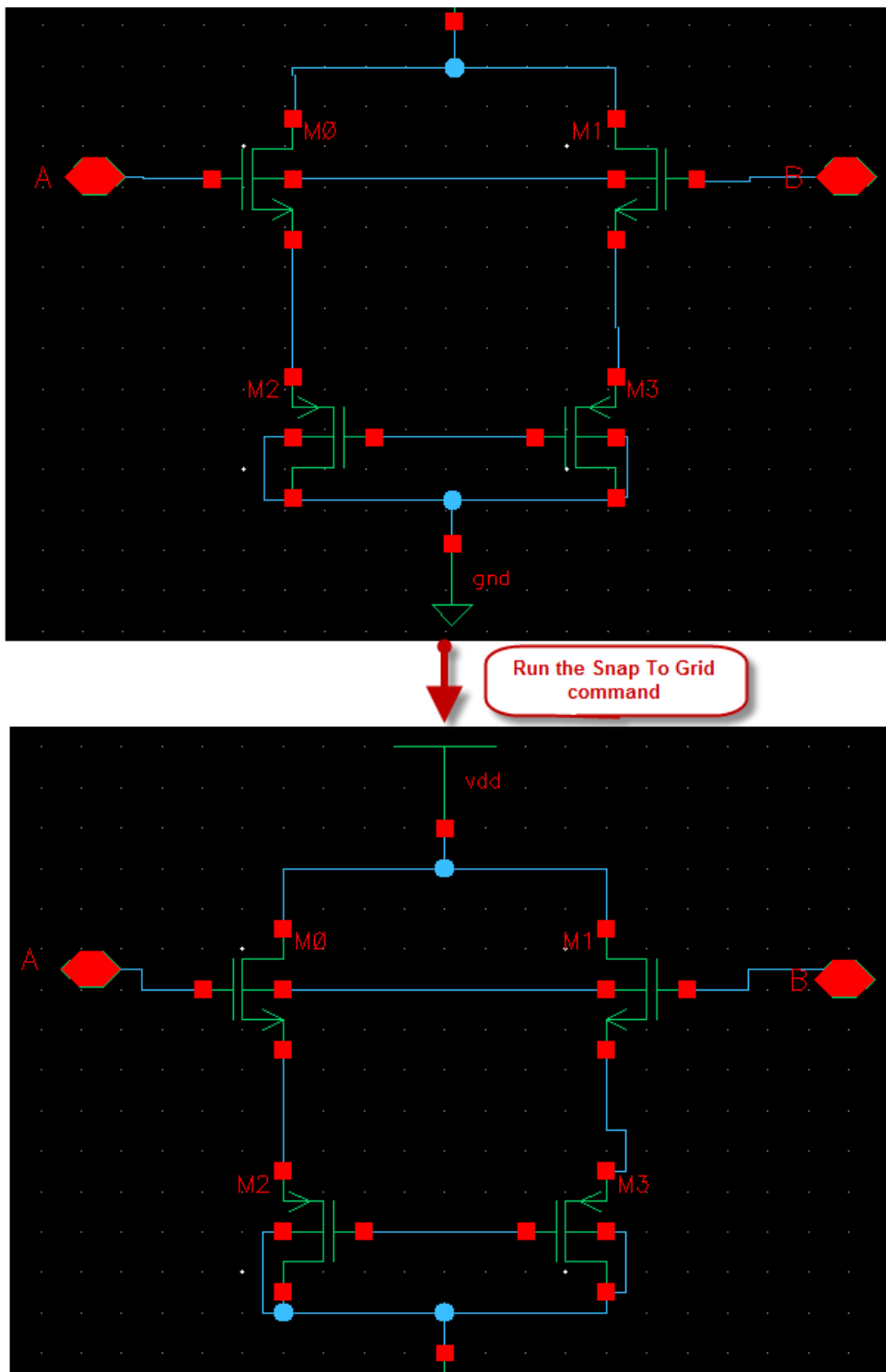
Editing Objects

To view the current snap spacing setting, click *Options – Display*.



Editing Objects

Mouse might also fail to select the objects if the schematic was created with one snap spacing setting and opened with a different snap spacing. Here is an example to show how the schematic view changes when the *Snap To Grid* command is applied.



Snapping Pre-Selected Objects to Grid

To snap the selected objects to a grid:

1. Select the objects in the schematic or symbol view.
2. Click *Edit – Align – Snap to Grid*.

Snapping Post-Selected Objects to Grid

To snap objects to a grid after enabling the *Snap to Grid* command:

1. Click *Edit – Align – Snap to Grid*.

The cursor changes to snap to grid mode () and the status bar displays the prompt to select objects.

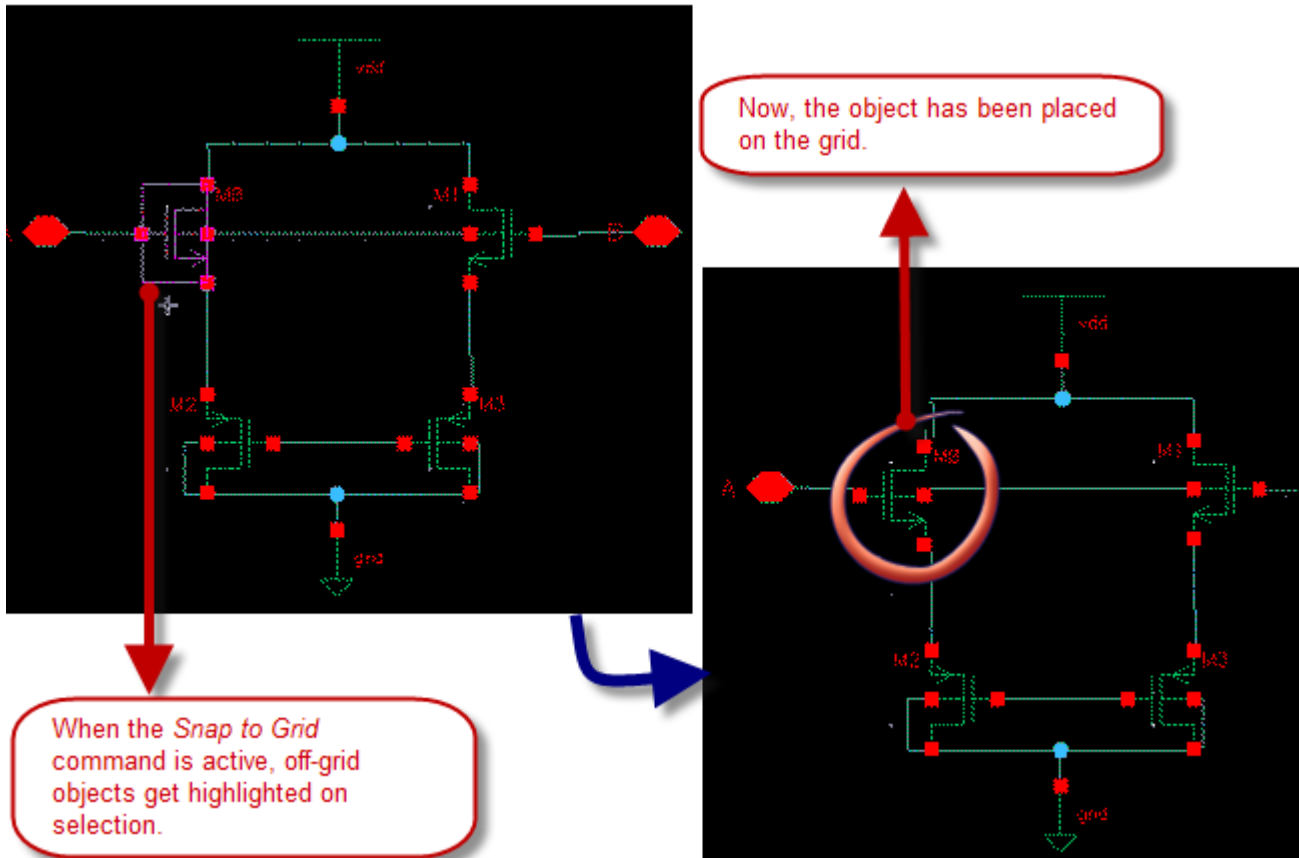


1(2) Select the objects for Snap To Grid (Double Click/Press Enter to Snap the objects to Grid) ...

Virtuoso Schematic Editor User Guide

Editing Objects

2. Select the objects in the schematic or symbol view. On selection, the off-grid objects get highlighted as shown below.



3. Double-click or press `Enter` to snap the selected objects to the grid.

Note: You can use the Selection Filter while running the *Snap to Grid* command.

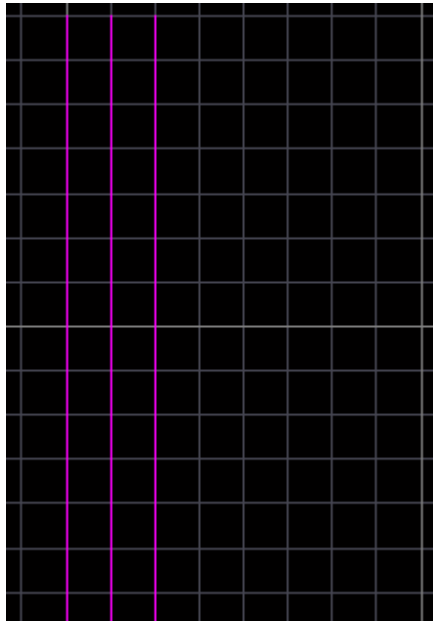
Limitations

The limitation of this functionality is that design objects, such as wires, pins, or instances, might overlap or routing and connectivity might break while changing the snap spacing setting. Therefore, to address such issues, you need to fix the design manually.

Example

Consider a scenario where the existing snap spacing is 0.0001 units. Therefore, three wires have been placed vertically equidistant at 1 snapSpacing = (0.0001 units) from each other as shown below.

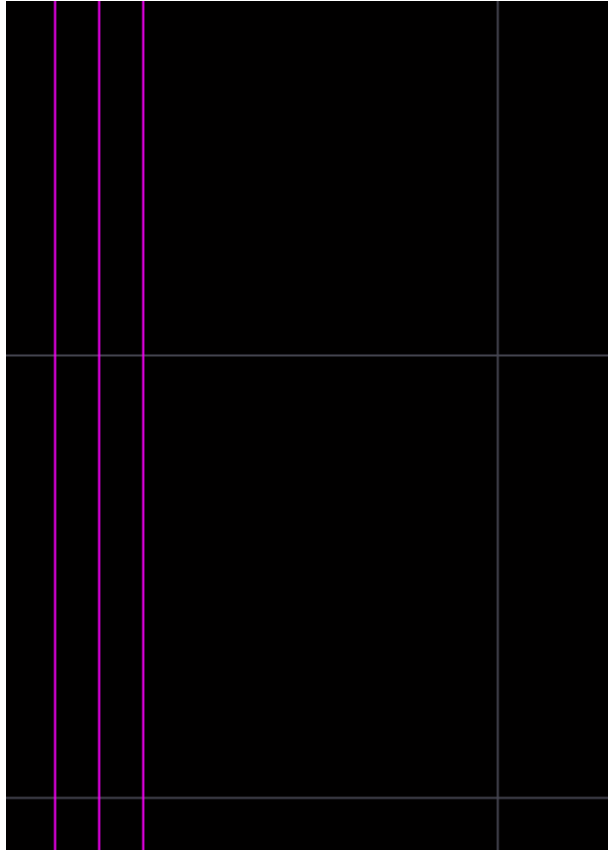
Note: The grid shown in this image is of 1 snapSpace.



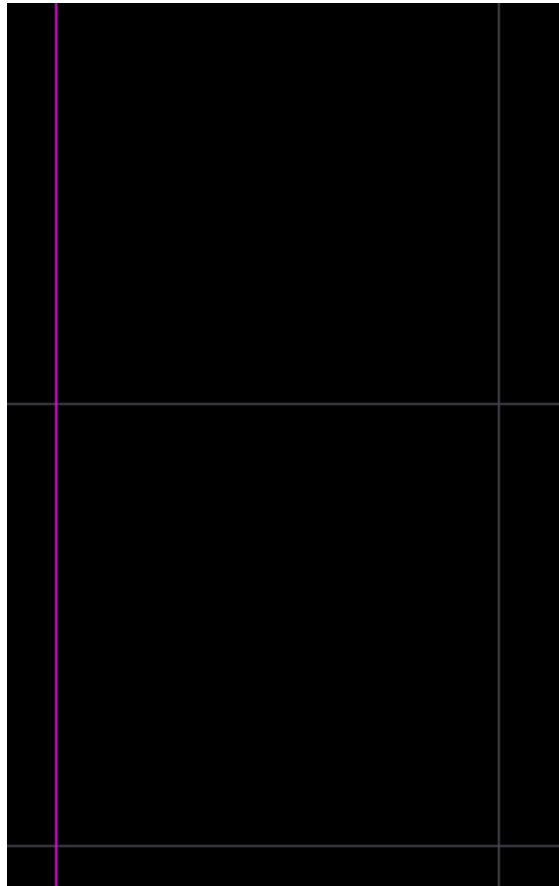
Virtuoso Schematic Editor User Guide

Editing Objects

Now, specify the new snapSpacing setting as 0.0625 units.



While snapping, all the wires being on the left half of the snapping grid coincide with the leftmost wire as shown below.



Snapping Instances to Grid Using SKILL Functions

You can use following SKILL functions to snap instances to a grid in the schematic:

- [`schHiSnapToGrid`](#)
- [`schSnapToGrid`](#)

Deleting

When you use the *Delete* command, keep in mind the following:

- You can select an object before you use the *Delete* command (preselection). When you preselect objects, the *Delete* command ends after the selected objects are deleted.
- You can select an object after you use the *Delete* command (postselection). When you postselect objects, press the `ESC` key to end the *Delete* command unless you want to delete another object. To be sure the *Delete* command is no longer active, check the prompt line at the bottom of the window in which the cellview is displayed.
- You can recover an object that you deleted by mistake by choosing the *Edit – Undo* command.
- You can delete an instance and the wires that connect to it by using extended selection with the *Delete* command.

You cannot use the *Delete* command to delete

- Sheet borders

To remove sheet borders, use the *Edit – Sheet Size* command.

- Selected visible properties

To delete properties, use the *Edit – Properties* command.

Note: When deleting objects from a cellview that contains design intents, to ensure there are no residual design intent items left on the canvas, remove the design intent from the object before deleting the object. For more details, see [Design Intent User Guide](#).

Deleting Preselected Objects

If you prefer to select an object before you delete it,

1. From the view, click the objects you want to delete.
2. Choose *Edit – Delete*.

The system removes the objects from your design.

Note: The *Delete* command always ends after it operates on preselected objects.

You can also delete objects by pressing the `Delete` key.

Deleting Postselected Objects

If you prefer to use the *Delete* command first and then select the object you want to delete,

1. From the view, choose *Edit – Delete*.
2. Click the objects you want to delete.

Note: The *Delete* command remains active until you explicitly cancel it by pressing the `Esc` key or by starting another command.

You can also delete objects by pressing the `Delete` key.

Deleting Sheet Borders

To delete the sheet border from a schematic sheet,

1. From the schematic window, choose *Edit – Sheet Size*.

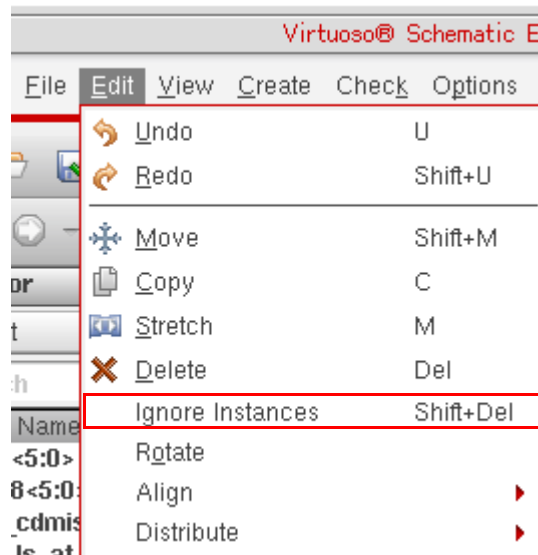
The Change Sheet Border Size form appears.



2. Set the *Border Type* to `none`.
3. Click *OK*.

Ignoring Instances

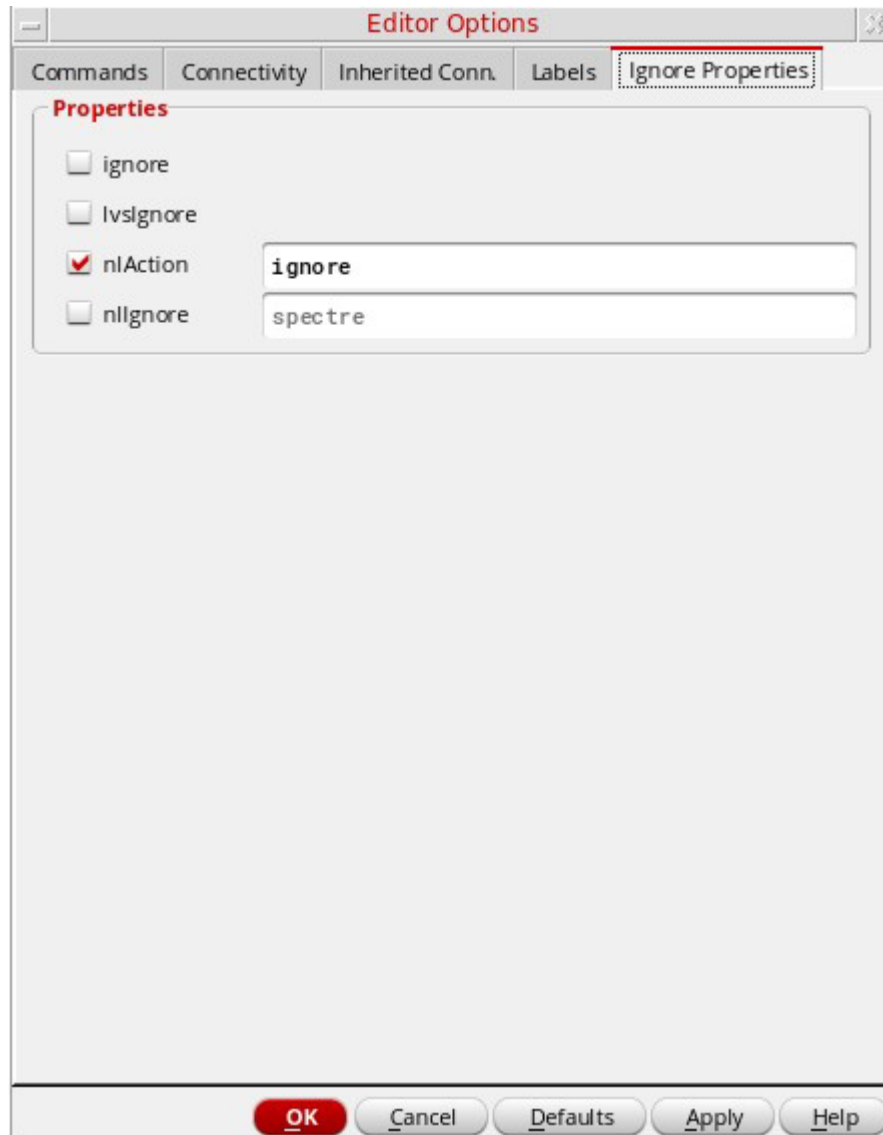
Select the instances on the schematic and choose *Edit – Ignore Instances*. It is used for ignoring, that is, adding ignore properties to the selected instance and removing ignore properties from the selected instance. Therefore, this command is used to toggle the ignored status of the given instance.



Virtuoso Schematic Editor User Guide

Editing Objects

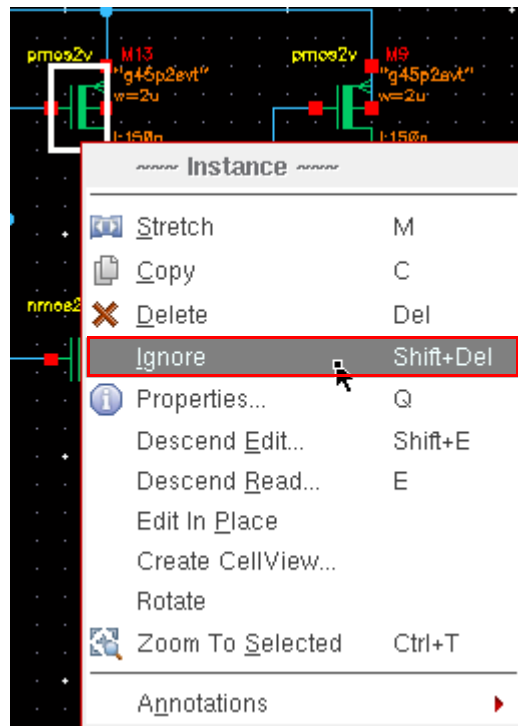
To view the registered and enabled ignore properties, use the *Ignore Properties* tab of the Editor Options form, which is displayed when you choose *Options – Editor*. You can also select or deselect the check boxes to enable or disable the ignore properties for an instance.



Virtuoso Schematic Editor User Guide

Editing Objects

Alternatively, to ignore a selected instance, right-click the instance and choose *Ignore* as shown below. This menu item also works as a toggle switch to change the status of an instance if it was previously ignored.



Note: If the *Ignored Instances* option is currently disabled on the Display Options form, the *View Ignored Instances* dialog box is displayed providing you with the option to enable ignored instances. You can enable the display of ignored instances at any time by selecting *View – Ignored Instances*.

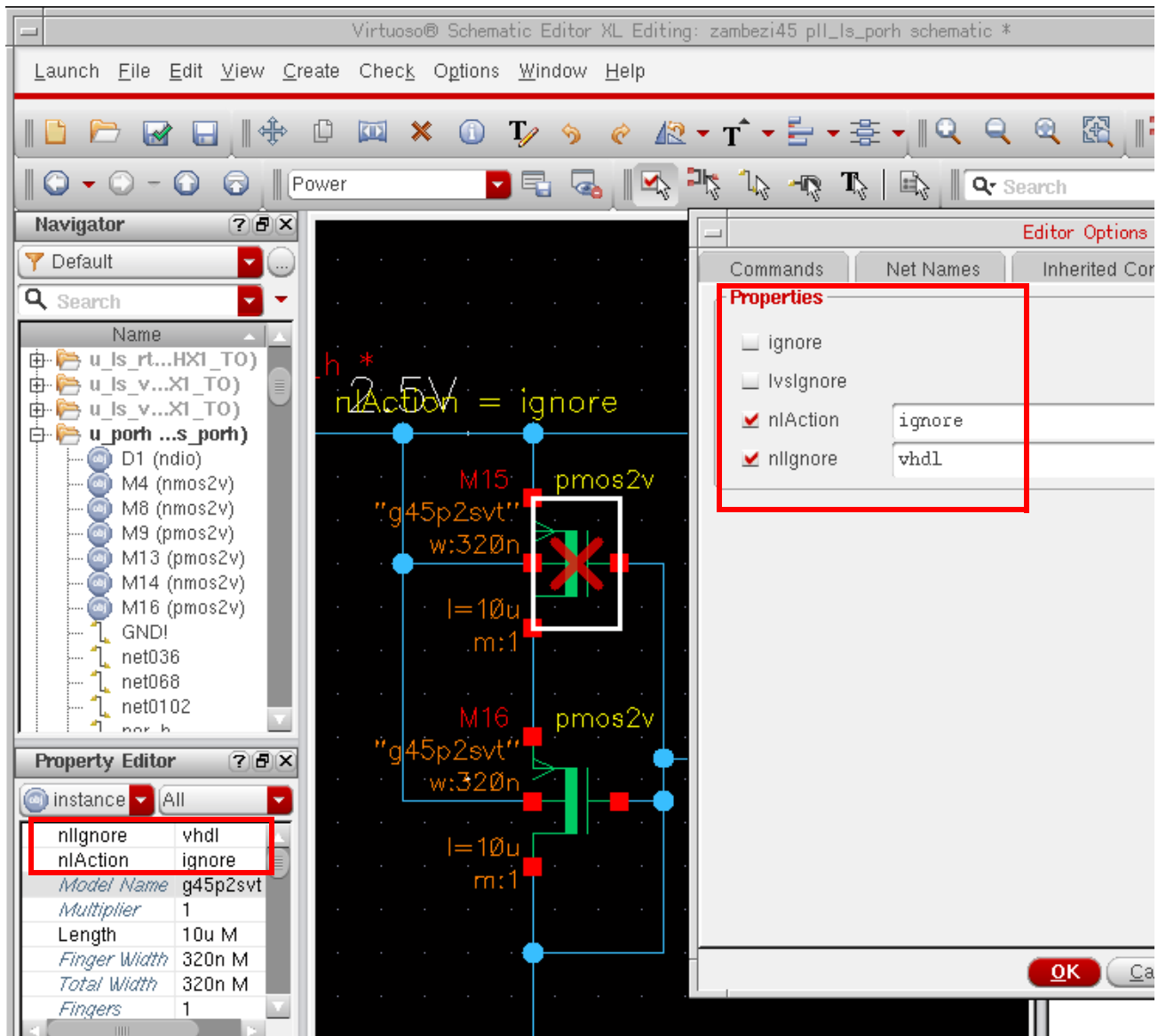
The ignored instances appear with a red cross mark drawn over them in the schematic. The cross mark that appears on the ignored instances has three states based on the ignore properties selected in the *Editor Options* form:

- No cross: None of the enabled ignore properties are found on the instance.

Virtuoso Schematic Editor User Guide

Editing Objects

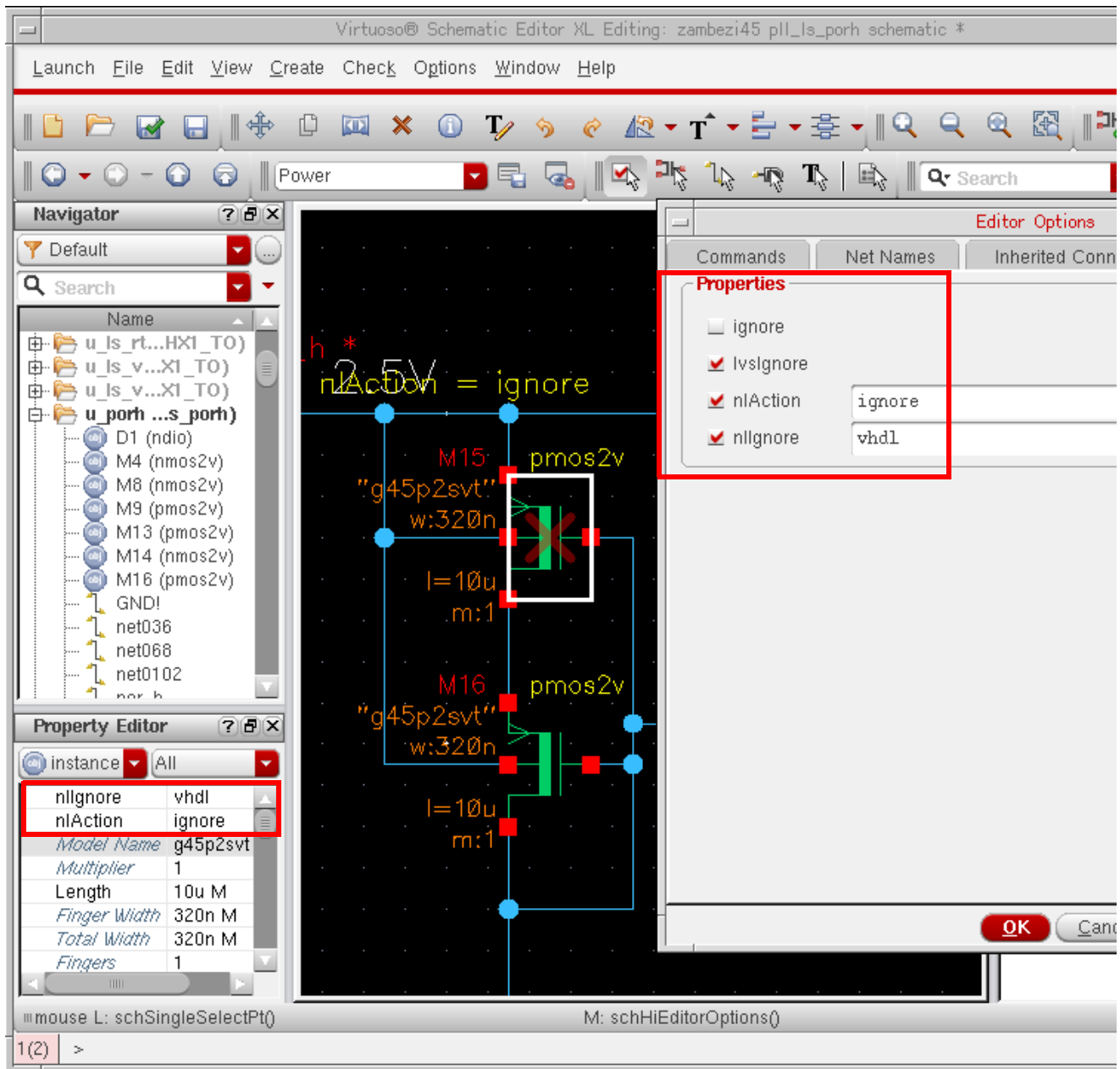
- Solid cross: All of the properties that are selected in the UI are also found on the instance, as shown below.



Virtuoso Schematic Editor User Guide

Editing Objects

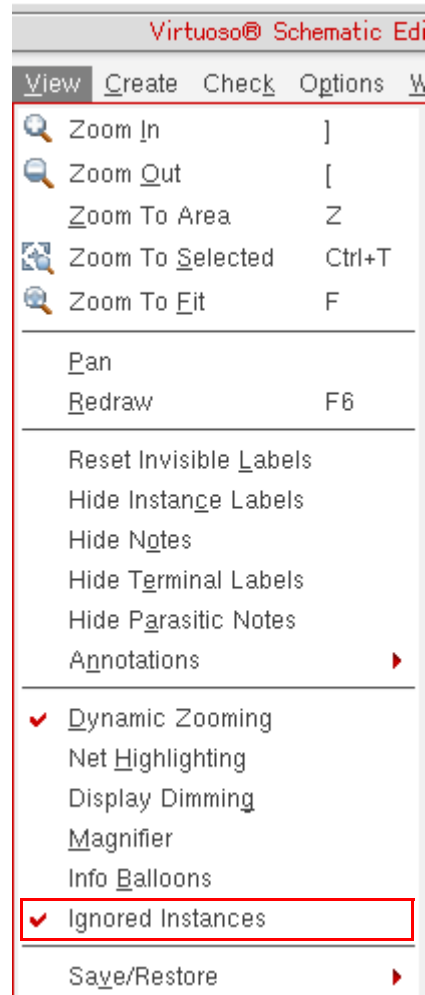
- Semi-transparent cross- Only few enabled ignore properties in the UI are found on the instance.



Virtuoso Schematic Editor User Guide

Editing Objects

The *View — Ignored Instances* menu item displays all ignored instances in the schematic.



Ignoring Pre-Selected Instances

To ignore instances using the ignore option:

1. Select the instance(s) in the schematic view.
2. Click *Ignore Instances* in the *Edit* menu.

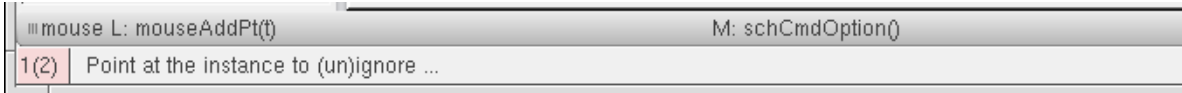
Ignoring Post-Selected Instances

1. Select *Ignore Instances* in the *Edit* menu.

Virtuoso Schematic Editor User Guide

Editing Objects

The cursor changes to ignore instance mode and the status bar displays the prompt to select instances for ignoring.



2. Select the instance(s).

3. Double-click or press `Enter` key to ignore the selected instances.

Note: When the ignore command is running, you can select only instances in the schematic.

Ignoring Instances Using SKILL Functions

You can use following SKILL functions to ignore instances in the schematic:

- [`schHilgnore`](#)
- [`schIgnore`](#)
- [`schRemoveIgnoreProp`](#)
- [`schGetAllIgnoreProps`](#)
- [`schAddIgnoreProp`](#)
- [`schFindIgnorePropByName`](#)
- [`schSetIgnorePropEnabled`](#)
- [`schGetIgnoredStatus`](#)

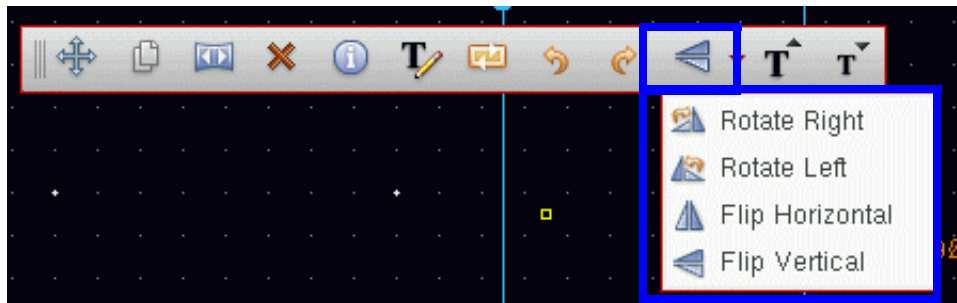
Rotating

The *Edit – Rotate* command rotates or flips objects

- *Rotate Right* turns objects 90 degrees clockwise.
- *Rotate Left* turns objects 90 degrees counter-clockwise.
- *Flip Horizontal* flips objects from side to side.
- *Flip Vertical* flips objects from top to bottom.

Rotating Using the Edit Toolbar

In addition to the *Edit – Rotate* option in the menu bar, you can access the rotation functions from the *Edit* toolbar.



To rotate an object using the rotate options on the *Edit* toolbar:

1. Select the rotation option that you want to use from the *Edit* toolbar.

Your cursor will change to be in object rotation mode, and the status bar will prompt you to *Point at reference point for rotate*.

2. Click the object that you want to rotate or flip.

The object will rotate as required.

3. Repeat as required (for *Rotate Right* and *Rotate Left*).

Rotating Preselected Objects

To rotate a preselected object,

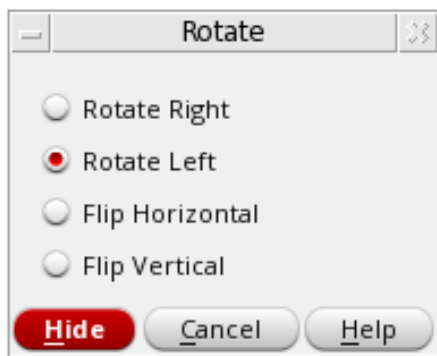
1. From the view, click the object you want to rotate.
2. Choose *Edit – Rotate*.
3. Point at a reference point in the design and click.

The selected object rotates and the command terminates.

If you want to change how the software rotates objects,

1. From the view, choose *Edit – Rotate*.

The Rotate form appears. If the form does not appear, press F3.



2. Change the option setting.

Rotating Postselected Objects

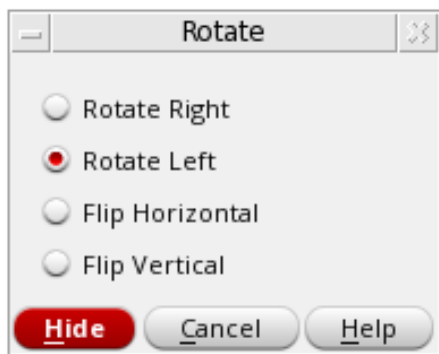
To rotate a postselected object,

1. From the view, choose *Edit – Rotate*.
2. Click an object.

If you want to change how the software rotates objects,

1. From the view, choose *Edit – Rotate*.

The Rotate form appears. If the form does not appear, press F3.



2. Change the option setting.

Rotating with Direct Manipulation

To rotate objects while in the *Move*, *Copy*, *Stretch*, *Create – Instance*, *Create – Pin*, *Create – Wire*, *Create – Note Text*, or *Create – Label* command,

- Click the right mouse button.

Rotating with Bindkeys

To rotate or mirror (flip horizontally or sideways) an object while in the *Move*, *Copy*, *Stretch*, *Create – Instance*, *Create – Pin*, *Create – Wire*, *Create – Note Text*, or *Create – Label* command,

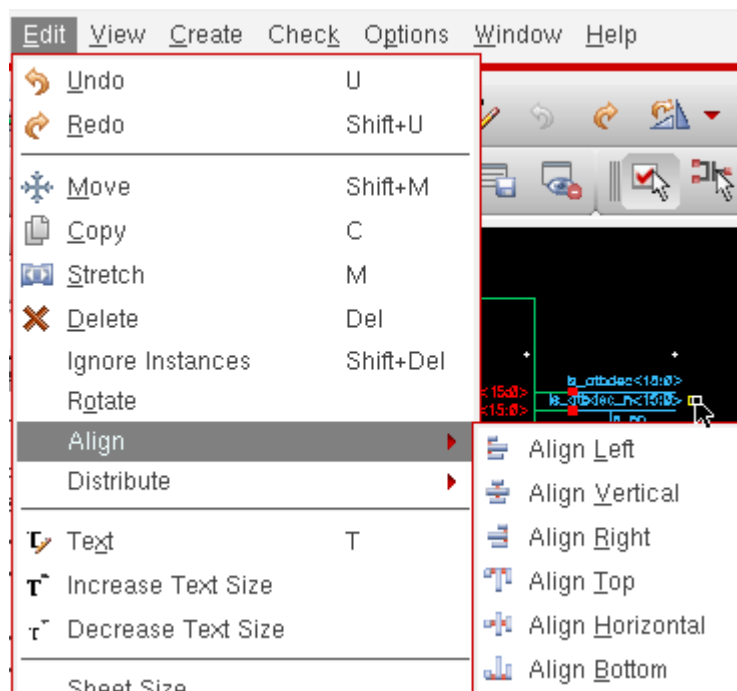
1. From the view, select an object.
2. Press one of the following:
 - `r`, to rotate 90 degrees counterclockwise
 - `Shift-r`, to flip horizontally/rotate sideways
 - `Control-r`, to flip vertically/rotate upside down

Aligning

The *Edit – Align* command aligns the selected objects in one of the following directions: right, left, bottom, top, horizontal, or vertical.

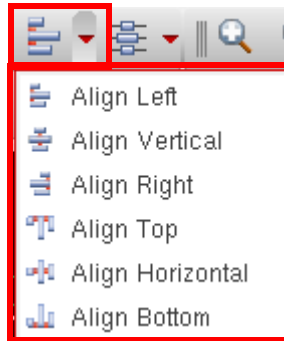
Video

For an overview of how alignment and distribution can be done in a schematic, see [*Editing Enhancements in Virtuoso Schematic Editor*](#) on Cadence Online Support.



Aligning Using the Edit Toolbar

Alternatively, you can access the align options from the *Edit* toolbar.



Aligning Pre-Selected Objects

To align pre-selected objects:

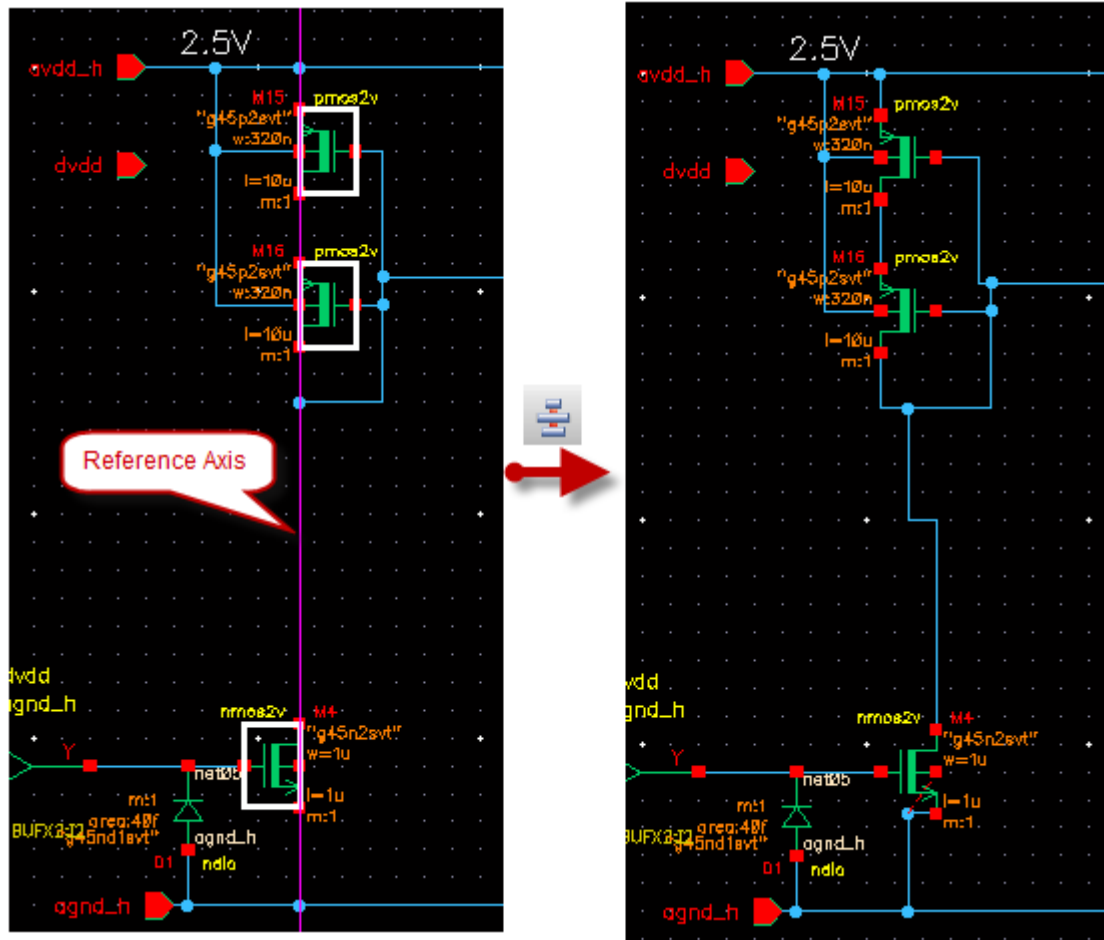
1. Select the objects in the schematic or symbol view.
2. Click the align option that you want to use from the *Edit* toolbar to get the objects aligned.

Align Left aligns all the selected objects vertically with the left-most edge of the object selection area.

Virtuoso Schematic Editor User Guide

Editing Objects

Align Vertical aligns the selected objects vertically with the reference axis on the schematic canvas, as shown below.



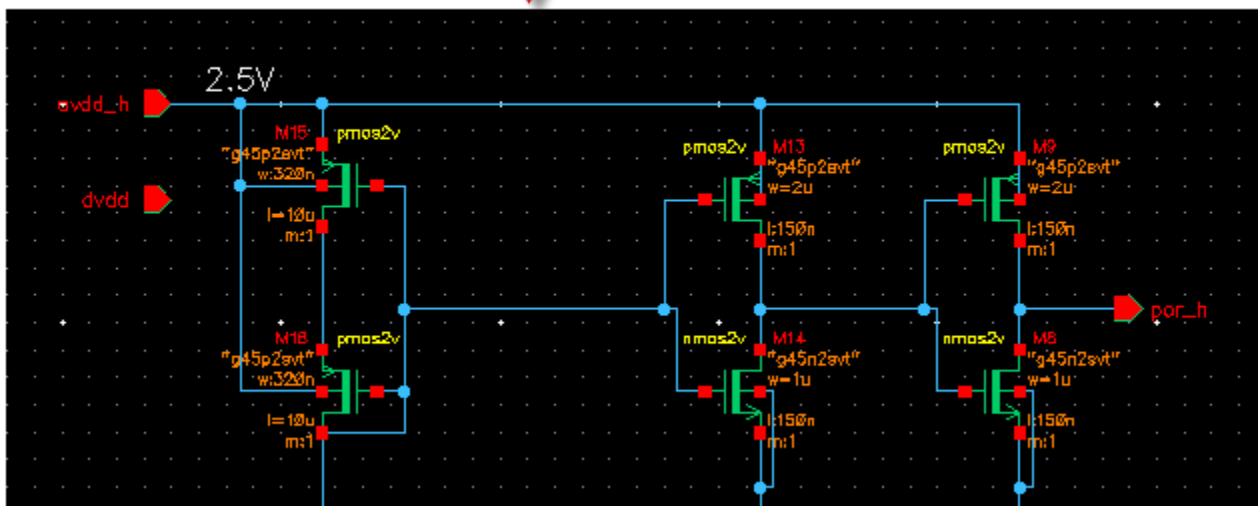
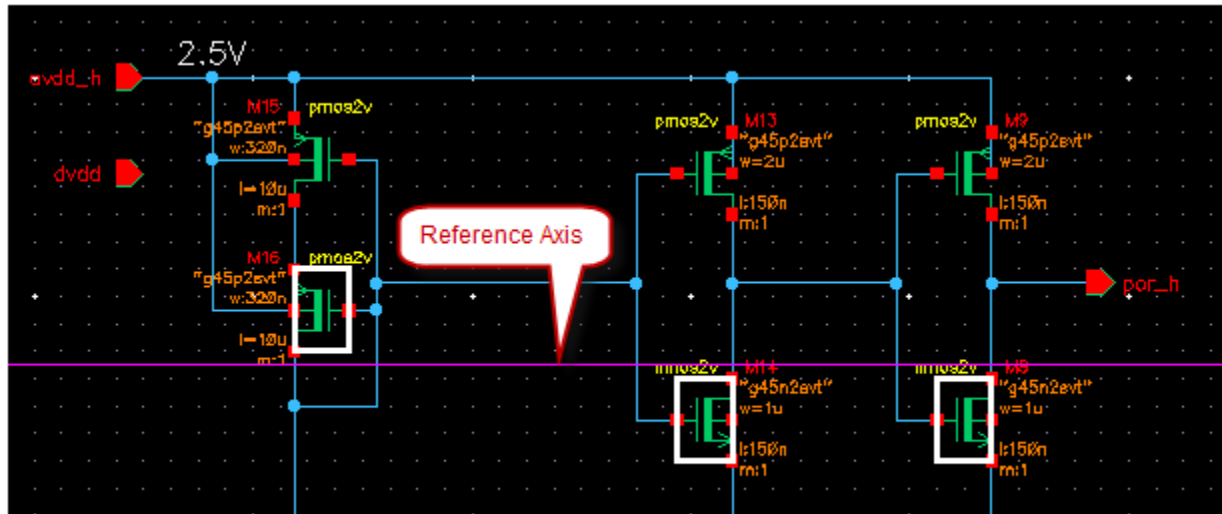
Align Right aligns all the selected objects vertically with the right-most object selected on the schematic canvas.

Align Top aligns all the selected objects horizontally with the top-most object selected on the schematic canvas.

Virtuoso Schematic Editor User Guide

Editing Objects

Align Horizontal aligns the selected objects horizontally with the reference axis on the schematic canvas, as shown below.



Align Bottom aligns all the selected objects horizontally with the bottom-most object selected on the schematic canvas.

Aligning Post-Selected Objects

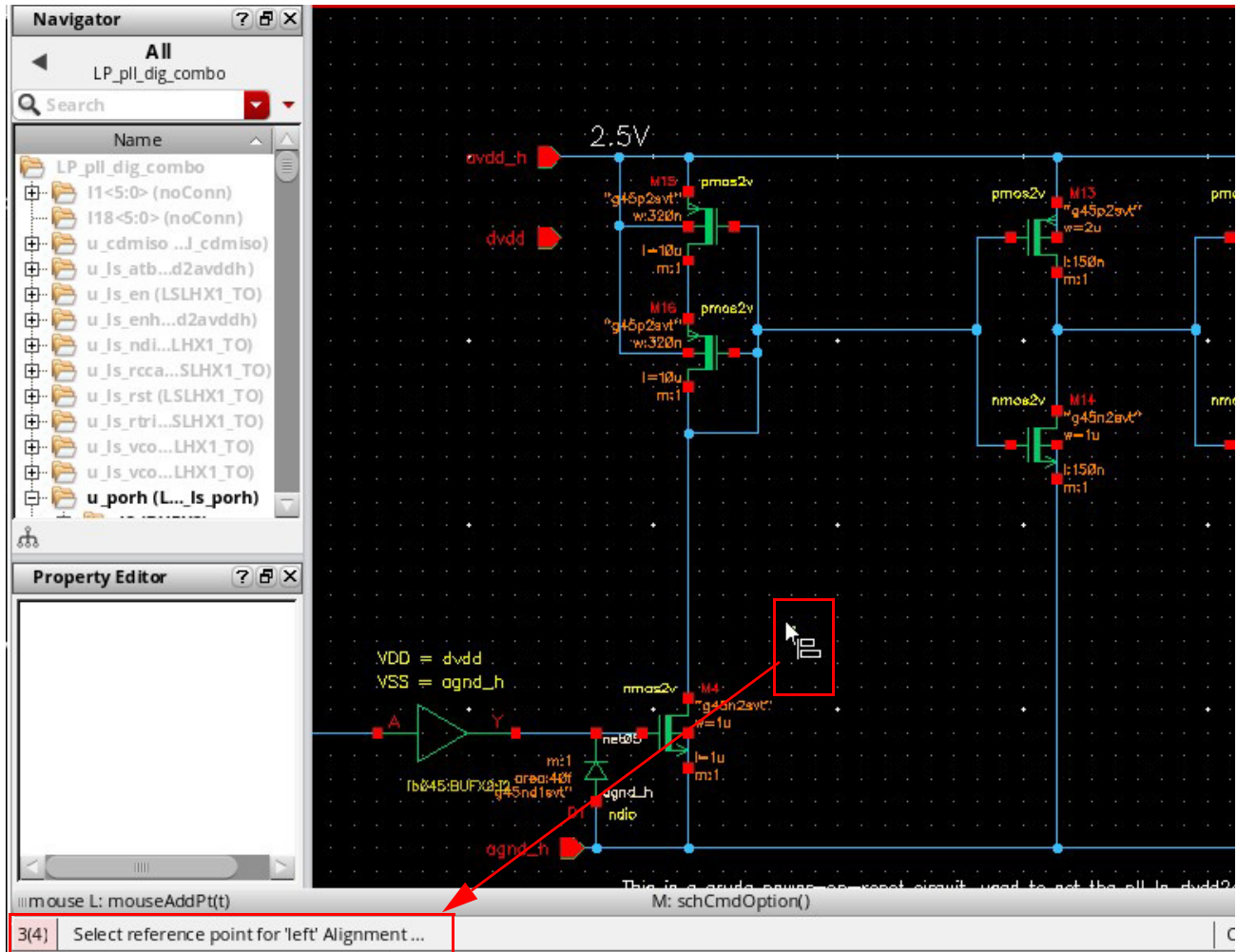
To align post-selected objects:

1. Select the align option that you want to use from the *Edit* toolbar.

Virtuoso Schematic Editor User Guide

Editing Objects

The cursor changes to object alignment mode and the status bar displays the prompt to select the reference point for alignment in the specified direction.

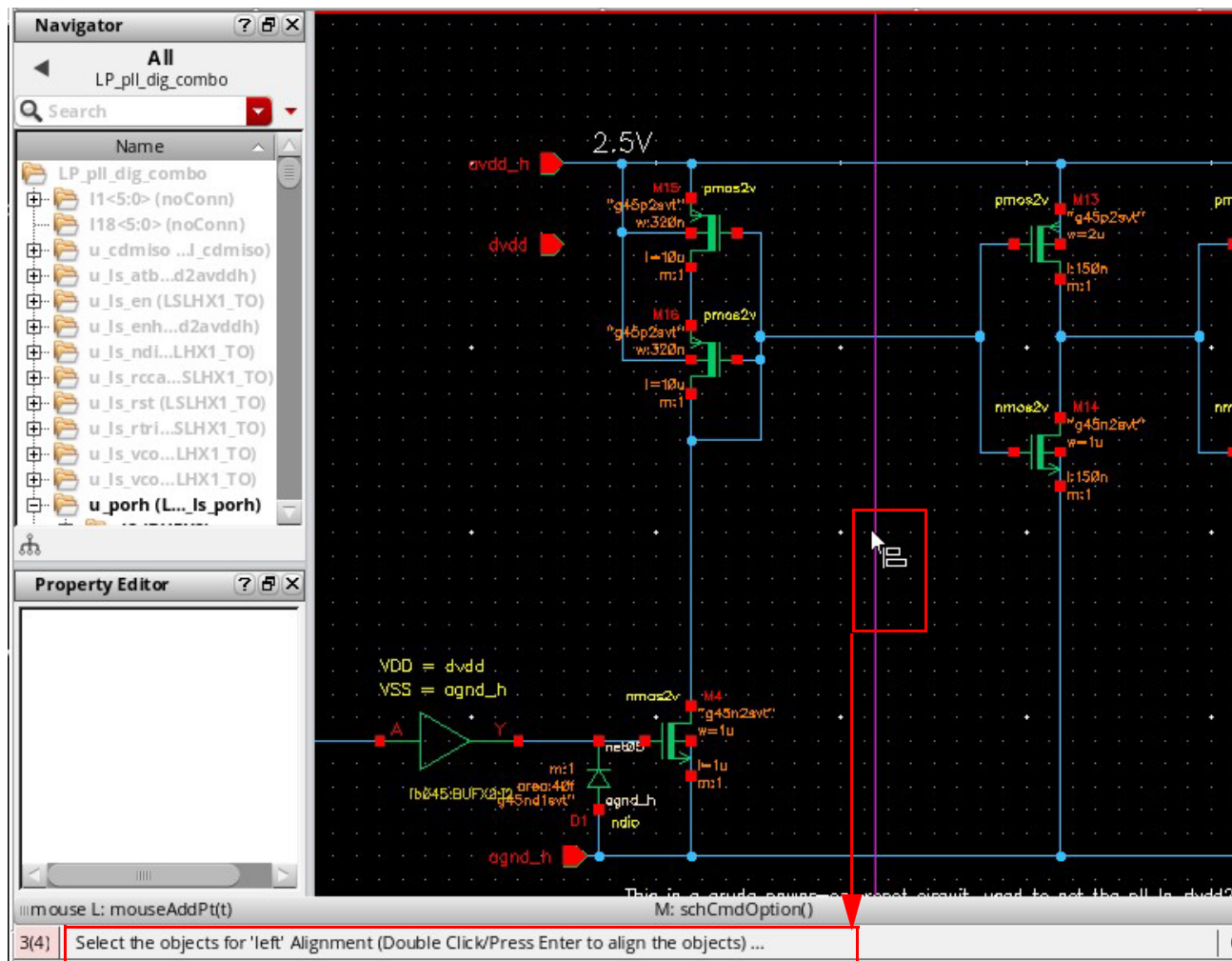


2. Select the reference point for alignment.

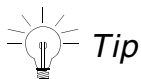
Virtuoso Schematic Editor User Guide

Editing Objects

The reference axis is displayed on the canvas according to the alignment direction, and the status bar displays the prompt to select objects for alignment in the specified direction.



3. Select the objects that you want to align.
4. Double-click or press the Enter key to align the objects in the required direction.



You can change the alignment direction anytime when the command is active. Use appropriate selection filter set to align the required objects and get the best results.

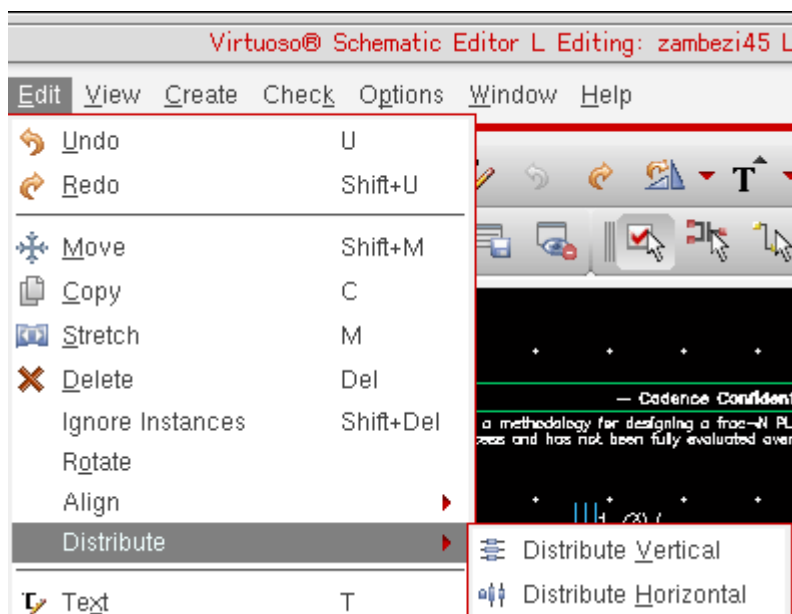
Aligning Using SKILL Functions

You can use the following SKILL functions to align objects in the schematic:

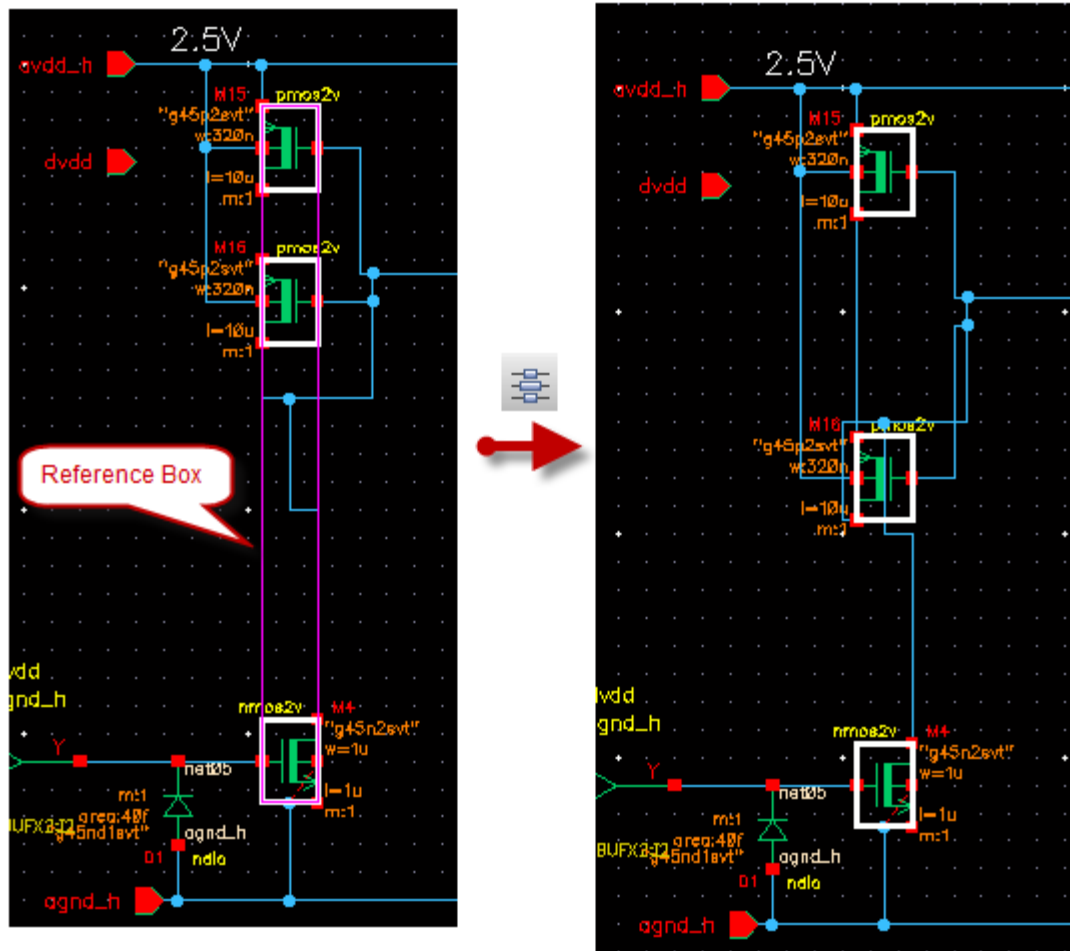
- `schHiAlign`
- `schAlign`

Distributing

The *Edit – Distribute* command arranges the selected objects at equal distance, vertically or horizontally.



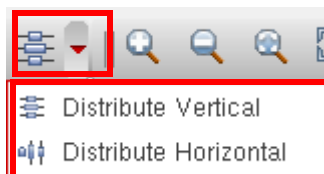
- *Distribute Vertical* distributes the objects vertically along the reference box, as shown below.



- *Distribute Horizontal* distributes the objects horizontally.

Distributing Using the Edit Toolbar

Alternatively, you can access the distribute options from the *Edit* toolbar.



Distributing Pre-Selected Objects

To arrange pre-selected objects:

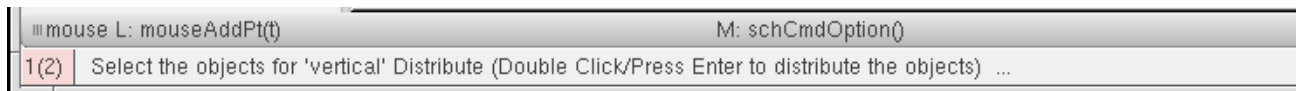
1. Select the objects in the schematic or symbol view.
2. Click the distribute option that you want to use from the *Edit* toolbar to place the objects at equal distance.

Distributing Post-Selected Objects

To arrange post-selected objects

1. Select the distribute option that you want to use from the *Edit* toolbar.

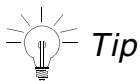
The cursor changes to object distribution mode and the status bar displays the prompt to select objects for distribution in the specified direction.



2. Select the objects to be distributed.

The reference rectangle is displayed on the canvas to encompassing all the selected objects.

3. Double-click or press the `Enter` key to distribute the objects in the specified direction.



Tip

You can change the distribution direction anytime when the command is active. Use appropriate selection filter set to distribute the required objects and get the best results.

Distributing Using SKILL Functions

You can use the following SKILL functions to align objects in the schematic:

- [schHiDistribute](#)
- [schDistribute](#)

Discarding Edits

To discard edits you made since the last save,

1. From the view, choose *File – Discard Edits*.

The Discard Edits dialog box appears.

2. Click *Yes* or *No*.

If you click *Yes*, the edits are discarded and the last version saved appears. If you click *No*, the edits are retained.

Alternating Symbol Views

When you create a schematic, you might find it convenient for a particular library cell to have more than one symbol representation. These different symbols might represent deMorgan equivalents or adhere to different drafting standards.

Use the *Edit – Alternate View* command to switch between the different symbol representations.

Changing the View of a Preselected Object

To change symbol views for a preselected object,

- From the schematic window, choose *Edit – Alternate View*.

Changing the View of a Postselected Object

To view symbol views for a postselected object,

1. From the schematic window, choose *Edit – Alternate View*.
2. Click an object in the schematic.

Continue clicking on the object to view all the alternative views. After displaying the last view, the command redisplay the first view.

Toggling Objects

Use bindkeys to toggle an object while in the *Create – Pin* or *Create – Wire* command.

Toggling Pin Direction Options

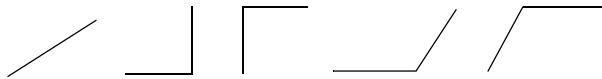
To toggle the direction of a pin (for example, from `input` to `inputOutput`),

1. From the view, choose *Create – Pin*.
2. Press the `Shift` key and click right to cycle through the options.



Toggling Wire Draw Mode Options

When you add a wire to your design, the wire segment is drawn using a draw mode that you specify on the Add Wire form. Use your mouse to cycle through the possible draw modes (shown below) to specify another mode.



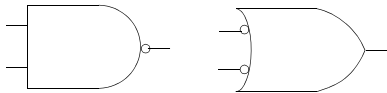
To toggle the draw mode,

1. From the schematic window, choose *Create – Wire*.
2. Use the right mouse button to toggle through the draw modes.
The wire will change form as you change the draw mode.
3. Click to indicate the first data point of the wire.
4. Move the cursor to the second point of the wire and click right.

Toggling Instance Symbol View Options

To toggle instance symbol views (for example, from a `nand2 symbol` view to a `nand2 symbolNeg` view),

1. From the schematic window, choose *Create – Instance*.
2. Press the `Shift` key and click right to cycle through the options.

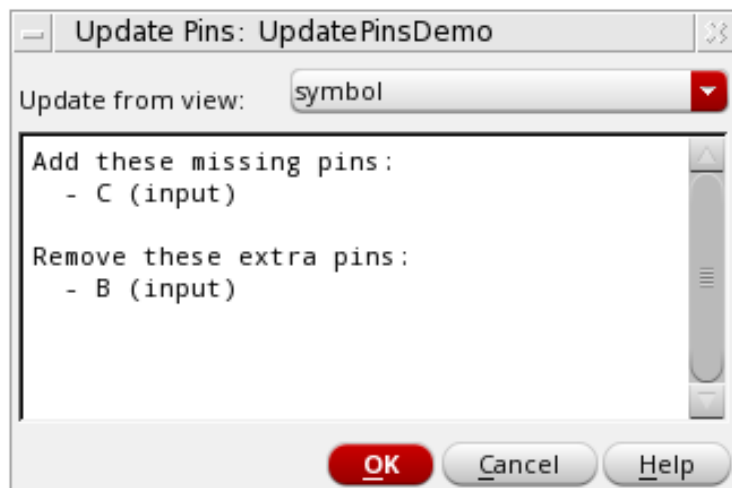


Updating Pins Across Cellviews

Use this form to update a schematic or symbol pin to match those in another view of the same cell. It is accessed using the *Edit – Update Pins From View* command.

1. From a schematic view, choose *File – Open Symbol* to open the corresponding symbol view.
2. Edit the symbol as required. For example, rename a pin from B to C.
3. Return to the schematic view and select *Edit – Update Pins From View*.

This will display the Update Pins form.



4. Select from the *Update from view* option the view from which you want to update the current view.

A default selection is automatically selected appropriate for the current cellview.

Note: The scrollable text area of the form displays the available updates from the selected view, in this example, the `schematic` view.

The updates that can be required to be made are:

- ☐ a list of pins that require to be added to the current view to sync-up with the selected view.
- ☐ a list of pins that require to be removed from the current view to sync-up with the selected view.
- ☐ specific pin directions that require to be changed in the current view to sync-up with the selected view.

5. Click OK to update the pin status of the current cellview.

Depending on what changes are to be made, the following updates will be performed:

- ☐ If there are pins to be added, the Add Pin form is invoked (select F3 to display and update the form if it is currently hidden). The *Pin Names* and *Direction* fields are automatically populated and you are prompted, at the status bar, to place the additional pins appropriately in the current cellview.
- ☐ Any pins to be deleted are automatically removed.
- ☐ Any pin direction changes required are automatically done.

You should check the CIW to confirm any additions, deletions, and changes made.

Note: You can also use the steps above to pull in the pins to the symbol view after changes have been made in a schematic view.

Annotating Data on the Schematic Window

You can annotate data onto the schematic to show the parameters, operating points, net names, currents and voltages of the design components. You can also change the existing annotations using the context menu or the *Annotation Setup* form. When you change the annotation settings for a design, the annotated data is updated at all levels of hierarchy in the design. You can configure the annotation data for editable as well as read only designs and the annotations of two separate designs are independent of each other even if there are common elements between them. In addition to displaying annotations on schematic canvas, you can use annotation balloons to display the annotation data on the schematic. Annotation balloons appear when you move the cursor over any instance on the schematic. The annotation balloons are useful for displaying the annotations on a dense schematic without

cluttering. For more information on annotation balloons, see [Using Annotation Balloons](#) on page 530. In order to annotate data onto the schematic canvas, the Label on the symbol view of the instance should be available. You can configure each Label individually to display different type of data using the *Annotation setup* form. For information on configuring data for annotation, see [Using Annotation Setup Form](#) on page 534. You can annotate the following data for each Label on the schematic:

- `cdsName()`: to display the instance name, cell name, library name, or any combination of library/cell/instance.
- `cdsTerm()`: to display the net name, pin name, DC voltage, Transient voltage, DC current, or Transient current. You can annotate the transient voltages and currents at the same time.
- `cdsParam()`: to display the component parameters, model parameters, and DC/Transient operating point parameters.

This section describes the following topics:

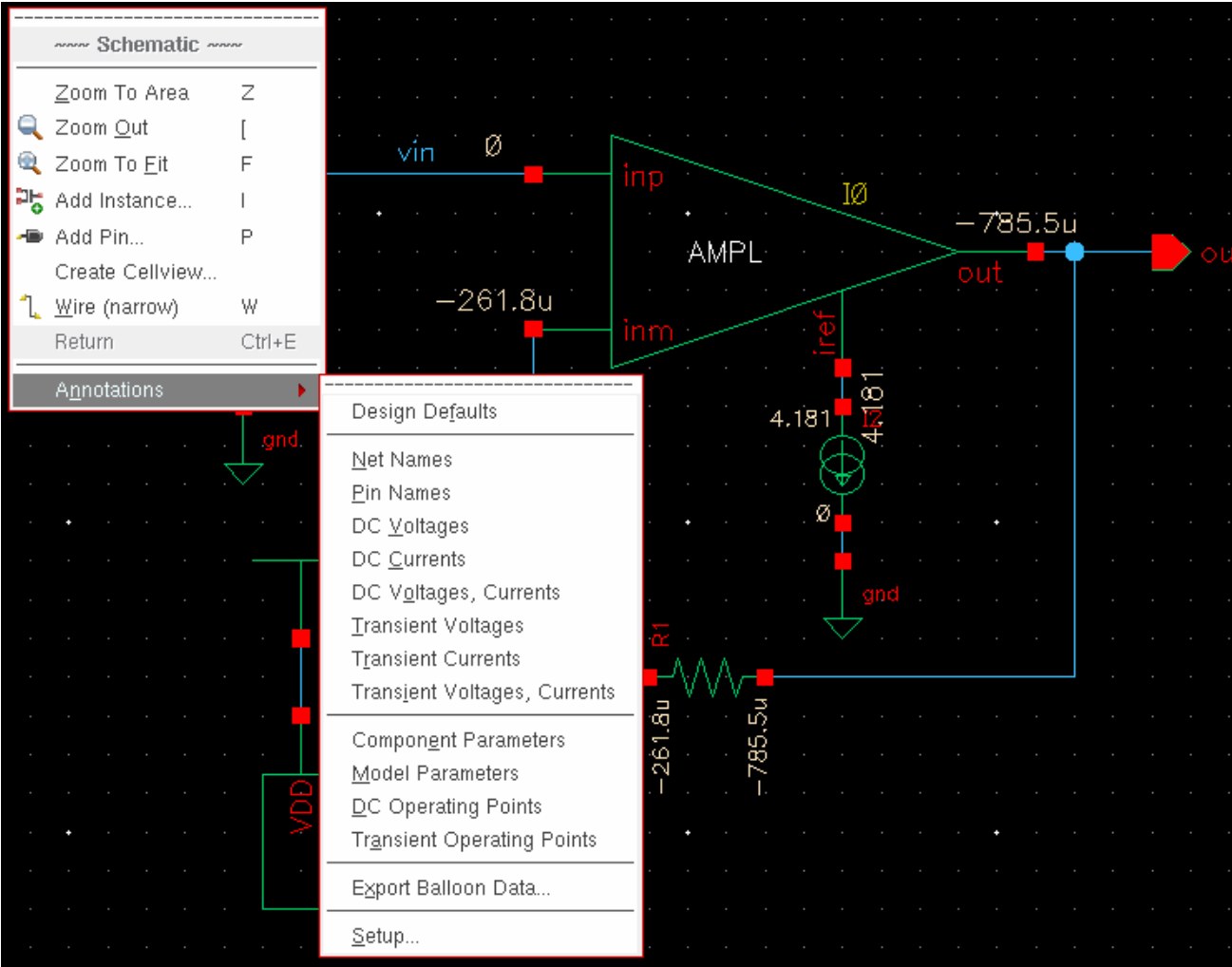
- [Annotation Context Menus on Schematic](#)
- [Using Annotation Balloons](#)
- [Using Annotation Setup Form](#)
- [Annotation Balloon Options](#)
- [Saving Annotation Settings](#)
- [Loading Annotation Settings](#)
- [Saving Data to User Cell-Level CDF](#)
- [Loading Data from Effective CDF](#)
- [Copying Annotation Settings](#)
- [Applying Global Annotation Settings](#)

Annotation Context Menus on Schematic

The annotation context menu enables you to:

- Select the data to be annotated on the schematic canvas and annotation balloon.
- Plot the currents and voltages, if the simulation results are available.

- Customize the annotation commands for a selected instance.



The options available on the annotation context menu vary based on the design component from which it is launched, as described in the following table:

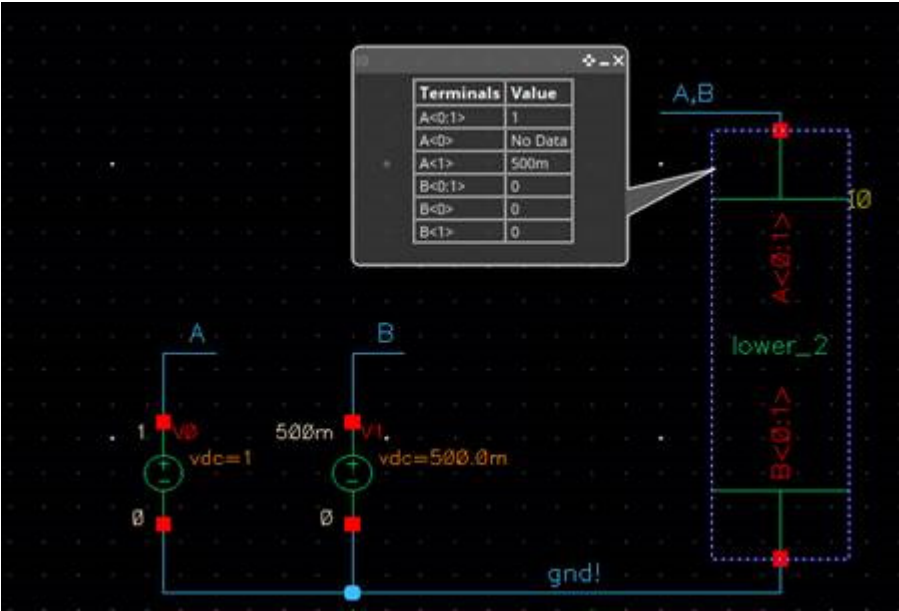
Design Component	Option
Empty area on the schematic	To change the annotation settings for labels and launch the <i>Annotation Setup</i> form. Note: The changed annotation settings are applied globally to the complete design.
Any Instance	To change the annotation settings to the selected instance only.

Design Component

Option

Net or wire

To plot voltages across the nets or wires. You can use the direct plot commands, such as DC, AC voltages, Transient minus DC, AC magnitude and phase, and so on to plot voltages across the nets or wires.



Instance pin or terminal

To plot terminal currents across the pins or terminals. You can use the direct plot commands, such as DC, AC current, Transient minus DC, AC magnitude and phase, and so on to plot currents across the pins or terminals.

Instance label

To change the annotation settings for the selected label of the instance. The context menu changes depending upon the label.

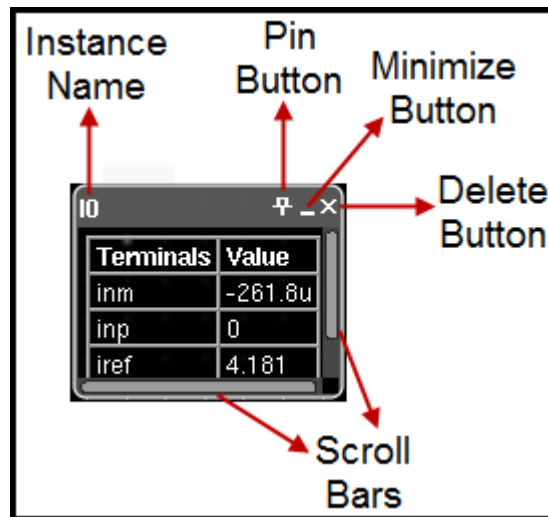
Using Annotation Balloons

You can use the annotation balloons to display CDF information and simulation results, such as voltages and currents, on the schematic. They appear when you move the cursor over any

Virtuoso Schematic Editor User Guide

Editing Objects

instance on the schematic canvas. The various components of the annotation balloon are shown below.



By default, the annotation balloons are configured to show the annotations for all the instances on the schematic canvas, and they are automatically updated when you change the annotation settings. The annotation settings can be configured using the *Annotation Setup* form as well as from the annotation context menu.

Annotation balloons eliminate the need of labels to be present on the symbol view of the instance. As a result, you can display any number of parameters (component, model, operating point parameters) on the annotation balloons. For more information, see [Annotating Different Values on Schematic Canvas and Annotation Balloons](#).

By default, the annotation balloons are not pinned on the schematic canvas and they fade away after the time specified in the *Fade Time* expires. You can pin these by:

- clicking the *Pin* button
- double-clicking the body of the annotation balloon
- using Shift+B, which pins only the visible annotation balloons

If the balloons are pinned, they do not fade away when you move the cursor away from the instance or when the *Fade Time* expires. You can remove the pinned balloons by clicking the *Delete* button on the balloon.

Note: The annotation balloons show *No Data*, if the results for a particular instance are not available.

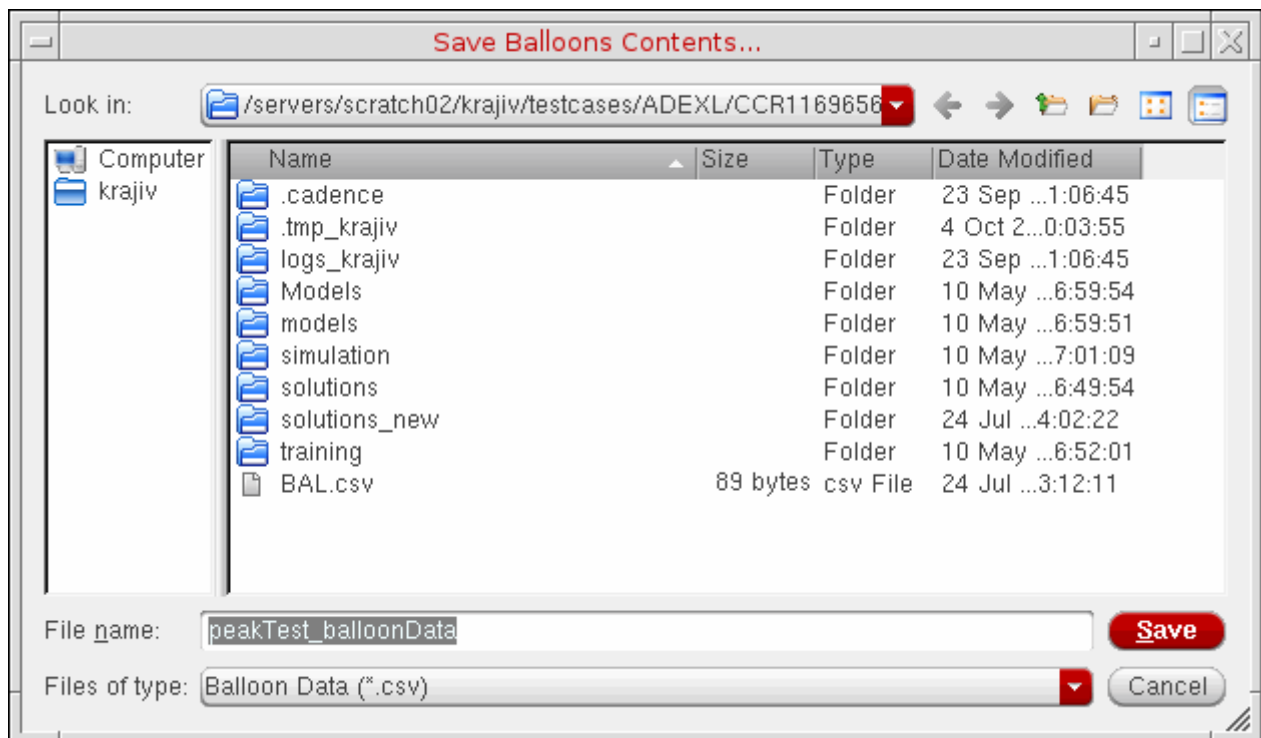
Exporting Annotation Balloon Data

You have the option of exporting the annotation information from the pinned balloons in the current hierarchy to a CSV file.

To export the annotations to a CSV file:

1. Choose *View – Annotations – Export Balloon Data*.

The *Save Balloons Contents* form appears.



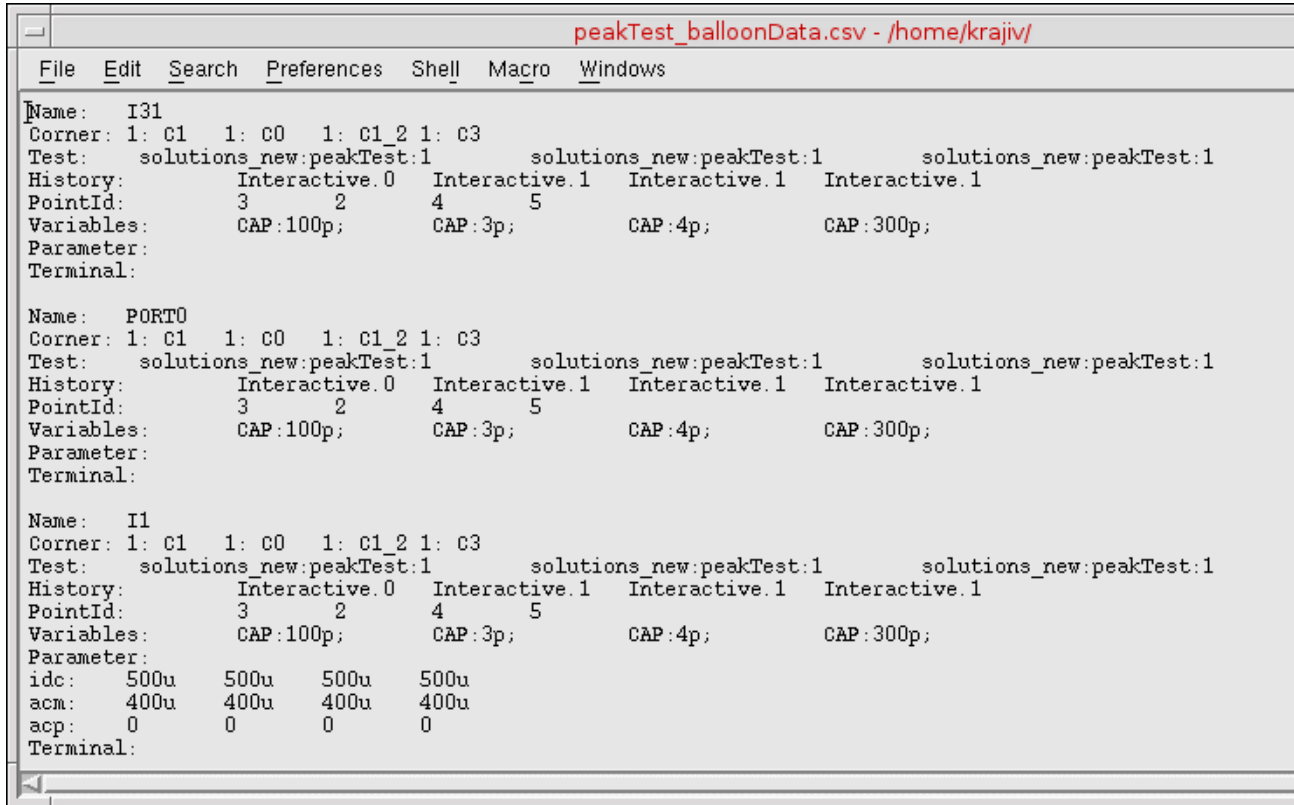
2. In the *File name* field, specify a name for the CSV file.
3. Click *Save*.

Note: The exported annotations are saved as tab-separated data in the CSV file. For information about the contents of the CSV file, see [CSV File Format](#) on page 534.

Virtuoso Schematic Editor User Guide

Editing Objects

After being exported to a CSV file, the annotations appear as shown in the following figure:



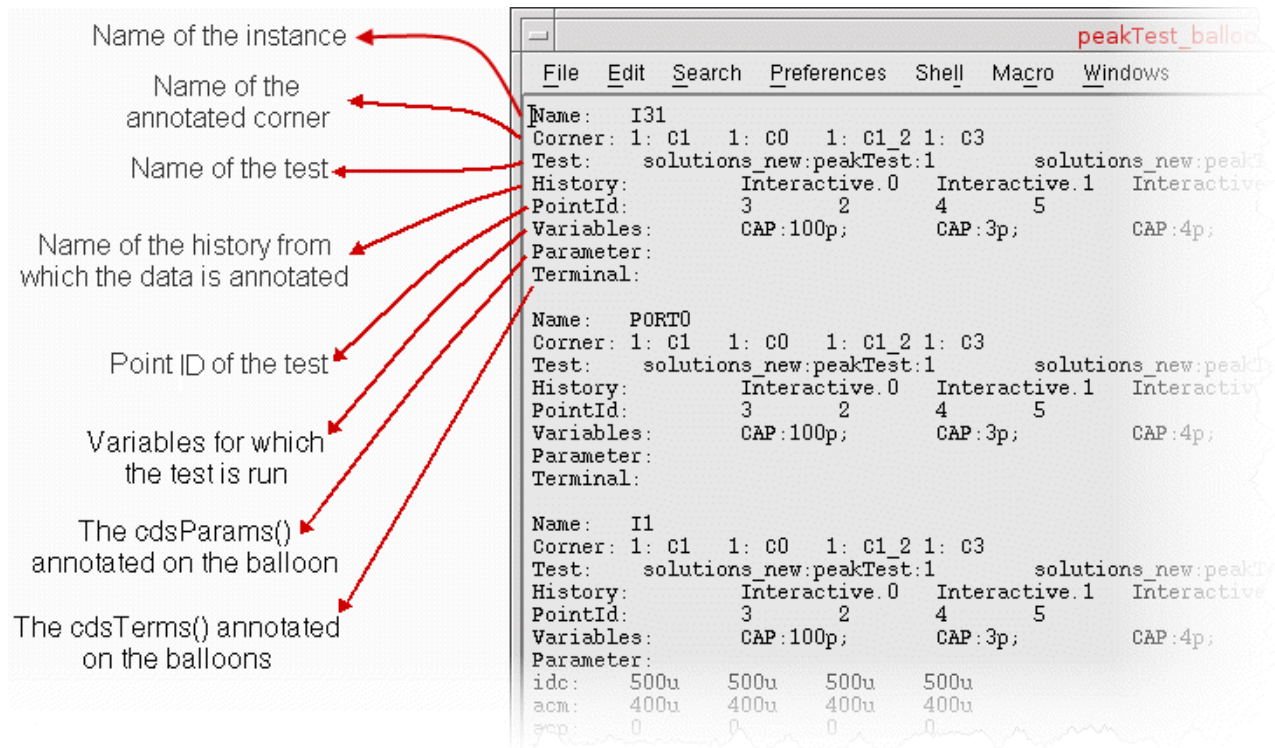
Note: You can open the CSV file in any text editor.

Virtuoso Schematic Editor User Guide

Editing Objects

CSV File Format

The format of the CSV file is as shown in the following figure:



Important Points to Note:

1. The annotations from only the pinned balloons in the current hierarchy are saved to the CSV file.
2. The *Export Balloon Data* option is also available from the context menu on the schematic.

Using Annotation Setup Form

To change the annotation settings of the design components using the *Annotation Setup* form, perform the following steps:

1. In the schematic window, choose *View – Annotations – Setup*.

Virtuoso Schematic Editor User Guide

Editing Objects

The Annotation Setup form appears.

Annotation Setup: training amplifier schematic (window:5)

File Edit Setup Global Help cadence

annotationSetup

Simulation Data Directory:

Annotation settings for:

Library: Cell: Instance: Selected List:

Label	Display Mode	Expression	Annotate	Balloon
Terminal:cdsTerm(B)	DC Voltage		☒	☒
Terminal:cdsTerm(D)	DC Voltage		☒	☒
Terminal:cdsTerm(G)	DC Current		☒	☒
Terminal:cdsTerm(S)	DC Voltage		☒	☒
Parameter:cdsParam(1)	Component Parameter	☐ model	☒	☒
Parameter:cdsParam(2)	Model Parameter	☒ kp	☒	☒
Parameter:cdsParam(3)	DC Operating Point	☒ vds+vgs	☒	☒
Parameter:cdsParam(4)	Component Parameter	☒ Vgs	☐	☒
Parameter:cdsParam(5)	DC Operating Point	☒ age	☐	☒
Parameter:cdsParam(6)	DC Operating Point	☒ vgs	☐	☒
Name:cdsName()	Instance Name		☒	☒

OK Apply Cancel

4 Set cell or instance specific data for each cds* label

2. Type the path to the simulation data directory in the Simulation Data Directory drop-down list to fetch the simulation results.

Alternatively, click the *Browse* button () or use the drop-down list box to select the simulation data directory. This loads the simulation data directly, and enables you to use the simulation results for managing annotation and plotting graphs.

Note: The *Simulation Data Directory* drop-down list box is automatically populated if the simulation data is available. If it is not available, then launch ADE and run a simulation to load the simulation data for the current session.

3. Select the library and cell to which you want to apply the annotation settings from the *Annotation settings for group* box. The table changes dynamically to list all the labels present in the selected library or cell.

You can use wildcards to select multiple cells that match the specified string pattern. The supported wildcards are *, ?, and []. For example, ?mos4 selects all the nmos4, pmos4, cmos4, and tmos4 cells; *mos* selects all the cells that contain the string mos, such as pmos1, nmos3, nmosrf, nmosTie, tmos2, pmos_cap; and [pn]mos4 selects all the pmos4 and nmos4 cells in the library.

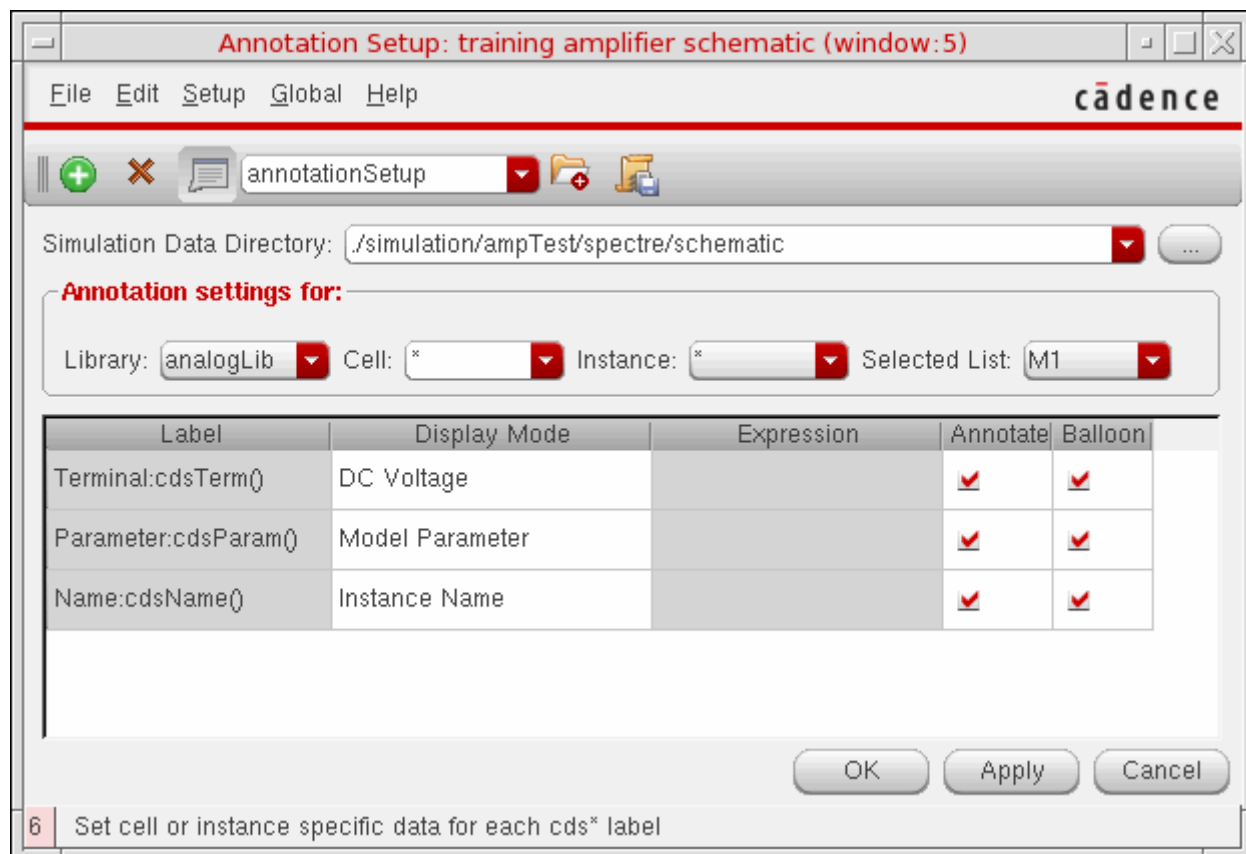
If you specify only * in the *Cell* drop-down list box, then the following three rows are displayed in the *Annotation Setup* form:

- ☐ *Terminal:cdsTerm()* for all the terminals
- ☐ *Parameter:cdsParam()* for all the parameters

Virtuoso Schematic Editor User Guide

Editing Objects

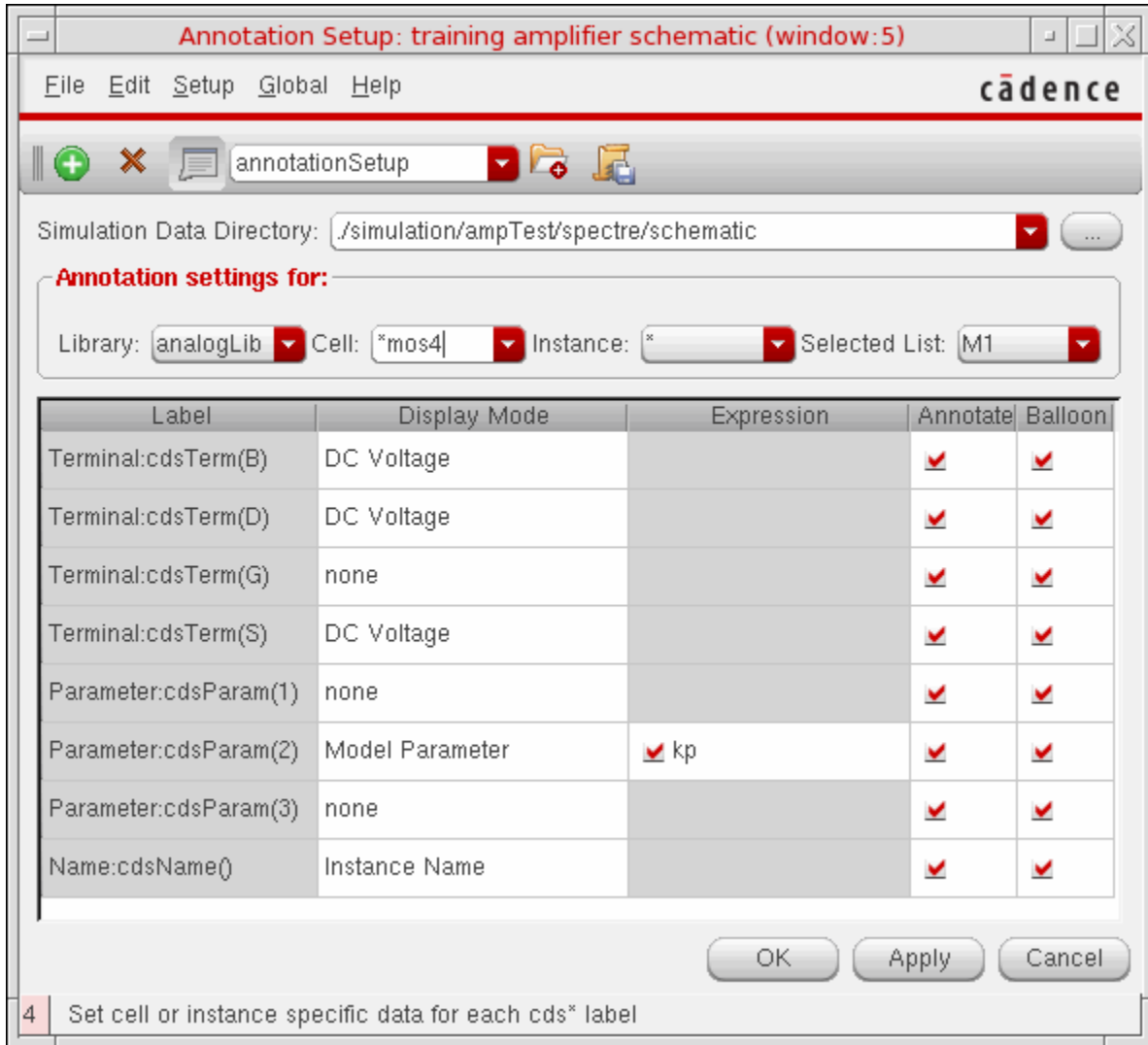
- ❑ *Name:cdsName()* for the instance name



Virtuoso Schematic Editor User Guide

Editing Objects

The following figure shows the usage of wildcards to select multiple cells:



- Double-click the *Display Mode* and *Expression* against each *Label* to change the annotation settings.

When you use wildcards in the *Cell* drop-down list box, the *Display Mode* and *Expression* for only the common Labels are displayed in the *Annotation Setup* form. The labels that are not common are set to *none*. As shown in the above image, when you specify **mos4* in the *Cell* drop-down list box, the *Display Mode* and *Expression* columns display information related to all common labels for *nmos4*, *pmos4*, *cmos4*, and *tmos4*. The labels that are not common are set to *none*.

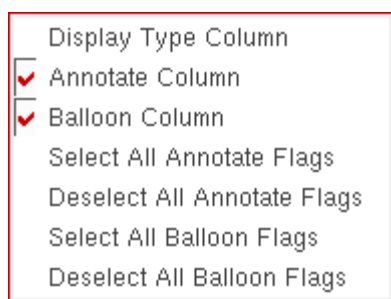
Note: For any instance of a particular cell, if there are no results corresponding to the selected *Display Mode*, the *Expression* list appears as disabled. This is depicted by

the grayed table cells under the *Expression* column in the image above.

5. Select the check box for each label for which you want to display the annotation data on the schematic canvas or on the annotation balloon. By default, all the annotations are displayed on the schematic as well as on the annotation balloons.

Annotation Setup Right-Click Menus

When you right-click any of the column headings on the Annotation Setup form, the following pop-up menu appears:



The various options available in the pop-up menu are:

- **Display Type Column:** Use this option to show the *Display Type* column to change the default display mode of the following labels:
 - ☐ **cdsTerm labels:** Displays the *Net Name* on schematic by default. You can choose to display the net name on schematic or on simulation from the drop-down list box.
 - ☐ **cdsParam label:** Displays the *Component Parameters* as literals. You can choose to display the component parameter as *Full*, *Suffix*, *Global*, and so on from the drop-down list box.
 - ☐ **cdsName:** Displays the *Instance Name* on the schematic. You can choose to display the instance name on schematic or on simulation from the drop-down list box.
- **Annotate Column:** Shows or hides the *Annotate* column.
- **Balloon Column:** Shows or hides the *Balloon* column.
- **Select All Annotate Flags:** Selects all the checkboxes in the *Annotate Column* on the *Annotation Setup* form.
- **Deselect All Annotate Flags:** Deselects all the checkboxes in the *Annotate Column* on the *Annotation Setup* form.

- **Select All Balloon Flags:** : Selects all the checkboxes in the *Balloon Column* on the *Annotation Setup* form.
- **Deselect All Annotate Flags:** Deselects all the checkboxes in the *Balloon Column* on the *Annotation Setup* form.

Adding Rows for Parameters

The *Annotation Setup* form enables you to add extra rows for `cdsParam()` for the selected library/cell element. The added `cdsParam()` rows can be configured to show the additional parameters. The added rows are colored to differentiate from the existing parameters in the design. As the added parameters are not present in the design, they can be configured and displayed using only the annotation balloons.

To add new rows for parameters:

1. In the *Annotation Setup* Form, choose *File – Add Row*.

A new `cdsParam()` row is added below the existing component parameters in the table in the *Annotation Setup* form.

2. Configure the *Display Mode* and *Expression* from the cyclic boxes of the added parameter to annotate the annotation balloons.

Choose *File – Delete Row* to remove the added `cdsParam()` row.

Alternatively, you can also add and delete the rows for parameters using  and  buttons on the *Annotation Setup* toolbar.

Annotating Different Values on Schematic Canvas and Annotation Balloons

You can configure the annotation settings from the *Annotation Setup* form to display different values on schematic canvas and annotation balloons. To configure the different annotation settings for schematic canvas and annotation balloons:

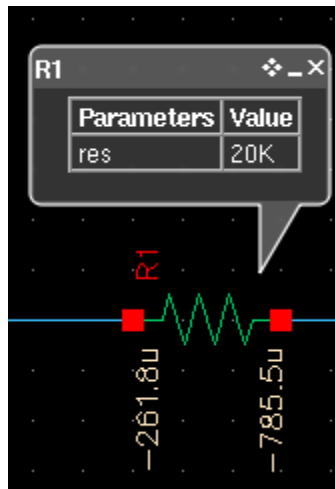
1. In the *Annotation Setup* form, select the instance for which you want to configure the annotation settings from the *Annotation settings for group* box.
2. Configure Labels in the table for annotation on the schematic canvas.
3. Deselect the Balloon check box for each Label so that the annotations are available only on the schematic canvas.
4. Add rows for parameters for the selected instance in the *Annotation Setup* form.

Virtuoso Schematic Editor User Guide

Editing Objects

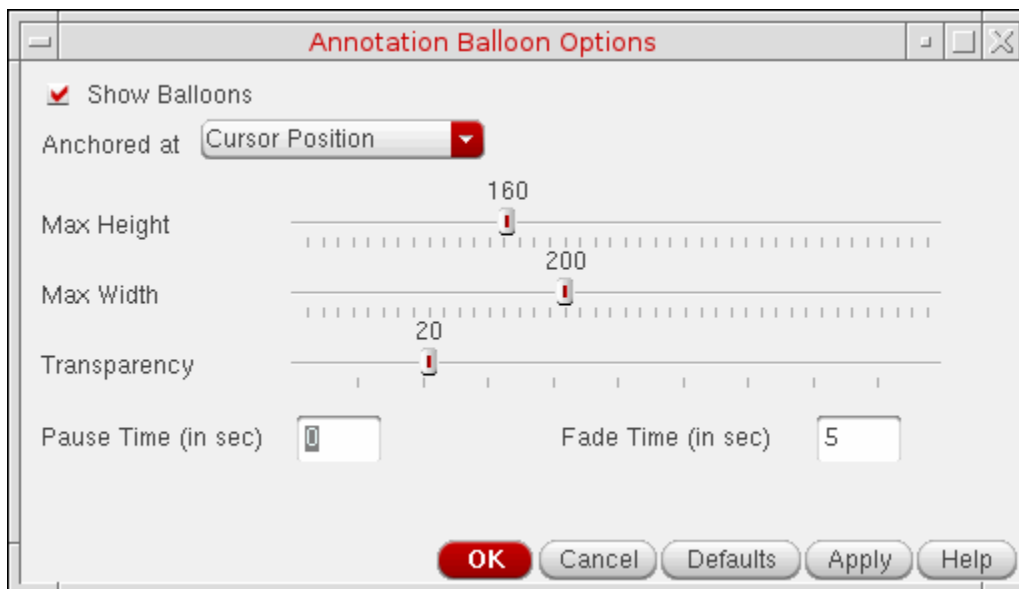
5. Configure the added parameter rows in the table for annotation on the annotation balloons. As the added parameter rows are not present in the design, they can be configured to display in the annotation balloons only.
6. Click *Apply*.

The annotations on the schematic canvas are different from the annotations on the annotation balloons.



Annotation Balloon Options

You can change the balloon options from the *Setup – Balloon Setup* option in the *Annotation Setup* form. The balloon setup contains the options shown in the following figure:



The options available on the balloon setup form are described below:

Field	Description
<i>Show Balloons</i>	If selected, enables the visibility of the annotation balloons on the schematic. Alternatively, click the  button on the <i>Annotation Setup</i> toolbar, or use Ctrl+B to toggle the visibility of the annotation balloons. Note: When you show the balloons, the pinned balloons are restored on the schematic. However, the balloons that are not pinned are visible only when you hover your cursor over any instance on the schematic canvas.

Virtuoso Schematic Editor User Guide

Editing Objects

Field	Description
<i>Anchored at</i>	Select the position where you want to display the annotation balloons when you position the cursor over any instance on schematic. The available options are: ■ Cursor Position ■ Top Left Corner ■ Top Right Corner ■ Bottom Left Corner ■ Bottom Right Corner
<i>Max Height</i>	Specifies the maximum height of the annotation balloon that can be displayed on the schematic canvas. If the data displayed in the balloon is less, it automatically shrinks to fit the data. Clicking the <i>Default</i> button sets the height of the annotation balloon to 160.
<i>Max Width</i>	Specifies the maximum width of the annotation balloon that can be displayed on the schematic canvas. If the data displayed in the balloon is less, it automatically shrinks to fit the data. Clicking the <i>Default</i> button sets the width of the annotation balloon to 200.
<i>Transparency</i>	Specifies the transparency of the annotation balloon. When set to 0, the annotation balloon appears <code>white</code> , whereas, if set to 100, the annotation balloon appears <code>transparent</code> . Clicking the <i>Default</i> button sets the transparency of the annotation balloon to 80.
<i>Pause Time (in sec)</i>	Specifies the time after which the annotation balloon should be visible on the schematic when you position the cursor over any instance.
<i>Fade Time (in sec)</i>	Specifies the time after which the annotation balloon should disappear from the schematic.

Important Points to Note:

- The size of the pinned balloons does not change dynamically even if the data displayed is changed.
- The annotation balloons that are not anchored at the cursor position are deleted only after the *Fade Time* expires.

Virtuoso Schematic Editor User Guide

Editing Objects

- The balloon settings are automatically saved in the `./ .cadence` directory when you exit the Annotation Balloon Options form, or click the *OK* or *Apply* button. These settings are loaded when you open a new Virtuoso session.

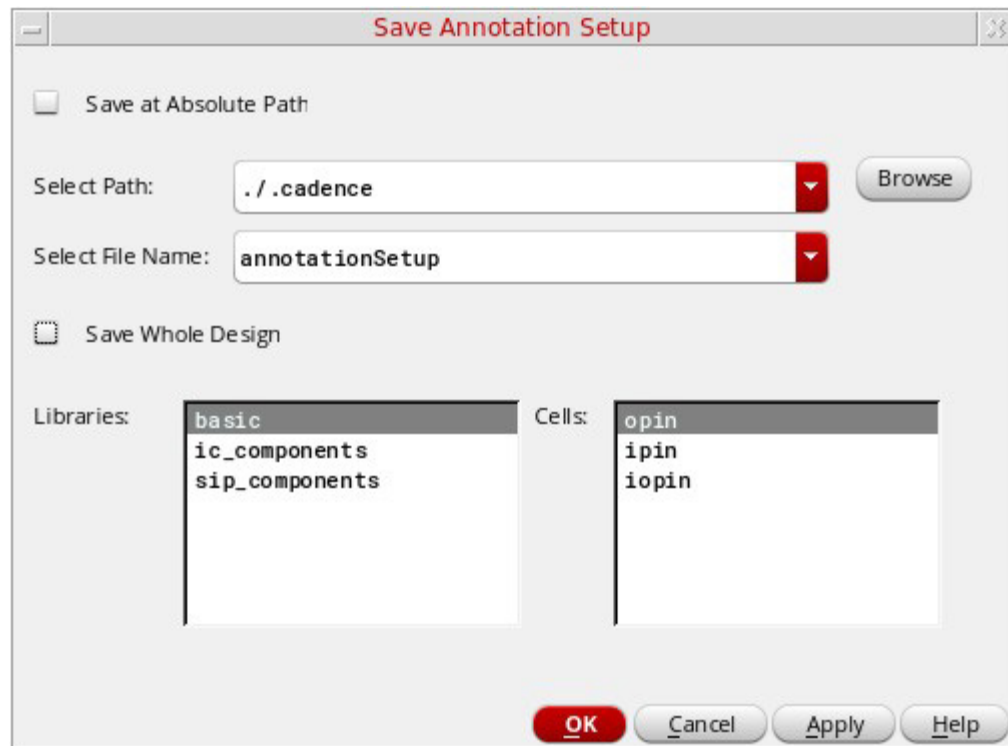
Saving Annotation Settings

You can use the *Save* option to save the annotation settings of the design into a file. Later if you run another simulation and click *Load*, the saved data can be loaded back and displayed as you specified while saving the annotation settings. You can load or save the settings from a different cellview too.

To save the annotation settings:

1. Choose *File – Save*.

The *Save Annotation Setup* form appears.

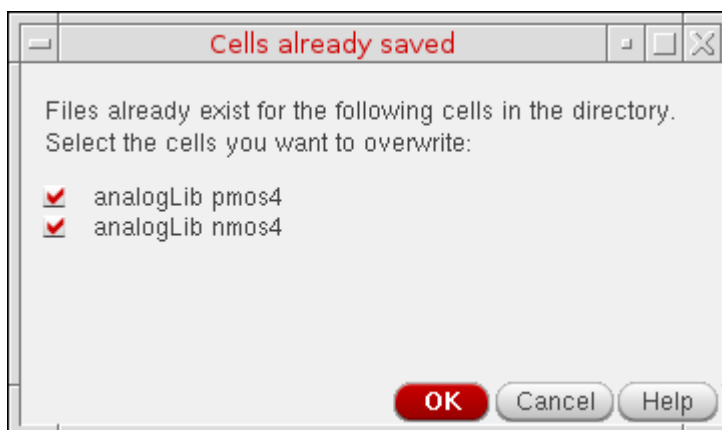


2. Select the *Save at Absolute Path* check box to save the annotation files in the `./ .cadence` directory. Deselect the check box to load the annotation files from the `./ .cadence/dfII/annotationSetups/<Lib>/<Cell>/<View>` directory. For additional details, refer to [Loading the Shared Annotation Settings Manually](#).

3. Select the location to which you want to save the file from the *Select Path* drop-down list. If you want to save the file at the Virtuoso installation directory, select `./ .cadence` from the *Select Path* drop-down list box, else select your home directory.
4. Type a filename in the *Select File Name* drop-down list box.
5. Select the *Save Whole Design* check box if you want to save the complete design. To save the annotation settings to a specific library or cell, make the required selections from the Libraries and Cells list boxes. To select multiple libraries or cells, keep the SHIFT or CTRL key pressed while making the required selections.
6. Click *OK* to save annotation settings to the specified file.

Alternatively, to quickly save the annotation settings, type the filename in the text box on the Annotations toolbar of the *Annotation Setup* form and click the  button. This option saves your complete design setup to the `./ .cadence` directory.

Note: If the annotation settings are already saved to a specific library or cell of a design, the following dialog box is displayed asking for the cells that you want to overwrite.



To save the annotation settings for cells:

1. Select the check box for which you want to overwrite the annotation settings.
2. Click *OK*.

Loading Annotation Settings

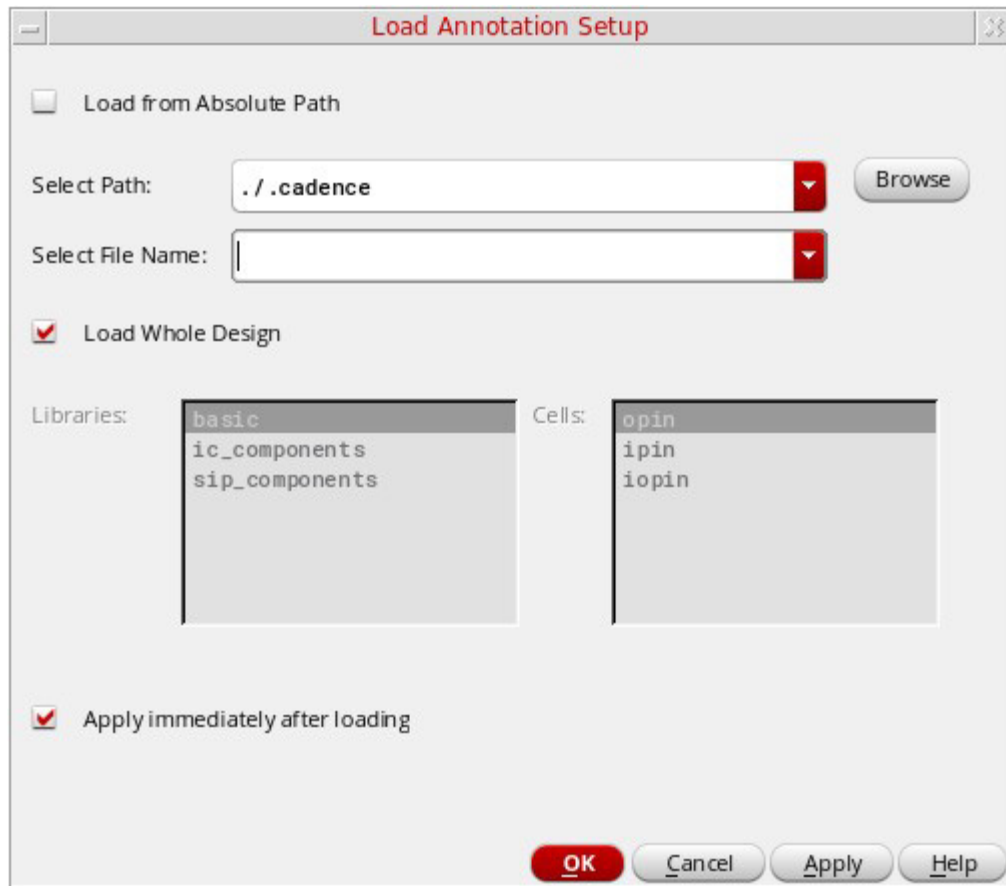
To load the saved annotation settings from a file:

1. Choose *File – Load* from the *Annotation Setup* form.

Virtuoso Schematic Editor User Guide

Editing Objects

The Load Annotation Setup form appears.



2. Select the *Load from Absolute Path* check box to search for the annotation files in the `./cadence` directory. Alternatively, click the *Browse* button to select the directory where the file is located. Deselect the check box to load the annotation files from the `./cadence/dfII/annotationSetups/<Lib>/<Cell>/<View>` directory. For additional details, refer to [Loading the Shared Annotation Settings Manually](#).
3. Select the path of the file from the *Select Path* drop-down list box.
4. Select the *Load Whole Design* check box if you want to load the complete design settings, else select the components that you want to load.
5. All the available annotation setup files are populated in the *Select File Name* drop-down list. Select the name of the setup file that you want to load.
6. Select the *Apply immediately after loading* check box if you want to apply the annotation settings to the complete design after loading the settings.
7. Click *OK* to load the specified file.

Alternatively, to quickly load the whole design setup that was saved in the `./cadence` directory during an earlier session, select the annotation settings file name from the drop-down list in the Annotations toolbar of *Annotation Setup* form and click the  button to load and apply the annotation settings. The drop-down lists all the annotation settings saved in the `./cadence` directory.

Loading the Shared Annotation Settings Automatically

To automatically load the specified annotation setup files in the schematic editor, set `annotationSetupFileList`. The setup files can point to the annotation setups of a different cellview. However, if the annotation setups are for the same cellview, the last annotation file read is loaded.

You can copy an existing annotation setup from the `./cadence/dfII/annotationSetups/<Lib>/<Cell>/<View>` directory to some central location to share it between designs.

Loading the Shared Annotation Settings Manually

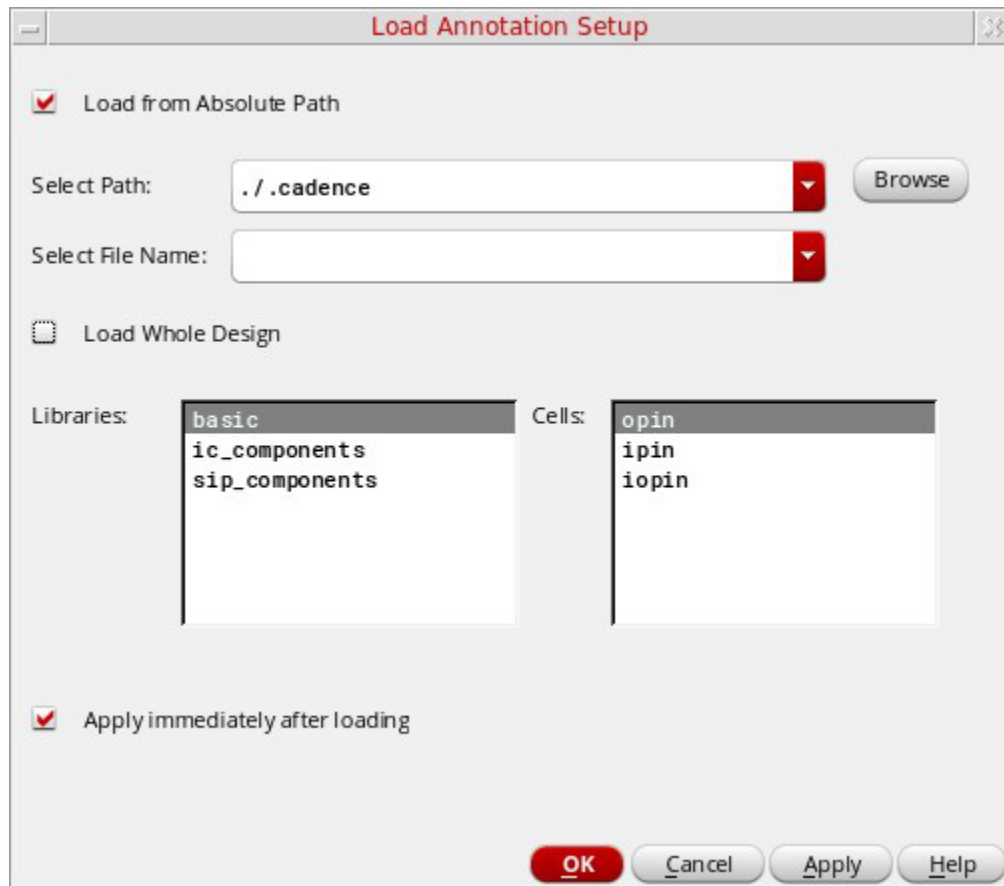
The annotation setups are not bound to a specific cellview. You can save the new annotation setup files in the home or project `./cadence` directory to share them between designs. To load the annotation setups manually,

1. Select the *Load from Absolute Path* check box to use globalized directory structure.

Virtuoso Schematic Editor User Guide

Editing Objects

2. Select the `.cadence` path in the Load Annotation Setup form.



Alternatively, you can use the following SKILL functions to manually load and save the annotation settings:

Related SKILL functions:

■ **[annLoadAnnotationData](#)**

To load the annotation setup:

```
annLoadAnnotationData(hiGetCurrentWindow()  
"<pathToAnnotationSetupFile">)
```

■ **[annSaveAnnotationData](#)**

To save the annotation settings:

```
annSaveAnnotationData(hiGetCurrentWindow()  
"<pathToNewAnnotationFile">)
```


Saving Data to User Cell-Level CDF

You can save the custom annotation settings of the design in the user level CDF of the cell. Choose *Setup – Save to User Cell CDF* option to save the annotation settings to the user CDF of the cell.

Loading Data from Effective CDF

You can load the custom annotation settings saved in the previous session from the effective CDF in the *Annotation Setup* form. When you load the effective CDF settings into the *Annotation Setup* form, the settings are applied to all the instances of the selected window. Choose *Setup – Retrieve from Effective CDF* option to load the *Annotation Setup* form and apply the CDF settings.

For information about base level CDF, user CDF and effective CDF, see the [*Levels of CDF Data*](#) section of the *Component Description Format User Guide*.

Copying Annotation Settings

You can use the copy and inherit options to copy the annotation settings of one instance to other instances of the same type. This option is useful when you want to apply the custom annotation settings from one instance to other instances of a cell. The *Annotation Setup* form provides you various options to copy the annotation settings, which are described below:

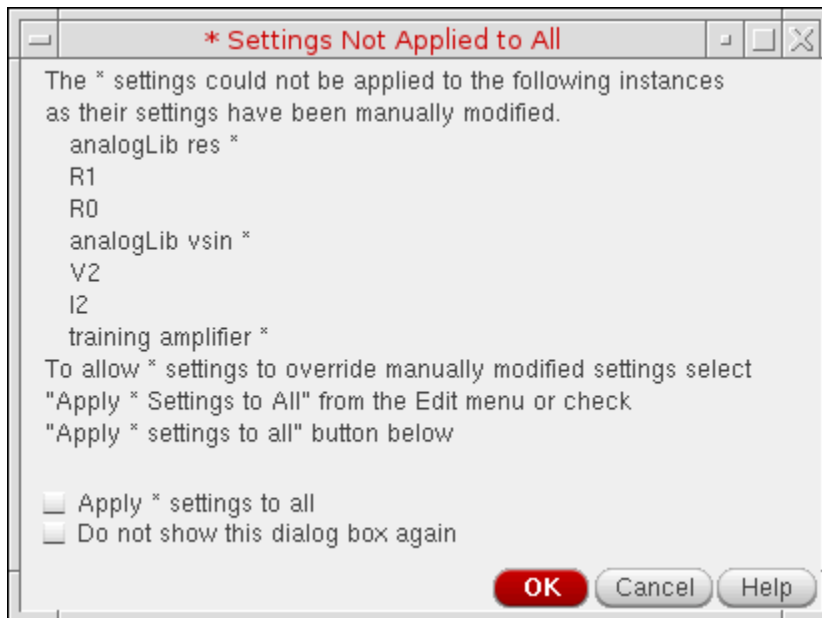
Option	Description
<i>Copy to All</i>	Copy the annotation settings of the selected instance to all the other instances of same type.
<i>Copy to Selected</i>	Copy the annotation settings of the selected instance to all the selected instances of the same type. For example, if you have selected the annotation settings for <code>pmos</code> transistor, and select other <code>pmos</code> transistors, namely M1, M4 and M5 from the schematic, then select <i>Copy to Selected</i> option to copy the annotation settings of the selected transistor to the other selected <code>pmos</code> transistors. Note: You can copy the annotation settings to the instances of only the same type. You cannot copy the annotation settings from one instance to another instance of a different type.

Virtuoso Schematic Editor User Guide

Editing Objects

Option	Description
<i>Inherit from Globals</i>	Inherit the annotation settings from the Global menu option to the instance selected in the <i>Annotation Setup</i> form.
<i>Inherit from Library</i>	Inherit the library level annotation settings to the selected instances.
<i>Inherit from Cell</i>	Inherit the cell level annotation settings to the selected instances.
<i>Apply * Settings to All</i>	Use this option to apply the annotation settings to the instances in the design globally. If this option is selected, the * annotation settings from the <i>Instances</i> drop-down list are copied to all the instances of the selected library or cell.

Note: The * Settings Not Applied to All dialog box (as illustrated below) is displayed when the Apply * Settings to All option is not selected from the *Annotation Setup* form, and you change the annotation settings after selecting * from the Cell or Instances drop-down lists and some instance have custom annotation settings.



On the displayed dialog box, you can select the *Apply * settings to all* check box to apply the * level annotation settings to all the instances of the selected library or cell.

Applying Global Annotation Settings

The *Annotation Setup* form enables you to apply the annotation settings globally to all instances of the same type at all levels of the hierarchy in the design. You can also use the *Annotation Setup* form to clear all the annotations from the design or to apply the default annotation settings. The annotations applied from the *Global* menu option on the *Annotation Setup* form are synchronized with the annotations applied using the context menu.

Note: The options on the *Global* menu and the context menu are available depending on the results. For example, if you do not have the results for DC simulation, the options to plot and annotate the DC voltages, DC currents, and DC operating points will not be available.

The *Annotation Setup* form provides you various options to apply the annotation settings, described below:

Option	Description
<i>Terminals</i>	Change the annotation settings on all the instance terminals in the selected window and all the descendant windows. Selecting the <i>Design Defaults</i> option removes the current CDF properties from all the instance terminals in the design and applies the Base Library CDF settings.
<i>Parameters</i>	Change the annotation settings of all the instance parameter in the selected window and all the descendant windows. Selecting the <i>Design Defaults</i> option removes the current CDF properties from all the instance parameters in the design and applies the Base Library CDF settings.
<i>Instances</i>	Change the annotation settings of all the instance names in the selected window and all the descendant windows. Selecting the <i>Design Defaults</i> option removes the current CDF properties from all the instance names in the design and applies the Base Library CDF settings.

Virtuoso Schematic Editor User Guide

Editing Objects

Option	Description
<i>Design Defaults</i>	Remove the current CDF properties from all the labels in the design and apply the Base CDF settings. This option also removes the cell-level settings and replaces them with the Base Library-level CDF settings overlaid by Base Cell CDF settings. When this option is selected, the values defined for the following user-defined CDF properties in the <i>Edit CDF</i> form are not overwritten: ■ paramLabelSet ■ opPointLabelSet ■ modelLabelSet ■ instDisplayMode ■ instNameType The design defaults automatically picks the settings explicitly set for the properties listed above.
<i>Clear All</i>	Clear the annotations from all the labels in the design.
<i>Apply Globals to All</i>	Apply all the cell/instance level annotation settings selected from the Global menu to all the instances in the design. If this option is not selected, then the annotation settings are applied to only those instances on which the custom annotation settings are not applied.

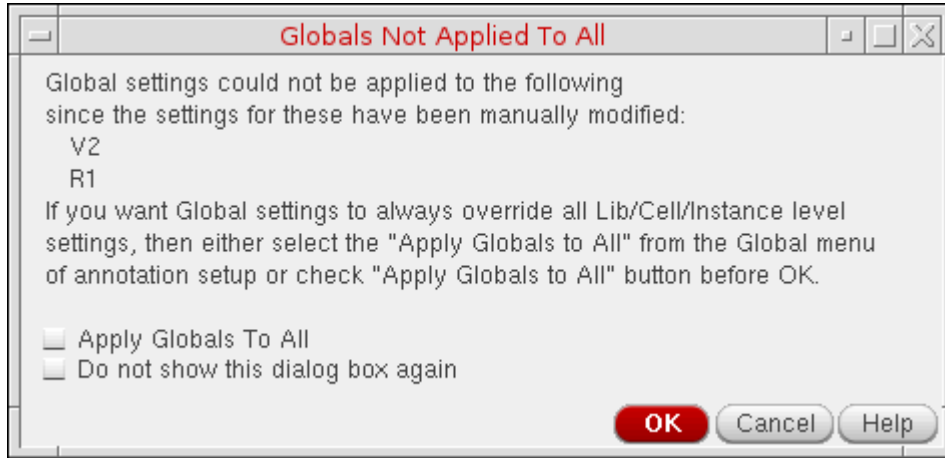
Note: The Globals Not Applied to All dialog box (as illustrated below) is displayed when the following conditions are true:

1. An option from the Global menu is selected to change the annotations of Terminals, Parameters, or Instances.
2. The *Apply Globals to All* option is not selected.

Virtuoso Schematic Editor User Guide

Editing Objects

3. Any instance in the design has custom annotation settings.



On the displayed dialog box, you can select the *Apply Globals To All* check box to apply the global annotation settings to all the instances.

Virtuoso Schematic Editor User Guide

Editing Objects

Editing Properties

This chapter describes how to find and edit object properties.

- [Using Command Shortcuts](#)
- [Searching for Objects That Have a Specific Property](#)
- [Replacing Properties](#)
- [Selecting Objects to Edit](#)
- [Selecting Objects by Filtering](#)
- [Object and Cellview Property Types and Values](#)
- [Editing Object Properties](#)
- [Editing Specific Object Properties](#)
- [Editing Cellview Properties](#)
- [Setting the Tool Filter](#)

Using Command Shortcuts

Direct editing (or direct manipulation) is the quickest way to edit or create an object. You can use your mouse to start the following command operations, rather than selecting a menu command or an icon:

- Copying using direct manipulation
- Moving using direct manipulation
- Create a wire from a schematic pin
- Create a wire from a block pin
- Repeat the last command
- Rotating using the mouse

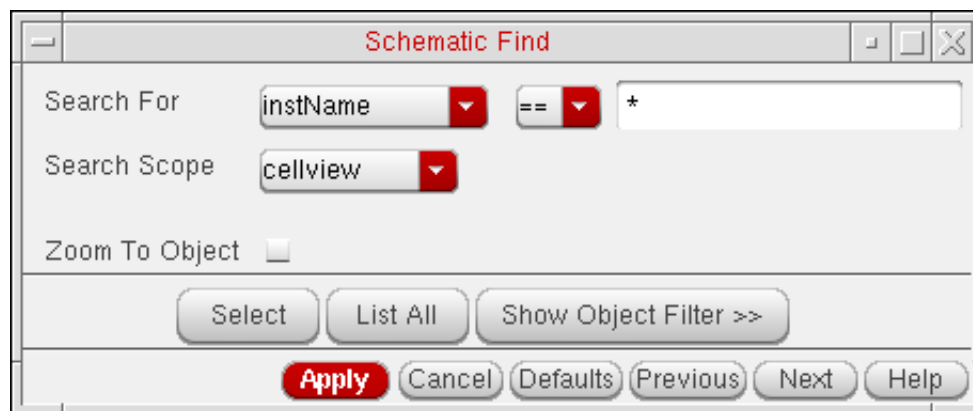
Searching for Objects That Have a Specific Property

Use the *Edit – Find* command to find objects that have a property that matches your specific search criteria.

To find objects that have a specific property,

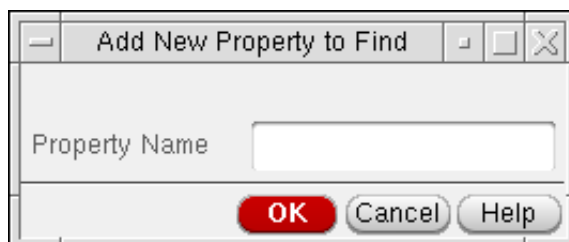
1. From the view, choose *Edit – Find*.

The Schematic Find form appears. The title bar reflects the editor name.



2. In the *Search For* cyclic field, choose the specific property to search for.

If you set the *Search For* cyclic field to *other*, the Add New Property to Find form appears.



- a. In the *Property Name* field, type the value of the property.

The asterisk (*) matches zero or more occurrences of any characters.

- b. Click *OK*.

The new name appears in the Schematic Find form.

3. Click *Apply*.

Virtuoso Schematic Editor User Guide

Editing Properties

When a match exists,

- ☐ A message reports the number of matching properties.
- ☐ All the properties that match the criteria specified on the Find form are added to a list. To display the list of matches, click the *List All* button.
- ☐ The software highlights the first matching object

If no match exists, a dialog box confirms no objects were found.

4. Click *Next*.

In the schematic window, the software highlights the next matching object.

5. To end the command, click *Cancel* or close the form window.

Search String Wildcards

Wildcards help you replace properties using partial string substitution. You can use one or more wildcards or tagged fields in the search and replace fields on the Schematic Find or Schematic Replace form.

Search String Wildcards	Description
* (asterisk)	Matches zero or more occurrences of any characters
. (period)	Matches any character
[c . .]	Matches one of the characters in the set
\ (backslash)	Referred to as the escape character, it causes the character following it to be treated normally, if it is: ■ { ■ * ■ , ■ \. Use \\ for a normal \.

The system counts each occurrence of an “*”, “.” or “[c . .]” as one single wildcard, except when the “.” and “*” appear together, in that order. In this case, the system counts the two together as one wildcard only.

Replacing Properties

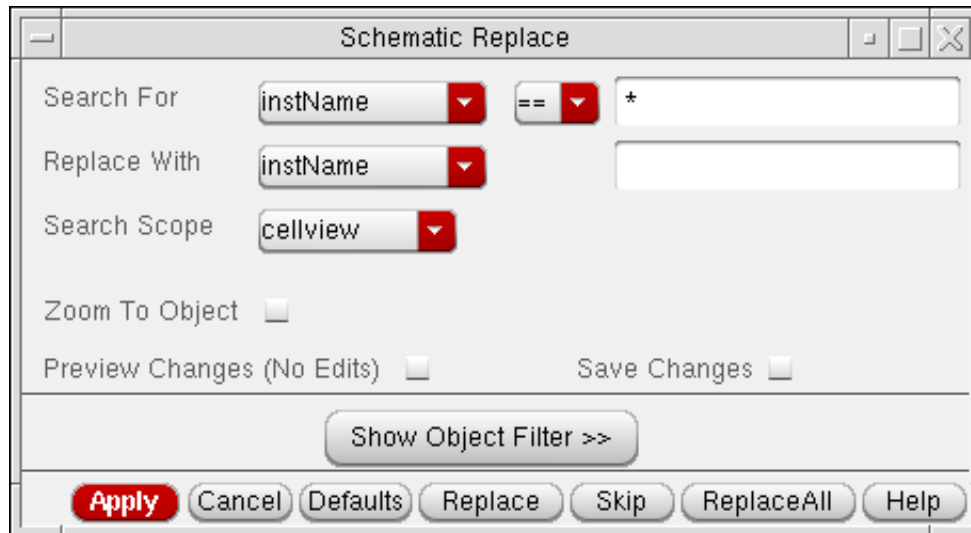
Use the *Edit – Replace* command to find objects that have a property that matches the search criteria you specify on the Replace form. You can then replace the property or the value with another property or value, which you also specify on the Replace form.

Note: You must have write access to the cellview that contains the property you search for and the cellview that contains the replacement property.

To find and replace properties,

1. From the view, choose *Edit – Replace*.

The Schematic Replace form appears. The title bar reflects the editor name.

The image shows a screenshot of the 'Schematic Replace' dialog box. The title bar reads 'Schematic Replace'. Inside the dialog, there are several fields and controls: 'Search For' with a dropdown menu showing 'instName', an operator dropdown showing '==', and a text field containing '*'; 'Replace With' with a dropdown menu showing 'instName' and an empty text field; 'Search Scope' with a dropdown menu showing 'cellview'; 'Zoom To Object' with an unchecked checkbox; 'Preview Changes (No Edits)' with an unchecked checkbox; and 'Save Changes' with an unchecked checkbox. Below these fields is a button labeled 'Show Object Filter >>'. At the bottom of the dialog is a row of buttons: 'Apply' (highlighted in red), 'Cancel', 'Defaults', 'Replace', 'Skip', 'ReplaceAll', and 'Help'.

2. In the *Search For* cyclic field, chose the specific property to search for.

When you set the *Search For* cyclic field to *other*, the Add New Property to Find/Replace form appears. When you click *OK*, the new name appears in the cyclic field on the Find form.

3. Check the object filter to make sure the property is selectable.
4. Click *Apply*.

If a match exists, a message reports the number of matching properties. If no match exists, a dialog box confirms no objects were found.

5. On the Replace form, do one of the following:

Virtuoso Schematic Editor User Guide

Editing Properties

- ❑ Click *ReplaceAll* to replace all property names and values in the list with the new property names and property values.
- ❑ Use the *Replace* and *Skip* buttons to view individual matches and apply the replacement property name and value as desired.

Replacing String-Tagged Characters

In the Virtuoso® Schematic Editor, you can use partial strings and wildcards in search-and-replace statements. The table below and the following examples describe the process.

Replace String-Tagged Characters	Description
{n}	Interprets <i>n</i> as an integer from 1 to 9. Tagged fields correspond to “ * ” or “ . ” or “[c . .]” or “ . * ” in the search string. The first nine wildcards in the search pattern are tagged. If a tagged field that is used in the replace string does not exist, it is replaced by a NULL string.
*nth " * "	Interpreted as {n}, <i>n</i> ranging from 1 to 9. If more than nine wildcards exist in the search string and more than nine *’s in the replace string, the system substitutes the first nine *’s in the replace string with the corresponding nine tagged fields from the search string. The remaining *’s in the replace string are substituted with a NULL string.
\ (backslash)	Causes the character following it to be treated normally, if it is { or * or \. Use \\ for a normal \. Referred to as the escape character.

Example 1

If you have instances I0, I1, I2,...I10, and you want to replace them with M1, M2,...M10, enter the following:

Search: I*

Replace: M* or M{1}

Example 2

You can specify the tagged fields in any order in the replace string. If you want to swap the start and end values for all buses with prefix ABC, enter the following:

Search: ABC<*: *>

Replace: ABC<{2} : {1}>

The {2} tag in the replace string corresponds to the characters represented by the second asterisk in the search string, while {1} corresponds to the characters represented by the first asterisk in the search string.

Example 3

In all buses with prefix ABC, keep the start value the same but change the end value to 20.

Search: ABC<*: *>

Replace: ABC<{1} : 20> or ABC<*: 20>

The {1} tag or * in the replace string corresponds to the characters represented by the first asterisk in the search string.

Example 4

Search for CLK, CLK_, CLK_IN and replace with CLOCK, CLOCK_, CLOCK_IN.

Search: CLK*

Replace: CLOCK{1} or CLOCK*

The {1} tag or * in the replace string corresponds to the characters represented by the asterisk in the search string.

Virtuoso Schematic Editor User Guide

Editing Properties

Example 5

You can use any combination of wildcards in the search string. To search for net names A1B1C1, A2B2C2, and so forth, and replace them with P1Q1R1, P2Q2R2, and so forth, you can enter any one of the following:

Search: A[0-9]B[0-9]C[0-9] or
A.B.C. or
A*B*C* or
A[0-9]B.C. or
A.B[0-9]C[0-9] or
A*B.C. or
A.B*C*

Replace: P{1}Q{2}R{3} or
P*Q*R*

Example 6

A tagged field can appear any number of times in the replace string. If you want to replace AfooB and AblahB by PfoofooQ and blahblahQ respectively, enter the following:

Search: A*B

Replace: P{1}{1}Q

Example 7

If the wildcards appear together in a sequence in the search string, each corresponds to a different tag except when a “.” and “*” appear together, in that order. In this case, the wildcards are treated together as one tag.

If you want to search for a string AxyzB and replace it with CxyzD, enter the following:

Search: A.[x-z]*B

Replace: C*D

The result is CxD and not CxyzD because the “*” in the replace string is interpreted as the first tagged field, which is the wildcard “.”. To achieve the desired result, you can use C***D or C{1}{2}{3} as the replace string, or use A*B as the search string.

However, suppose you enter the following:

Search: A.*[x-z]B

Replace: C*D

Virtuoso Schematic Editor User Guide

Editing Properties

The result is C_{xy}D, because the “ * ” in the replace string is interpreted as the first tagged field in the search string, which is the sequence “ . * ”.

To achieve the desired result, enter the following:

Search: A*B

Replace: C**D or C{1}{2}

Example 8

The character \ can be used to escape wildcard characters in the search-and-replace strings. If you want to replace A.B by A*B, enter the following:

Search: A*B or A.B

Replace: A*B

The \ before “ * ” in the replace string indicates that the asterisk be treated as a literal and not as a wildcard.

If you had an array A[1], A[2], A[3] ... A[n] and you wanted to replace all square brackets by curly braces, enter the following:

Search: A\[*] or A\[[0-9]]

Replace: A\{{1}} or A\{*}

The \ before the first { in the replace string means treat the { as a literal and not as a tagged field.

If you want to replace ABC by A\B\C, enter the following:

Search: A[A-Z][A-Z]

Replace: A** or A\\{1}\\{2}

Use \\ for each \ that you want to be treated as a normal character. The asterisk following the second backslash in the replace pattern is treated as a wildcard. If the replace string is A** instead, then the result is A**.

Virtuoso Schematic Editor User Guide

Editing Properties

Example 9

If the search string does not have a wildcard character and the replace string does, a substitution is not achieved. Suppose you had

Search: A1

Replace: P* or P{1}

In this case, the “*” and {1} in the replace string are treated as normal characters and not as wildcards.

Example 10

If you had more than nine wildcards in the search-and-replace strings, only the first nine in the replace string can be substituted correctly. The rest are substituted with empty strings. Suppose you want to search for a string A1B2C3D4E5F6G7H8I9JKL by P1Q2R3S4T5U6V7W8X9YKZ:

Search: A*B*C*D*E*F*G*H*I*J*L

Replace: P*Q*R*S*T*U*V*W*X*Y*Z

The result is P1Q2R3S4T5U6V7W8X9YZ. The last “*” in the replace string gets substituted by an empty string and not the substring corresponding to the last “*” in the search string.

Selecting Objects to Edit

You can select objects before or after you start an editing command.

If you select an object before you start an editing command, the system prompts you for a starting point called the reference point for the edit. The reference point is usually in the middle of the object.

When you select an object after you start a repeatable command (such as move, delete, copy), the command remains active (default) until you cancel the command by pressing the `ESC` key.

To change the default to nonrepeating,

1. From the view, choose *Options – Editor*.

Virtuoso Schematic Editor User Guide

Editing Properties

The Editor Options form appears.

The screenshot shows the 'Editor Options' dialog box with the 'Command' tab selected. The dialog has five tabs: 'Commands', 'Connectivity', 'Inherited Conn.', 'Labels', and 'Ignore Properties'. The 'Command Controls' section includes the following settings:

- Add Instance Browser Type: ☒ library ☐ component
- Show Schematic Views in Browser: ☒ off ☐ on
- Block Pin Direction Rule: ☒ edge ☐ order of entry
- Wire Snapping: ☐ off ☒ on
- Automatic Solder Dots: ☐ off ☒ on
- Solder Dot Wire Overlap: 0.03125
- Symbol Generator Pin Sorting: ☐ alphanumeric ☒ geometric
- Add Symbol Pins as Instances: ☒ off ☐ on
- Repeat Commands: ☐ off ☒ on
- Cross Selection: ☒ off ☐ on
- Gravity: ☐ off ☒ on
- Minimum Marker Size: 0.05
- View Name List: schematic symbol
- Auto Wire Stub for Bus Terminals: ☒ wide ☐ narrow

The 'Constraint Notes Controls' section includes the following settings:

- Auto Constraint Notes: ☐ enable ☒ disable
- Sync on Design Open: ☐ enable ☒ disable
- Sync on Design Check: ☐ enable ☒ disable

At the bottom of the dialog are buttons for 'OK', 'Cancel', 'Defaults', 'Apply', and 'Help'.

2. In the *Command* tab, Turn *Repeat Commands* off.
3. Click *OK*.

Using Repeating Commands

You do not have to repeatedly select a command if you plan to perform the same task several consecutive times. For example, editing commands automatically repeat when the *Repeat Commands* option is set to *on* (the default).

Selecting Objects Automatically by Type

You can set the software to select all the objects of one or more object types automatically. For example, you might want to select all the pins in your design.

To select all the objects of one or more object types,

1. From the view, choose *Edit – Select – Select By Filter*.

Note: This also works for deselection.

If you are in the schematic window, the Schematic Select By Filter form appears.

The screenshot shows the 'Select By Filter' dialog box with the title bar 'Select By Filter'. Inside, there are two radio buttons for 'Form Action': 'select' (which is selected) and 'deselect'. Below this is a section for 'Schematic Objects' with a grid of checkboxes: 'wire', 'pin', 'instance', 'note shape', 'wire name', 'pin name', 'marker', 'note text', 'flight', and 'label'. All these checkboxes are checked. To the right of this grid are two buttons: 'all' (checked) and 'none' (unchecked). Below the schematic objects is a section for 'Instance Objects' with a grid of checkboxes: 'name', 'pin', 'pin name', and 'label'. Only the 'pin' checkbox is checked. To the right of this grid are two buttons: 'all' (unchecked) and 'none' (unchecked). At the bottom of the dialog are five buttons: 'OK' (highlighted in red), 'Cancel', 'Defaults', 'Apply', and 'Help'.

If you are in the symbol window, the Symbol Select By Filter form appears.

The screenshot shows the 'Select By Filter' dialog box with the title bar 'Select By Filter'. Inside, there are two radio buttons for 'Form Action': 'select' (which is selected) and 'deselect'. Below this is a section for 'Symbol Objects' with a grid of checkboxes: 'pin', 'device shape', 'selection box', 'note shape', 'pin name', 'label', 'marker', and 'note text'. All these checkboxes are checked. To the right of this grid are two buttons: 'all' (checked) and 'none' (unchecked). At the bottom of the dialog are five buttons: 'OK' (highlighted in red), 'Cancel', 'Defaults', 'Apply', and 'Help'.

2. On the form, specify the object types you want to select.

The option *none* turns off all object types.

3. Click *OK*.

The software highlights the specified objects in the design window.

Note: This command does *not* modify the selection filter settings that you set using the *Options – Select Filter* command.

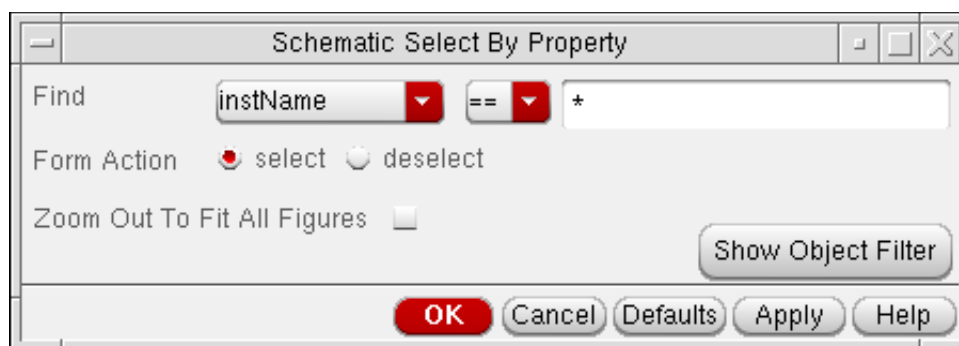
Selecting Objects by Property

You can limit the selection set by specifying that the software select only those objects with certain property values.

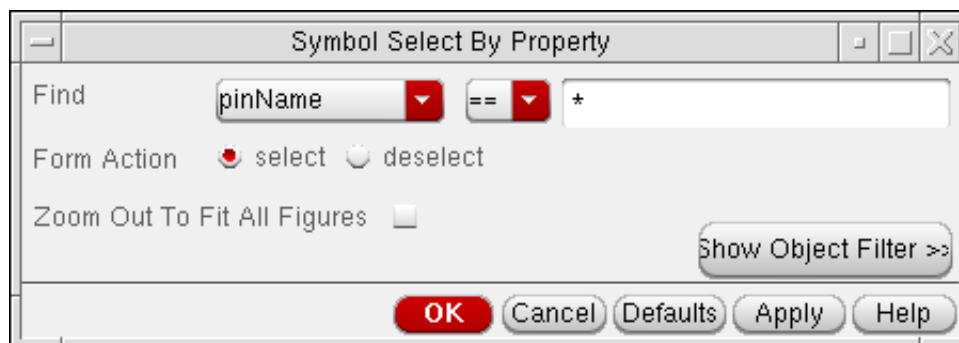
To limit a selection set to object types with certain property values,

1. From the view, choose *Edit – Select – Select By Property*.

If you are in the schematic window, the Schematic Select By Property form appears.



If you are in the symbol window, the Symbol Select By Property form appears.



2. In the *Find* fields, specify the property name, relational operator, and property value.

You usually look for objects that have a particular property value. You can also use the relational operator cyclic field to control which objects are selected.

3. Click *Show Object Filter* to make sure the object type is selectable.
4. Click *OK*.

The software highlights the specified objects in the design window.

Adding a Property Name to the Search List

In either the schematic window or the symbol window, you can add a property name to the search list using the following procedure.

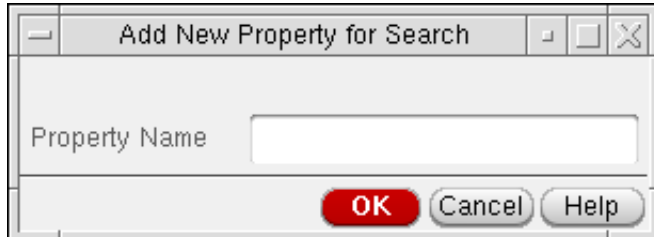
1. From the view, choose *Edit – Select – By Property*.

If you are in the schematic window, the Schematic Select By Property form appears.

If you are in the symbol window, the Symbol Select By Property form appears.

2. In the *Find* cyclic field, choose *other*.

The Add New Property for Search form appears.



3. In the *Property Name* field, type the property name that you want to add to the search list.
4. Click *OK*.

The new property name appears in the *Find* cyclic field on the Schematic or Symbol Select By Property form.

Selecting Objects by Filtering

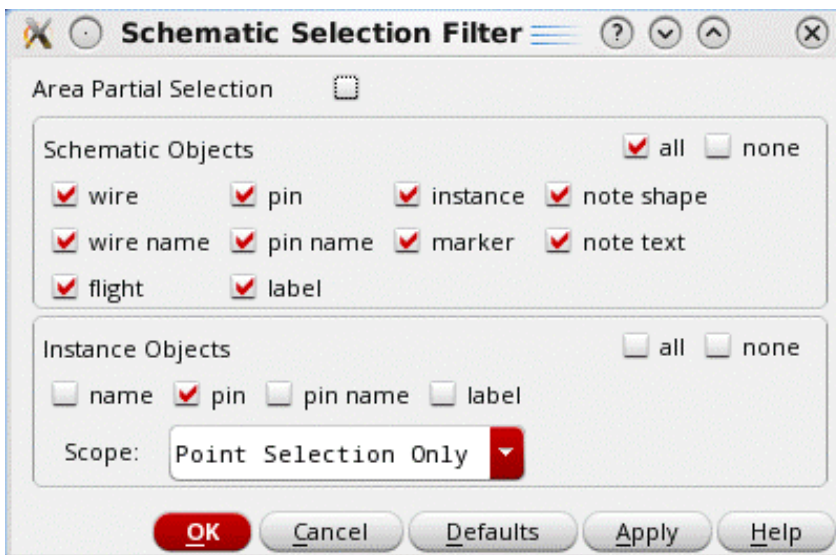
The Selection Filter form lets you specify which object types you can select manually for editing. This is useful when you have many object types, or overlapping object types, in a small area of your design.

For example, if you want to work exclusively with pins, you can turn on selection capabilities for pins only. As a result, if you click another type of object that is near a pin, the other object is not selected.

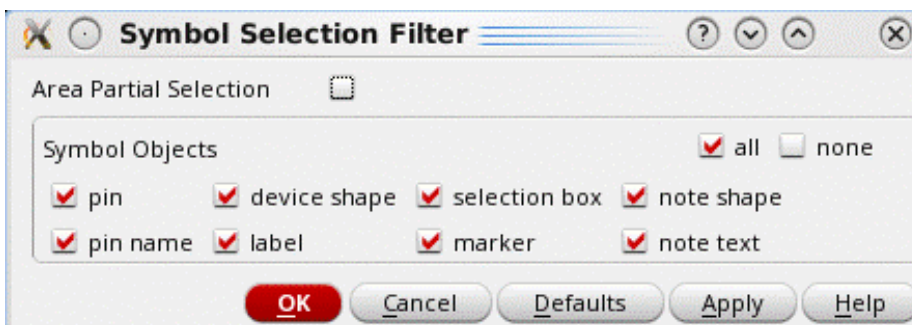
To specify the objects you want to filter for selection,

1. From the view, choose *Options – Select Filter*.

If you are in the schematic window, the Schematic Selection Filter form appears.



If you are in the symbol window, the Symbol Selection Filter form appears.



2. In the *Schematic* (or *Symbol*) *Objects* section, specify the object types you want to select for editing.

The option *none* turns off all object types.

3. Click *OK*.

Note: You can also control which instance objects are available for selection using the *Instance Select Filter*, *Pin Select Filter*, *Wire Select Filter*, or *Text Select Filter* available on the *Select* toolbar. In the schematic view you can use the *Text Select Filter* option to reposition symbol labels.

Object and Cellview Property Types and Values

The following table lists and describes each property type that appears in the *Type* cyclic field on the Add Property, Delete Property, or Modify Property subform.

Property Type	Description
int	Integer
float	Floating-point number, such as 5.32e3
string	Character string
boolean	TRUE, true, t, yes, FALSE, false, nil, or no
ILExpr	SKILL expression
ILList	SKILL list
NLPExpr	Expression evaluated by the netlister substitution language
fileName	Name of a file (relative and absolute paths are OK).
time	Time and date
hierProp	The parent to a list of other properties. Its value is a list of properties.

Object and cellview property values must correspond to the property type shown in the *Type* cyclic field; for example,

- If you use an integer type for a rise time property, you can type 10 for 10 nanoseconds.
- If you select *int* or *float* for the property type, you can type a minimum and maximum value to define a range.

Virtuoso Schematic Editor User Guide

Editing Properties

- The minimum and maximum values must be valid for the *int* or *float* type. The value can be 0 or any positive or negative number. You can also specify *infinity* (no limit over 0), *-infinity* (no limit under 0), or leave the value blank to reflect no limit on the value.
- If you select *string* for the property type, you can specify a list of valid strings separated by spaces.
- The individual word specified represents the only possible values for this property. If the *Choice* field is left blank, any string is valid.
- If you select *time* for the property type, use the following format:

`mmm dd hh:mm:ss yyyy`

where

`mmm` is the month, such as Jan

`dd` is the day of the month, from 01 to 31

`hh` is the hour from 00 to 23

`mm` is the number of minutes from 00 to 59

`ss` is the number of seconds from 00 to 59

`yyyy` is the year, such as 2000

For example, to set the valid time for July 26, 2000, 7:23 a.m., type

`Jul 26 07:23:00 2000`

Editing Object Properties

Properties describe specific characteristics of a design object, such as values of wires and pins. You can add, delete, or modify properties.

Note: You can use [The Property Editor Assistant](#) to compare property values from multiple windows.

This section provides information about the following:

- [Applying Edits to Objects and Displaying Object Types](#)
- [Applying One or More CDF Parameter Edits to Multiple Objects](#)
- [Adding Object Properties](#)
- [Deleting Object Properties](#)
- [Modifying Object Properties](#)

Applying Edits to Objects and Displaying Object Types

To change properties of specific objects,

1. From the view, choose *Edit – Properties – Objects*.

The Edit Object Properties – Basic form appears.



2. Set the *Apply To* options.

The *Apply To* option consists of up to three cyclic fields that you use to specify the group of objects whose properties you want to edit.

A current object exists when you select multiple objects in a selected set. The current object is highlighted in a color different from the selection highlight.

When you configure the *Apply To* cyclic fields to apply changes to multiple objects, the system highlights all affected objects (other than the current object) in a different color.

3. To display the types of properties and attributes that you can edit, set the *Show* options.

The *Show* options let you view and edit properties:

- ☐ Select *system* to display the Cadence properties
- ☐ Select *user* to display the properties that you create
- ☐ Select *CDF* to display predefined properties in Component Description Format (CDF)

4. Select the object in your design to view its properties.

The Edit Object Properties form expands.

Applying One or More CDF Parameter Edits to Multiple Objects

You can select a number of objects in a schematic, and then access the Edit Object Properties form to change CDF parameters for all selected objects.

Note: Only the modified CDF parameters will be set on the selected instances, there will be no unspecified master or parameter changes.

When applying CDF parameter edits to objects, the following CDF callbacks are used:

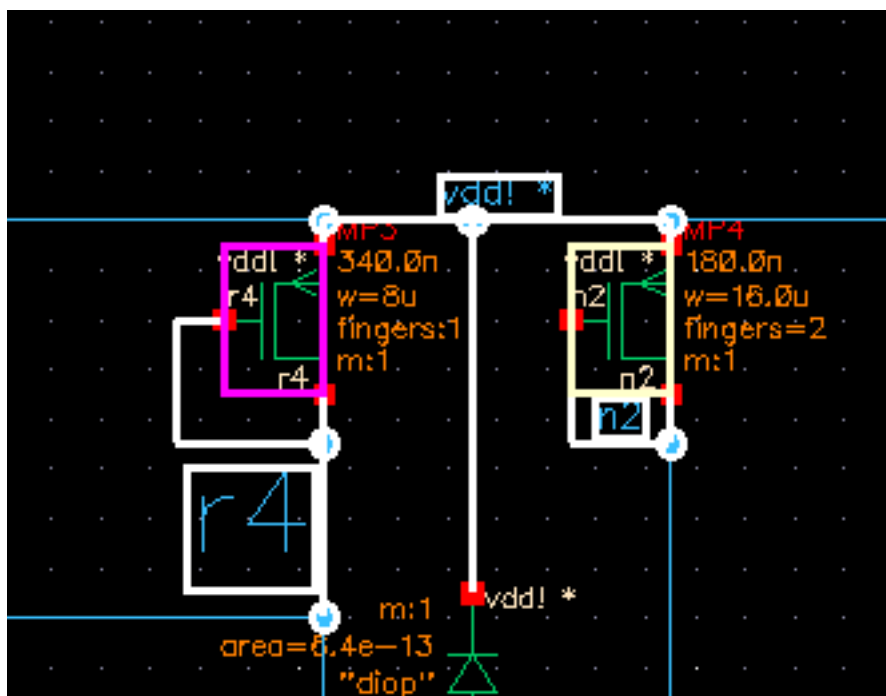
- `formInitProc` will be called for each object, prior to the modification of any parameter.
- `doneProc` will be called for each object, following the modification of any parameter.

Note: Any implicit changes made to object properties, as a result of a callback, are not applied as they are automatically triggered by the callback. Only those changes that are explicitly (manually) made will be applied (in the order they were made).

For more information see the [Component Description Format User Guide](#).

To apply one or more CDF parameter edits to multiple objects,

1. Select the objects on the schematic whose parameters you want to collectively change.



Note: When you select more than one object in the schematic, the first, or lead, object that you selected will have its parameter values displayed in the Edit Object Properties form. The lead object is highlighted in the schematic, in this case MP3.

2. Select *Edit – Properties – Objects* to display the Edit Object Properties form.

Virtuoso Schematic Editor User Guide

Editing Properties

3. In the Edit Object Properties (EOP) form, change the required *CDF Parameter(s)* and *Value(s)* as required.
4. At the top of the EOP form, now choose to, for example, apply your single/multiple CDF parameter update(s) to all selected instances of the same master.
5. Click the *OK* button.

The parameter(s) that you modified will be set on all of the originally selected objects.

Note: A CDF parameter will only be set on a specific object if it is a valid parameter for that object.

For example, consider an initial selection of a resistor that has $r/l/w$ parameters and a capacitor that has $c/l/w$ parameters, with the resistor being the *lead* object displayed in the EOP form. If you now decide to change the r and w resistor parameter values, only the w parameter will be changed on the capacitor once these updates are applied.

Adding Object Properties

To add properties to an object,

1. From the view, choose *Edit – Properties – Objects*.

The Edit Object Properties – Basic form appears.

2. In your design, select an object to which you want to add a property.

3. In the Edit Object Properties form, click *Add*.

The Add Property – Edit Object Properties form appears.

The form varies depending on the object selected.

4. Type the name of the property you want to add and the value of the property.

Note: You should also select the appropriate type of property to be added.

5. Click *OK* to add the properties to the Edit Object Properties form.

6. On the Edit Object Properties form, click *OK* to add the properties to the database object.

Adding Hierarchical Object Properties

Hierarchical properties (hierProps) are properties that can have other (sub) properties attached to them (including other hierProps). This allows you to build up data structures.

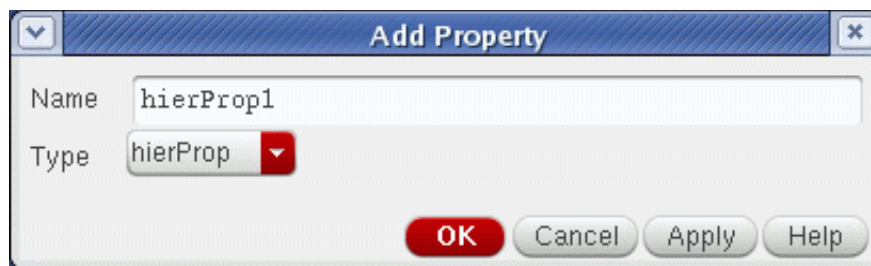
To add hierarchical object properties to an object:

1. From the view, choose *Edit – Properties – Objects*.

The Edit Object Properties – Basic form appears.

2. In your design, select an object to which you want to add a property.
3. In the Edit Object Properties form, click *Add*.

The Add Property – Edit Object Properties form appears.



4. Enter the *Name* of the hierarchical property.
5. From the *Type* pull-down select *hierProp*.
6. Click the *OK* button.

The hierarchical property will be added to the selected object in the *Edit Object Properties* form.

7. Click the *Expand* button.

This will display a properties form for the hierarchical property.



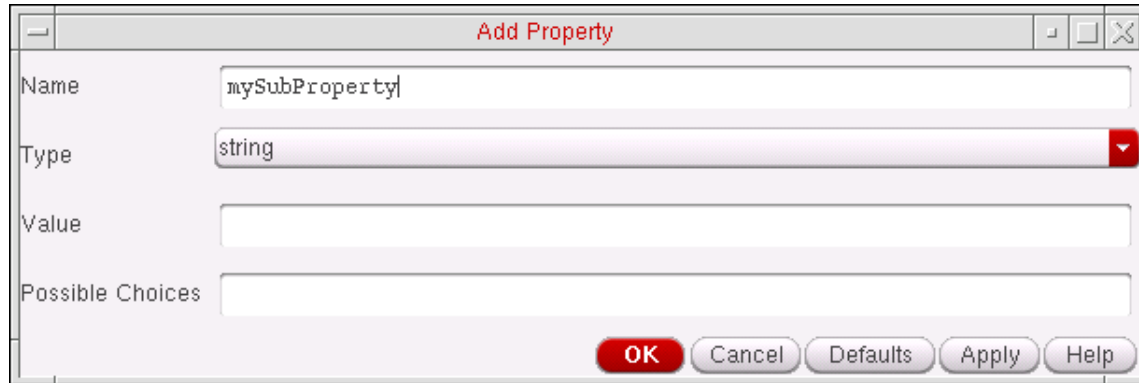
8. Click the *Add* button in the hierarchical properties form.

By clicking the *Add* button on the hierarchical property form you can begin to add sub-properties to the current hierarchical property.

Virtuoso Schematic Editor User Guide

Editing Properties

The Add Property form is displayed.



9. Complete the Add Property form as required:

- ☐ *Name* specifies a name for the new property.
- ☐ *Type* specifies a property type.
- ☐ *Value* specifies a value for the property.
- ☐ *Choices* can be set when *Type* is set to *string* you have the option of setting a list of possible choices for the string value (not mandatory). If the string can have more than one value, type the possible values. Entries in this field must match an entry in the *Value* field.
- ☐ *Minimum* and *Maximum* can optionally be set when *Type* is set to *int*, *float* or *time*. Specify a range for the minimum and maximum range for the integer or floating-point value.

10. Click the *OK* button.

The sub-property will be added to the selected object's hierarchical property in the sub-properties form.



11. You can repeat steps 8-10 to add further sub-properties as required.

Deleting Object Properties

To delete object properties,

1. From the view, choose *Edit – Properties – Objects*.

The Edit Object Properties – Basic form appears.

2. In your design, select an object that has a property you want to delete.

3. In the Edit Object Properties form, click *Delete*.

The Delete Property – Edit Object Properties form appears.

4. From the list box, select the property that you want to delete.

You can press `Shift` and click to select multiple properties.

5. Click *OK*.

After selecting *OK*, the properties will be shown as deleted in the Edit Object Properties form.

6. On the Edit Object Properties form, click *OK* to delete the properties from the database object.

Modifying Object Properties

To modify object properties,

1. From the view, choose *Edit – Properties – Objects*.

The Edit Object Properties – Basic form appears.

2. In your design, select an object that has a property you want to modify.
3. In the Edit Object Properties form, click *Modify*.

The Modify Property – Edit Object Properties form appears.

4. From the list box, select the property that you want to modify.
5. Modify the properties.
6. Click *OK*.

The modified properties appear on the Edit Object Properties form.

7. On the Edit Object Properties form, click *OK* to modify the properties on the database object.

Editing Specific Object Properties

Before you can edit object properties, you must be in edit mode.



Tip

You can also directly edit certain object properties directly on the canvas. For more information, see Editing Object Properties (Text Edit) Directly on the Design Canvas.

You can edit object properties that describe the following objects:

Instances and blocks

Pins

Pin names

Wires

Wire names

Instance pins

Labels

Note text

Note shapes

Symbol shapes

Symbol pins

Symbol labels

Symbol selection box

Net expressions

You can also edit

System properties

User-defined properties

CDF properties

Cellview properties

VHDL properties

Analog Design Environment properties



Caution

To avoid confusion when text, labels, or names contain underscore characters, Cadence does not recommend that you enable the Overbar display option. It is provided only for backward compatibility with prior releases.

Editing Object Properties (Text Edit) Directly on the Design Canvas

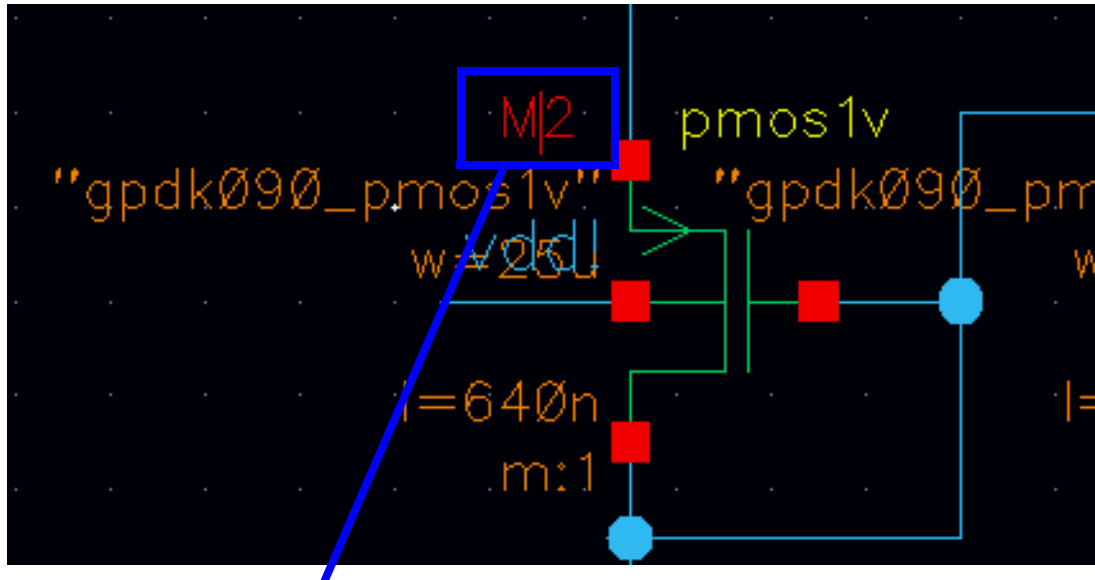


Figure 8-1 Text Label M2 in Direct Text Edit Mode

You can directly edit a text string (including multi-line notes) in the design canvas, if you are in edit mode, by using one of the following text selection mechanisms:

- Selecting the *Direct Text Edit* enter function on *Edit* toolbar.

The status bar will then prompt you to click the text to be edited.



Direct Text Edit

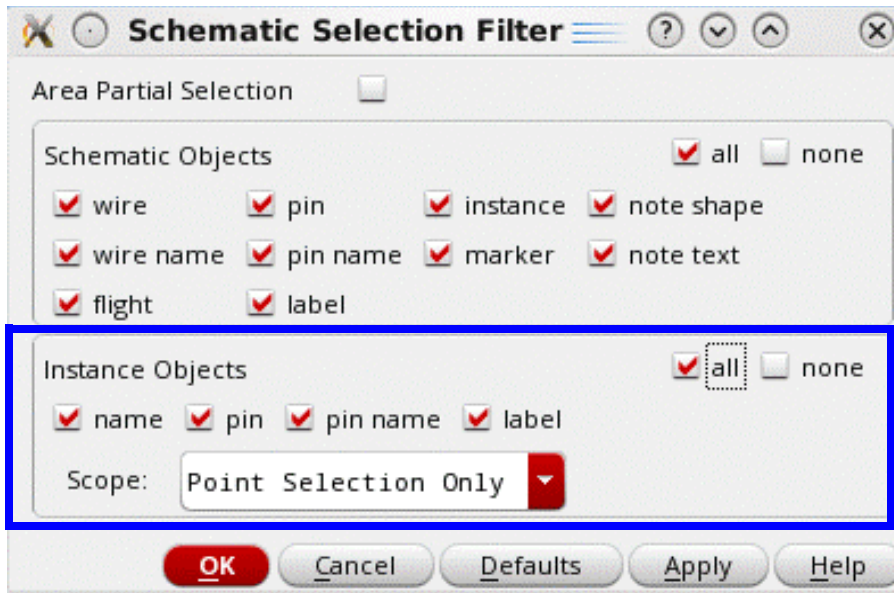
Note: Press the `Return` key to finish your changes or the `Esc` key to cancel editing and revert back to the original text. While typing, you can also use standard text editing shortcuts, for example, you can use the `left/right/Home/End` keys to move the cursor around, to select text you can use `Ctrl-A`, and to select all text you can left-drag the mouse cursor over the text to select. For more information see [Direct Text Edit Bindkeys](#).

- Selecting *Edit – Direct Text Edit* (bindkey `T`).
- Double-clicking on the text (when the text is selectable).

Note: You must ensure that the instance object type has been selected in the [Schematic Selection Filter](#) form (select *Options – Select Filter*).

Virtuoso Schematic Editor User Guide

Editing Properties



Note: When you are in direct text edit mode, the cursor will change to reflect this mode change.

- Clicking the right-mouse button and selecting *Direct Edit Text* from the displayed content-menu.

If they are selectable and editable, the following object properties can be edited directly in any schematic or symbol view that is currently active:

- instance names
- instance labels
- instance parameters
- wire labels
- pin names
- single and multi-line notes

Importantly, you can use direct text edit to change the values of, for example, `cdsParam()` instance labels without having to open the Edit Object Properties form.

Note: If a callback has been attached to the particular string being edited, it will be run to ensure that any new value entered does not exceed any set thresholds.



Figure 8-2 Editing parameter values without using the Edit Objects Properties form

To directly edit object properties on the design canvas:

1. Optionally, zoom in closer to the object text string that you want to edit.
2. Use one of the direct text edit activation methods outlined above to select a text string for editing (for example, a pin name text string as shown below).

The text string will be selected.

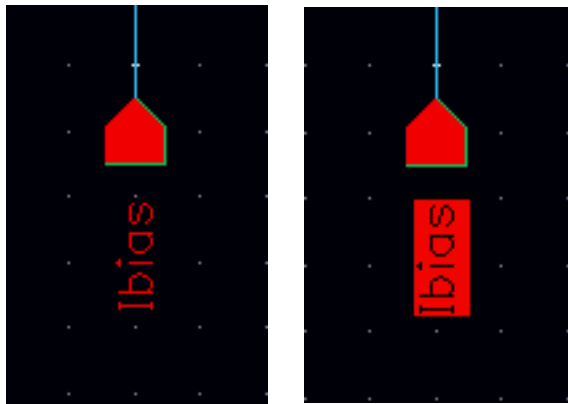


Figure 8-3 Pin name “Ibias” before and after double-click selection for direct text edit

3. Start to type the new string name. The text will update directly.

A cursor is displayed as you type, indicating the current position of the next text character to be inserted (the cursor position can be changed using the cursor keys). If you need to remove any text, use the `delete` key to edit the text string again.

4. When you have completed editing the text string, press the `Return` key to register the updated string name.



Figure 8-4 The updated pin name after direct text edit

Note: The edited object name is reflected in any design intent associated with the object the next time connectivity is updated.

Changing Text Size in the Canvas

To increase or decrease the text size in the design canvas;

1. Choose *Edit – Direct Text Edit*.

Your cursor is now in direct text edit mode.

2. Click the text in the design canvas whose size you want to change.

The text is highlighted.

3. Click incrementally on the *Increase Text Size* or *Decrease Text Size* options on the *Edit* toolbar to alter the selected text as required.

4. Once you have made the text size alteration, press the `Enter` key to register the text size change.

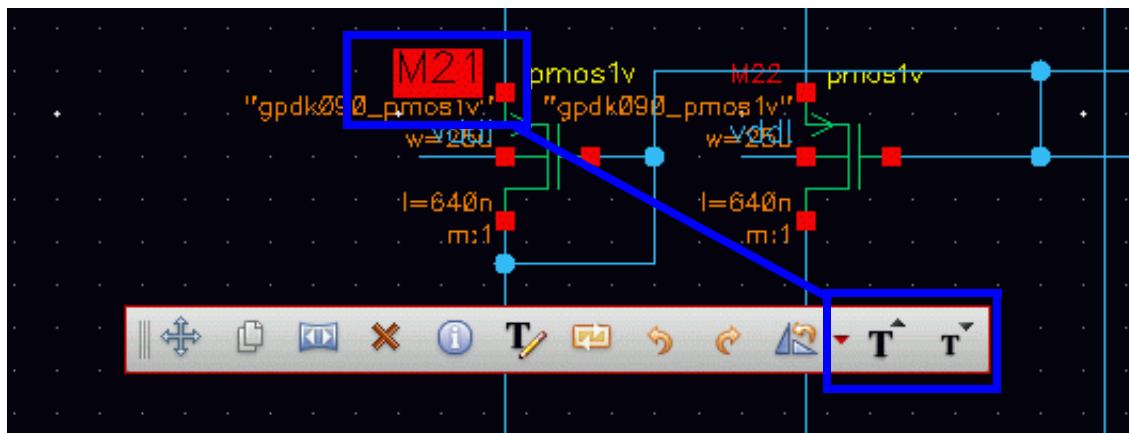


Figure 8-5 Changing the Text Size of Instance M21

Virtuoso Schematic Editor User Guide

Editing Properties

Direct Text Edit Bindkeys

The following bindkeys can be used when in Direct Text Edit mode:

Bindkey	Action
Up/Down arrow keys	Moves up and down one line.
Left/Right arrow keys	Moves one character to the left or right.
Shift-Up/Down/Left/Right	Selects additional text from the placement area of the cursor.
Home/End	Moves to the beginning or end of a text note.
Ctrl+Insert	Copies selected text to the clipboard.
Ctrl+A	Selects whole note.
Ctrl+X	Cuts selected text.
Ctrl+V	Pastes selected (clipboard) text.

Note: You can also paste clipboard text using the middle-mouse button.

Note: If you want to paste text from one label to another label:

1. Put one label in edit mode.
2. Highlight the text you want to copy.
3. Use `Ctrl+Insert` to copy the selected text.
4. Put the second label into edit mode.
5. Click or move the cursor to where you want to insert the text (or highlight the text you want to replace).
6. Select the middle-mouse-button or `Ctrl+V` to paste the text.

Ctrl+Z	Undo edits.
Shift+Enter	Creates a new line of text.
Enter	Validates and text edits made and quits the note edit.
Esc	Cancels and quits note edit mode.

Text That Cannot Be Edited

If text cannot be edited, a *No Stop* icon will be displayed.

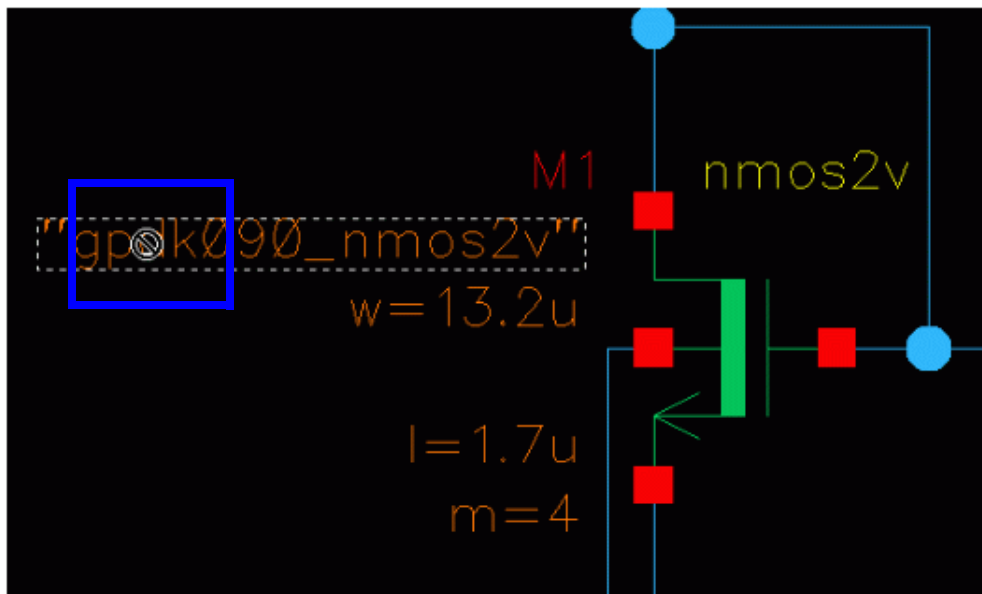


Figure 8-6 Text Edit Not Permitted

Virtuoso Schematic Editor User Guide

Editing Properties

Editing Instance and Block Properties

To edit instance and block properties and their parameters and values,

1. From the view, click an instance or block that has a property you want to edit.
2. Choose *Edit – Properties – Objects* to display the Edit Object Properties – Instance and Block form.

Edit Object Properties

Apply To:

Show: ☐ system ☒ user ☒ CDF

Property	Value	Display
Library Name	sip_components	off
Cell Name	filter_bp_smd	off
View Name	symbol	off
Instance Name	I109	value

User Property	Master Value	Local Value	Display
\$REFDES		D1	off
\$SEC		1	off
REFDES		D1	off
SEC		1	off
interfaceLastCha..	15 07:18:44 1998		off
lastSavedAutoExtr..	15 15:18:28 1998		off
partName	chebyshev_bp		off
vendorName			off

CDF Parameter	Value	Display
Filter Order	3	off
Input impedance	50	off
Output impedance	50	off
Relative bandwidth	0.1	off
Center frequency(Hz)	1e+09	off

If you clicked on an instance in your design,

- ❑ The *Magnification* and *Origin* fields appear when you enable the *Show system* field
- ❑ The *Display* cyclic field controls whether the value of a property appears in your schematic

If you clicked on a block in your design,

- ❑ The *Origin*, and *FreezePinCreate* fields appear when you enable the *Show system* field.

To automatically add a pin when you add a wire to a block, turn on *FreezePinCreate*.

You can change the direction and name of a block pin when you enable the *Show system* button.

To change the *Instance Name* default field, see Changing the Default Instance Name Prefix.

3. Click *OK*.

Editing Pin Properties

To edit pin properties and their parameters and values,

1. From the schematic window, click a pin that has a property you want to edit.
2. Choose *Edit – Properties – Objects*.

The Edit Object Properties – Pin form appears.

3. Change the *Direction* or *Usage* cyclic field of the pin.
4. Click *OK*.

Designating the Order of Pins

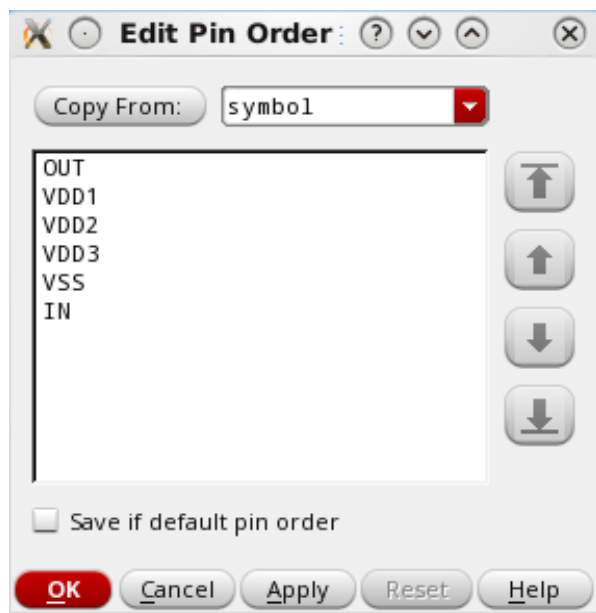
A pin order property is often required for netlisting. You can update or swap the pin order either manually or by copying.

Note: CDLOUT, Verilog-XL Integration, and VHDL Integration support the portOrder property.

To manually redefine the logical pin ordering for netlisting,

1. From the view, choose *Edit – Properties – Pin Order*.

The Edit Pin Order form appears.



2. Select a pin name by clicking on it in the list box.
3. Click one of the move buttons to move the selected pin name up or down in the list.
4. If the order of pins you set is the same as the default order, select the *Save if default pin order* check box to save it.

Note: The default port order is output input inputOutput, and the pins in each type are sorted alphanumerically.

5. Click *OK*.

Copying a Pin Order from Another Cellview

To copy a pin order from another cellview,

1. From the view, choose *Edit – Properties – Pin Order*.

The Edit Pin Order form appears.

2. Select a pin name by clicking on it in the list box.
3. Choose a view name from the *Copy From* pull-down from which to copy the pin order from.
4. If the order of pins you copied is the same as the default order, select the *Save if default pin order* check box to save it.

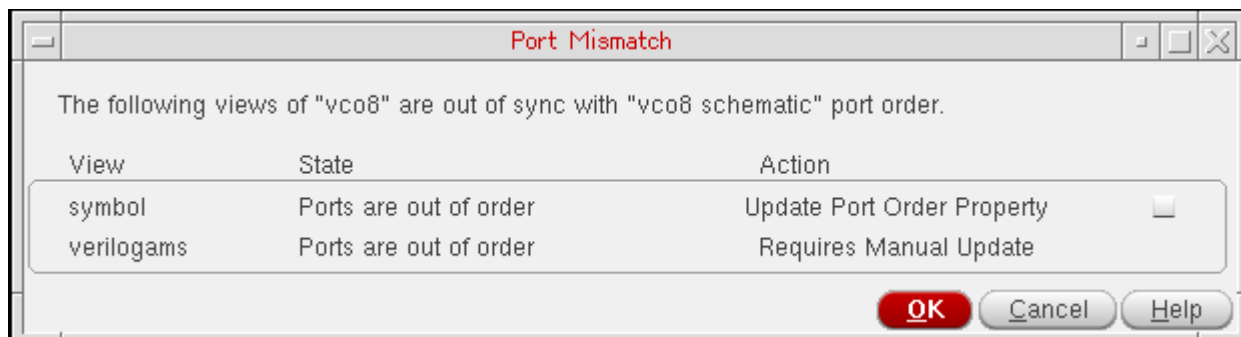
Note: The default port order is output input inputOutput, and the pins in each type are sorted alphanumerically.

5. Click *OK*.

The pin order in the view list window changes to match the selected view.

Resolving Pin Mismatch

If the pins you want to copy are out of sync with the pins in the existing view, the Port Mismatch form appears.



In the Port Mismatch form, you are given details of:

- The view name that has mismatched ports (*View*).
- The reason the view has mismatched ports (*State*).

The reasons could be:

- ☐ The port order of the view is different from the current view.
- ☐ The port names are different from that of the current view, which means either an extra port or a missing port has been detected.

- The action you must take to resolve the mismatch (*Action*).

Here, the following recommended actions could be:

- ☐ *Update Port Order Property*
- ☐ *Regenerate Symbol*
- ☐ *Requires Manual Update*

The first two of these recommended actions can however be automated using the `disablePortOrderPopup` environment variable.

Setting the boolean envar `disablePortOrderPopup` to `t` instructs VSE to perform these actions without displaying the Port Mismatch form. If a manual update is recommended, the form still displays irrespective of the environment variable setting.

Note: By default, the Port Mismatch form is displayed (`disablePortOrderPopup=nil`).

If you want to disable only the port order check, set the environment variable `vicCheckPinOrder`. However, the port name mismatch issues in the cellviews are still reported. By default, the port order check is run (`vicCheckPinOrder=t`).

Verifying Pin Order

To verify pin order validity of your most recently defined pin ordering,

- From the view, choose *File – Check and Save*.

If you added, deleted, or renamed pins since you last defined the logical pin ordering, this command opens the Edit Pin Order form so you can verify or redefine the new pin ordering.

Editing Pin Name Properties

To edit pin name properties and pin name parameters and values,

1. From the view, click a pin.
2. Choose *Edit – Properties – Objects*.

The Edit Object Properties – Pin Name form appears.

Virtuoso Schematic Editor User Guide

Editing Properties

3. Change the *Height*, *Font*, or *Justification* fields.
4. Click *OK*.

Editing Wire Properties

To edit the properties of a wire:

1. From the view, click a wire.
2. Choose *Edit – Properties – Objects*.

The Edit Object Properties – Wire Segment form appears.

3. Edit the values in the *Width*, *Signal Type*, *Color*, or *Line Style* fields.
4. Click *OK*.

Editing Wire Name Properties

To edit wire name parameters and values:

1. Select a wire name/label on the design canvas.

Note: If you select a wire segment, the net name will be displayed in the Edit Object Property form, but it is not editable. You can only edit the wire name/label by selecting the wire label directly.

2. Choose *Edit – Properties – Objects*.

The Edit Object Properties – Wire Name form appears.

3. Edit the *Label*, *Height*, *Font*, *Justification* or *Bundle Display* fields as appropriate.
4. Click *OK*.

The connectivity automatically updates when any changes are applied.

Editing Instance Pin Properties on an Instance of a Block

To edit instance pin properties on an instance of a block,

1. From the view, click an instance pin.

Note: You can only edit basic properties such as *name* and *direction*. User properties are editable for instance pins of normal instances.

2. Choose *Edit – Properties – Objects*.

The Edit Object Properties – Instance and Block form appears.

3. Click *Add*, *Delete*, or *Modify*.
4. Edit the form that appears.
5. Click *OK* on both forms.

Editing Label Properties

To edit label properties and label parameters and values,

1. From the view, click a label whose properties you want to edit.
2. Choose *Edit – Properties – Objects*.

The Edit Object Properties – Label form appears.

3. Change the *Height*, *Font*, or *Justification* fields.
4. Click *OK*.

Resetting Invisible Labels

If you turned off label display on the Edit Object Properties form while editing a label, you can reset invisible labels either manually or automatically.

To manually reset (make visible) individual labels,

1. From the view, choose *Edit – Select – Filter*.

Make sure the labels are selectable.

2. Choose *View – Reset Invisible Labels*.

All the invisible labels appear as yellow (or the color of the marker/warning lpp).

All the visible labels appear as nonbinding highlighted objects.

3. Click each of the highlighted labels that you want to make visible.

When you select an invisible label, the label color changes to white to indicate that this label would be made visible.

When you click again on a label to deselect it, it returns to being yellow again, indicating it will remain invisible when you quit the command.

4. Press the `ESC` key to end the command.

The selected labels remain visible in your design.

To automatically reset all invisible labels,

1. From the view, choose *View – Reset Invisible Labels*.

All invisible labels appear as blinking highlighted objects.

2. Press the `F3` key.

The Reset Invisible Labels form appears.

3. Turn on *visible*.

All the invisible labels remain highlighted but stop blinking.

4. Click *Hide*.

The Reset Invisible Labels form closes.

5. Press the `ESC` key to end the command.

The selected labels remain visible in your design, regardless of the Selection Filter settings.

Showing/Hiding Instance Labels

You can toggle the display of instance labels in any cellview that is open in the current session by selecting *View – Show/Hide Instance Labels*.

Note: Hiding instance labels can make schematics easier to read, and can also allow you to take a snapshot of the schematic without parameter values being displayed.

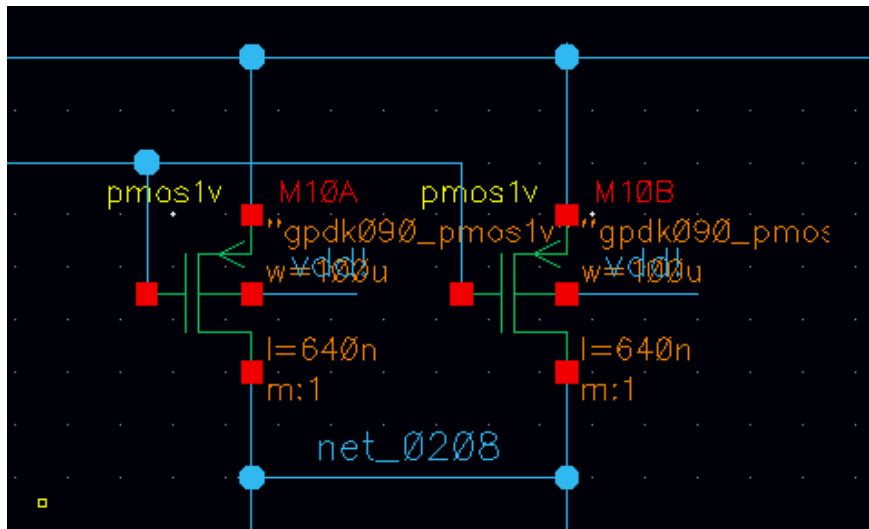


Figure 8-7 Schematic with Instance Labels Displayed

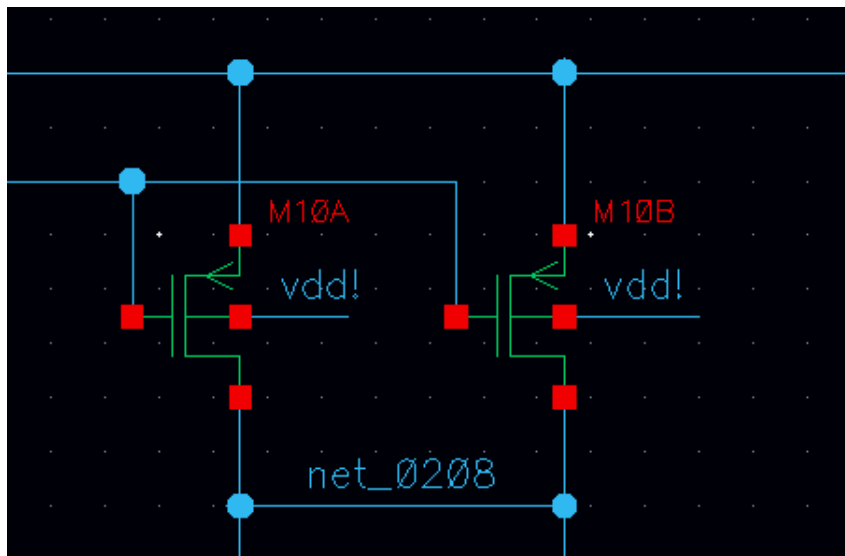


Figure 8-8 Schematic with Instance Labels Hidden

Editing netSet Properties

Refer to [Adding Net Expression Labels to Create an Inherited Connection](#) and [Adding netSet Properties to Create an Inherited Connection](#) for information about the `netSet` property.

Editing Note Text Properties

To edit text of a note after you add it to your design,

1. From the view, select the note text.
2. Choose *Edit – Properties – Objects*.

The Edit Object Properties – Note Text form appears and displays text in the *Note Text* field.

3. Make your changes using the default keystrokes (see link below).

Important

Virtuoso IC6.x uses a Qt-based UI. As part of this UI system, the default bindings for text fields were updated (from Xt/Motif-based GUI in IC5.1.41). For more information, see Default Keybindings for Text Fields in the *Virtuoso Design Environment User Guide*.

4. Click *OK*.

Editing Note Shape Properties

To change the line property of a note shape from a solid line to a dashed line or to reposition a note shape using coordinates,

1. From the view, click a note shape.
2. Choose *Edit – Properties – Objects*.

The Edit Object Properties – Note Shapes form appears.

3. Change *Line Style* or the position of the note shape.
4. Click *OK*.

You can also use the *Move* or *Stretch* commands for repositioning.

Editing Symbol Shape Properties

To change the width property of a symbol shape from narrow to wide or to reposition a symbol shape using coordinates,

1. From the view, click a symbol shape.
2. Choose *Edit – Properties – Objects*.

The Edit Object Properties – Symbol Shape form appears. The form appearance changes to show all properties for the type of object selected.

3. Edit the appropriate fields.
4. Click *OK*.

You can also use the *Move* or *Stretch* commands for repositioning.

Editing Symbol Pin Properties

To edit symbol pin properties and symbol pin parameters and values,

1. From the view, click a symbol pin.
2. Choose *Edit – Properties – Objects*.

The Edit Object Properties – Symbol Pin form appears. The form appearance changes to show all properties for the type of object selected.

3. Change the *Direction* cyclic field of the symbol pin.
4. Click *OK*.

Editing Symbol Label Properties

To edit symbol label properties and symbol label parameters and values,

1. From the view, click a symbol label.
2. Choose *Edit – Properties – Objects*.

The Edit Object Properties – Symbol Label form appears. The form appearance changes to show all properties for the type of object selected.

3. Change the properties of the label.
4. Click *OK*.

Editing Symbol Selection Box Properties

To change the position of the selection box,

1. From the symbol window, click the selection box.
2. Choose *Edit – Properties – Objects*.

The Edit Object Properties – Selection Box form appears.

3. Change the position of the selection box by editing the coordinates.

You can also use the *Move* or *Stretch* commands for repositioning.

4. Click *OK*.

Editing User-Defined Object Properties

To edit visible user-defined object properties, values, and their display attributes,

1. From the view, click the object property.
2. Choose *Edit – Properties – Objects*.

The Edit Object Properties – Other Object form appears. The form's appearance depends on whether you click the property master or a property instance. If you select a user-defined property instance, the Edit Object Properties form displays values for *Name* and *Value*.

If you select a user-defined property master, the Edit Object Properties form displays values for *Master Value* and *Local Value*. You cannot edit the *Master Value* field.

3. Change the property values of *Local Value*, *Height*, *Font*, or *Justification*.
4. Click *OK*.

Passing Parameters in a Design

To pass a parameter from the symbol view to the schematic, set up a CDF parameter for the parameter under consideration.

1. From the schematic window, choose *Edit – Properties – Objects* on the instance and change the value of the instance parameter from a fixed value.
2. Choose *Create – Create Cellview – From Cellview* and make appropriate drawing shape changes to the symbol.
3. Save the symbol view.
4. Edit the CDF for this component by choosing *CIW – Tools – CDF – Edit*.

The Edit Component CDF form appears.

5. Complete the form as required, referring to CDF Commands in the Component Description Format User Guide if required.

Editing Cellview Properties

While using the *Edit – Properties – Cellview* command, you can add, delete, and modify the properties that appear on the form.

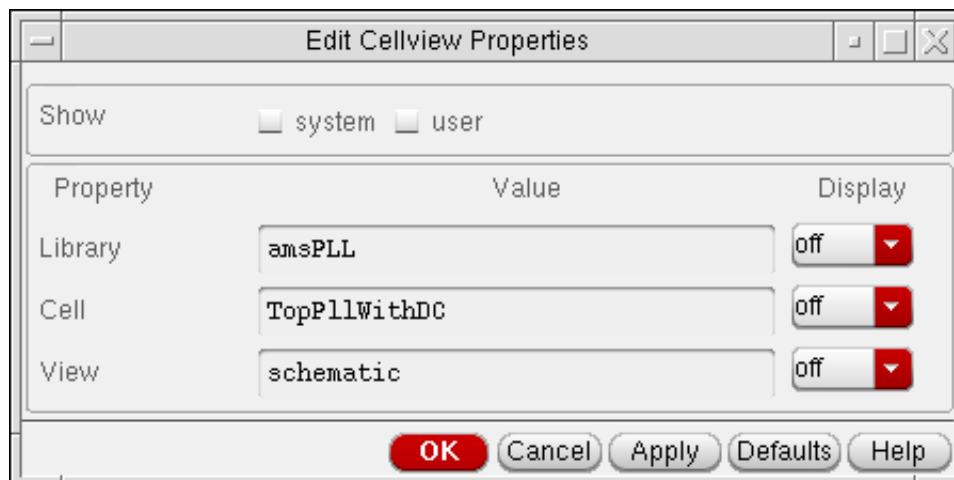
To add, modify, or delete properties, open a subform. You use the subform to specify the property type and value. When you click *OK* on the subform, the form reflects the change.

When you click *Apply* or *OK* on the form, the change is applied to the specified object in the design window.

To display cellview properties that you want to edit,

1. From the view, choose *Edit – Properties – Cellview*.

The Edit Cellview Properties form appears.



2. Set the *Show* option as follows:

- ☐ Select *system* to display attributes that are less frequently used.
- ☐ Select *user* to display properties that you create.

3. Click *OK*.

Adding Cellview Properties

The *Add* option lets you add cellview properties.

To add a cellview property,

1. From the view, choose *Edit – Properties – Cellview*.

The Edit Cellview Properties form appears.

2. Set the *Show* option to *user*.

3. At the bottom of the Edit Cellview Properties form, click *Add*.

The Add Property – Edit Object Properties form appears.



4. Type a new property name, and give it a value.

Note: You should also select the appropriate type of property being added.

5. Click *OK*.

6. In the Edit Cellview Properties form, click *OK*.

Deleting Cellview Properties

To delete a cellview property,

1. From the view, choose *Edit – Properties – Cellview*.

The Edit Cellview Properties form appears.

2. Set the *Show* option to *user*.

3. Click *Delete*.

The Delete Property – Edit Object Properties form appears.

4. Choose the property you want to delete.

You can *Shift*-click to select multiple properties

5. Click *OK*.

6. In the Edit Cellview Properties form, click *OK*.

Modifying Cellview Properties

The *Modify* option lets you modify cellview properties.

To modify a cellview property,

1. From the view, choose *Edit – Properties – Cellview*.

The Edit Cellview Properties form appears.

2. Set the *Show* option to *user*.

3. Click *Modify*.

The Modify Property – Edit Object Properties form appears.

4. Choose the property you want to modify.
5. Change the property values for *Type*, *Name*, and *Value*.
6. Click *OK*.
7. In the Edit Cellview Properties form, click *OK*.

Editing VHDL Properties

To display VHDL properties in a cellview so you can set specific netlisting options,

1. From the view, choose *Edit – Properties – VHDL*.

The VHDL Properties form appears.

The screenshot shows the 'VHDL Properties' dialog box. It features a 'Generics' section with a table for defining generic parameters. The table has three columns: 'Name', 'Type', and 'Value'. Below the table are buttons for 'Add', 'Modify', and 'Delete'. Below these are input fields for 'Scalar Data Type', 'Vector Data Type', and 'Use Clause'. A large text area for 'User Comment' is at the bottom. The dialog is styled with a standard Windows-style title bar and buttons.

2. Type editing commands in the *User Comment* text box.

Note: To set netlister options for each cellview for VHDL, refer to the Virtuoso VHDL Toolbox User Guide.

Adding VHDL Properties

The *Add* button lets you add properties that represent VHDL generics.

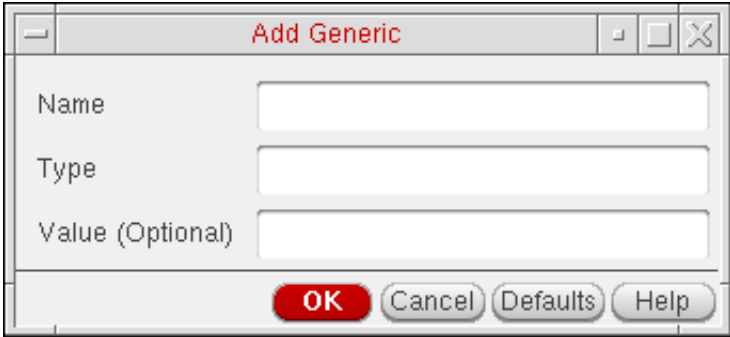
To add a VHDL property,

1. From the view, choose *Edit – Properties – VHDL*.

The VHDL Properties form appears.

2. Click *Add*.

The Add Generic – VHDL Properties form appears.



3. Type a generic name, a type, and a value.
4. Click *OK*.
5. In the VHDL Properties form, click *OK*.

Deleting VHDL Properties

The *Delete* button lets you delete properties that represent VHDL generics.

To delete a VHDL property,

1. From the view, choose *Edit – Properties – VHDL*.

The VHDL Properties form appears.

2. Click *Delete*.

The Delete Generic – VHDL Properties form appears.

3. Type the generic name you want to delete.

4. Click *OK*.

5. In the VHDL Properties form, click *OK*.

Modifying VHDL Properties

The *Modify* button lets you modify properties that represent VHDL generics.

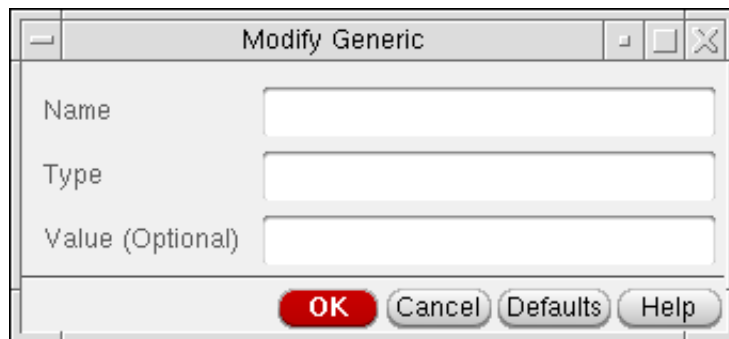
To modify a VHDL property,

1. From the view, choose *Edit – Properties – VHDL*.

The VHDL Properties form appears.

2. Click *Modify*.

The Modify Generic – VHDL Properties form appears.



3. Type a generic name, a type, and a value.
4. Click *OK*.
5. In the VHDL Properties form, click *OK*.

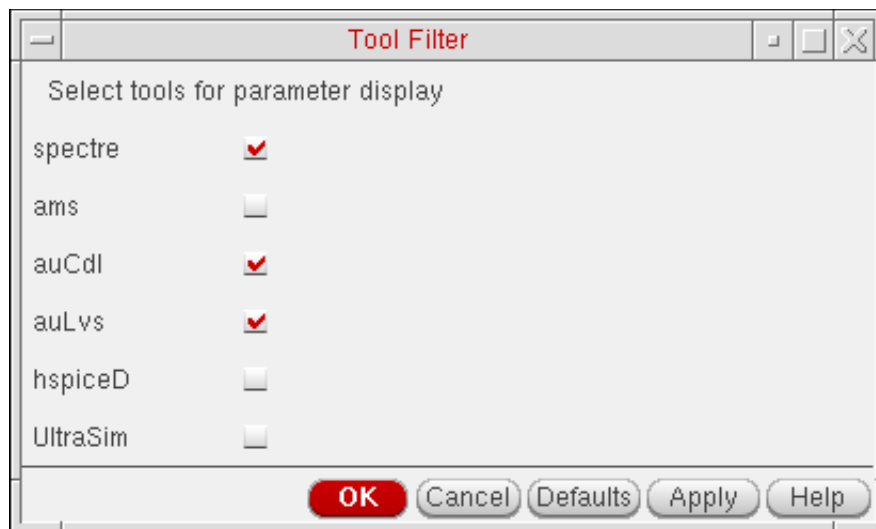
Setting the Tool Filter

This section explains how to edit analog circuit design properties using the tool filter.

To display the analog circuit design simulators that are active in your cellview,

1. From the schematic window, choose *Options – Tool Filter*.

The Tool Filter form appears.



2. Save your cellview.
3. Click the *Defaults* button.

The system displays the simulators that were active when you last saved your cellview.

By default, UltraSim is selected on the Tool Filter form. You can override this default by setting `auCore.toolFilter defaultTools` in your `$HOME/.cdsenv` file. You can also change the default within your entire site by changing the `dfII/etc/tools/auCore/.cdsenv` file. For example, if you want to specify only `spectre` as the active default, add the following to your `.cdsenv` file:

```
auCore.toolFilter defaultTools string "spectre" nil
```

4. Select the tools for the Tool Filter display.
5. Click *OK*.

Activating the Auto-Update Tool Filter

To activate the auto-update tool filter,

- In your `.cdsenv` file, set the `autoUpdate` environment variable as follows:

```
auCore.toolFilter autoUpdate 'boolean t
```

When auto-update is activated, the tool filter setting is updated to show CDF parameters for the simulator selected in ADE. This situation occurs when you

- Bring up ADE from a schematic window
- Change the simulator in an ADE session to which a schematic window is attached
- Bring up a schematic window from ADE
- Select a new design in an ADE session to which a schematic window is attached

When auto-update is activated, the tool filter setting does not change when you traverse through the hierarchy in the schematic window. That is, the same tool filter setting applies for any instance at any level in the hierarchy.

Note: If you have the *Create New Window When Descending* option set, you loose the tool filter setting as you descend down the hierarchy.

You can change the default of the `autoUpdate` environment variable by setting `auCore.toolFilter autoUpdate` in your `$HOME/.cdsenv` file or in the `dfII/etc/tools/auCore/.cdsenv` file to change it sitewide. To deactivate this feature, set `autoUpdate` to `nil`.

Virtuoso Schematic Editor User Guide

Editing Properties

Traversing the Design Hierarchy and Creating a Design Configuration View

This chapter provides information about the following topics:

- [About the Design Hierarchy](#) on page 624
- [Exploring the Design Hierarchy](#) on page 625
- [About the Configuration View of a Design Hierarchy](#) on page 630
- [About the Hierarchy Editor](#) on page 631
- [Traversing a Design Hierarchy](#) on page 632
- [Displaying the Current Hierarchical Scope](#) on page 641
- [Displaying Cellview Listings Using Print Tree](#) on page 642
- [Creating a New Configuration View for the Design Hierarchy](#) on page 644
- [Opening an Existing Design Configuration View](#) on page 645
- [About the Hierarchy Editor within the Virtuoso Schematic Editor](#) on page 646
- [Working with a Design Configuration within the Virtuoso Schematic Editor](#) on page 647

About the Design Hierarchy

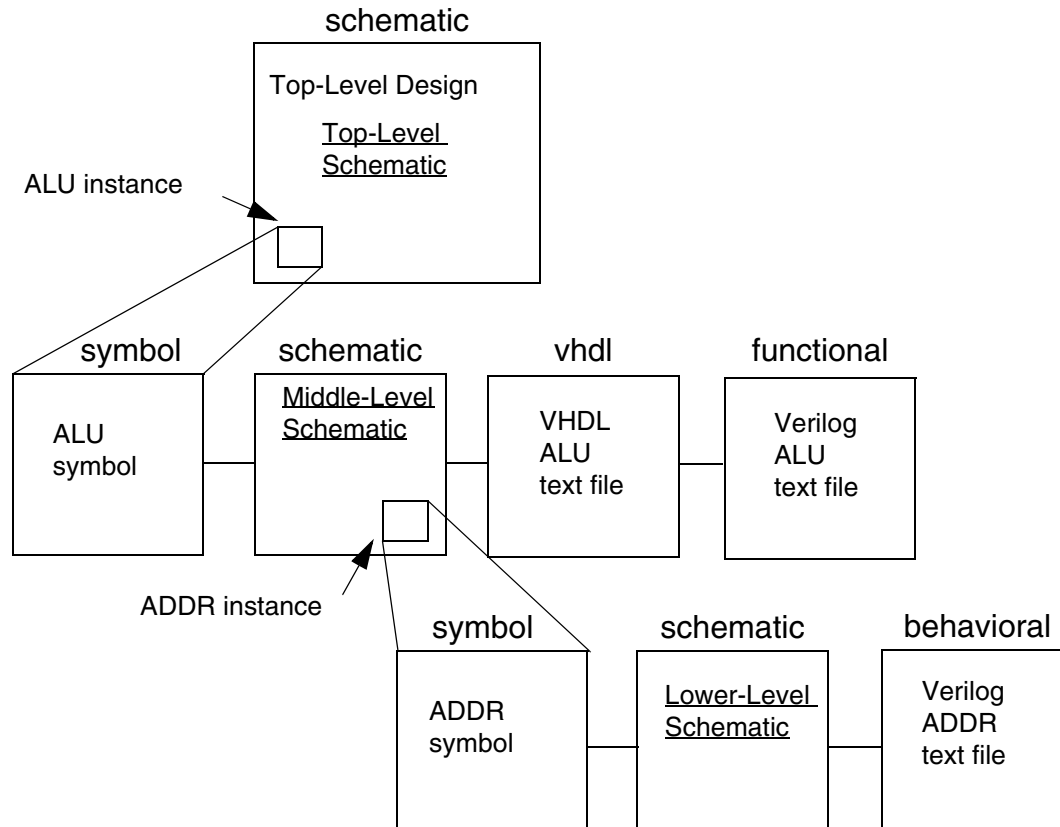
A design hierarchy comprises many levels of a single design. Each time you descend into the design hierarchy, you view a smaller division of the larger design in greater detail. Working in smaller divisions of a design lets you more effectively distribute work across multiple design engineers or groups.

A single instance in a hierarchical design typically represents multiple types of different models. For example, an ASIC may include an underlying gate-level schematic, a layout, or a number of different behavioral descriptions, or a model. A model card is a collection of default parameters for a device. A model file is a collection of model cards. The specification of model cards is different based on the tool and the interface you use.

You have the flexibility to define the relationship between an instance and its various representations. You can descend into any particular lower-level design representation for examination or editing.

Exploring the Design Hierarchy

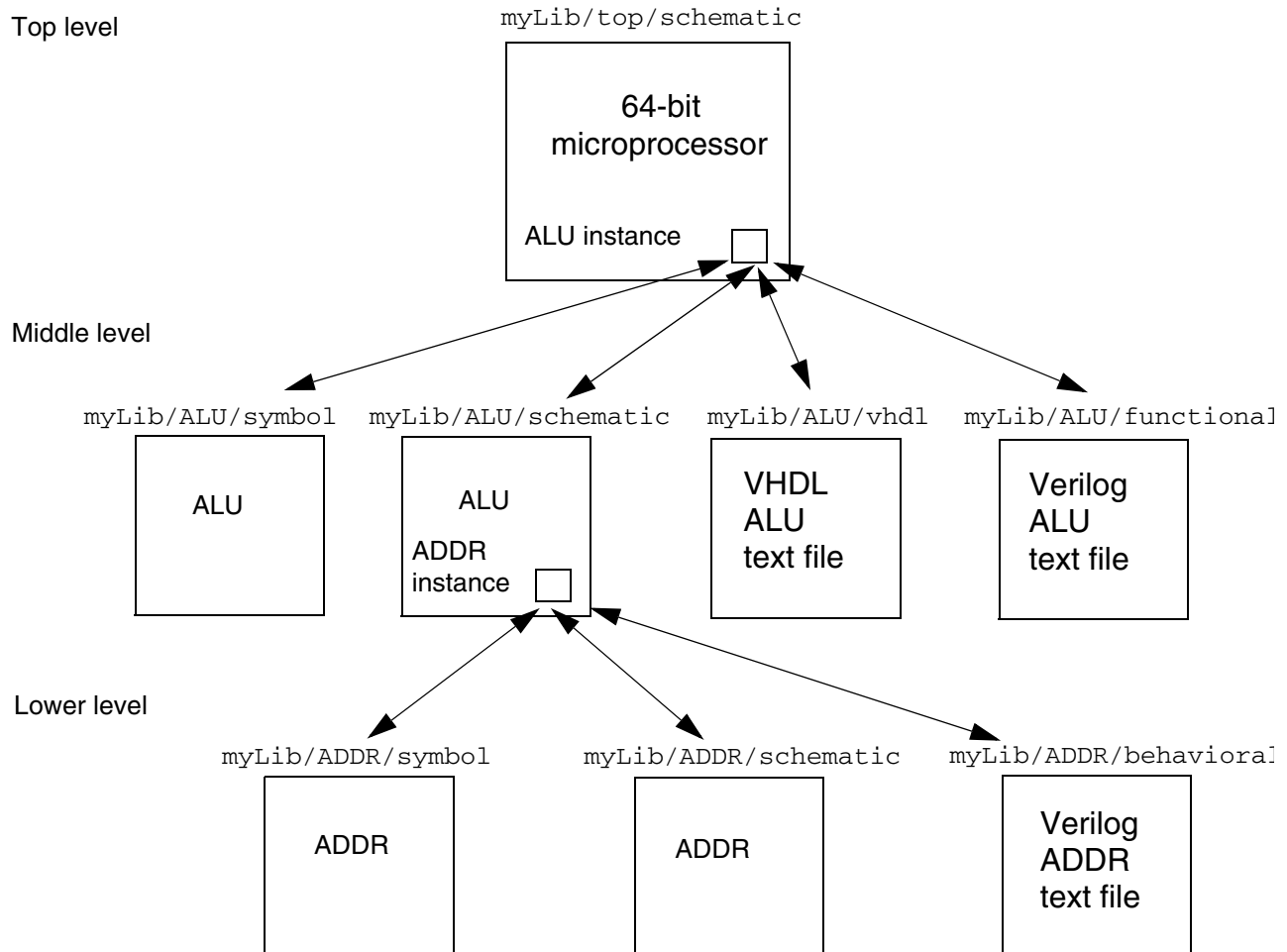
The following diagram shows an ALU and an ADDR symbol view instantiated in a schematic view.



Virtuoso Schematic Editor User Guide

Traversing the Design Hierarchy and Creating a Design Configuration View

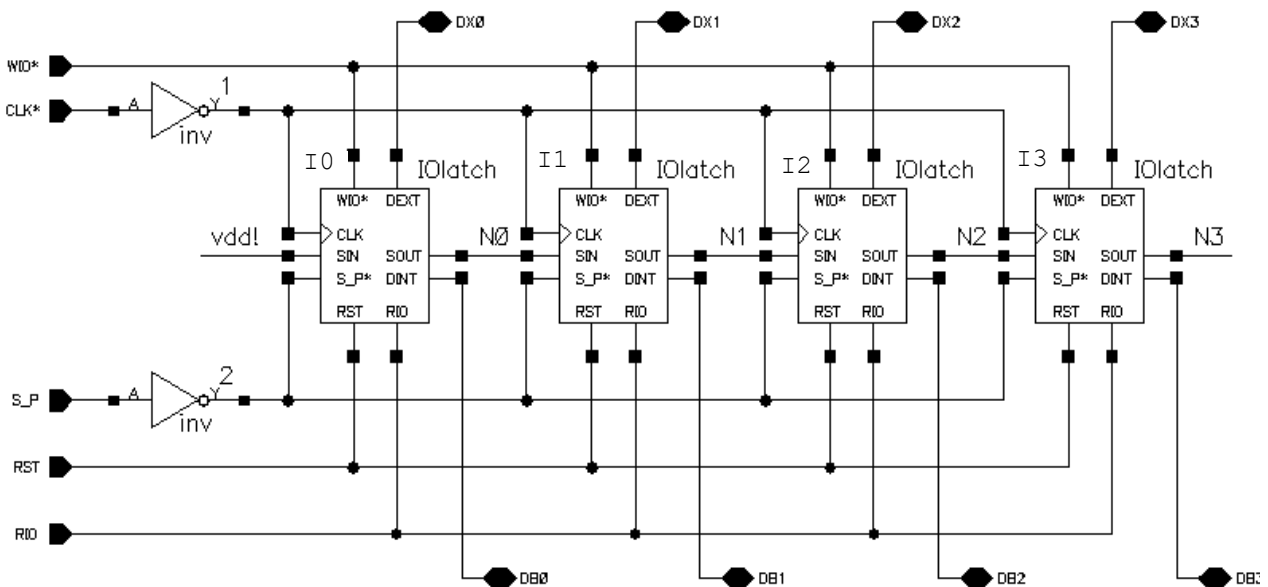
The following diagram shows the path (library, cell, and view name) for each view. You can traverse into any existing view of the cell that is referenced by an instance and whose view name has a corresponding view type in a registered tool.



Top-Level Schematic View

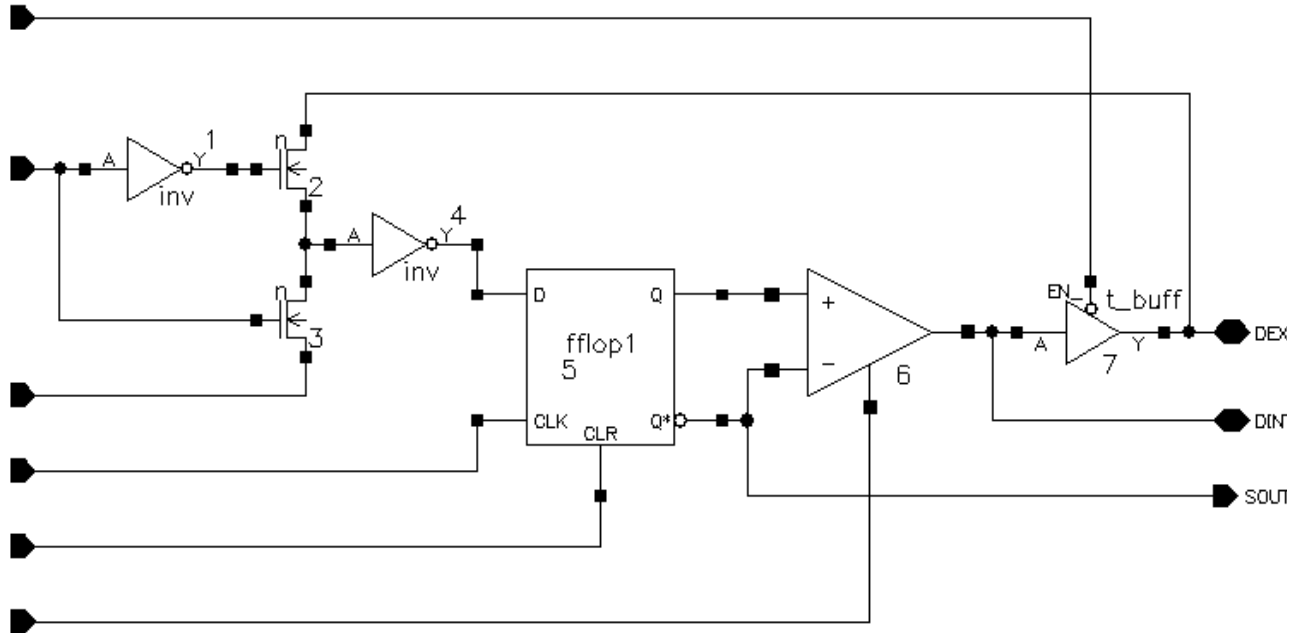
The following illustration emulates descending from a top-level I/O register schematic into a middle-level I/O latch schematic.

If you click the `I3 IOlatch` instance shown below, you descend into the schematic view (see next page) of an I/O latch for the instance named `I3`.



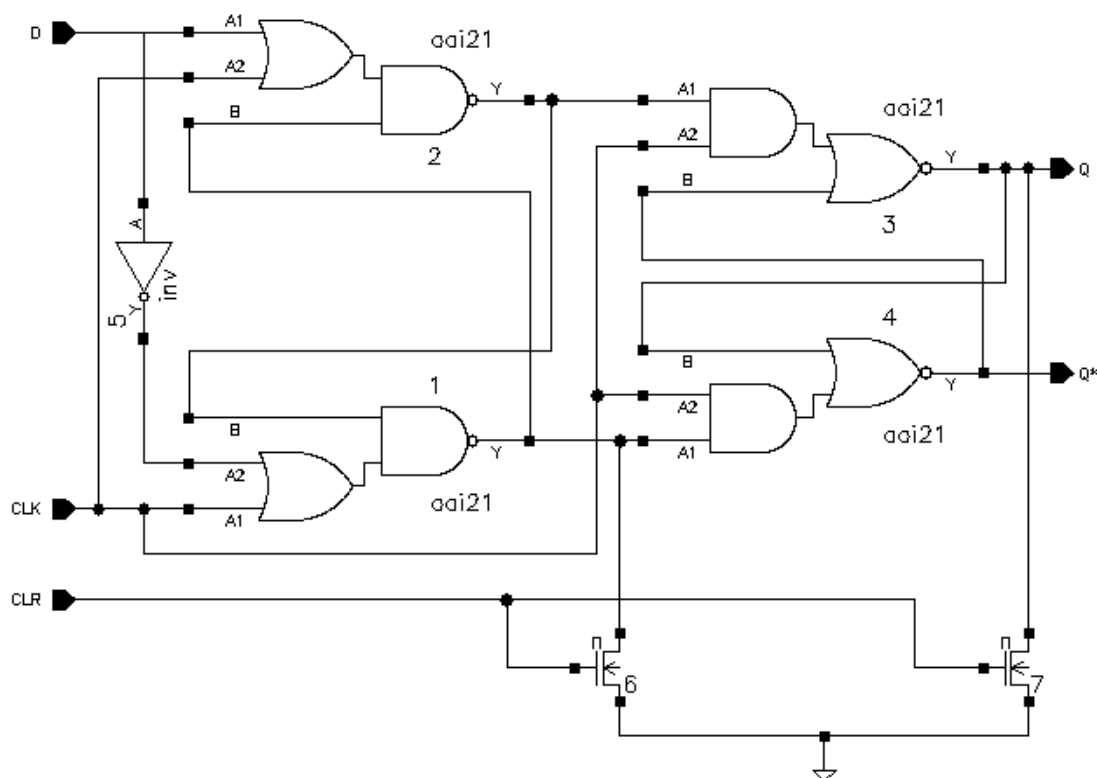
Middle-Level Schematic View

If you click the `fflop1` component shown below, you descend from a middle-level I/O latch schematic into the lower-level `fflop1` schematic (see next page).



Lower-Level Design Hierarchy

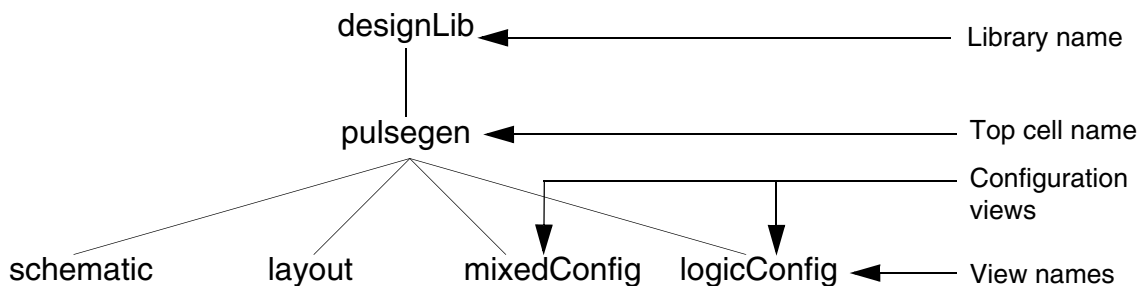
The following illustration shows the lowest level schematic of a design in the hierarchy of this particular design, the `fflop1` schematic. The `fflop1` schematic instantiates the primitive logic devices (or leaf nodes).



About the Configuration View of a Design Hierarchy

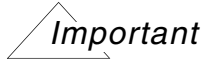
Setup and netlisting processes and simulations should be run from the configured view rather than the original schematic so that you can access the expansion information.

The Cadence® hierarchy editor creates a configuration view that provides expansion information controlling how the design hierarchy can be traversed.



In the above example, the library *designLib* contains two configuration definitions with the view names `mixedConfig` and `logicConfig`. You can use the hierarchy editor to open these views.

About the Hierarchy Editor



The hierarchy editor is a tool you use to create and edit a configuration view.

A configuration is a set of rules that defines which cellviews under a top-level cell are to be considered part of the design for netlisting, simulation, design partitioning for mixed signal simulation, and analysis.

The hierarchy editor lets you view the hierarchy of a design using a table or a tree format to manage multiple components and views.

You use the hierarchy editor to do the following:

- Change the global bindings
- Add cell-based bindings and cell-based view lists
- Add instance-based bindings and instance-based view lists
- Update your configuration to see the effect of your work
- Change instance bindings inside a block
- Save your rules

For more details about using the hierarchy editor, refer to the [Cadence Hierarchy Editor User Guide](#).

Traversing a Design Hierarchy

You can traverse a design hierarchy to view or edit your design using the hierarchy editor. When you descend into a schematic, the schematic window opens. When you descend into a symbol, the symbol window opens. When you descend into a text view, the text editor specified by the `$EDITOR` variable in your `.cshrc` file opens.

Note: If you attempt to descend into a cellview that is already being used, a warning message is displayed informing you of the current user of the cellview and the current host machine name. From here you can choose to either open the cellview as read-only or not to open the cellview.

When you descend from within a configuration design, you might descend into a view that is not defined in the configuration expansion definition. The new scope is then out of context, and the title bar indicates you are not in a configuration design.

This section discusses two ways to move down the design hierarchy.

- Use the *Edit – Hierarchy – Descend Edit / Read* command to descend into any view.

When you choose the View Name List option, you descend directly into the first available view in the list.

Note: If you double-click an instance the Descend form will be displayed.

- Use the *Edit – Hierarchy – Edit In Place* command to access the symbol editor from within the schematic editor environment. The schematic, while not editable, shows you the relationship of your lower-level symbol to the entire schematic.

Important

For information on how to quickly traverse up and down a design hierarchy using the Virtuoso Schematic Editor *Go* toolbar, see Navigating Cellviews and Hierarchies in the Virtuoso Design Environment User Guide.

Descending Using the Descend Command

To descend into a lower-level view of an instance,

1. From the view, choose *Edit – Hierarchy – Descend Edit / Read*.

If you do not have edit permission to descend, you are prompted to use read mode.

2. Select an instance in your design to descend into.

When you postselect the instance using the default settings, you descend directly into the lower-level view, usually the symbol view.

The *Descend* command behaves differently depending on data and options.

Data

- The Descend form appears if the instance you select has several views available for descending into. The form asks you to choose the view to descend into.
- If the instance is a symbol and the symbol represents a multisheet design, a form lets you choose to go to the index sheet or a specific sheet number.
- In the case of a preselected instance, the Descend form opens immediately.
- In case infix mode is enabled or you started the *Descend* command with a bindkey, the Descend form opens immediately.
- If the instance is iterated, the Descend form lets you choose the iteration number you want to descend into.



The Descend Edit/Read form above allows you to descend and open a chosen view in a *new tab*, the *current tab* (default), or in a *new window*.

You can also specify whether to open the selected view in *edit*, *read*, or *auto* mode.

Where:

edit opens the view in edit mode.

read opens the view in read mode.

auto opens the view dependent upon current edit permissions for the view being descended into.

Note: A thumbnail will also be displayed in the Descend form if there is one available for the current cellview. For more information on the generation and use of thumbnails see [Thumbnail View Functions](#) in the *Cadence User Interface SKILL Reference* in the first instance.

Options

- Press the **F3** key in the active *Descend* command to bring up the options form.

Target Cellview Specify the descend target cellview, either *query user* or *use viewNameList*.

Environment variable: descendTarget

If the descend target is *query user* (default), the Descend form opens, asking you which view you want to descend into.

If the descend target is *use viewNameList*, type view names in a specific order in the View Name List field in the *Command* tab of the Editor Options form.

The following three forms show data for different instances.

- ☐ Using a single instance representing a single-sheet schematic on the Descend Edit/Read form.
- ☐ Using an iterated instance representing a single-sheet schematic on the Descend Edit/Read form.
- ☐ Using a single instance representing a multisheet schematic on the Descend Edit/Read form.

If the Descend Edit/Read form opens, do the following:

1. Complete the form depending on your data and options.
2. Click *OK*.

Virtuoso Schematic Editor User Guide

Traversing the Design Hierarchy and Creating a Design Configuration View

If you select a graphic view to descend into, either the schematic window or the symbol window opens.

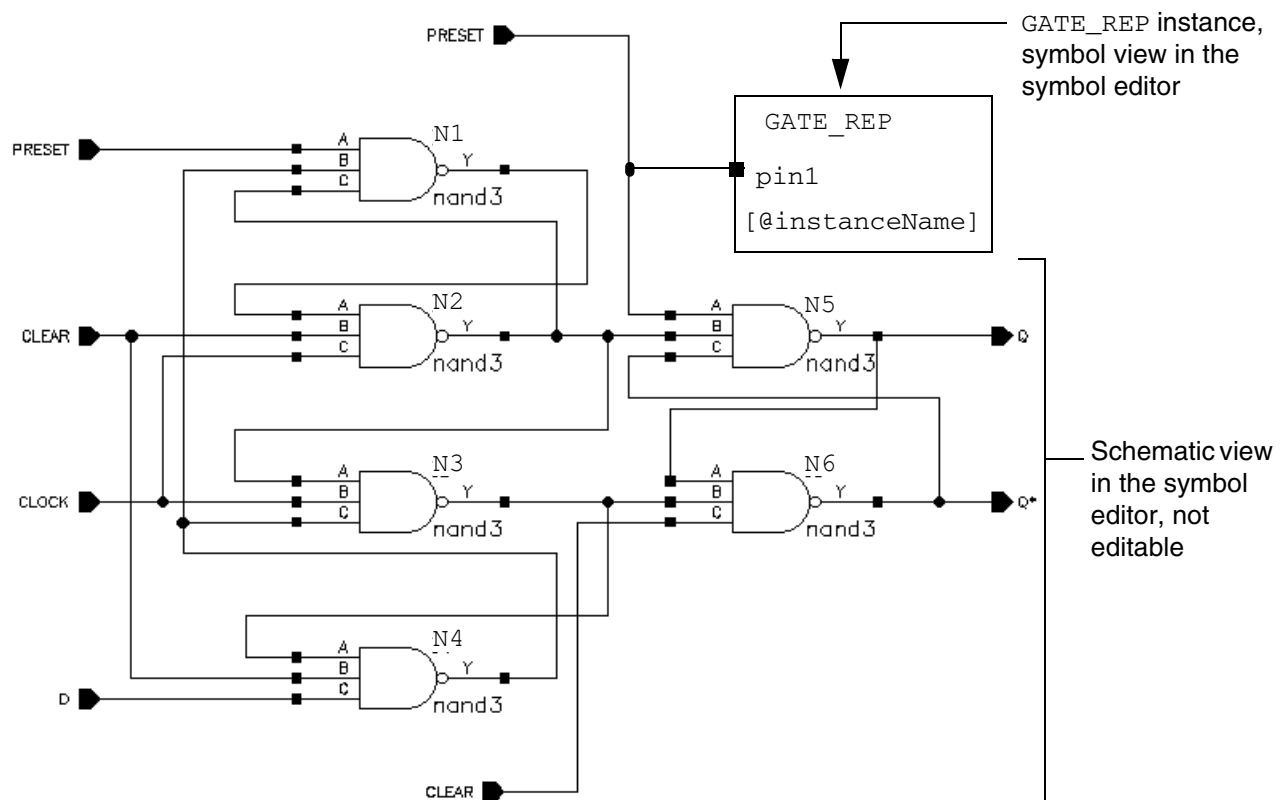
If you select a text view, a text editor window opens. Some text editors automatically check your text file for syntax errors. If errors are found, the Parser Error/Warnings form appears and asks if you want to view the error file and re-edit it.

Editing an Instance in Place

To edit an instance in place within the context of the higher-level schematic,

- From the view, choose *Edit – Hierarchy – Edit In Place*.

You descend into the symbol view of the instance to use the symbol editor while still viewing the graphics of the higher-level schematic. The schematic, while not editable, shows you the relationship of your symbol to the entire schematic.



Note: If you attempt to edit in place an instance of a cellview that is already being used, a warning message will be displayed informing you of the current user of the cellview and the current host machine name. From here you can choose to either open the cellview as read-only or not to open the cellview.

Descending Using the View Name List

By default, when you descend into another view, the system prompts you for the destination view. If you want to bypass this step to descend directly into a view in a certain order, you can use the View Name List option when you are out of context of a design hierarchy.

To change the default so that you automatically descend into the first available view in the view name list box,

1. From the view, choose *Edit – Hierarchy – Descend Edit / Read*.

The Descend Edit/Read form is displayed.

2. Turn on *use viewNameList*.

This options works with the View Name List option (see also Options).

3. Select an instance in the schematic.

You automatically descend into the first view listed in the *viewNameList* for that instance. To set up the list of views in a specific order, see the View Name List option.

Setting the View Precedence Order

You can arrange the views for descending for data not within a configuration scope.

To rearrange the views in the *View Name List* option to set up the precedence order for the view to be used as the descend target,

1. From the view, choose *Options – Editor*.

The Editor Options form appears.

2. In the *Command* tab, type the names in the *View Name List* field in the order in which you want to descend. For example:

```
schematic symbol vhdl functional abstract layout synthesis
```

The arrangement sets up the precedence order of views to use when you descend into another view.

3. Click *OK*.

The hierarchy configuration is more powerful and flexible than the View Name List method for specifying design comparisons. You edit a hierarchy configuration using the hierarchy editor. The *View Name List* option does not work in configuration mode.

Returning Up the Design Hierarchy

After descending, you can return by levels or all at once.

To return to the previous level,

- From the view, choose *Edit – Hierarchy – Return*.

To return to the top level,

- From the view, choose *Edit – Hierarchy – Return To Top*.

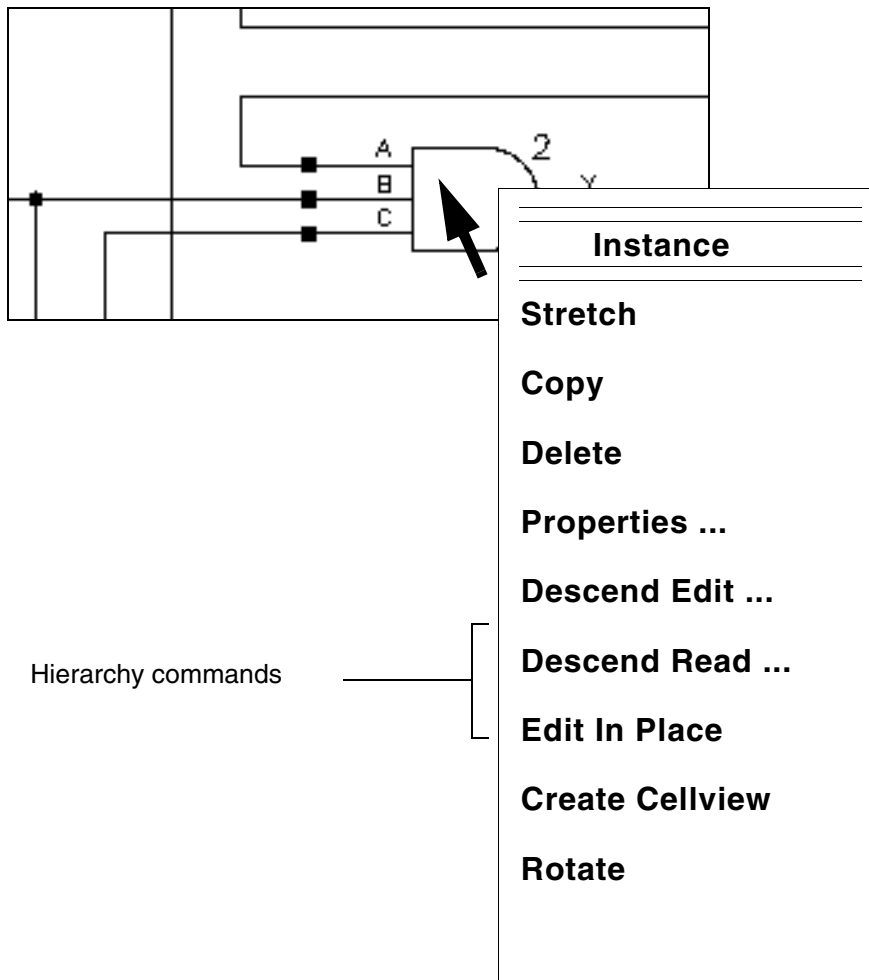
Using Objective-Sensitive Menus to Traverse the Design Hierarchy

To use object-sensitive menus (OSMs), make sure you turned on the OSM option. (For details, refer to *Virtuoso Design Environment User Guide*.) The default is *off*.

1. Place the cursor over an instance.
2. Press the middle mouse button.

The OSM pops up.

3. Select a *Descend* command.
4. With the cursor over an open area in the descended view, use the same process to return to the previous level.



Using Bindkeys to Traverse the Design Hierarchy

Before you use bindkeys, verify that the bindkey file is installed properly.

Note: For more information see Bindkeys in the *Virtuoso Design Environment User Guide*

To use bindkeys, select one of the following:

Command	Bindkey
Descend Edit	<Key>e
Descend Read	Shift<Key>-e
Return	Control-e

Displaying the Current Hierarchical Scope

When you display the hierarchical scope of your design, you can

- View all the instance names and corresponding cell names of the current window
- View a list of the instance names you have descended into and in which cell name and view name they are located

To display the current hierarchical scope,

1. From the view, choose *Edit – Hierarchy – Show Scope*.

A dialog box appears that lists the instance names and the corresponding cell names of the current window.

2. Click *Close* to close the dialog box.

Displaying Cellview Listings Using Print Tree

The Print Tree utility lists cellviews in a tree structure, allowing a design hierarchy to be captured by displaying the instances in the current cellview.

To list the hierarchy of instances in the current cellview:

1. Select *Edit – Hierarchy – Print Tree* (or use the `Shift+T` bindkey).

The Tree Input form is displayed.

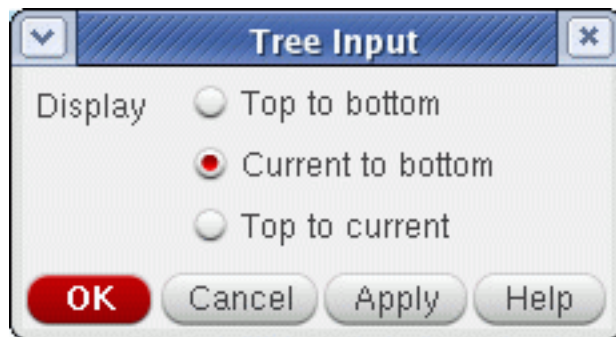


Figure 9-1 The Tree Input Form

See also [`schHiTree`](#) in the *Virtuoso Schematic Editor SKILL Functions Reference*.

2. Select the *Display* type to specify the instances that you want to display.

Here:

- ☐ *Top to bottom* lists the entire hierarchy of instances present in the cellview.
- ☐ *Current to bottom* lists the hierarchy of instances from the current level to the stop level.
- ☐ *Top to current* lists the hierarchy of instances from the top level to the current level.

3. Click *OK*.

The Tree Output window is displayed detailing the structure of the current design hierarchy.

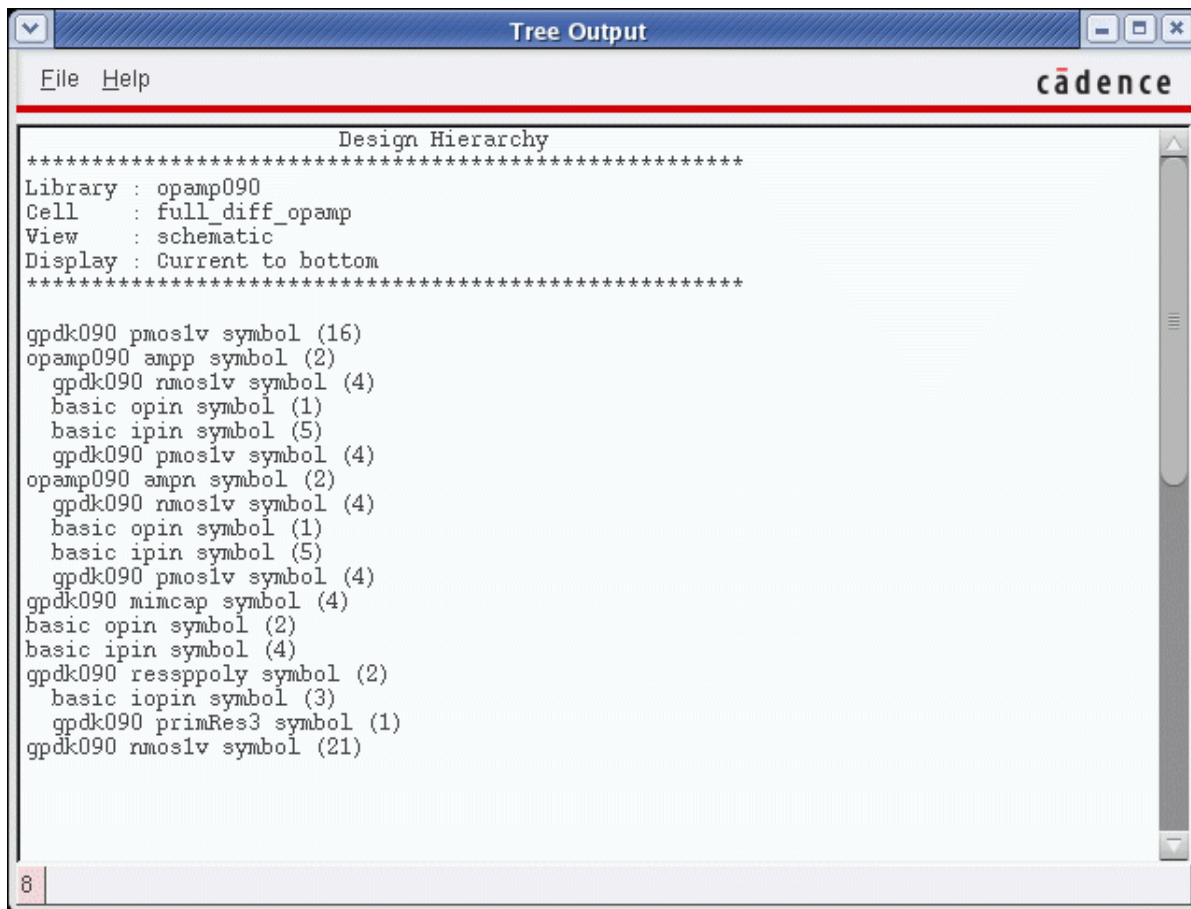


Figure 9-2 Example Tree Output Window

The cellview instances' information lists the hierarchy of instances in the current cellview, or the instances that are above or below the edit-in-place cell. The number in parentheses () represents how many times that particular instance appears in the design.

Note: If indents are displayed in the output list, they represent that instances of the indented cells are inside the instance listed above it.

4. Choose *File – Close* to close the window. If required, you can save (*File – Save*) the contents displayed in the window to a text file. You can also view (*File – Open*) in the Tree Output window a text file save earlier.

Creating a New Configuration View for the Design Hierarchy

To create a new configuration view of a design and open the hierarchy editor,

1. From the Command Interpreter Window (CIW), choose *File – New – Cellview*.
2. Change *View Name* to `config` and *Tool* to *Hierarchy-Editor* and click *OK*.

For details on creating a configuration view, refer to the [*Cadence Hierarchy Editor User Guide*](#).

Note: You can set up a config cellview using Virtuoso Schematic Editor or Virtuoso Text Editor. For details, refer to [Setting Up a Config Cellview using Virtuoso Schematic Editor or Virtuoso Text Editor](#).

Opening an Existing Design Configuration View

To open an existing design configuration view,

1. In the CIW, choose *File – Open*.

The Open File form appears.

2. Set *View Name* to *config* for the schematic design you want to open.

3. Click *OK*.

The Open Configuration or Top CellView form appears.

When you set the configuration to *yes*, you can open the hierarchy editor.

When you set the top cell view to *yes*, you can open the top schematic for this configuration.

4. Click *OK*.

Notice the title bar changes to indicate you opened the schematic view in the context of this configuration:

```
Virtuoso Schematic Editing: mixedVHDL designA    schematic    Config: mixedVHDL  
designA config
```

For details on opening a configuration view, refer to the [*Cadence Hierarchy Editor User Guide*](#).

About the Hierarchy Editor within the Virtuoso Schematic Editor

The hierarchy editor integration with the Virtuoso® Schematic Editor provides a convenient way to synchronize the configuration database with the Virtuoso Design Environment database. From the *Hierarchy-Editor* menu on the schematic window, you can open the hierarchy editor, synchronize the database, set instance binding, and view the current set of view binding information.

The following products use the hierarchy editor within the schematic editor: mixed-signal circuit design environment, analog circuit design environment, Verilog®-XL simulator, and VHDL integration.

For more information, see the [Cadence Hierarchy Editor User Guide](#).

Working with a Design Configuration within the Virtuoso Schematic Editor

This section covers the commands available from the *Hierarchy-Editor* menu within the schematic editor.

- [Editing a Design Configuration within the Virtuoso Schematic Editor](#) on page 648
- [Setting Instance Bindings](#) on page 649
- [Updating a Design Configuration](#) on page 651
- [Building a New Inherited View List](#) on page 652
- [Showing Views Found](#) on page 653

To open the hierarchy editor menu within the schematic editor,

- Choose *Launch – Plugins – Hierarchy Editor*.

The *Hierarchy-Editor* menu title appears in the schematic editor menu bar.

Editing a Design Configuration within the Virtuoso Schematic Editor

To edit a configuration view,

1. From the schematic window, choose *Launch – Plugins – Hierarchy Editor*.

If *Hierarchy-Editor* is not visible, do the following:

- a. In the CIW, type

```
cdsHierEditor &
```

- b. From the schematic window, choose *Launch – Plugins – Hierarchy Editor*.

The *Hierarchy-Editor* menu title appears in the schematic editor menu bar.

2. Choose *Hierarchy-Editor – Edit Configuration*.

The hierarchy editor appears with information pertaining to your configured schematic.

For more details about using the hierarchy editor, refer to the [Cadence Hierarchy Editor User Guide](#).

Setting Instance Bindings

The *Set Instance Binding* command lets you define the view to use and lets you choose the inherited view list. You can also set the view to use and the inherited view list from the hierarchy editor, but the *Set Instance Binding* command from the schematic editor gives you more flexibility on the scope of changes.

To define the view to use and to choose the inherited view list,

1. From the schematic window, choose *Hierarchy-Editor – Set Instance Binding*.

Note: You may need to initially select *Launch – Plugins – Hierarchy Editor*.

You are prompted to open the Hierarchy Editor form if it is not already open.

The Set Instance Binding form appears.

Note: For more details about using the hierarchy editor, refer to the [Cadence Hierarchy Editor User Guide](#).

2. Select one or more instances in the design that you want changed.

The current instance is highlighted and its name is displayed in the *Instance Name* field.

3. Change the *Apply To* cyclic field to display *current instance* (the main selection), *all selected instances* (in a group selection, including the main selection), or *all instances* (in the cellview).

If you choose *all selected* or *all instances*, another cyclic field appears on the form. Choose *same master* or *any master*.

4. Select *View To Use* to be bound to these instances.

Views common to all of the selected instances are available for selection. If you want to revert to the default bindings for your configuration in the hierarchy editor, set the *View To Use* field to *<Default>*.

5. To keep the inherited view list as is and not build a new inherited view list (default), turn on the *As Is* field.

You can change the inherited view list. To build a new inherited view list, see [Building a New Inherited View List](#) below.

6. Click *Apply* to make your changes.

Your changes are reflected in the *View To Use* field in the hierarchy editor for the instances you selected.

7. To propagate your changes to subordinate instances and save your changes, choose *Hierarchy-Editor – Update* in the schematic window.

All instances that are affected by your *View To Use* selection are changed in the hierarchy editor and saved.

Updating a Design Configuration

After you edit the configuration, such as defining a new instance binding, you need to update the information in the hierarchy editor. To update a configuration from the schematic editor,

- From the schematic window, choose *Hierarchy-Editor – Update*.

If you are using the schematic editor to edit cellviews, you must save them to have the edits made reflected in your configuration.

The hierarchy editor is updated with the following information:

- The cellviews you selected are saved
- The hierarchy editor display is updated
- The schematic editor is updated with the new configuration

Building a New Inherited View List

After choosing *Hierarchy-Editor – Set Instance Binding* and completing the steps in the previous section,

1. Turn the *As Is* button off in the *Inherited View List* field.
2. Type in the inherited view list or click *Build* to construct the view list.

The Build Inherited View List form appears.

The *Available Constants/Views* list box shows the combined views from the selected cells.

Note: For more details about using the hierarchy editor, refer to the *Cadence Hierarchy Editor User Guide*.

3. Select the appropriate views and constants, and click the right arrow to add them to the *Inherited View List* list box.
4. Use the *Up* and *Down* buttons to reposition entries for using the appropriate views in a specific order.
5. (Optional) To specify a view name that is not listed, type the name in the field below the *Inherited View List* list box and click *Add*.
6. Click *OK*.
7. In the *Set Instance Binding* form, click *Apply*.

The new view list is added to the appropriate instances in the hierarchy editor.

8. From the schematic window, choose *Hierarchy-Editor – Update*.

Your changes are propagated through subordinate instances in the hierarchy editor and are saved in both the configured schematic and in the hierarchy editor configuration view.

Showing Views Found

In Building a New Inherited View List, you either set the instance binding or built a new inherited view list. This command graphically shows the current view being used for each instance.

To show views used by the instance,

1. From the schematic window, choose *Hierarchy-Editor – Show Views Found*.

The Show Views Found form appears.

The *View Name(s)* option shows all the views bound to all the instances in the current schematic cellview. The *Highlight Layers* option shows the colors used to highlight the instances with the corresponding views. The command graphically shows the current view being used for each instance.

2. (Optional) Change the highlight layer for each view name.
3. Click *Apply*.

The schematic display is updated with the appropriate highlight color showing the views used by all the instances.

Virtuoso Schematic Editor User Guide

Traversing the Design Hierarchy and Creating a Design Configuration View

Checking Designs

This chapter describes the procedures you use for checking your design.

- [Features of the Check Commands](#)
- [Setting the Check Options](#)
- [Creating Custom Checks](#)
- [Checking a Design](#)
- [Bypassing Checks](#)
- [Resolving Check Errors](#)

Features of the Check Commands

Before you can pass the design data to any other tools, you must check your design for connectivity, errors, and rule violations.



By default, the connectivity of a design is updated automatically as you create and edit the design. For more details, see [Updating Connectivity](#).

The Cadence netlisters cannot generate the netlist until you check your design and, if necessary, correct any errors found during the check process.

You do not need to correct any warnings found during the check process, but you must review them before you attempt to generate a netlist.

The check commands are as follows:

- *File – Check and Save*
- *Check – Current Cellview*
- *Check – Hierarchy*
- *Check – Label Attachment*
- *Check – CrossView*

You can check schematics using the following commands.

- The *File – Check and Save* command runs most of the check commands simultaneously, and

- ☐ Checks either a schematic or a symbol cellview
- ☐ Extracts [connectivity information](#) from the graphics in the schematic

Note: When using the schematic editor as part of a Layout XL session, the Layout XL binder engine is reinitialized to take account of any changes in connectivity. This can create new bindings between schematic and layout components. For details see [Device Correspondence](#) in the *Virtuoso Layout Suite XL User Guide*.

- ☐ Runs specified options using the [schematic rules checker](#) (SRC), such as logical and physical checks
- ☐ Runs the [cross-view checker](#) (VIC) on views you can specify
- ☐ Saves the design

- The *Check – Current Cellview* command checks all the features that the *File – Check and Save* command does, but it does not save the design.
- The *Check – Hierarchy* command is identical to the *Check – Current Cellview* command, but it checks the entire hierarchy.
- The *Check – Label Attachment* command
 - ❑ Shows the association between the labels and objects by drawing highlighted lines
 - ❑ Checks for unattached labels
- The *Check – Cross View* command checks the interface of specified views.
 - ❑ Checks interface consistency between a list of specified views
 - ❑ Reports signals exported in one view but not in the other
 - ❑ Reports signals whose pins have different directions in the multiple views
 - ❑ Reports terminal order mismatch between views
 - ❑ Checks if inherited terminals match between views
 - ❑ Checks if terminal net expressions match
 - ❑ Checks if supply sensitivity of terminals matches

Highlighting Errors and Warnings in the CIW

You can use the following environment variables to control whether or not the CIW is raised to the front of other windows to display any errors or warnings that have been issued.

```
ui_raiseCIWonError boolean t/nil
ui_raiseCIWonWarning boolean t/nil
```

Setting the Check Options

Before you run the check commands, you can set options and rules using the following commands:

- *Options – Check*

Sets options to be run when you choose *File – Check and Save*, *Check – Current Cellview*, or *Check – Hierarchy*.

- *Options – Check Rules Setup*

Sets the severity levels for checking logical, physical, name, and various other rules.

Note: *Check* and *Check Rules Setup* are accessible from both the *Options* menu and the *Check* menu.

Setting User Preferences

To set user preferences,

1. From the schematic window, choose *Options – Check*.

The Schematic Check Options form appears.

The screenshot shows the 'Schematic Check Options' dialog box with the following settings:

- Connectivity**
 - ☒ Extract connectivity during schematic checking
 - ☒ Beep When 0 Errors/Warnings
 - ☐ Set Signal Type from Net Name and Type Registration
 - Check and Save Action on Error: ☐ Save ☐ No Save ☒ Ask Me
 - From a Multisheet Index, Check: ☐ all ☒ only if needed
 - Wire Name Glue Distance:
- Schematic Rules Checks**
 - ☒ Run Schematic Rules Checks (with 'Rules Setup ...' button)
- Cross View Checks**
 - Cross View Check Severity: ☐ ignored ☒ warning ☐ error
 - Views To Check:
 - ☐ Match Pin Names Exactly
 - ☒ Check Pin Order
 - ☒ Only check views with explicit pin order
 - ☒ Match Inherited Terminals
 - ☒ Ignore Terminals with Net Expression and no Terminal in the Other View
 - Ignore Terminals for Specified Views
 - View Names:
 - ☒ All Terminals ☐ Terminal Names
 - ☒ Match Terminal Net Expressions
 - ☒ Match Terminal Supply Sensitivity
 - ☒ Match Terminal Signal Type

Buttons at the bottom: OK, Cancel, Defaults, Apply, Help.

Virtuoso Schematic Editor User Guide

Checking Designs

2. Specify the options you want to apply to your check.
3. Click *OK*.

Virtuoso Schematic Editor User Guide

Checking Designs

- ☐ Logical checks the high-level connectivity data
- ☐ Physical checks for unconnected wires, offset labels, and overlapping instances
- ☐ Name checks for syntax and collisions for the Cadence VHDL verification tool or the Verilog[®]-XL simulator
- ☐ Inherited Connection checks existing or missing inherited connections
- ☐ Constraints checks constraints (for example, mismatched parameters)
- ☐ Signal Type checks input, output and I/O pins
- ☐ RF Implementation checks transmission lines
- ☐ Coloring checks the CDF colors
- ☐ (optional) Custom Checks lets you create your own custom check tabs and associated rules checks

4. Click *OK*.

Used to create and register a new schematic rule checker.

Virtuoso Schematic Editor User Guide

Checking Designs

Logical Checks

For more information see the [Schematic Rules Checks Setup](#).

The screenshot shows the 'Schematic Rules Checks Setup' dialog box with the 'Logical' tab selected. The dialog has a title bar with a close button. Below the title bar are tabs: 'Logical', 'Physical', 'Name', 'Inherited Conn.', 'AMS', 'Constraints', 'Signal Type', and 'Coloring'. The 'Logical' tab is active and contains a list of checks with radio buttons for 'ignored', 'warning', and 'error'. The 'warning' option is selected for most checks, except for 'Offsheet Connectors', 'Plain Globals', and 'Pin Direction Collision', which have 'ignored' selected. At the bottom, there is a 'Packaged Checks' section with a dropdown menu set to 'None'. To the right of this are five buttons: 'OK', 'Cancel', 'Defaults', 'Apply', and 'Help'.

Check	ignored	warning	error
Floating Nets	☐	☒	☐
Floating Input Pins	☐	☒	☐
Floating Output Pins	☐	☒	☐
Floating I/O Pins	☐	☒	☐
Floating Switch Pins	☐	☒	☐
Shorted Output Pins	☐	☒	☐
Offsheet Connectors	☒	☐	☐
Plain Globals	☒	☐	☐
Pin Direction Collision	☒	☐	☐

Packaged Checks: None

Buttons: OK, Cancel, Defaults, Apply, Help

Virtuoso Schematic Editor User Guide

Checking Designs

Physical Checks

For more information see the [Schematic Rules Checks Setup](#).

The screenshot shows the 'Schematic Rules Checks Setup' dialog box with the 'Physical' tab selected. The dialog has a title bar with a close button. Below the title bar are tabs: 'Logical', 'Physical' (selected), 'Name', 'Inherited Conn.', 'AMS', 'Constraints', 'Signal Type', and 'Coloring'. The main area contains several settings:

Check Name	Radio Buttons
Unconnected Wires	☒ ignored ☐ warning ☐ error
Solder On CrossOver	☐ ignored ☒ warning ☐ error
Overlapping Instances	☐ ignored ☒ warning ☐ error
Percent Overlap Allowed	18
Maximum Label Offset	☒ ignored ☐ warning ☐ error
Maximum Offset From Wire	0.045

At the bottom, there is a 'Packaged Checks' section with a dropdown menu set to 'None'. Below this are five buttons: 'OK' (highlighted in red), 'Cancel', 'Defaults', 'Apply', and 'Help'.

Name Checks

For more information see [Schematic Rules Checks Setup](#).

The screenshot shows the 'Schematic Rules Checks Setup' dialog box with the 'Name' tab selected. The dialog has several tabs: Logical, Physical, Name, Inherited Conn., AMS, Constraints, Signal Type, and Coloring. The 'Name' tab contains the following settings:

Check Name	Severity Level
Connection By Name	☒ ignored ☐ warning ☐ error
Connection By Name Ignores	
Unnamed Connection	☒ ignored ☐ warning ☐ error
Instance/Net Name Collision	☐ ignored ☒ warning ☐ error
Pin/Net Name Collision	☐ ignored ☒ warning ☐ error
Verilog HDL Syntax	☒ ignored ☐ warning ☐ error
System Verilog HDL Syntax	☒ ignored ☐ warning ☐ error
VHDL Syntax	☒ ignored ☐ warning ☐ error
Instance Name Syntax	☒ ignored ☐ warning ☐ error
Instance Name Expression	
Pin Name Syntax	☒ ignored ☐ warning ☐ error
Pin Name Expression	
Net Name Syntax	☒ ignored ☐ warning ☐ error
Net Name Expression	
Sparse Index Bus	☒ ignored ☐ warning ☐ error
Sparse Index Bus Only Terms	☐ yes ☒ no

At the bottom, there is a 'Packaged Checks' dropdown menu set to 'None', and buttons for 'OK', 'Cancel', 'Defaults', 'Apply', and 'Help'.

The following specifics apply to this form:

- **Connection By Name** field. To run this check on a net marked as being connected by name, set the severity level to *error*. If your design has errors after checking it, you can highlight all wire segments for that net by choosing *Edit – Find* and searching on that net name. Your design will show you which net is disjointed and where you need to add wire segments.
- **Connection By Name Ignores** field. To run this check on a net marked as being connected by name, set the severity level to *error*. If your design has errors after checking it, you can highlight all wire segments for that net by choosing *Edit – Find* and searching on that net name. Your design will show you which net is disjointed and where you need to add wire segments. to accept a list of net names that should be ignored. The names may be provided as a space separated list of POSIX Extended regular expressions in the Schematic Rules Checks form.

Virtuoso Schematic Editor User Guide

Checking Designs

- Text fields *Instance Name Expression*, *Pin Name Expression*, and *Net Name Expression*. To use these fields, set the corresponding syntax severity level to either *warning* or *error*. In the text fields, use the regular expression syntax as documented in the UNIX `man` pages for the `ed 1` command. If a given instance, pin, or net name matches the specified regular expression, an appropriate message is reported:

```
pin name "name" does not match expression "regular expression"
```

You can precede the expression in these text fields with an exclamation point (`!`) to report names that do not match a specified regular expression. The message takes the form

```
net name "name" does not match expression "regular expression"
```

If these text fields consist of more than one signal or bundle, the name check is applied only to the base name portion of the name.

Use of Regular Expressions for Schematic Rules Checks

Only one regular expression is allowed when setting the *Instance Name Expression*, *Pin Name Expression* or *Net Name Expression* fields.

It is also recommended that you describe the allowed syntax, and negate it with an `!` at the front so that you can specify any exception as a violation.

For example, the following regular expression allows only names that start with an alpha character, and are followed by any occurrence of an alphanumeric character or an underscore, and end with an alphanumeric character, an underscore, or an exclamation mark:

```
!^[A-Za-z][A-Za-z0-9_]*[A-Za-z0-9_!]+$
```

Note: Further information on regular expression syntax can be found in the public domain.

Virtuoso Schematic Editor User Guide

Checking Designs

Inherited Connections Checks

For more information see the [Schematic Rules Checks Setup](#).

The screenshot shows the 'Schematic Rules Checks Setup' dialog box with the 'Inherited Conn.' tab selected. The dialog has a title bar with a close button. Below the title bar are tabs: Logical, Physical, Name, Inherited Conn. (selected), AMS, Constraints, Signal Type, and Coloring. The main area contains five check items, each with three radio buttons for 'ignored', 'warning', and 'error'. The 'Packaged Checks' section at the bottom has a dropdown menu set to 'None'. At the bottom right are buttons for OK, Cancel, Defaults, Apply, and Help.

Check Item	ignored	warning	error
Inherited Pin/Net Connections	☒	☐	☐
Missing Override Nets	☐	☒	☐
Inherited Pin/Net Name Collision	☒	☐	☐
Implicit Inherited Terminals	☒	☐	☐
Inherited Instance Pin/Pin Connections	☒	☐	☐

Packaged Checks: None

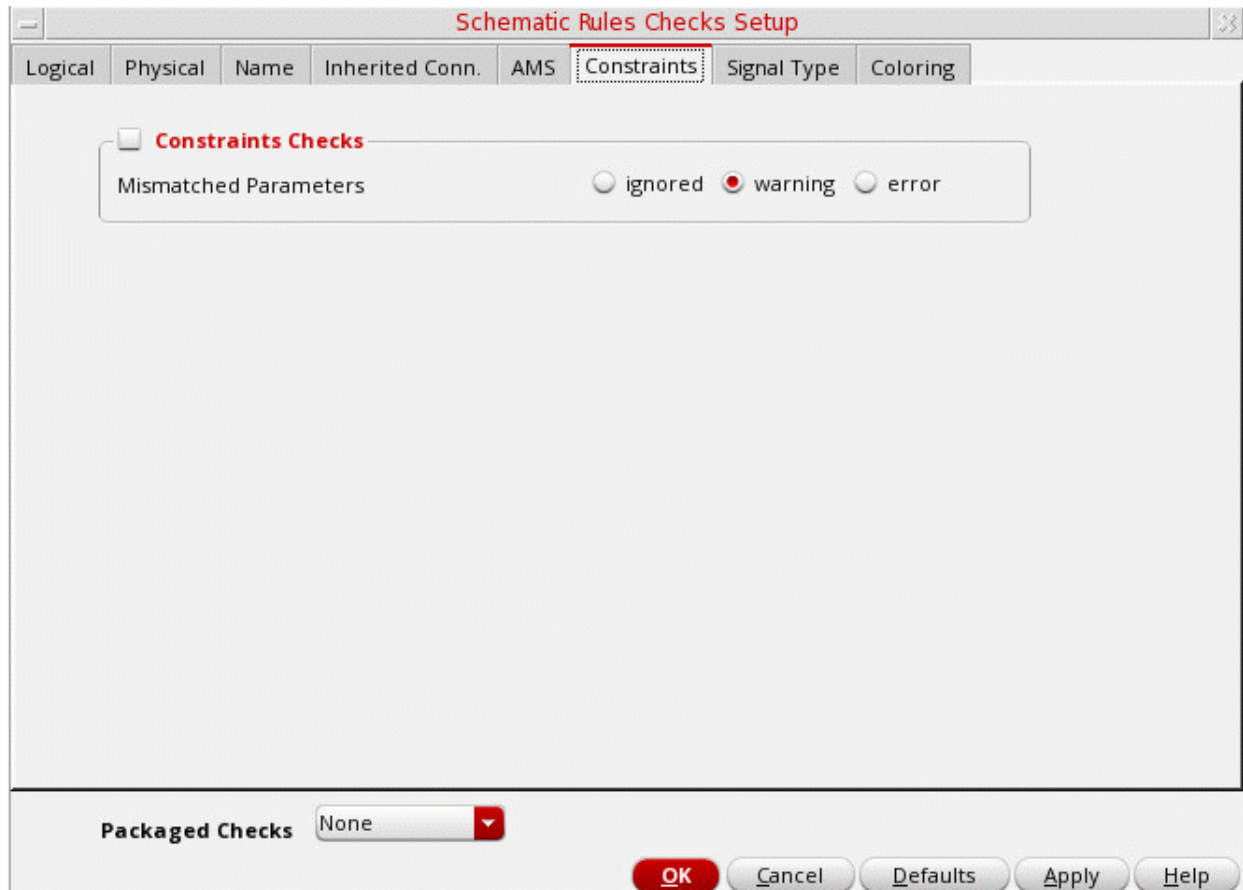
Buttons: OK, Cancel, Defaults, Apply, Help

Virtuoso Schematic Editor User Guide

Checking Designs

Constraints Checks

Note: Constraints functionality is only available in Virtuoso Schematic Editor XL. For more information on Constraints, see the *Virtuoso Unified Custom Constraints User Guide*.



Mismatched Parameters

The *Constraints* tab on the *Schematic Rules Checks Setup* can be set to check for *Mismatched Parameters*.

If you set *Constraints Checks* to *yes*, the constraints rules checker will test devices that are members of the *matchedParameters* constraint, confirming, or not, that the parameter has the same value on the devices.

You can choose to check for a mismatched parameter *warning* or *error*, or to have this check *ignored*.

Virtuoso Schematic Editor User Guide

Checking Designs

- If the check passes, the `matchedParameters` constraint is marked as *enforced* and appears as *enforced* in the *Constraint Manager* after a check and save is performed (see [Checking and Saving a Design](#)).
- If the check fails, the `matchedParameters` constraint is marked as *impossible* and appears as *impossible* in the *Constraint Manager*. Markers will also be placed on the canvas devices that are referenced by the constraint.

Note: As the *Virtuoso Schematic Editor* allows parameters to be edited, this *Mismatched Parameters* check acts as a check as to whether these edits may invalidate a `matchedParameters` constraint. This will therefore help guard against mismatched parameters impacting simulations or layouts.

The *Virtuoso Schematic Editor* checker (see [Checking and Saving a Design](#)), being cellview based, can be run on either the current cellview or through the design hierarchy. The *Mismatched Parameters* check looks at all members (that is, all the various cellviews), but only reports violations, by adding markers, on the check cellview. Any cellview that exhibits failure markers will cross-highlight to the appropriate `matchedParameters` constraint in the *Constraint Manager* when selected, allowing you to identify other failed devices across cellviews.

Therefore, on each invocation of the schematic checker, all constraint views will be examined for matched parameters, but markers will only be added to the edit cellview.

Note: The hierarchical checker (see [Checking a Design Hierarchy](#)) can also be used to add markers to multiple cellviews from a single command.

Signal Types Checks

For more information see the [Schematic Rules Checks Setup](#).

Pin Signal Type \ Net Signal Type	Signal	Power	Ground	Reset	Clock	Analog	Scan	tieHi	tieLo	tieOff
Signal	✓	✗	✗	✓	✓	✓	✓	✗	✗	✗
Power	⚠	✓	✗	⚠	⚠	⚠	⚠	✓	✗	⚠
Ground	⚠	✗	✓	⚠	⚠	⚠	⚠	✗	✓	⚠
Reset	✓	✗	✗	✓	⚠	⚠	⚠	✗	✗	✗
Clock	✓	✗	✗	⚠	✓	⚠	⚠	✗	✗	✗
Analog	✓	✗	✗	⚠	⚠	✓	⚠	✗	✗	✗
Scan	✓	✗	✗	⚠	⚠	⚠	✓	✗	✗	✗
tieHi	⚠	✓	✗	⚠	⚠	⚠	⚠	✓	✗	⚠
tieLo	⚠	✗	✓	⚠	⚠	⚠	⚠	✗	✓	⚠
tieOff	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	✓

Packaged Checks: None

OK Cancel Defaults Apply Help

Note: The Signal Type tab has three sub-tabs (*Input pins*, *Output pins*, and *I/O pins*) that contain the same *Net Signal Type* and *Pin Signal Type* options. These tabs are grayed out unless the *Signal Type Checks* checkbox option is selected.

Optical Checks (Virtuoso Photonics Option)

Note: Virtuoso Photonics Solution - You need the *Virtuoso_Photonics_Option* license to use the functionality discussed in this section. For information about obtaining the required license, contact your local Cadence representative.

The optical checks are controlled by the options on the *Optical* tab of the *Check – Rules Setup – Schematic Rules Checks Setup* form. The severity can be updated as carried out for the other signal types.

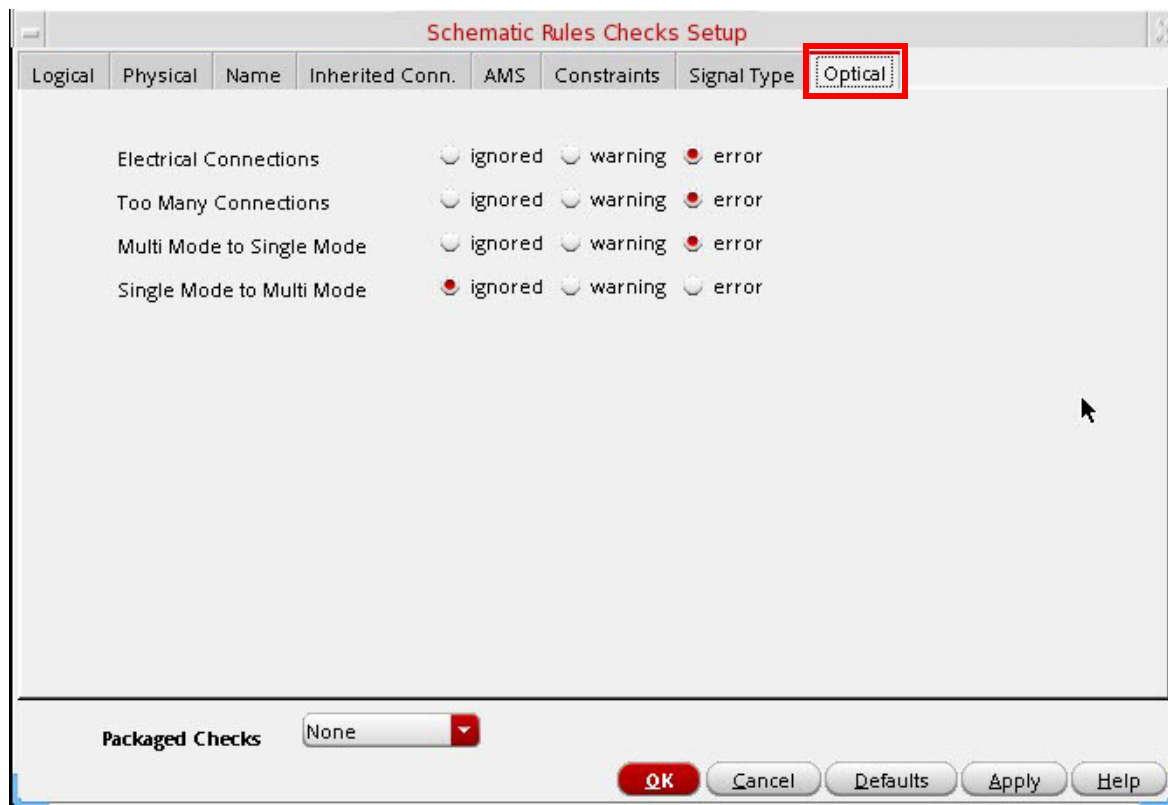
Virtuoso Schematic Editor User Guide

Checking Designs

Important

A connection between an optical net and electrical net results in an error.

Optical nets must have only two connections. Multi mode nets or outputs should not connect to single mode inputs or nets. Conversely, single mode nets or outputs can connect to multi mode inputs or nets.

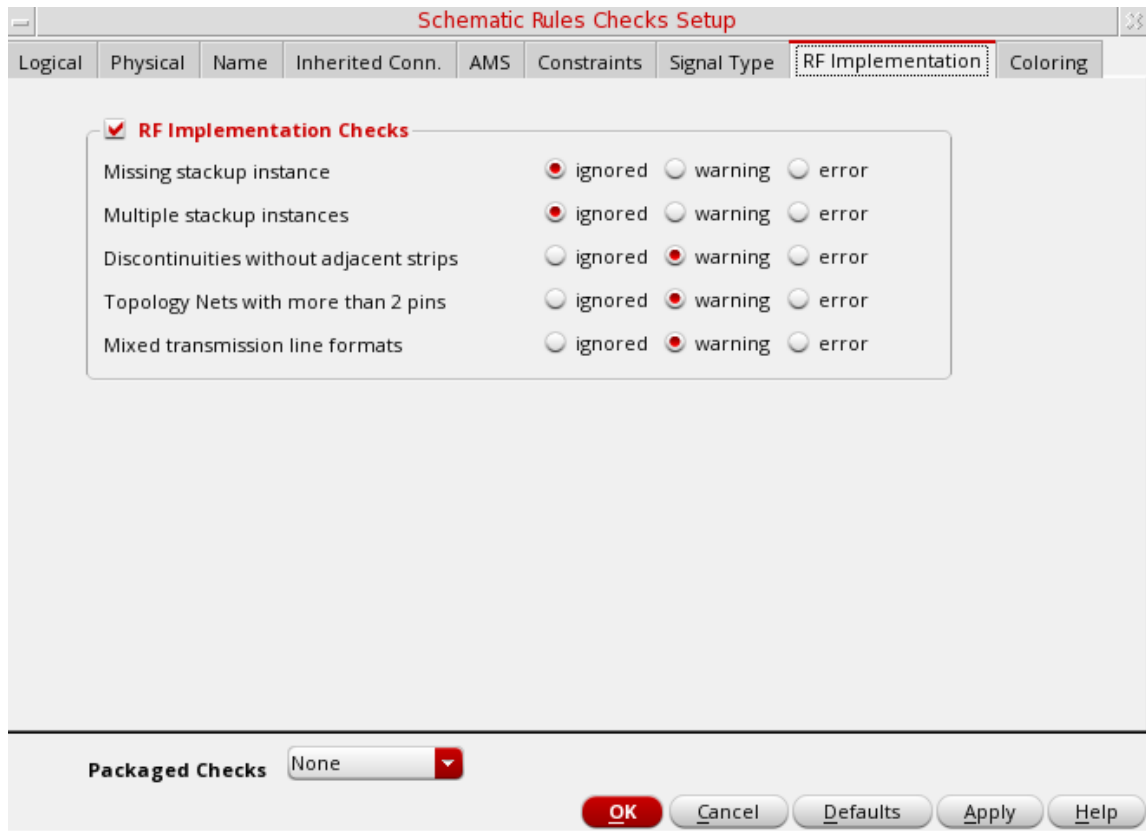


Virtuoso Schematic Editor User Guide

Checking Designs

RF Implementation (Virtuoso RF Solution Option)

For more information see the [Schematic Rules Checks Setup](#) and [RF Transmission Lines](#).

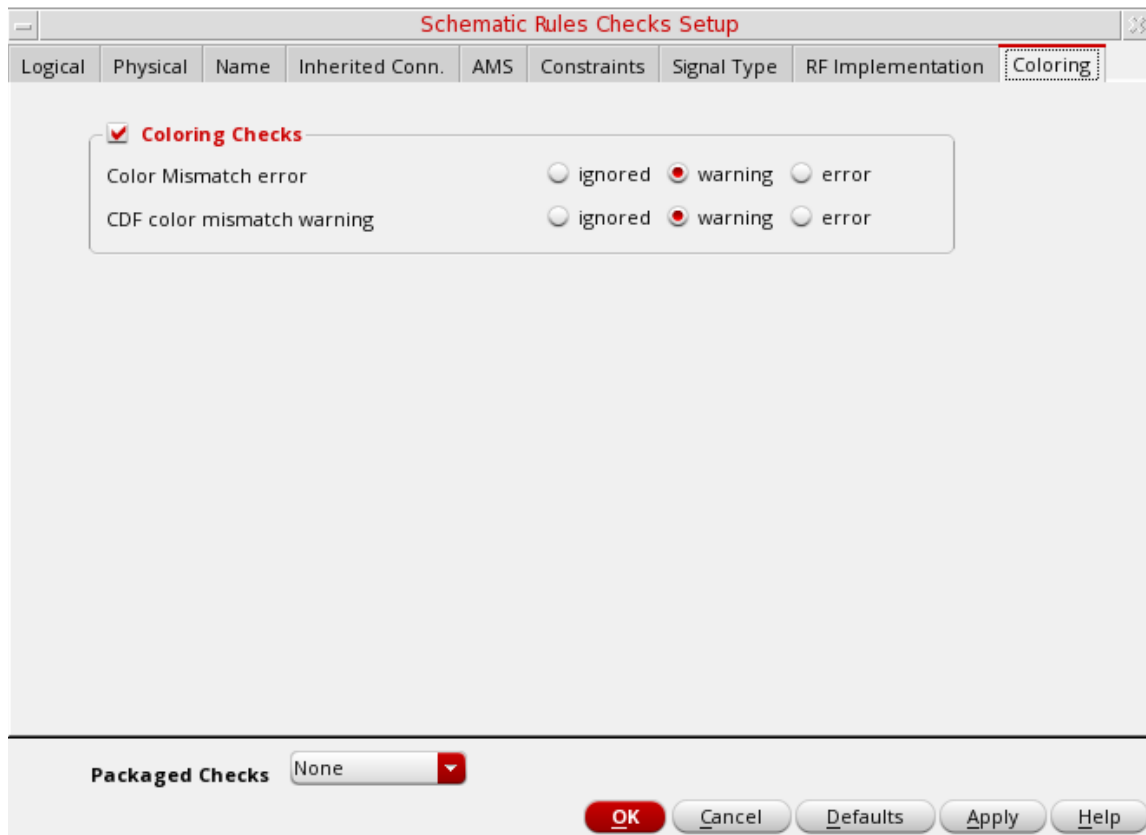


Virtuoso Schematic Editor User Guide

Checking Designs

Coloring

For more information see the [Schematic Rules Checks Setup](#).



Creating Custom Checks

You can create your own custom check tabs (see [Example of How to Create a Custom Check Tab](#)) and associated rules checks, for use within the Schematic Rules Checks Setup form, using the following SKILL functions:

- [`schRegisterCheckGroup`](#)

Use to create and register a new checker group. This will create a custom tab in the Schematic Rules Checks Setup form (see also [`schGetCheckGroups`](#)).

- [`schRegisterCheckRule`](#)

Use to create and register a new schematic rule checker option.

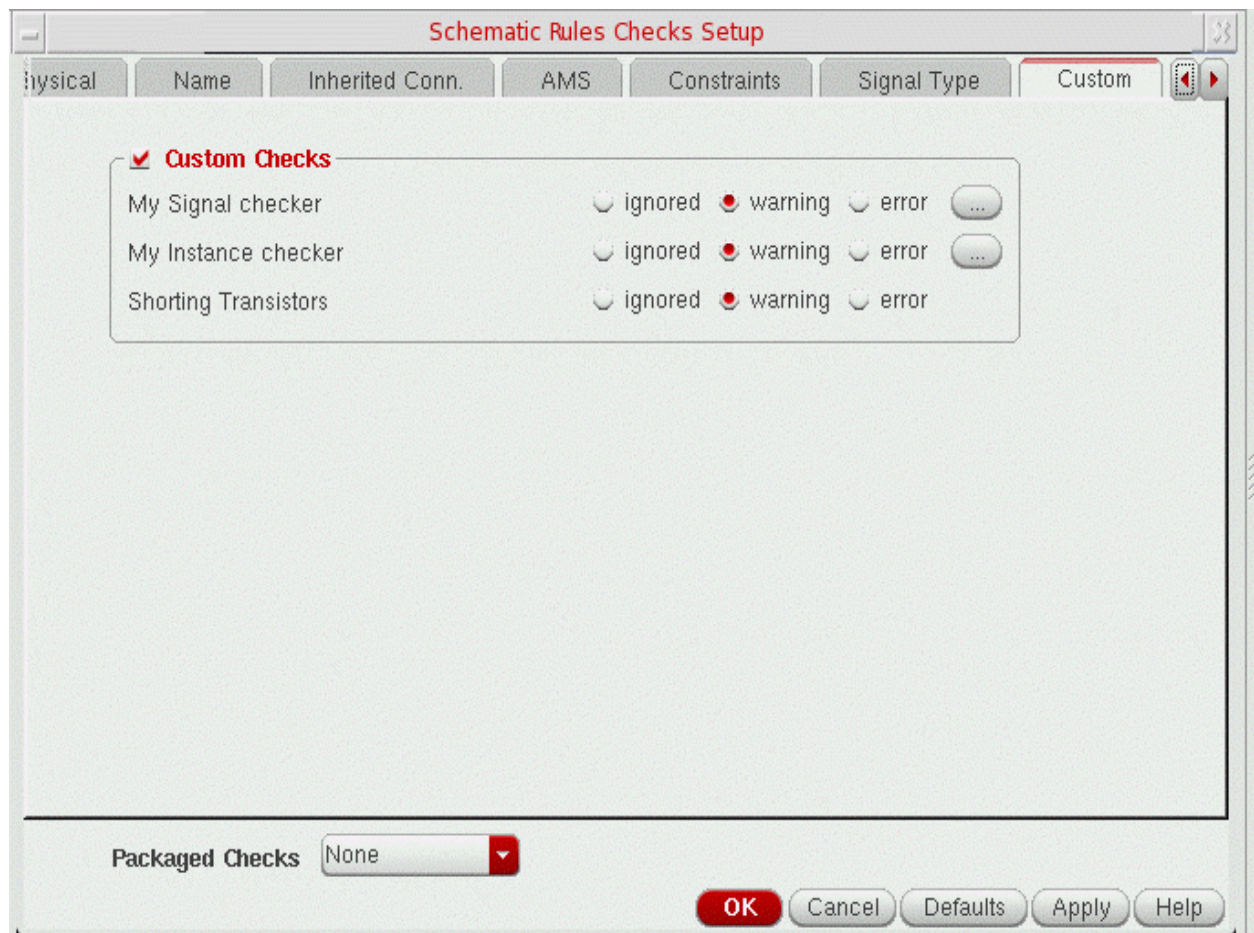


Figure 10-1 Customized “My Checks” Tab Added to Schematic Rules Checks Setup Form

Virtuoso Schematic Editor User Guide

Checking Designs

Example of How to Create a Custom Check Tab

The following is an example of how to create a customized tab in the Schematic Rules Checks Setup form:

```
procedure(CustomCheckCB(cellView ruleObject)
  let((msg "The gate of instance \"%s\" is connected to a high voltage net.))
  foreach(instance cellView->instances
    ;; Perform checks on instance and report failures...
    schReportCheckFailure(
      ?object      instance
      ?checkRule   ruleObject
      ?message     sprintf(nil msg instance->name)
    )
  )
)

procedure(ConfigCheckCB(groupName checkName)
  hiDisplayAppDBox(
    ?name          'ConfigCheckDialogBox
    ?dboxBanner    "Configure Custom Check"
    ?dboxText      sprintf(nil "Configure the %L check from the %L tab..." checkName
                             groupName)
  )
)

schRegisterCheckGroup(
  ?name 'CustomChecks
  ?title "Custom"
)

schRegisterCheckRule(
  ?title          "No HV on Gate of Thin Oxide MOS Tx"
  ?name           'CustomNoThinOxideMOSTxGateHV
  ?groupName      'CustomChecks
  ?checkCB        'CustomCheckCB
  ?configCB       'ConfigCheckCB
  ?description    "The gate should not have a direct or indirect path to a high voltage
net"
)
```

Checking a Design

There are two modes in which the connectivity of a design is updated: incremental and batch.

Incremental connectivity mode is the default mode. The connectivity in a schematic design is updated automatically as you create and edit a design. Batch connectivity mode is also available.

Regardless of the connectivity mode used, before you can pass the design data to any other tools, you must check your design for connectivity, errors, and rule violations.

You have several options for checking a design:

- [Checking and Saving a Design](#)
- [Checking the Current Cellview](#)
- [Checking a Design Hierarchy](#)
- [Checking a Design Hierarchy in the Configuration Context](#)
- [Checking Multisheet Schematics](#)
- [Checking a Label Attachment](#)
- [Cross-View Checking](#)
- [Checking Read-Only Cellviews](#)

Related Topics

[Updating Connectivity](#)

Checking and Saving a Design

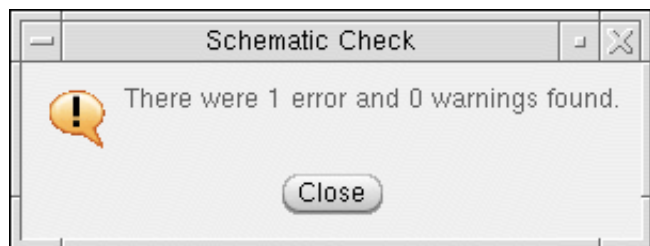
The *File – Check and Save* command does the following:

- Checks either a schematic or a symbol cellview
- Extracts connectivity information from the graphics in the schematic. Use the following SKILL functions to specify the Equivalent Connectivity Time information in a cellview.
 - ☐ dbCellViewHasEquivalentConnectivityTime
 - ☐ dbGetCellViewEquivalentConnectivityTime
 - ☐ dbSetCellViewEquivalentConnectivityTime
 - ☐ dbUnsetCellViewEquivalentConnectivityTime
- Runs specified options using the schematic rules checker (SRC)
- Runs the cross-view checker (VIC)
- Saves the design

To check and save either a schematic or symbol,

1. From the view, choose *File – Check and Save*.

After the check is completed, the Command Interpreter Window (CIW) shows the results of the check, and a dialog box appears.



Flashing highlighted markers in your schematic indicate the objects that have errors or warnings.

2. On the dialog box, click *Close*.

The Cadence netlisters cannot generate the netlist until you check the design and, if necessary, correct any errors found during the check process.

You do not need to correct any warnings found during the check process, but you should review them before you attempt to generate a netlist.

If no errors exist, your design is automatically saved.

If errors exist, you are prompted to save the cellview. You can change this default behavior by altering the value of the *Check and Save Action on Error* option on the Schematic Check Options.

3. Read the check results and reply to the form.

To find the errors, choose *Check – Find Marker*.

Note: If the input and output terminals of a device are connected, the *Check and Save* option returns a warning message in the CIW. However, if the input and output terminals of a device are connected to the `noConn` device, the *Check and Save* option does not give any warning message.

Using the Progress Bar to Measure Processing State

As an aid to establishing the current processing state for a number of specific tasks, a *Progress Bar* is displayed in the bottom right corner of the VSE status bar.

The *Progress Bar* will be displayed when processing the following actions:

- *File – Check and Save*

Note: The *Progress Bar* will always display when a *Check and Save* is performed, irrespective of the task duration.

- *Check – Current Cellview*

- *Check – Hierarchy actions*

Note: The *Progress Bar* for the action of checking a hierarchy will display progress in terms of the number of cellviews that have undergone extraction and the number that still have to be extracted (*Processing cellview 1 of n*).



Figure 10-2 The Progress Bar in Use for a Check and Save Action

Checking the Current Cellview

The *Check – Current Cellview* command does the following:

- Checks either a schematic or a symbol cellview
- Extracts connectivity information from the graphics in a schematic
- Runs specified options using the schematic rules checker (SRC)
- Runs the cross-view checker (VIC) on views you can specify

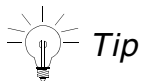
Before checking a cellview, review the check options.

To check only the current cellview,

1. From the schematic window, choose *Check – Current Cellview*.

The software performs the check. After the check is completed, the CIW displays the results of the check, and a dialog box appears.

2. Read the check results and reply to the form.
3. Choose *Check – Find Marker* to view and resolve each error, if applicable.



See also Using the Progress Bar to Measure Processing State.

Note: You can register a function to be called before or after a schematic is checked using the SKILL functions [schRegPreCheckTrigger](#) or [schRegPostCheckTrigger](#).

Checking a Design Hierarchy

Note: If the design is not opened in the context of a configuration, `viewNameList` will be used for hierarchy traversal, and `pc.db` would be used for text views.

See also [Using the Progress Bar to Measure Processing State](#).

The *Check – Hierarchy* command does the following:

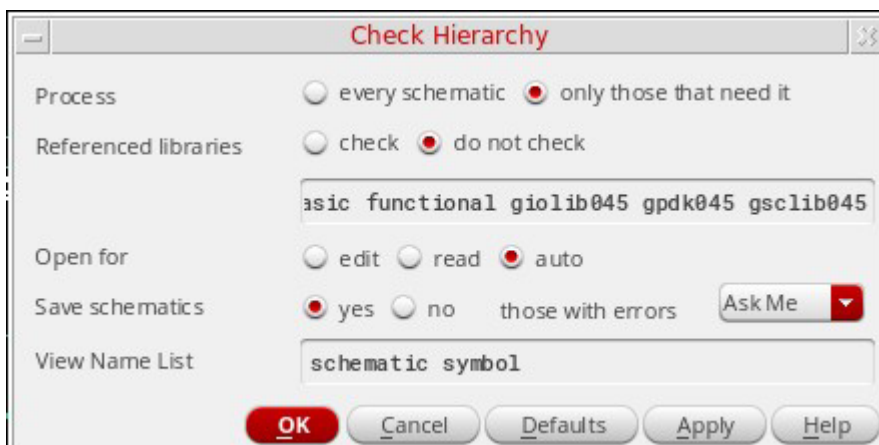
- Checks all schematic and symbol cellviews in a design hierarchy
- Extracts [connectivity information](#) from the graphics in the hierarchy
- Runs specified options using the [schematic rules checker](#) (SRC)
- Runs the [cross-view checker](#) (VIC) on views you can specify
- Checks the cellviews in the read-only mode to examine the connectivity and markers. For details, refer to [Checking Read-Only Cellviews](#).

Before checking a hierarchy, review the [check options](#).

To check all schematics in a design hierarchy,

1. From the schematic window, choose *Check – Hierarchy*.

The Check Hierarchy form appears.



Note: The appearance of this form is different when the schematic is in the context of a design configuration or a hierarchy configuration.

2. Set the options for your particular needs.
3. Click *OK*.

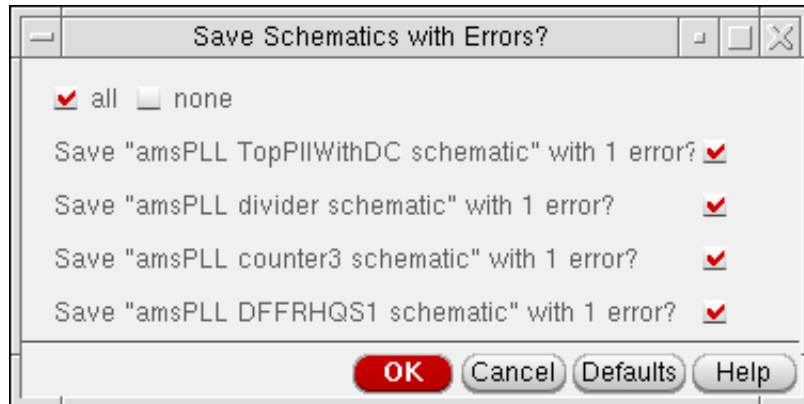
Virtuoso Schematic Editor User Guide

Checking Designs

The system runs the SRC, VIC, and any options you specified on the [Schematic Check Options form](#) or [Schematic Rules Checks Setup form](#).

If errors are found, they are listed in the CIW.

If you have errors, the [Save Schematics with Errors](#) form appears.



Note: To save time, you can check only those cellviews that have not been checked since the last time you saved your cellview. To customize the view name list, refer to the [schCycleViewNameExclusionList](#) variable.

You can register a function to be called before or after a schematic is checked using the SKILL functions [schRegPreCheckTrigger](#) or [schRegPostCheckTrigger](#).

Checking a Design Hierarchy in the Configuration Context

If you open a schematic in the context of a hierarchy configuration (configured schematic), the *Check – Hierarchy* command does not use the view name list. Instead, the command uses the schematic associated with its configuration data through partitioning. Partitioning is the process of separating the cells or blocks of a design into sets that will be simulated in either the digital or analog domain. Refer to “Partitioning Your Design” in *Virtuoso Analog Design Environment L User Guide*.

If you are checking a hierarchy in the design configuration context, the Check Configuration Hierarchy form appears, and the *View Name List* field is replaced with two fields that display the configuration name and the name of the top-level cellview.

Checking Multisheet Schematics

To check the current sheet of a multisheet schematic,

- From the schematic window, choose *Check – Current Cellview*.

To check all the cellviews of a design hierarchy,

- From the schematic window, choose *Check – Hierarchy*.

To check all sheets of a multisheet schematic,

- Run the check from the index schematic.

The offsheet connector check used in multisheet schematics performs the following operations:

- Runs only when the cellview being processed is an index of a multisheet schematic
- Flags and lists each offsheet connector specified in any of the sheet instances on the index schematic

Flags occur only when the connector imports or exports a signal that does not connect to another offsheet connector or hierarchical pin on another schematic sheet
- Flags signals that do not connect to other sheets using offsheet connectors
- Sends an error message if you use an offsheet connector incorrectly instead of using a schematic pin

If you modify some of the sheets in your design so that an offsheet connector on a specific sheet becomes disconnected, an error message appears. If your signals implicitly attach across page boundaries, an error message appears; signals need to attach across page boundaries explicitly.

- Highlights the error pin when you set the severity for the check to either *warning* or *error*, and if the offsheet connector in sheet 1 does not connect to any other offsheet connector with the same name on another sheet

Note: Whenever a multisheet schematic is checked using the *Check – Current Cellview* or *File – Check and Save* command, an `msymbol` cellview is created or updated.

Checking a Label Attachment

To highlight the association between a label and its owning object (that is, a wire or an instance):

1. From the schematic window, choose *Check – Label Attachment*.

The Check Label Attachment form appears.



2. Select which labels that you want to check as being attached to particular objects. Choose from *wire*, *instance*, *both*, or *none*.

The system will now apply a highlighted line between the displayed label text and its associated object. A flashing box appears around any labels not attached.

3. When required, select *Close* to close the form and remove all label attachments from the canvas.

Note: This action will also reset the selection to *none*.

Cross-View Checking

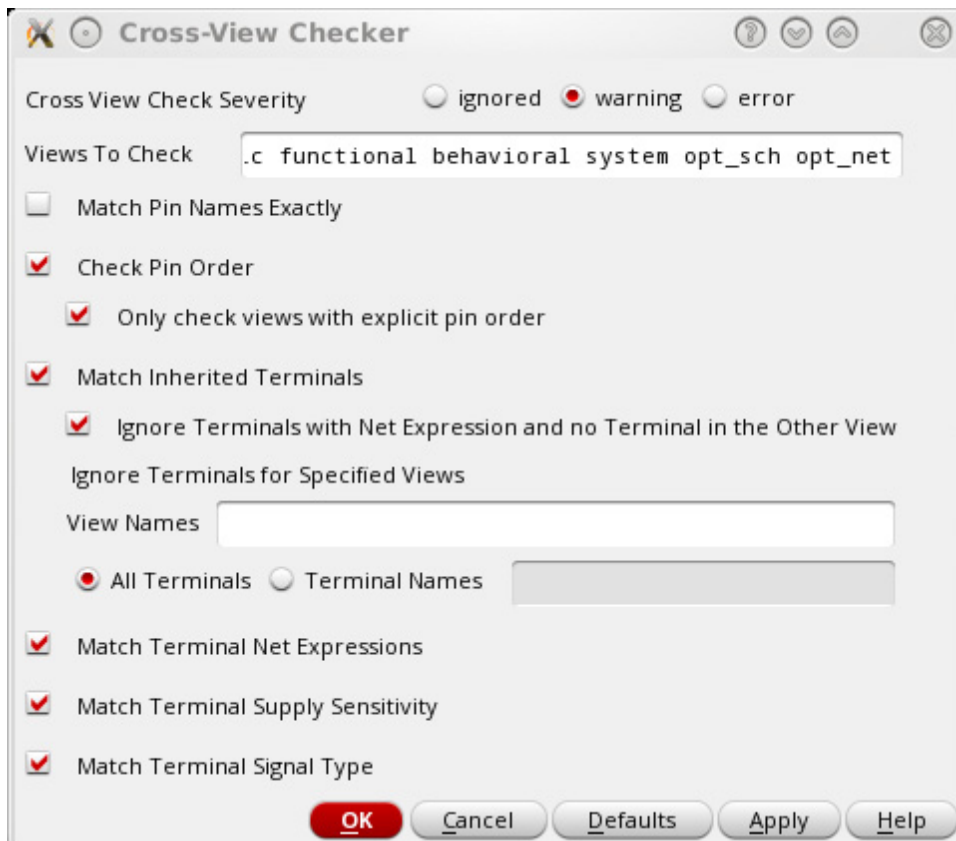
You can check symbols using the *Check – Cross View* command. It provides the following capabilities:

- Checks interface consistency between a list of specified views
- Reports signals exported in one view but not in the other
- Reports signals whose pins have different directions in the multiple views
- Reports terminal order mismatch between views
- Checks if inherited terminals match between views
- Checks if terminal net expressions match

Checks if supply sensitivity of terminals matches. To check only the interconnection consistency between views,

1. From the symbol window, choose *Check – Cross View*.

The Cross-View Checker form appears.



2. Specify the options you want to apply to the check.
3. Click *OK*.
4. View your errors or warnings.

When the check completes,

- ☐ A Cross-View Checker dialog box displays the number of errors and warnings found in the symbol.
- ☐ The software displays blinking highlighted markers in your symbol to indicate the objects with errors or warnings.
- ☐ The CIW displays a one-line explanation of each error and warning.

5. Click *Close*.
6. Choose *Check – Find Marker* to view and resolve each object error.

Checking Expanded and Unexpanded Buses Across Views

The cross-view checker flags mismatches of expanded and unexpanded buses in symbol and schematic cellviews. If, for example, you place input pins in the schematic editor for bus A as A<1>, A<2>, and A<3>, but in the symbol cellview you unexpand the bus as A<3 : 1>, cross-view checker issues the following warning message:

Warning: Pin "A<3>" doesn't have an exact match in "cell1 symbol"

To correct this warning, set up the environment variable as follows:

```
schSetEnv("vicExactPinMatch" nil)
```

Alternatively, deselect the *Match Pin Names Exactly* option in the Cross-View Checker form to avoid displaying the warning.

Ignoring Explicitly Inherited Terminals for Text Cellviews

The Cross-View Checker form lets you ignore explicitly inherited terminals for text cellviews, such as `functional`, `systemverilog`, `verilogams`, or `symbol cellviews`.

For example, consider that you have a cell `cellA` and it has three views, `schematic`, `symbol`, and `SystemVerilog`.

Here,

- The `symbol` view has one port `a`.

- The `schematic` view has two ports `a` and `b`, where `b` is an explicitly inherited terminal.
- The `systemverilog` view has two ports `a` and `b`.

To ignore inherited terminals:

1. From the schematic window, open the Cross-View checker form.
2. Select *Match Inherited Terminals*.
3. Select *Ignore Terminals with Net Expression and no Terminals in the Other View*.
4. Specify a text view type in the *View Names* field. For example, specify the `functional` view.
5. Select the Terminal Names option and specify the name of the terminal in the adjacent field.
6. Click *OK*.
7. Open the view in the Text Editor window and click *Check and Save*.

The inherited terminals are matched while the explicitly inherited terminals with pin mismatches are ignored for the specified cellviews.

Checking Read-Only Cellviews



The functionality of checking read-only cellviews is available only in Virtuoso Schematic Editor XL.

You can check a schematic or symbol when the cellview is opened in read-only mode. While checking, the temporary markers are placed on the canvas and the files are not checked out in your Design Management system. The error and warning messages are written to the log file. You can only examine a cellview in read-only mode and cannot do any further changes or corrections unless it is editable.

Select one of the following options in the Check Hierarchy to specify the mode for opening cellview during the checks:

- *auto*: Checks the hierarchy in the same mode as the top cellview. A similar option is available in the Descend form, where it descends in the same mode as the current cellview.
- *edit*: Checks the hierarchy in editable mode.

Virtuoso Schematic Editor User Guide

Checking Designs

- *read*: Checks the hierarchy in read-only mode. This mode is used for read-only checks.

You can also use the checkHierOpenMode environment variable to specify the mode in which the cellview is opened for hierarchy checks.

Bypassing Checks

Besides changing check options to *ignored*, you can bypass floating pin checks and unconnected wire checks.

Bypassing Floating Pin Checks

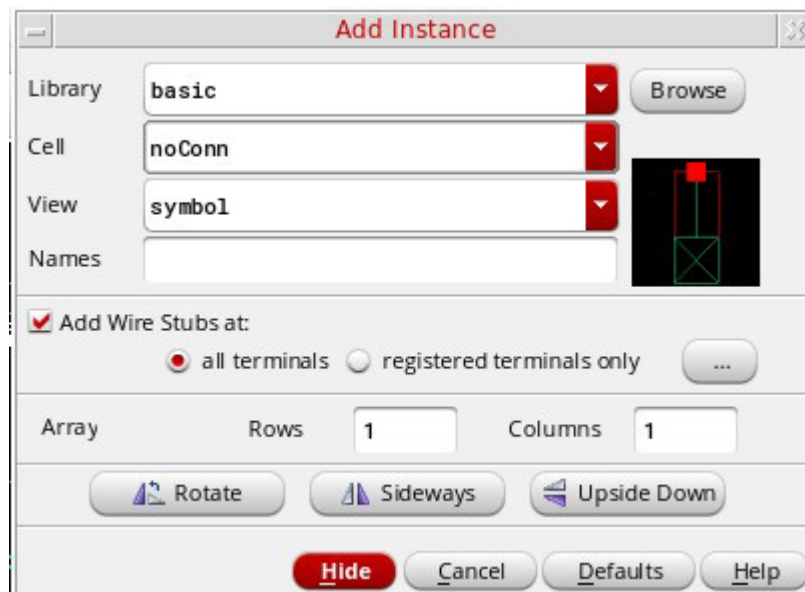
There are two methods you can use to bypass the floating pin check:

- Connect the `noConn` symbol to terminals that are unconnected
- Manually change the checks to the ignored state on the SRC setup form

To attach the `noConn` symbol to unconnected terminals,

1. From the schematic window, choose *Create – Instance*.

The Add Instance form appears.

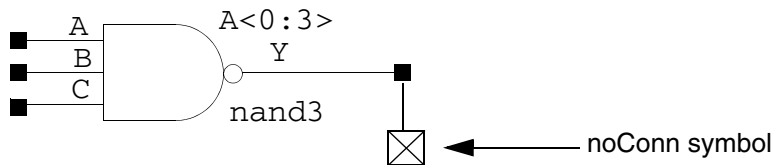


2. In the *Library* field, type `basic`.
3. In the *Cell* field, type `noConn`. The `noConn` symbol is attached to the pointer.

Virtuoso Schematic Editor User Guide

Checking Designs

- Click to place the `noConn` symbol to an unconnected terminal.



SRC does not highlight the dangling terminal as an error.

Bypassing Unconnected Wire Checks

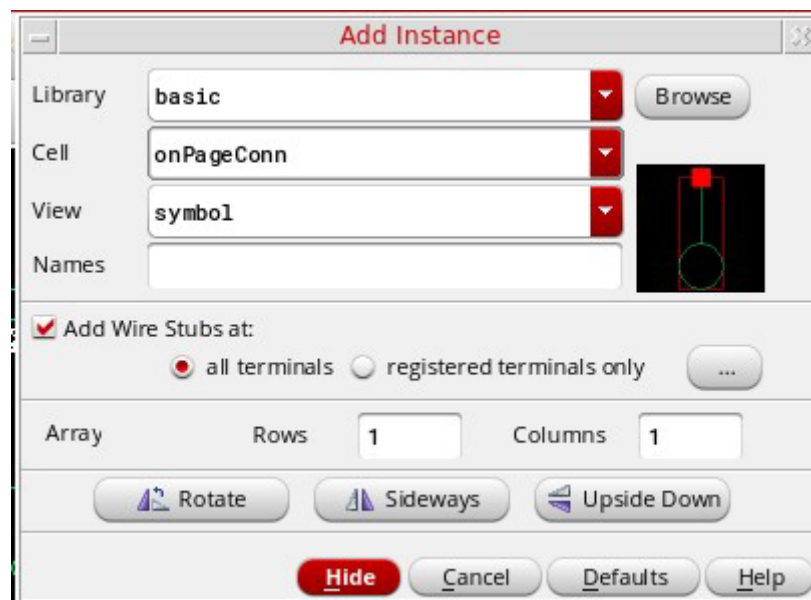
There are two methods you can use to bypass the unconnected wire check:

- Connect the `onPageConn` symbol to the wire vertex
- Manually change the check to *ignored* state on the SRC setup form

To connect the `onPageConn` symbol to the wire vertex,

1. From the schematic window, choose *Create – Instance*.

The Add Instance form appears.



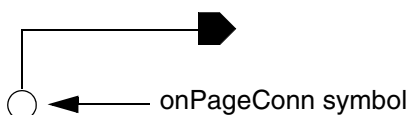
2. Click *Browse*.

The appropriate browser appears.

3. Select the `onPageConn` symbol from the *basic* library.

The `onPageConn` symbol is attached to the pointer.

4. Click to place the `onPageConn` symbol to a wire vertex.



SRC does not highlight the dangling wire as an error.

Drawing a Custom noConn Symbol

A custom noConn symbol must contain two specific cellview properties, both with the property type `string`. To define these properties, do the following:

1. Without anything selected in Virtuoso, type `geQuerySelSet()` in the CIW.

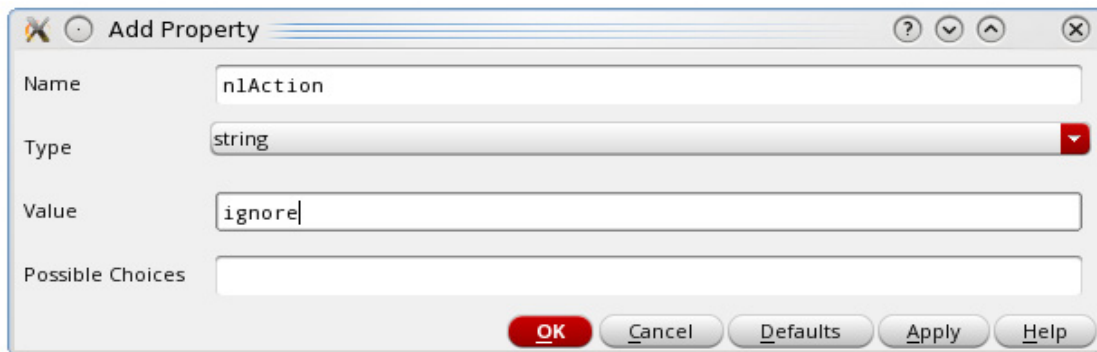
The Cellview Property Editor form appears.

2. Click *Add*.

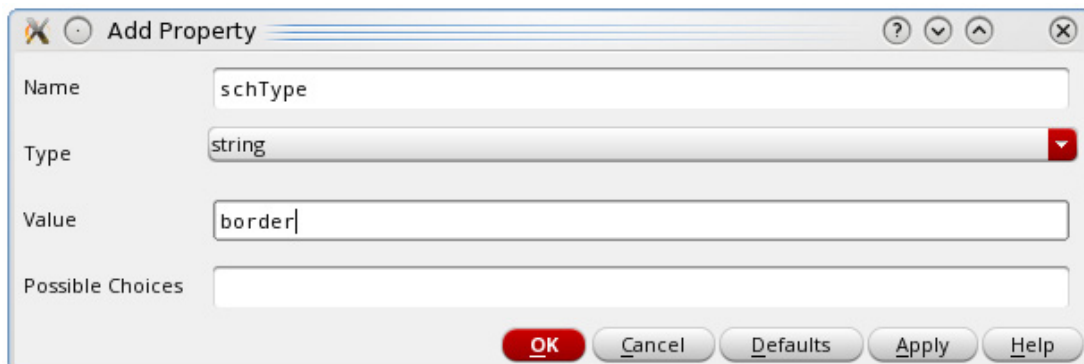
The Add Property form appears.

3. Add the following `string` type properties:

☐ `"nlAction" = "ignore"`



☐ `"schType" = "border"`



4. Draw the custom noConn symbol, ensuring it has exactly one pin named `noConn`.

The custom noConn symbol is created.

Resolving Check Errors

When a check command results in errors or warnings, a flashing highlighted marker is displayed on each object that generated an error or warning.

If your design has errors or warnings, the *Check – Find Marker* command lets you view them by listing the corresponding error and warning messages in the Find Marker form.

You can use the Find Marker form to narrow your search by limiting the list box display to a particular severity level. You can also use the Find Marker form to expand your search by expanding the scope.

For more information on the connectivity error messages issued, see [Troubleshooting Schematic Connectivity](#).

This section provides information about the following:

- [Finding Errors and Warnings in a Design](#)
- [Mapping an Error Message to a Marker](#)
- [Managing the Error Message List](#)
- [Assigning an ignoreCheck Property](#)
- [Suppressing Message Displays](#)
- [Deleting an ignoreCheck Property](#)
- [Deleting All ignoreCheck Properties](#)
- [Deleting Individual Markers](#)
- [Deleting All Markers](#)

Related Topics

[Connectivity and Naming Conventions](#)

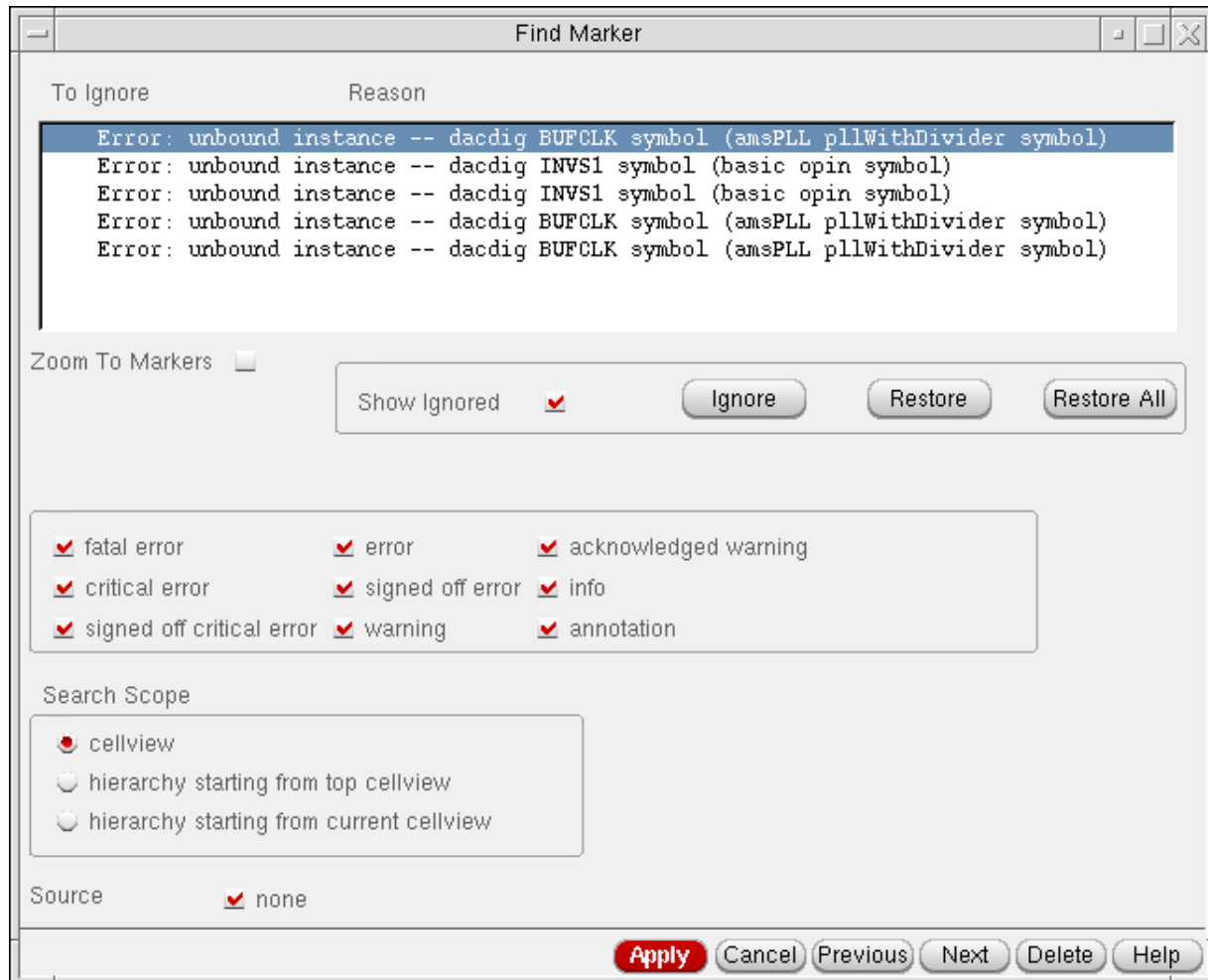
[Troubleshooting Schematic Connectivity](#)

Finding Errors and Warnings in a Design

To search for error markers in your design after you run one of the *Check* commands,

1. From the view, choose *Check – Find Marker*.

The Find Marker form is displayed.



The list box displays a list of all the error and warning messages generated by the check. By default, the first message and the associated marker are highlighted in the design window.

2. Click the *Next* and *Previous* buttons to navigate through the message list.

If you selected the *Zoom To Markers* button, the schematic editor zooms in on each associated marker and highlights the marker in the design window.

Mapping an Error Message to a Marker

To find the message that corresponds to a specific marker in the design window,

1. From the view, choose *Check – Find Marker*.

The Find Marker form is displayed.

2. Click the flashing marker in the schematic cellview.

The corresponding message is highlighted in the *Find Marker* list box.

Managing the Error Message List

After you run a check operation, you can use the Find Marker form to assign an *ignoreCheck* property to a net, an instance, or an instance terminal in your design. The *ignoreCheck* property operates in conjunction with the *Show Ignored* option on the Find Marker form. When you assign the *ignoreCheck* property to an object and also turn off the *Show Ignored* option, the software suppresses the display, in the Find Marker form list box, of error and warning messages associated with the object when the following checks are run:

- Floating input
- Floating output
- Floating net
- Floating switch
- Floating I/O
- Shorted output pin

Assigning an ignoreCheck Property

To assign an *ignoreCheck* property to a net, an instance, or an instance terminal,

1. From the view, choose *Check – Find Marker*.

The Find Marker form is displayed.

2. Click the message in the list box.
3. Click the *Ignore* button.

The software assigns an *ignoreCheck* property to the object and displays an asterisk next to the message in the column in the list box.

Suppressing Message Displays

To suppress the display of all messages that have an *ignoreCheck* property,

1. From the view, choose *Check – Find Marker*.

The Find Marker form is displayed.

2. Turn off the *Show Ignored* option.

All the messages that have an *ignoreCheck* property disappear from the list box.

Deleting an ignoreCheck Property

To delete an *ignoreCheck* property from an object,

1. From the view, choose *Check – Find Marker*.

The Find Marker form is displayed.

2. Turn on the *Show Ignored* option.

The list box displays all error and warning messages.

3. Click an error or a warning message in the list box.

4. Click the *Restore* button.

The software removes the *ignoreCheck* property and deletes the asterisk from the message in the list box.

Deleting All ignoreCheck Properties

To delete all *ignoreCheck* properties from all objects,

1. From the view, choose *Check – Find Marker*.

The Find Marker form is displayed.

2. Turn on the *Show Ignored* option.

The list box displays all error and warning messages.

3. Click the *Restore* button.

The software removes all the *ignoreCheck* properties and deletes the asterisks from the messages in the list box.

Deleting Individual Markers

To delete individual markers in the design window,

1. From the view, choose *Check – Delete Marker*.
2. Click the marker you want to delete.

You can also choose the *Find – Marker* command and click the *Delete* button at the top of the form to delete the current marker.

Deleting All Markers

To find and delete all the markers of a particular severity level, and in a particular cellview or hierarchy,

1. From the view, choose *Check – Delete All Markers*.

The Delete All Markers - OpenAccess version form appears.

Delete All Markers

Severity

☐ all ☐ none

☒ fatal error ☒ error ☒ acknowledged warning

☒ critical error ☒ signed off error ☒ info

☒ signed off critical error ☒ warning ☐ annotation

Search Scope

☒ cellview

☐ hierarchy starting from top cellview

☐ hierarchy starting from current cellview

Options

Cellview Access Mode: ☐ read ☒ edit

Confirm Access Mode Change: ☐

Sources

☒ all ☐ select

2. Select the *Severity* types, *Search Scope*, and the *Sources*.

3. In the *Options* section, select the *Cellview Access Mode*.
4. Click *OK*.

Plotting Designs

You can plot libraries, cellviews, or a portion of your schematic using one command form. Another form sets the plot options.

This chapter provides information about

- [Setting Up System Options to Plot](#)
- [Setting Up Printer Options to Use Various Paper Sizes](#)
- [Plotting a Design](#)
- [Using a Plot Template File](#)
- [Checking the Queue Status](#)
- [Canceling a Plot Job](#)

Note: See also [File – Export Image](#).

Setting Up System Options to Plot

To set up your system to plot to either a plotter or a laser printer,

1. Install the hardware.
2. Configure the system.

Refer to your operating system documentation for specifics about your plotter.

For details about configuring your operating system for plotting, refer to *Plotter Configuration User Guide*. It discusses the `.cdsplotinit` file that the Virtuoso® Schematic Editor uses.

3. Set up the software.

Setting up the Cadence® software includes defining the correct plotting device (output device), which you define in your operating system.

4. Create a site-specific version of the plot configuration file as detailed in *Plotter Configuration User Guide*.

Copy the `cdsplotinit.sample` file from the `your_install_dir/tools/plot/samples` directory to a new file called `.cdsplotinit` in either your working or home directory.

The `.cdsplotinit` file is used to define the list of available printers and plotters and their characteristics.

Setting Up Printer Options to Use Various Paper Sizes

You can set up printer options to use various paper sizes either by using PostScript printer definition (PPD) files or by adding printer size options to the `.cdsplotinit` file.

For more information, see [*Plotter Configuration User Guide*](#).

Setting Various Paper Sizes Using the `.cdsplotinit` File

To plot to a different paper size, you need to add a definition for another paper size to the `.cdsplotinit` file. Paper size choices appear in the Plot Options form when you plot using the *File – Print* command (then select *Plot Options*). The choices listed are specific to the selected plotter, and the information is read from the `~/ .cdsplotinit` file. For example, the following `.cdsplotinit` file has only one paper size, 8 1/2" x 11", defined:

```
Myprinter|Myprinter postscript: \  
:spool=lpr: \  
:query=lpq: \  
:type=postscript2: \  
:maximumPages#10: \  
:resolution#300: \  
:paperSize="8.5 X 11" 2400 3150 75 75:
```

Plotting a Design

You can plot any of the following:

- Library
- Current cellview
- Entire hierarchy starting from the current cellview or as many levels down as you want
- All sheets of a multisheet schematic by plotting from an index schematic cellview

Virtuoso Schematic Editor User Guide

Plotting Designs

Plotting a Library

To plot a library,

1. From the view, choose *File – Print*.

The Submit Plot form appears.

The screenshot shows the "Submit Plot" dialog box with the following fields and options:

- Plot**: Radio buttons for ☐ library, ☒ cellview, and ☐ viewing area.
- Library Name**: Text field containing "amsPLL" with a "Browse" button.
- Cell Name**: Text field containing "TopPllWithDC".
- View Name**: Text field containing "schematic".
- Area to Plot**: Text field containing "((-5.1875 -2.1125) (2.1875 2.23125))" with a "(Full Size)" label and "Full Size" and "Select" buttons.
- Plot Scope**:
 - Hierarchy: Radio buttons for ☐ hierarchy, ☒ current level, ☐ hierarchy, and ☐ levels down, followed by a text field containing "1".
 - View Name List**: Text field containing "schematic".
 - Skip Libraries**: Text field containing "basic sample".
 - Skip Cells Below**: Empty text field.
- Plot With**: Checkboxes for ☒ header, ☐ notes, and ☐ grid/axes.
- Notes**: A large text area for notes.
- Template File**: Text field with "Load" and "Save" buttons.
- Plotter Name**: Text field containing "No Plotters".
- Paper Size**: Text field containing "No Size".
- Total Pages**: Text field containing "1".
- Copies**: Text field containing "1".
- Plot To File**: Text field containing "Not Selected".
- Plot Options ...**: Button.
- Buttons**: "OK", "Cancel", "Defaults", "Apply", and "Help" buttons at the bottom.

Virtuoso Schematic Editor User Guide

Plotting Designs

2. In the *Plot* field, turn on *library*.
3. In the *Library Name* and *View Name List* fields, type the library you want to plot and its views. You can also click *Browse* to select the library.
4. Change other fields or options as needed.
5. Click *OK*.

Plotting a Cellview

To plot a cellview,

1. From the view, choose *File – Print*.

The Submit Plot form appears.

2. In the *Plot* field, turn on *cellview*.

3. In the *Library Name*, *Cell Name*, and *View Name* fields, type the complete path to the cellview you want to plot.

You can also click *Browse* to select the cellview.

4. Change other fields or options as needed.
5. Click *OK*.

Plotting Part of a Schematic

To plot only the viewing area of a cellview,

1. From the view, zoom in on only the object or area you want to plot by choosing *View – Zoom In*.
2. Choose *File – Print*.

The Submit Plot form appears.

3. In the *Plot* field, turn on *viewing area*.

When you plot using the viewing area, you plot the cellview from which you started the *Plot* command. Only the area displayed in the graphic window is plotted.

4. Change other fields or options as needed.
5. Click *OK*.

Plotting Without a Header Page for Every Session

To plot without a header page from a layout or a schematic,

1. From the view, choose *File – Print*.

The Submit Plot form appears.

2. Turn off *Plot With header*.
3. In the *Template File* field, type a fully-qualified template file name.
4. Click *Save*.
5. Click *Cancel*.
6. Add the following commands in the `.cdsinit` file to point to the saved template file:

For layout:

```
lePlotTemplate = path_to_the_template_file/name_of_the_template_file
```

For schematic:

```
schPlotTemplate = path_to_the_template_file/name_of_the_template_file
```

Every Cadence session will now load the plot template files from the `.cdsinit` file and set the header off by default in the Submit Plot form.

Adding Notes to a Plot

To add notes to any type of plot,

1. From the view, choose *File – Print*.

The Submit Plot form appears.

2. In the *Plot With* field, click *notes*.
3. Type in the *Notes* field.

Specific editing commands apply.

Note: See Default Keybindings for Text Fields in the *Virtuoso Design Environment User Guide*

4. Click *OK*.

Setting Plot Options

You can set several plot options: plotter name, paper size, page orientation, size and position of plotting, area, number of copies, plot time, and plot to file.

To set plot options,

1. From the view, choose *File – Print*.

Virtuoso Schematic Editor User Guide

Plotting Designs

The Submit Plot form appears. Specify the cellview (in the Library Name, Cell Name, and View Name fields).

The **Submit Plot** dialog box is shown with the following fields and options:

- Plot**: Radio buttons for **library**, **cellview** (selected), and **viewing area**.
- Library Name**: Text field with **zambezi45** and a **Browse** button.
- Cell Name**: Text field with **LP_pll_dig_combo**.
- View Name**: Text field with **symbol**.
- Area to Plot**: Text field with **((-1.025 -3.5) (4.23125 1.10625))** and a **(Full Size)** label.
- Full Size** and **Select** buttons.
- Plot Scope**:
 - Hierarchy**: Radio buttons for **current level** (selected), **hierarchy**, and **levels down**.
 - View Name List**: Text field with **schematic**.
 - Skip Libraries**: Text field with **basic sample**.
 - Skip Cells Below**: Empty text field.
- Plot With**: Checkboxes for **header** (checked), **notes**, and **grid/axes**.
- Notes**: A large text area for notes.
- Template File**: Text field with **Load** and **Save** buttons.
- Plotter Name**: **No Plotters**
- Paper Size**: **No Size**
- Total Pages**: **1**
- Copies**: **1**
- Plot To File**: **Not Selected**
- Plot Options ...** button.
- OK**, **Cancel**, **Defaults**, **Apply**, and **Help** buttons at the bottom.

2. At the bottom of the Submit Plot form, click *Plot Options*.

Virtuoso Schematic Editor User Guide

Plotting Designs

The Plot Options form appears.

The screenshot shows the 'Plot Options' dialog box with the following settings:

- Display Type:** display (dropdown)
- Plotter Name:** No Plotters (dropdown)
- Paper Size:** No Size (dropdown)
- Orientation:** portrait (radio), landscape (radio), automatic (radio, selected)
- Scale:** 1.0000 (text), Center Plot (checkbox), Fit to Page (checkbox)
- Plot Size:** 0.1000 X 0.1000 (text), inches (dropdown)
- Offset:** 0.0000 X 0.0000 (text)
- Total Plot Size:** 0.0 X 0.0 (text)
- Total Pages:** 1 (text)
- Number Of Copies:** 1 (text)
- Local Tmp Directory:** /usr/tmp (text)
- Queue Plot Data At:** 10 : 00 PM Monday (dropdowns)
- Send Plot Only To File:** (checkbox, unchecked)
- Mail Log To:** gblack (text, checked)

Buttons at the bottom: OK, Cancel, Defaults, Apply, Help.

3. Set the options.
4. Click *OK* on the Plot Options and Submit Plot forms.

Example of Use - Creating a Plot PostScript File

You can choose to plot as a PostScript file, then use a PostScript-to-PDF converter when you require to view or print.

For the following plotting to PostScript file example script to work you must:

- Have a `.cdsplotinit` file in your home directory or the current directory.
- Check the *Send Plot Only To File* option in the Plot Options form (and enter the file location).

Example script:

```
Colour PS|Hewlett-Packard LaserJet, PS: \  
:manufacturer=Hewlett-Packard: \  
:type=postscript2: \  
:spool=lp -d $PRINTER: \  
:query=lpstat -o $PRINTER: \  
:remove=cancel $1 $PRINTER: \  
:maximumPages#30: \  
:resolution#600: \  
:paperSize="A4" 4758 6846 90 90:
```

Plotting a Color Schematic in Black and White

To plot designs in black and white instead of in grayscale,

- For PostScript color plotters

In the `.cdsplotinit` file, set the type entry to

```
type=postscript1:\
```

Note: To change plotting back to grayscale, set the type entry to

```
:type=postscript2:\ .
```

- For inkjet pen plotters

The standard entry in the `.cdsplotinit` file denotes each color pen and its position as follows:

```
white#x:black#x:red#x:yellow#x:green#x:cyan#x:blue#x:magenta#x:\
```

`x` represents the pen number.

A pen plotter maps colors to the pens in its pen carousel. The Cadence Plotting Services software supports eight colors for lines and stipple patterns. A pen plotter selects colors by the index (pen) number. The index number is the location of the pen in the carousel.

To plot in black and white, determine the position of the black pen and specify that position for all the other pens, except white, in the `.cdsplotinit` file. The software then sends all the non-white color requests to the black pen for black-and-white plotting.

For example, if the position of the black pen is 5, specify the following in the `.cdsplotinit` file:

```
white#0:black#5:red#5:yellow#5:green#5:cyan#5:blue#5:magenta#5:\
```

Changing the Background Color of a Schematic Window

To change the background color of the schematic window,

1. Open the `.Xdefaults` file in edit mode.
2. Change the background color using the `defaultEditorBackgroundColor` environment variable.
3. At the UNIX prompt, type

```
xrdb -load ~/.Xdefaults
```
4. Restart your Cadence software.

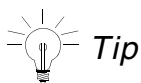
If you do not know where your `.Xdefaults` file is located, you can change the background color of the schematic and layout windows by doing the following:

1. At the UNIX prompt, type

```
xrdb -query > xyz
```

This stores all the current system `.Xdefaults` variable settings in the file `xyz`.
2. In the `xyz` file, change the background color using the `defaultEditorBackgroundColor` environment variable.
3. At the UNIX prompt, type

```
xrdb -load xyz
```
4. Restart your Cadence software.



If changing the default background color makes it more difficult to see wires of interest during interactive editing, you can change the default selection and highlight set color used by the schematic editor using the `defaultDragColor` environment variable. For example, to change the default drag color to dark red, type the following in the CIW:

```
envSetVal("ui" "defaultDragColor" 'string "darkRed")
```

Using a Plot Template File

If you want to use the same plot options for other schematics, you can store the options in a plot template file. The plot template file stores plot options in property list format. You can use a template file to plot schematics in batch mode.

Creating a Plot Template File

To create a plot template file,

1. From the view, choose *File – Print*.

The Submit Plot form appears.

2. In the *Template File* field, type the path to the directory in which you want to store the file.
3. Complete the rest of the form to indicate all settings you want to save for plotting.
4. Click *Save*.
5. Click *OK*.

You can also create a plot template file using a text editor. The following list shows plot options that you can set. All of them map to a field on the Submit Plot form.

```
library          ; text field
cell             ; text field
view            ; text field
version         ; text field
scope           ; text field, "/" should be the default.
bBox            ; not set when "Area Fit" is selected.
multisheet      ; set when "All Sheets" or "Start Sheet" is set.
startsheet
endsheet
indexsheet      ; set when "Plot Index Sheets" is set.
header          ; set when "Plot Header" is set.
hierarchy       ; set when "Hierarchy" or "Levels Down" is set.
hierleveldown
viewlist        ; set when hierarchy is set.
ignorelibs      ; set when hierarchy is set.
stopcells       ; set when hierarchy is set.
time
copy
outputfile

; plotter options form

plotter
papersize
plotsize
offset
unit
```

Virtuoso Schematic Editor User Guide

Plotting Designs

```
orientation
display
tmpdir
mail
```

```
; new options
```

```
plot          ; current value of plotAreaRadio
area          ; current value of plotAreaCyclic
notes         ; current value of plot notes
noteText      ; current value of user header
```

You can put the following sample information in the `.cdsinit` file and use it as a single cellview plot template file:

```
schPlotOptions = '(nil
library        "mylib"
cell           "mydesign"
view           "schematic"
version        "0.0"
multisheet     nil
indexsheet     nil
hierarchy      nil
copy           1
outputfile     ""
plotter        "MP Imagen"
papersize      "A"
plotsize       (200.0 165.0)
offset         (0.0 0.0)
unit           "mm"
orientation    "Automatic"
display        "display"
tmpdir         "/usr/tmp"
mail           t
)
```

The following two files are samples for a hierarchical plot template file:

```
schPlotOptions = '(nil
library        "mylib"
cell           "mydesign"
view           "schematic"
version        "0.0"
multisheet     nil
indexsheet     t
hierarchy      t
hierleveldown  2
viewlist       "schematic"
ignorelibs     "basic sample"
stopcells      ""
copy           1
outputfile     ""
plotter        "MP Imagen"
papersize      "A"
plotsize       (8.0 6.5)
offset         (0.0 0.0)
unit           "inches"
orientation    "Landscape"
display        "display"
tmpdir         "/usr/tmp"
```

Virtuoso Schematic Editor User Guide

Plotting Designs

```
mail          nil
)

schPlotOptions = ' (nil
hierleveldown 0
area           "whole"
version        "0.0"
multisheet     nil
indexsheet     nil
hierarchy      nil
copy           1
outputfile     ""
plotter        ""
papersize      "A"
plotsize       (10.5 8.0)
offset         (0.0 0.0)
unit           "inches"
orientation    "Automatic"
display        "display"
tmpdir         "/usr/tmp"
mail           nil
header         nil
notes          nil
grid           nil
noteText       ""
scope          "/"
)
```

Loading a Plot Template File

When you load a template file, the system reads in plot setup information that you have previously saved.

To load a template file,

1. From the view, choose *File – Print*.

The Submit Plot form appears.

2. In the *Template File* field, type the path to the file you want to load.

You can also type the path to one of the sample plot template files:

```
your_install_dir/tools/dfII/samples/plot/schPlot.il
your_install_dir/tools/dfII/samples/plot/schMetPlot.il
```

3. Click *Load*.
4. Click *OK*.

Creating a Default Plot Template File

To create a default plot template file,

- Add the following line to your `.cdsinit` file:

```
schPlotTemplate = "/usr/myPath/myTemplate"
```

The system loads the default plot template file when you use the *File – Print* command for the first time. Once you change the path in your `.cdsinit` file, you can use the *Load* button on the Submit Plot form to load other template files.

Checking the Queue Status

To check the status of your plot in the queue,

1. From the view, choose *File – Print Status*.

The Plot Job Queue Status/Removal form appears.

2. In the *Select Plotter* cyclic field, choose the plotter to which you sent your schematic.

The system displays the plot job queue in the text box.

Canceling a Plot Job

To cancel a job,

1. From the view, choose *File – Print Status*.

The Plot Job Queue Status/Removal form appears.

2. In the text box, click the job you want to cancel.
3. Click *Cancel Selected Plot Jobs*.
4. Click *OK*.

Virtuoso Schematic Editor User Guide

Plotting Designs

Setting Virtuoso Schematic Editor Options

You can change the appearance and the behavior of the Virtuoso® Schematic Editor by using the *Options* menu in the schematic window, symbol window, or Command Interpreter Window (CIW). In general, you can choose a preference once you are in a session. When you find a desired behavior, you can make it the default for the product for that session.

This chapter provides information about the following:

- [Overview of Virtuoso Schematic Editor Options](#)
- [Changing User Preferences](#)
- [Saving the Current Window Setup](#)
- [Viewing Bindkey Settings](#)
- [Changing Browser and CIW Preferences](#)
- [Customizing Editor Options](#)
- [Changing Display Options](#)
- [Selecting Objects Using the Filter Option](#)
- [Setting Plot Options](#)
- [Setting Schematic Check Options and Rules](#)
- [Setting the Tool Filter](#)
- [Saving Form Field Default Settings](#)
- [Loading Form Field Default Settings](#)

Overview of Virtuoso Schematic Editor Options

The following options are available from the CIW:

Command (CIW)	Description
User Preferences	Options to change window and command preferences
Save Session	Save the display of your design window
Bindkey	Option to change the set of bindkeys for the application
File Preferences	Options to change the opening preferences

The following options are available from the schematic window:

Command (schematic)	Description
Editor	Options to edit a design which applies to all edit commands
Objects	Options to change object preferences
Display	Options to display a design
Select Object Filter	Options to control selection in a design
Plot Options	Options to setup the plotter device
Check	Options to do a schematic design connectivity inspection (no options available for a symbol cellview)
Rules Setup	Options to set up the rules to do a schematic design inspection (no options available for a symbol cellview)
Tool Filter	Select tools for parameter display (no options available for a symbol cellview)
Component Display	Options to display each object's label in a design
Save Defaults	Save the current schematic editor environment variables to a file
Load Defaults	Load the schematic editor environment variable from a file to set the current environment

Changing User Preferences

You can change window and command preferences from the CIW by using *Options – User Preferences*, described in *Virtuoso Design Environment User Guide*. To make changes permanent, from the CIW choose *Options – Save Defaults* or edit the `.cdsinit` file.

Minimizing Mouse Clicks

When *Infix* is *on*, you are not prompted to specify the first data point. Instead, when you start a command, the current location of your pointer is used as the first data point.

To turn *Infix* on,

1. From the CIW, choose *Options – User Preferences*.

Virtuoso Schematic Editor User Guide

Setting Virtuoso Schematic Editor Options

The User Preferences form appears.

The screenshot shows the 'User Preferences' dialog box with three main sections: Window Controls, Command Controls, and CIW Controls. The 'Infix' checkbox under Command Controls is highlighted with a red box.

User Preferences

Window Controls

- ☐ Place Manually
- ☐ Create New Window When Descending
- ☐ Scroll Bars
- ☒ Tear-Off Menus
- ☒ Menu Shortcuts
- Mouse Prompts: **Bottom**

Command Controls

- ☐ Infix (No Click is necessary for first point)
- ☐ Options Displayed When Commands Start
- Undo Limit: **128** (range 0 to 200)
- Nest Limit: **5** (range 0 to 5)
- Double Click Time: **0**
- Beep Volume: **0**
- Web Browser: **netscape**

CIW Controls

- Output History: **1000**
- Input Buffer Lines: **0**
- Input Area Lines: **1**
- ☒ Enter Key Executes Command

Buttons: OK, Cancel, Defaults, Apply, Help

2. Select the *Infix* check box.
3. Click *OK*.

Changing the Default Display Setting for Command Option Forms

You can reconfigure your system to display the command options forms automatically whenever you select a command that has an associated form. This reconfiguration is helpful when you are unfamiliar with schematic editor commands and how they work.

To set your system to display associated options forms automatically whenever you choose commands that have an associated form,

1. From the CIW, choose *Options – User Preferences*.

The User Preferences form appears.

2. In the *Command Controls* section of the form, turn on *Options Displayed When Commands Start*.
3. Click *OK*.

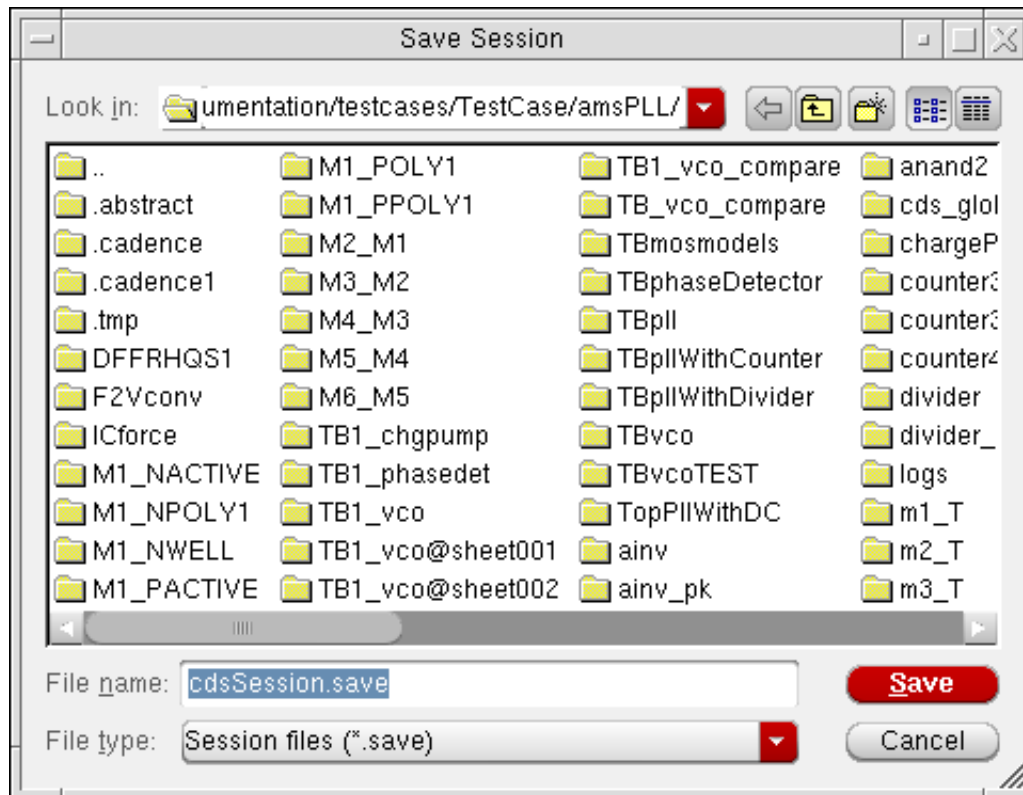
The User Preferences form closes. Command option forms appear automatically when you next start a command.

Saving the Current Window Setup

To save the display of your design window (including all window positions, customized settings, contents, window sizes, zoom levels, and design hierarchies) when you exit the current session,

1. From the CIW, choose *Options – Save Session*.

The Save Session form appears.



2. In the *File Name* field, type a filename in which to save the session data.
3. Click *OK*.

Restoring a Saved Window Setup

To restore a design session from a saved session,

- In a UNIX window, type

```
<cds> -restore <sessionFileName>
```


Virtuoso Schematic Editor User Guide

Setting Virtuoso Schematic Editor Options

where

cds is the command for initializing the system; for example, `virtuoso`.

sessionFileName is the name of the file you want to restore.

The design window appears, showing the same display setup for the windows, contents, size, zoom levels, and design hierarchies.

Viewing Bindkey Settings

After you start a session, you can view the current bindkey file settings for various application types. For more information, see [Bindkeys](#) in the *[Virtuoso Design Environment User Guide](#)*.

Changing Browser and CIW Preferences

You can change browser and CIW preferences by using the *Options – Browser Preferences* command from the CIW, described in *[Virtuoso Design Environment User Guide](#)*.

To set the browser in conjunction with the *Add Instance* command,

1. From the view, choose *Options – Editor*.

The Editor Options form appears.

2. Choose the appropriate browser in the *Add Instance Browser Type* field.

You can also change the appropriate editor option by setting the environment variable `browserType`.

Other options let you change browser preferences:

- [The Component Browser](#)
- [Changing the Component Browser Display](#)

Customizing Editor Options

You can change general options for the schematic editor by using the *Options – Editor* command, which opens the Editor Options form.

Canceling the Repeat Command Option

To turn off the *Repeat Commands* option, which stops the active command from automatically repeating:

Virtuoso Schematic Editor User Guide

Setting Virtuoso Schematic Editor Options

1. From the view, choose *Options – Editor*. The Editor Options form appears.

Editor Options

Commands | Connectivity | Inherited Conn. | Labels | Ignore Properties

Command Controls

Add Instance Browser Type ☒ library ☐ component

Show Schematic Views in Browser ☒ off ☐ on

Block Pin Direction Rule ☒ edge ☐ order of entry

Wire Snapping ☐ off ☒ on

Automatic Solder Dots ☐ off ☒ on

Solder Dot Wire Overlap

Symbol Generator Pin Sorting ☐ alphanumeric ☒ geometric

Add Symbol Pins as Instances ☒ off ☐ on

Repeat Commands ☐ off ☒ on

Cross Selection ☒ off ☐ on

Gravity ☐ off ☒ on

Minimum Marker Size

View Name List

Auto Wire Stub for Bus Terminals ☒ wide ☐ narrow

Constraint Notes Controls

Auto Constraint Notes ☐ enable ☒ disable

Sync on Design Open ☐ enable ☒ disable

Sync on Design Check ☐ enable ☒ disable

Buttons: OK Cancel Defaults Apply Help

2. Turn *Repeat Commands* off.

3. Click *OK*.

To set *Repeat* permanently on for future sessions, change the environment variable `modalCommands`.

Changing Display Options

To change the window display and grid controls,

1. From the view, choose *Options – Display*.

The Display Options form appears.

The screenshot shows the 'Display Options' dialog box with the following settings:

- Display Controls:**
 - Background Color: default
 - Net Expression Display: value only
 - Drag Limit: 100
 - Axes: off
 - Cross Cursor: off
 - Object-Sensitive Pop-Up Menus: on
 - Dynamic Highlighting: on
 - Dynamic Net Highlighting: off
 - Connect By Name Highlighting: on
 - Show Direct Connections: off
 - Show Text Justification: off
 - Show Text Indicator: off
 - Ignored Instances: on
 - Display Status Banner: off
 - Coordinates on Status Banner: off
- Grid Controls:**
 - Type: dotted
 - Spacing: 0.125
 - Multiple: 8
 - Snap Spacing: 0.0625
- Zoom / Pan Controls:**
 - Dynamic Zooming: on
 - Mode: Pan To Selected
 - Zoom Scale (%): 90
- Dimming Controls:**
 - Display Dimming: off
 - Mode: Automatic
 - Intensity: 50
 - Grayscale: off

Buttons at the bottom: OK, Cancel, Defaults, Apply, Help.

2. Edit the appropriate options.

Changing Canvas Color

To change the color of the canvas while retaining good canvas visibility:

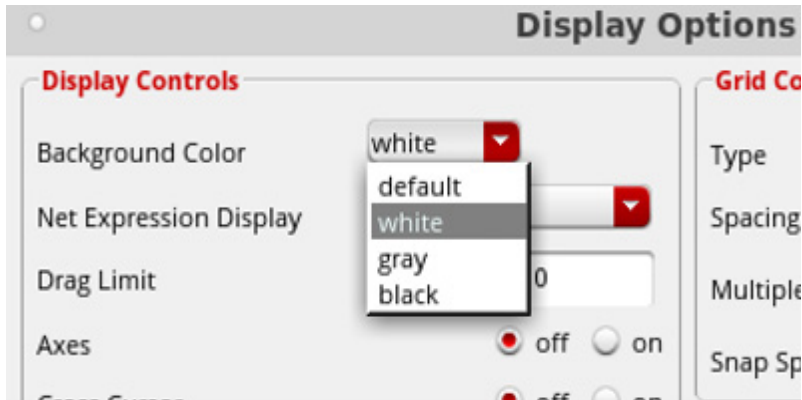
1. From the view, choose *Options – Display*.

The Display Options form appears.

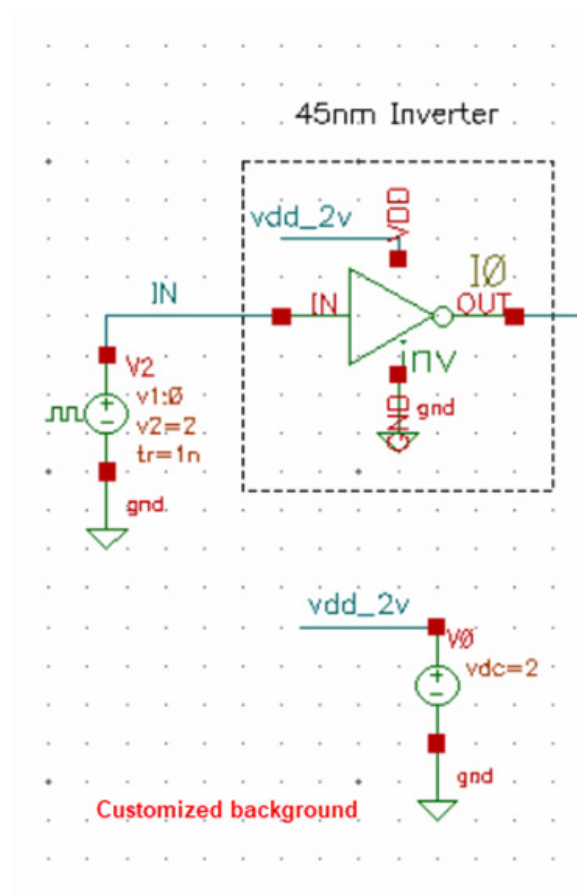
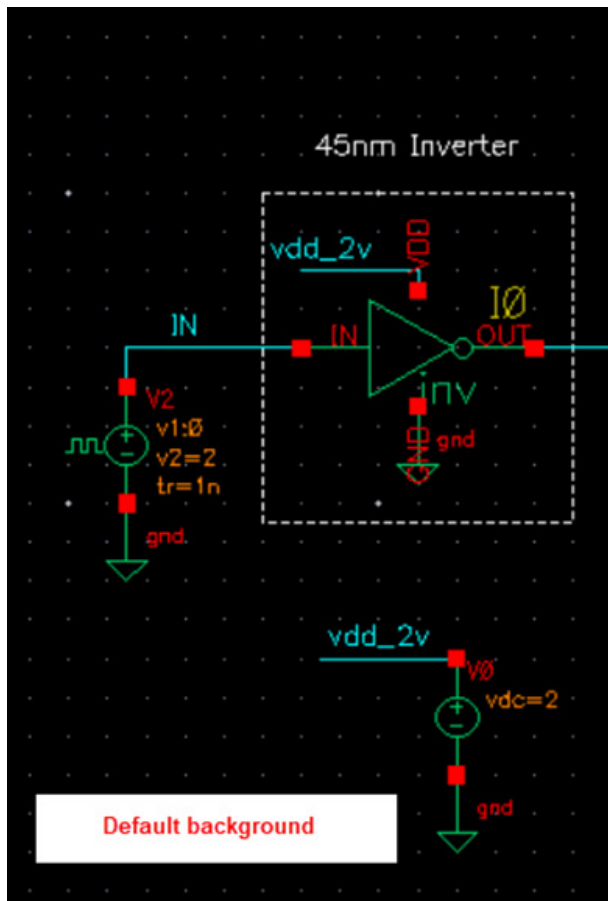
Virtuoso Schematic Editor User Guide

Setting Virtuoso Schematic Editor Options

2. Select an option from the *Background Color* drop-down menu.



The background color is changed and the color of the objects on the canvas is adjusted automatically.



Selecting Objects Using the Filter Option

The Selection Filter form lets you specify which object types you can select manually for editing. This is especially useful when you have many object types, or overlapping object types, in a small area of your design.

The *Select Filter* command is discussed in [Editing Properties](#).

Setting Plot Options

Setting plot options is discussed in [Plotting Designs](#).

Setting Schematic Check Options and Rules

Setting schematic check options and rules are discussed in [Checking Designs](#).

Setting the Tool Filter

Editing analog circuit design simulator properties using the parameter (tool) filter is discussed in [Editing Properties](#).

Saving Form Field Default Settings

To save the current environment variables and form defaults to a file,

1. From the view, choose *Options – Save Defaults*.

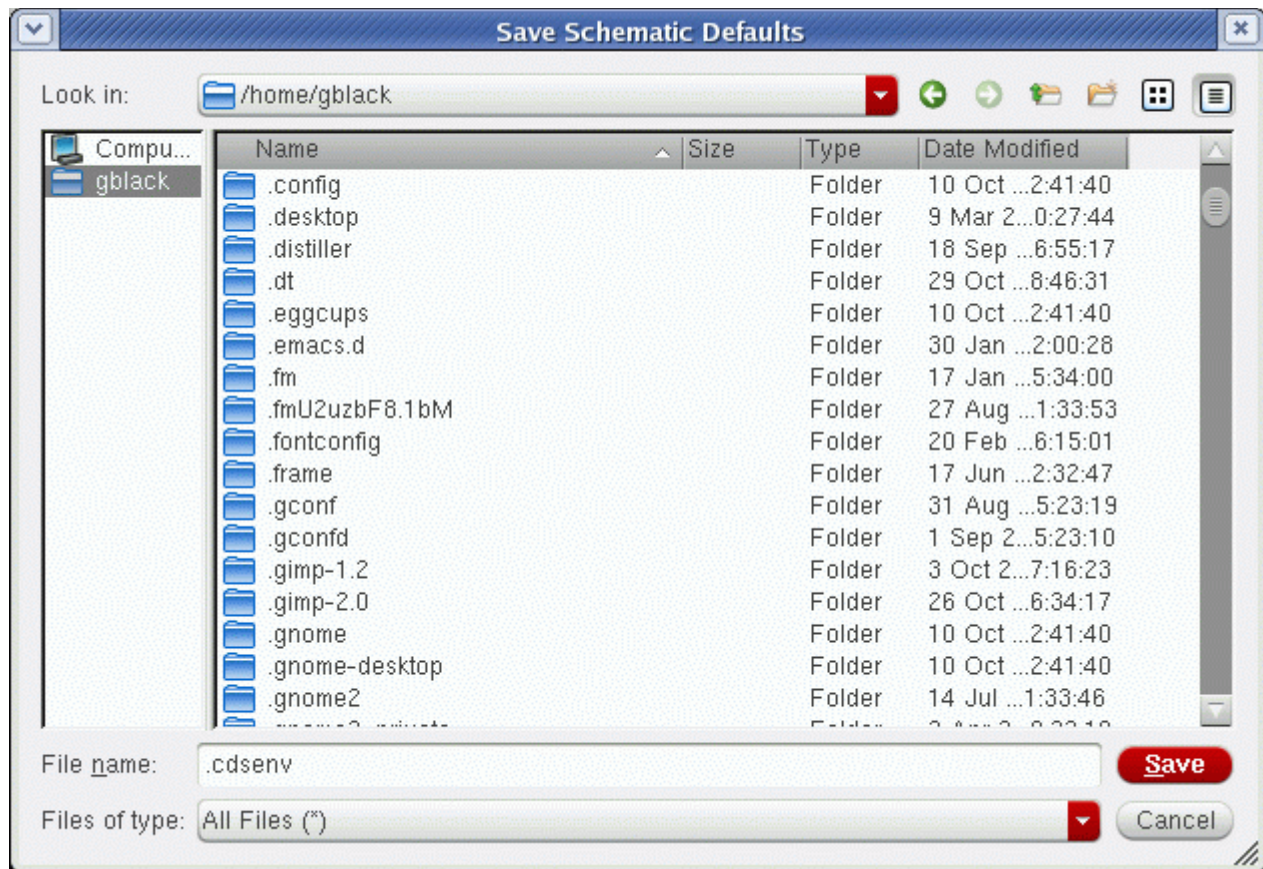
The [Save Schematic Defaults](#) form is displayed.

2. (Optional) To change the default file name, type a new name in the *File name* field.

Note: If you have an existing `.cdsenv` file, you might not want to overwrite it. Instead, you can write it to a different file and then copy the lines you want to the `.cdsenv` file.

Virtuoso Schematic Editor User Guide

Setting Virtuoso Schematic Editor Options



3. Click *Save*.

The filename specifies the name of the file where the variables and form defaults are saved. The default is a file named `.cdsenv` in your home directory. If you use this filename, the saved variables are automatically loaded the next time you start the editor.

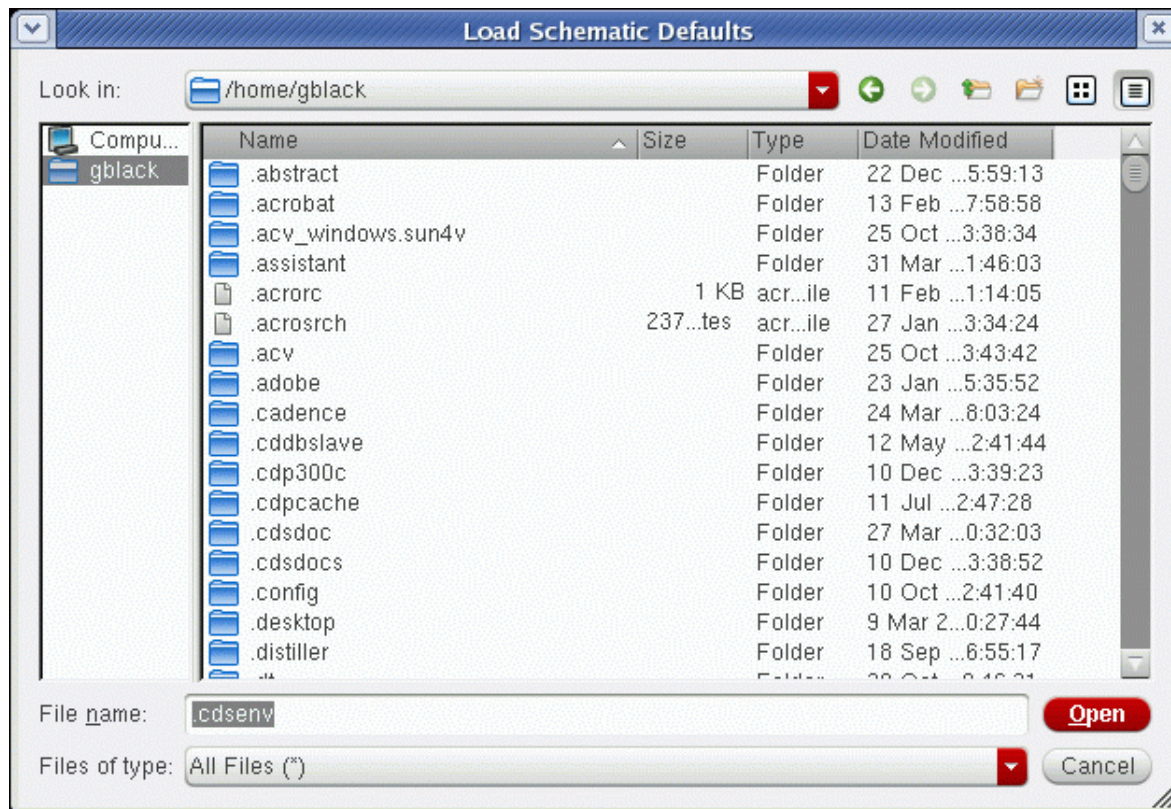
Note: This form saves environment variables only within the schematic editor environment. The same command in the CIW saves all environment variables.

Loading Form Field Default Settings

To load editor variables and form defaults from a specified file,

1. From the view, choose *Options – Load Defaults*.

The Load Schematic Defaults form appears.



2. In the *File name* field, type a new filename.
3. Click *Open*.

You can use a text editor to edit the variables in the defaults file. You can store the display of your design window (including all window positions, customized settings, contents, window sizes, zoom levels, and design hierarchies) when you exit the current session. You can store the current session by selecting a command from the CIW. You can also preset session defaults and variables by editing the `.cdsenv` file.

Note: This form loads environment variables only within the schematic editor environment. The same command in the CIW loads all environment variables.

Virtuoso Schematic Editor User Guide

Setting Virtuoso Schematic Editor Options

Customizing the Virtuoso Schematic Editor

The Virtuoso® Schematic Editor is an open application that you can customize. You can change its appearance and behavior. In general, you can choose a preference or customization once you are in a session. When you find a desired behavior or customization, you can make it the default for the product. For further details on file setup, see [Cadence Application Infrastructure User Guide](#).

This chapter provides information about the following:

- [Setting Default Schematic Application](#)
- [Customizing Pull-Down and Pop-Up Menus](#)
- [Customizing Object-Sensitive Menus](#)
- [Understanding Initialization and Setup Files](#)
- [Customizing Global Editor Variables for Form Fields](#)
- [Customizing the Plot Configuration](#)
- [Customizing Schematic Object Attributes](#)

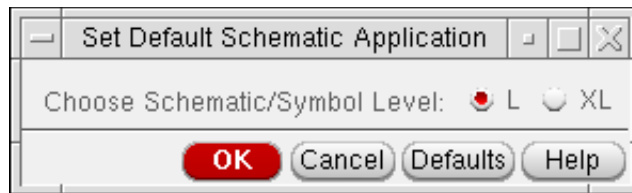
 Important

For information on customizing and creating your own workspace configurations in Virtuoso® Schematic Editor see [Working With Workspaces](#) in the [Virtuoso Design Environment User Guide](#).

Setting Default Schematic Application

You can set the default application to be invoked when viewing a particular cellview by selecting *File – Set Default Application*.

For example, you can choose to always display Schematics L or Schematics XL when viewing a schematic.



For more information see [Setting the Default Application for a Cellview](#) in the *[Virtuoso Design Environment User Guide](#)*.

Customizing Pull-Down and Pop-Up Menus

You interact with the schematic editor through pull-down menus, pop-up menus, the icon bar, and bindkeys. You can customize all schematic editor menus. In general, you copy files and edit them using a text editor.

For general information about customizing menus, see the file

`your_install_dir/tools/dfII/etc/tools/menus/README`

For an example of how to customize schematic editor menus, see the file

`your_install_dir/tools/dfII/etc/tools/schematic/README_menus.txt`

The menu files make enable you to define each menu item (for example, the menu itemSymbol, its menu text, and the `callBack` function) and to organize these definitions into the respective pull-down and pop-up menus. You can change menu definitions without having to rebuild the Cadence® SKILL language context to see the change.

You or your technical administrator can customize the menus. The menu file definitions are easier to maintain from one release to the next because they are independent of the code that implements the menus. You do not need to know the SKILL symbol name of a menu item to change its text. Instead, you modify the definition in the menu file.

Default Menu Definitions

The default menu definitions are in

`your_install_dir/tools/dfII/etc/tools/menus/schematic.menus`

For information regarding the customization of these menu items, refer to

`your_install_dir/tools/dfII/etc/tools/schematic/menus.sample`

Sample Menu File

You can use the sample customization file as a starting point for the schematic editor. It redefines the banner menu's order and labels.

For site customization, make your changes and copy the file or files into

```
your_install_dir/local/menus/schematic.menus
```

or

```
your_install_dir/local/menus/schView.menus
```

It is more likely that you will customize only the `schematic.menus` file mentioned above showing default menu definitions, but `schView.menus` is listed for completeness.

You can also customize menus for different projects or work areas. Copy the customization files to

```
your_install_dir/local/workOrProjectDir/menus/schematic.menus
```

or

```
your_install_dir/local/workOrProjectDir/menus/schView.menus
```

For personal customization, copy the files to

```
~/menus/schematic.menus
```

or

```
~/menus/schView.menus
```

The definitions in the menu files are prefixed with `schc`; for example, `schcSaveItem`. Change the contents of the definitions rather than changing the variables holding the definitions because some of the schematics menu code processes specific definitions.

Customizing Object-Sensitive Menus

You can customize object-sensitive menus (OSMs) in the following ways:

- Reassigning the OSM to a different mouse button

The default bindkey file assigns OSMs to the middle mouse button.

- Registering a custom menu with the pop-up manager
- Issuing a Cadence SKILL command to register a `menuHandle` for a specific category

This procedure customizes which menu pops up when you place the cursor over specific objects.

You can type the SKILL functions in the Command Interpreter Window (CIW), in which case they remain active for the session only. Or you can type the SKILL functions in your `.cdsinit` file, in which case they become active whenever you start the schematic editor.

Example of Customizing the Instance Object-Sensitive Menu

Use the following procedure to customize an Instance OSM to display these commands: *Stretch*, *Copy*, *Edit Properties*, *Descend Read*, and *Fit*.

1. Define each command in the new menu.

```
; newItem = hiCreateMenuItem(  
; define new variable  
; ?name 'newMenuItem  
; use same variable name  
; ?itemText "Menu String"  
; text displayed on menu  
; ?callback "MenuCommand() "  
; skill command to be executed  
; )  
newPopUpStretch = hiCreateMenuItem(  
  ?name      'newPopUpStretch  
  ?itemText  "Stretch"  
  ?callback  "schHiStretch() "  
)  
newPopUpCopy = hiCreateMenuItem(  
  ?name      'newPopUpCopy  
  ?itemText  "Copy"  
  ?callback  "schHiCopy() "  
)  
newPopUpEditInstance = hiCreateMenuItem(  
  ?name      'newPopUpEditInstance  
  ?itemText  "EditInstance"  
  ?callback  "schHiObjectProperty() "  
)  
newPopUpDescendRead = hiCreateMenuItem(  
  ?name      'newPopUpDescendRead  
  ?itemText  "DescendRead"  
  ?callback  "schHiDescendRead() "  
)
```

Virtuoso Schematic Editor User Guide

Customizing the Virtuoso Schematic Editor

```
?name      'newPopUpDescendRead
?itemText  "DescendRead"
?callback  "schHiDescendRead()"
)
newPopUpFitPlus = hiCreateMenuItem(
  ?name      'newPopUpFitPlus
  ?itemText  "Fit"
  ?callback  "hiZoomAbsoluteScale(getCurrentWindow() 0.9)"
)
```

2. Put the following menu items into a single menu.

```
; newMenu = hiCreateMenu(
; create new menu 'newMenu
; use same variable name "Title String"
; text displayed at top edge of menu
; list(
;   menuItem1      : list of menus as
;   menuItem2
; defined by . . .
; hiCreateMenuItem
; )
; ""
; )
newInstPopUpMenu = hiCreateMenu(
  'newInstPopUpMenu
  "Instance"
  list(
    newPopUpStretch
    newPopUpCopy
    newPopUpEditInstance
    newPopUpDescendRead
    newPopUpFitPlus
  )
  ""
)
```

3. Register the new menu with the schematic pop-up manager.

```
; schRegisterPopUpMenu( "objectType" menuName)
schRegisterPopUpMenu("instance" newInstPopUpMenu)
```


Understanding Initialization and Setup Files

You can customize elements of the user interface by modifying these files.

Filename	Customizing Ability
<code>schConfig.il</code>	The configuration file specifies various schematic editor configurations
<code>schBindKeys.il</code>	The bindkey file sets the bindkeys for the schematic editor
<code>schFixMenu.il</code>	The fixed menu file sets the icon bars
<code>.cdsinit</code>	The Cadence initiation file sets environment variables that contain user or site SKILL code definitions You can set these variables by calling <code>schSetEnv</code> or <code>envSetVal</code> in the <code>.cdsinit</code> file.
<code>.cdsenv</code>	The Cadence environment file sets default values for environment variables in Virtuoso Design Environment II. You can create the <code>.cdsenv</code> file by using the <i>Options – Save Defaults</i> command.

Converting a Schematic from a 0.125 Grid to a 0.1 Grid

Setting a new grid spacing can cause some of your existing symbols, pins, and wires to go off grid. The best way to convert a schematic from a 0.125 grid to a 0.1 grid, is to change the values in the `.cdsenv` file by doing the following:

Note: Change the default in your `.cdsenv` file to set the default for every library. It is not advisable to set up different grids or different snap spacing for different libraries.

1. In your `.cdsenv` file, change

```
schematic symGridSpacing float 0.125000 nil  
schematic schGridSpacing float 0.125000 nil
```

to

```
schematic symGridSpacing float 0.100000 nil  
schematic schGridSpacing float 0.100000 nil
```

2. To change the snap spacing, set

```
schematic symSnapSpacing float 0.050000 nil  
schematic schSnapSpacing float 0.050000 nil
```

Customizing Global Editor Variables for Form Fields

These IL variables describe the “indirection map” used by the schematic editor. You can customize these public global SKILL variables, which relate directly to the form field settings. These variables are stored in the `schConfig.il` file.

To edit these variables, you need to open the configuration file.

The following tables describe variables for customizing form fields.

Schematic Editor Variables

Schematic Form Field Description	Variable Name
block edge to pin type and direction	<u>schBlockPinAttrs</u>
block sizes and samples	<u>schBlockTemplate</u>
schematicSymbol views exclusion	<u>schCycleViewNameExclusionList</u>
schematic pin type	<u>schPinMasters</u>
default schematic plot template file	<u>schPlotTemplate</u>
edit properties user section	<u>schPropertiesFilter</u>
severity and value check	<u>schSRCPackagedChecks</u>
property name list selection	<u>schSelPropertyNameList</u>
sheet border sizes and types	<u>schSheetMasters</u>
simulation display format	<u>schSimProbeMasters</u>
view translation to and from	<u>schViewMasters</u>
view translation options function	<u>schViewMastersOptionFunc</u>
pin cross reference format	<u>schZoneFormatString</u>
pin cross reference format	<u>schZoneFormatValue</u>

Virtuoso Schematic Editor User Guide

Customizing the Virtuoso Schematic Editor

Symbol Editor Variables

Symbol Form Field Description	Variable Name
symbol label attributes	<u>schSymbolLabelChoices</u>
symbol pin direction and type	<u>schSymbolPinMasters</u>

Symbol Generation (tsg) Variables

The following IL variables describe the “indirection map” used by the schematic editor user interface. You can customize these public global SKILL variables used on command forms. These variables are stored in the `schConfig.il` file.

Symbol Form Field Description	Variable Name
pin connection type	<u>tsgConnectorMasters</u>
symbol pin graphic characteristics	<u>tsgPinGraphicMasters</u>
	Note: You can edit the <code>tsgPinGraphicMasters</code> variable using the appropriate <i>Attributes - List</i> option in the Symbol Generation Options form.
default tsgTemplateType	<u>tsgTemplateMasters</u>

To edit these variables, you need to open the configuration file.

HDL (Verilog and Verilog-A) Variables

Use the following variables for customizing form fields for HDL.

Symbol Form Field Description	Variable Name
enables/disables cross-view checking or checks views against edited view	<u>schHdlCrossViewCheck</u>
specified set of views to check	<u>schHdlCrossViewString</u>
controls editing and processing	<u>schHdlEditAfterError</u>
controls HDL printing	<u>schHdlPrintCommand</u>
configures the characteristics of symbols built by the text-to-symbol generator (TSG)	<u>schHdlUpdateSymbolAfterEdit</u>

AMS Netlisting Variables

You can set severities for the following AMS checks:

Form Field Description	Variable Name
illegal identifiers	<code>allowLegalIdentifiers</code>
name collisions	<code>allowNameCollisions</code>
conflicting bus ranges	<code>allowDeviantBuses</code>
sparse buses	<code>allowSparseBuses</code>

schBlockPinAttrs

```
schBlockPinAttrs = list(  
    pinType  
    list( edgeDirection terminalDirection ) ...  
)
```

Description

Sets the pin direction implied by connecting to a specific edge of a block. This IL structure describes the mapping of the block edge to pin type and direction. This map is used by the schematic user interface to provide a simplified approach for selecting the pin direction implied by connecting a wire to a specific edge of a block.

To edit this variable, you need to [open the configuration file](#). You can add or delete choices any time before you execute any schematic functionality in an interactive session.

Arguments

<i>pinType</i>	The name of the pin “type.”
<i>edgeDirection</i>	Valid Values: left, right, bottom, top
<i>terminalDirection</i>	The terminal direction.

pinType and *terminaldirection* uniquely identify a pin master in the `schSymbolPinMasters` map.

Example

```
schBlockPinAttrs = list(  
    "block"  
    list("left"    "input")  
    list("right"   "output")  
    list("bottom"  "inputOutput")  
    list("top"     "inputOutput")  
)
```

schBlockTemplate

```
schBlockTemplate = list(  
    list( blockName blockSize  
        list( label label  
            ...  
        )  
    )  
)
```

Description

Sets the block sample sizes during top-down block creation. See [Add Custom Pin](#).

To edit this variable, you need to [open the configuration file](#). You can add or delete choices any time before you execute any schematic functionality in an interactive session.

Put choices here for the block sample names, sizes, and labels that are used during block creation.

Arguments

<i>blockName</i>	A string identifying blockSample.						
<i>blockSize</i>	A list of points forming a block's border in the form <pre>list(<i>pt1</i> <i>pt2</i> <i>pt3</i> <i>pt4</i> ...)</pre>						
<i>label</i>	A list of label attributes in the form <pre>list(<i>text</i> <i>choice</i> <i>height</i> <i>font</i> <i>type</i>)</pre> where <table><tr><td><i>text</i></td><td>The label string</td></tr><tr><td><i>choice</i></td><td>The type of label; must be enclosed in quotation marks. Valid Values: instance label, logical label, physical label, pin label, device annotate, pin annotate</td></tr><tr><td><i>height</i></td><td>Label height in user units.</td></tr></table>	<i>text</i>	The label string	<i>choice</i>	The type of label; must be enclosed in quotation marks. Valid Values: instance label, logical label, physical label, pin label, device annotate, pin annotate	<i>height</i>	Label height in user units.
<i>text</i>	The label string						
<i>choice</i>	The type of label; must be enclosed in quotation marks. Valid Values: instance label, logical label, physical label, pin label, device annotate, pin annotate						
<i>height</i>	Label height in user units.						

Virtuoso Schematic Editor User Guide

Customizing the Virtuoso Schematic Editor

<i>font</i>	A dbFontStyle; must be enclosed in quotation marks.
<i>type</i>	A dbLabelType; must be enclosed in quotation marks. Valid Values: normalLabel, NLPLabel, ILLabel

Example

```
schBlockTemplate = list(  
  list("freeform"  
    list(0:0 0:0)  
    list(  
      list("[@instanceName]" "instance label" .08125  
        "stick" "NLPLabel")  
      list("[@cellName]" "logical label" .08125  
        "stick" "NLPLabel"))  
    list("small"  
      list(0.0:0.0 1.0:1.0)  
      list(  
        list("[@instanceName]" "instance label" .075  
          "stick" "NLPLabel")  
        list("[@cellName]" "logical label" .075  
          "stick" "NLPLabel"))  
      ...  
    )  
  )  
)
```

schCycleViewNameExclusionList

```
schCycleViewNameExclusionList = list(  
    excludedView ...  
)
```

Description

Sets the exclusion for views in the list from being displayed on the Add Instance form.

To edit this variable, you need to [open the configuration file](#). You can add or delete choices any time before you execute any schematic functionality in an interactive session.

Arguments

<i>excludedView</i>	The views to exclude; must be enclosed in quotation marks.

schHdlEditAfterError

```
schHdlEditAfterError = { t | nil }
```

Description

Edits Verilog[®]-A or Verilog simulator descriptions again if a syntax error is found (t).

Example

```
schHdlEditAfterError = nil
```

schHdlCrossViewCheck

```
schHdlCrossViewCheck = { t | nil }
```

Description

Enables (t) or disables cross-view checking after editing Verilog or Spectre[®] HDL. If turned on, cross-view checks are performed after editing Verilog or Spectre HDL.

schHdlCrossViewString

```
schHdlCrossViewString = checkView
```

Description

Specifies a particular set of views to check against the edited view.

Arguments

<i>checkView</i>	The particular view to check; must be enclosed in quotation marks. Setting <i>checkView</i> to <code>nil</code> or <code>" "</code> checks all views.

Example

```
schHdlCrossViewString = "schematic symbol"
```

schHdlPrintCommand

`schHdlPrintCommand = command`

Description

The UNIX command used by the *File – Print* command in the Verilog or Spectre HDL view window.

Arguments

command Must be enclosed in quotation marks.

Example

`schHdlPrintCommand = "lw -Plw1"`

schHdlUpdateSymbolAfterEdit

`schHdlUpdateSymbolAfterEdit = flag`

Description

After you edit Verilog or Verilog-A text, the symbol associated with the cell is conditionally updated to match the `pinout` of the Verilog module.

Arguments

flag

Controls the updating of the symbol.

Valid Values: `t` (update symbol after editing, if required), `nil` (do not update symbol after editing), `"query"` (display dialog box asking for symbol)

Example

`updatedschHdlUpdateSymbolAfterEdit = "query"`

schPinMasters

```
schPinMasters = list(  
    list( pinUsage  
        ...  
        list( pinDirection terminalDirection pinMaster  
            ...  
            )  
        )  
    )  
)
```

Description

Sets the pin masters choices that are used when creating a pin. Pin masters are selected based on their usage and their pin and terminal directions. Sample pin masters are in the *basic* library.

To edit this variable, you need to [open the configuration file](#). You can add or delete choices any time before you execute any schematic functionality in an interactive session.

You can list as many choices as you want; for example, *schematic*, *input*, *offsheet*, and *iopin*.

Use *offsheet* only when you are going to connect the added pins to another sheet in a multisheet schematic.

Arguments

<i>pinUsage</i>	Each pin usage value is a list of two elements: the first element is a member of pin usage as it appears on the Create Pin form; the second element indicates the pin is used as an offsheet connector when its value is 't.
<i>pinDirection</i>	Pin direction as it appears on the Create Pin form.
<i>terminalDirection</i>	The terminal direction.
<i>pinMaster</i>	Each pin master value is a list of three elements that define the library, cell, and view names of a pin master. There must a <i>pinMaster</i> for each <i>pinUsage</i> .

Virtuoso Schematic Editor User Guide

Customizing the Virtuoso Schematic Editor

Example

```
schPinMasters = list(  
  list(list("schematic" nil) list("offSheet" t))  
  list("input" "input"  
    list("basic" "ipin" "symbol")  
    list("basic" "ipin" "symbolrOff"))  
  list("output" "output"  
    list("basic" "opin" "symbol")  
    list("basic" "opin" "symbollOff"))  
  list("inputOutput" "inputOutput"  
    list("basic" "iopin" "symbolr")  
    list("basic" "iopin" "symbolrOff"))  
  list("switch" "switch"  
    list("basic" "iopin" "symbolr")  
    list("basic" "iopin" "symbolrOff"))  
)
```

schPinNetExprPlacement

```
schPinNetExprPlacement = list(  
    list (position justify xoffset yoffset)  
    ...  
)
```

Description

Stores label settings that determine pin net expression placement. It holds an association list with pin directions as keys, and disembodied property lists (DPLs) as values. The fields of each DPL stores the `position`, `xOffset`, `yOffset`, and `justify` values required for the associated pin direction. Position and justification are stored as strings, while offsets are specified in user units.

Note: The `schPinNetExprPlacement` variable can be set in `install_dir/tools/dfII/samples/local/schConfig.il`.

Arguments

<i>position</i>	Determines position of net expression placement. Must be enclosed in quotation marks. Valid Values: upperLeft, upperCenter, upperRight, centerLeft, centerCenter, centerRight, lowerLeft, lowerCenter, lowerRight.
<i>justify</i>	Determines the pin label placement. Must be enclosed in quotation marks. Valid Values: upperLeft, centerLeft, lowerLeft, upperCenter, centerCenter, lowerCenter, upperRight, centerRight, lowerRight
<i>xoffset</i>	Specifies a horizontal offset from a chosen position.
<i>yoffset</i>	Specifies a vertical offset from a chosen position.

Virtuoso Schematic Editor User Guide

Customizing the Virtuoso Schematic Editor

Example

```
schPinNetExprPlacement = ` (
    ("input" (nil
        position "upperLeft"
        xOffset 0.0
        yOffset 0.0
        justify "lowerRight"))
    ("output (nil
        position "lowerRight"
        xOffset 0.0
        yOffset 0.0
        justify "lowerLeft"))
    ("inputOutput" (nil
        position "upperCenter"
        xOffset 0.0
        yOffset 0.03125 / * for example */
        justify "lowerCenter"))
    )
```

schPinNetExprGenFunc

```
schPinNetExprGenFunc = list(  
    list( termName)  
    ...  
)
```

Description

Returns, from a terminal name, a list consisting of the generated net expression property and the default global net name.

The `schPinNetExprGenFunc` global variable can be set in `install_dir/tools/dfII/samples/local/schConfig.il`.

The default `schConfig.il` points `schPinNetExprGenFunc` to a default function. Given a terminal name, this function returns the same value as a property name, and depends upon a “!” character for the default net name (see the example below).

Arguments

<i>termName</i>	The terminal name whose net expression property and default global net name you want to return.

Example

```
schPinNetExprGenFunc = list(  
    list(termName strcat (termName "!"))  
)
```

schPlotTemplate

`schPlotTemplate = filename`

Description

This IL variable describes the default schematic plot template file that is loaded when the *Plot* command is started for the first time.

To edit this variable, you need to [open the configuration file](#). You can add or delete choices any time before you execute any schematic functionality in an interactive session.

Arguments

<i>filename</i>	The default plot template file to be loaded.

Example

```
schPlotTemplate = "~/plot.tf"
```

schPropertiesFilter

```
schPropertiesFilter = list(  
    list( object  
    list( propName ... )  
    )  
    ...  
    )
```

Description

Filters out the parameter property values on the Edit Properties form. Using the following nested list format, you can list the property names you want the Edit Properties command to skip over.

Arguments

object

An object; must be enclosed in quotation marks.

Valid Values: cellView, wire segment, pin, instance,
wire name, wire segment inherited, pin inherited

propName

The property to assign to *object*; *propName* can contain a wildcard character (*) to filter out a group of properties.

schSelPropertyNameList

```
schSelPropertyNameList = list(  
    list( viewName  
        list(  
            list( aliasName realName [ attrList ] )  
            ...  
        )  
        ...  
    )  
    ...  
)
```

Description

Sets the property names for the *Select By Property* and the *Replace and Find* commands.

To edit this variable, you need to [open the configuration file](#). You can add or delete choices any time before you execute any schematic functionality in an interactive session.

Arguments

<i>viewName</i>	Must be enclosed in quotation marks. Valid Values: <code>schematic</code> , <code>schematicSymbol</code>
<i>aliasName</i>	The name as it appears in the cyclic field on the form; must be enclosed in quotation marks.
<i>realName</i>	The actual property name to be searched for in the database; must be enclosed in quotation marks.
<i>attrList</i>	List defining the database attribute names and the expected values, in the form <i>name value</i> <i>name</i> and <i>value</i> must be enclosed in quotation marks.

Example

```
schSelPropertyNameList = list(  
    list("schematic"  
        list(  
            list("instName" "name"  
                list('("objType" "inst") '("purpose" "cell"))  
            list("netName" "name" list('("objType" "net"))  
        )  
    )  
)
```

Virtuoso Schematic Editor User Guide

Customizing the Virtuoso Schematic Editor

```
list("pinName" "name" list('("objType" "term")))
list("master" "master")
list("libName" "libName")
list("cellName" "cellName")
list("fontStyle" "font")
list("fontHeight" "height")
list("orient" "orient")
list("partName" "partName")
list("phyPartName" "phyPartName")
list("power" "power")
list("refDes" "refDes")
list("technology" "technology")
list("simMonitor" "schSimSignalName" list('("objType" "inst")))
)
)
)
```

schSheetMasters

```
schSheetMasters = list(  
    list( sheetChoice ... )  
    list( sizeName  
        sheetMaster  
        ...  
    )  
    ...  
)  
...
```

Description

Sets the choices for sheet border sizes and types on the Create Schematic Sheet form that used for a new sheet.

To edit this variable, you need to [open the configuration file](#). You can add or delete choices any time before you execute any schematic functionality in an interactive session.

Arguments

<i>sheetChoice</i>	A sheet type that appears as an element in a cyclic field; must be enclosed in quotation marks. You can list multiple choices.
<i>sizeName</i>	The first element of the sheet reference; must be enclosed in quotation marks. This is also the name that appears on the user interface form.
<i>sheetMaster</i>	A list of three elements that define the library, cell, and view names of a sheet, in the form <pre><i>library cell view</i></pre> Each must be enclosed in quotation marks. There must be a <i>sheetMaster</i> entry for each <i>sheetChoice</i>. To specify a choice that is valid but does not create a sheet border, use a list of null strings: <pre>list(" " " " " ")</pre>

Example

```
schSheetMasters = list(  
    list("basic" "continue")  
    list("A"  
        list("US_8ths" "Asize" "symbol")  
        list("US_8ths" "Asize_c" "symbol"))  
)
```

Virtuoso Schematic Editor User Guide

Customizing the Virtuoso Schematic Editor

```
list("B"  
    list("US_8ths" "Bsize" "symbol")  
    list("US_8ths" "Bsize_c" "symbol"))  
)
```


schSimProbeMasters

```
schSimProbeMasters = list(  
    list( probeFormat probeMaster)  
    ...  
)
```

Description

Sets the simulation probe (schematic monitors) to be used for a specified format.

To edit this variable, you need to [open the configuration file](#). You can add or delete choices any time before you execute any schematic functionality in an interactive session.

Arguments

<i>probeFormat</i>	The format string that appears in the cyclic field of the create display box form; must be enclosed in quotation marks. It has three special format variables: <i>name</i> , <i>state</i> , and <i>time</i> .
<i>probeMaster</i>	A list of three elements that define the library, cell, and view names of a probe symbol, in the form <pre><i>library cell view</i></pre> Each must be enclosed in quotation marks.

Example

```
schSimProbeMasters = list(  
    list("<state>"      list("basic" "simState" "symbol"))  
    list("<time>:<state>" list("basic" "simState" "symbol"))  
    list("<name>=<state>" list("basic" "simState" "symbol"))  
    list("<name>=<time>:<state>" list("basic" "simState" "symbol"))  
)
```

schSRCPackagedChecks

```
schSRCPackagedChecks = list(  
    list( value  
        list( checkType severity )  
        ...  
    )  
)
```

Description

Sets the severities and values for the various SRC checks. Added `offsheetConnCheck` to Normal, Logical Only, and Physical Only package groups.

To edit this variable, you need to [open the configuration file](#). You can add or delete choices any time before you execute any schematic functionality in an interactive session.

Arguments

<i>value</i>	Must be enclosed in quotation marks. Valid Values: Normal, Logical Only, Physical Only
<i>checkType</i>	The type of check whose value is being reset; must be enclosed in quotation marks.
<i>severity</i>	The new severity level; must be enclosed in quotation marks.

Example

```
schSRCPackagedChecks = list(  
    list("Normal"  
        list("floatingNets"      "warning")  
        list("floatingInput"     "warning")  
        list("floatingOutput"    "warning")  
        list("floatingIO"        "warning")  
        list("floatingSwitch"    "warning")  
        list("shortedOutputs"     "warning")  
        list("offsheetConnCheck" "ignored")  
        list("unconnectedWires"  "ignored")  
        ...  
    )  
)
```

schSymbolLabelChoices

```
schSymbolLabelChoices = list(  
    list( choice list( text type layerName purpose ) )  
    ...  
)
```

Description

Sets the symbol label attributes on the Add aSymbol Label form.

Arguments

<i>choice</i>	An identifier used in forms to access this list to elect the label attributes; must be enclosed in quotation marks.
<i>text</i>	The label string; must be enclosed in quotation marks.
<i>layer</i>	The layer name used for the label; must be enclosed in quotation marks.
<i>purpose</i>	The purpose of the label.
<i>type</i>	A dbLabelType; must be enclosed in quotation marks. Valid Values: normalLabel, NLPLabel, ILLLabel

Example

```
list("instance label"  
    list("[@instanceName]" "NLPLabel" "instance" "label"))  
list("device annotate"  
    list("[@refDes]" "NLPLabel" "device" "annotate"))  
list("logical label"  
    list("[@partName]" "NLPLabel" "device" "label"))  
list("physical label"  
    list("[@userPartName:%:[@phyPartName]]" "NLPLabel" "annotate" "drawing2"))  
list("pin name"  
    list("{pinName}" "normalLabel" "pin" "label"))  
list("pin annotate"  
    list("[@p_{pinName}]" "NLPLabel" "pin" "annotate"))  
list("art instance label"  
    list("cdsName()" "ILLLabel" "annotate" "drawing7" ))  
list("art pin annotate"  
    list("cdsTerm(\"{pinName}\")" "ILLLabel" "annotate" "drawing8" ))
```

Virtuoso Schematic Editor User Guide

Customizing the Virtuoso Schematic Editor

```
list( "art device annotate"  
list( "cdsParam(1)" "ILLabel" "annotate" "drawing" ))  
)
```

schSymbolPinMasters

```
schSymbolPinMasters = list(  
    list( offsetIncrement maxIncrementCount )  
        list( pinType  
            list( terminalDirection pinMaster )  
            ...  
        )  
    ...  
)
```

Description

Sets the direction and type of pin masters on the Create Pin form for creating a symbol pin.

The first element in the `schConfig.il` file lists only one entry for the number of increments and size.

To set the default *Type* to *square*, refer to the `schSymbolPinUsage` environment variable.

Arguments

<i>offsetIncrement</i>	Defines the basis unit of the label offset increment
<i>maxIncrementCount</i>	Defines the maximum basis units a label can be offset by the user. Controls the maximum value of a maximum increment slider on the user interface form.
<i>pinType</i>	The pin type as appeared on the user interface form; must be enclosed in quotation marks. Each pin type can have pins of different directions.
<i>terminalDirection</i>	The terminal direction; must be enclosed in quotation marks.
<i>pinMaster</i>	A list of three elements that define the library, cell, and view names of a pin master, in the form <i>library cell view</i> Each must be enclosed in quotation marks.

Virtuoso Schematic Editor User Guide

Customizing the Virtuoso Schematic Editor

Example

```
schSymbolPinMasters = list(  
  list(0.03125 32)  
  list("square"  
    list("input"      list("basic" "sympin" "symbolNN"))  
    list("output"     list("basic" "sympin" "symbolNN"))  
    list("inputOutput" list("basic" "sympin" "symbolNN"))  
    list("switch"     list("basic" "sympin" "symbolNN"))  
  list("round"  
    list("input"      list("basic" "circle" "symbol"))  
    list("output"     list("basic" "circle" "symbol"))  
    list("inputOutput" list("basic" "circle" "symbol"))  
    list("switch"     list("basic" "circle" "symbol"))  
  )  
)
```

schViewMasters

```
schViewMasters = list(  
    list( viewName translateTo translateFrom )  
    ...  
)
```

Description

Sets the default view used for translation on all Cellview From... forms.

To edit this variable, you need to [open the configuration file](#). You can add or delete choices any time before you execute any schematic functionality in an interactive session.

Arguments

<i>viewName</i>	The view name to be translated from or to.
<i>translateTo</i>	The name of the SKILL function for translating from this view to a pin list.
<i>translateFrom</i>	The name of the SKILL function for translating from a pin list to this view.

If a translation routine is `nil`, translation in that direction is undefined.

Synopsis of Translation Function

View to pinlist: `funcName(t_libName t_cellName t_viewName) => l_pinList`

Pinlist to view: `funcName(t_libName t_cellName t_viewName l_pinList) => t/nil`

t_libName, *t_cellName*, *t_viewName*: Defining the library, cell, and view names.

l_pinList: A DPL list describing the cellview ports, cellview properties, and port properties in the following format:

```
<l_pinList> := ( nil ports <portList> [props <propList>] )  
<portList> := ( <port> [ <portList> ] )  
<port>      := ( nil name "termName" direction "termDir"  
                [prop <propList>] [pins <pinList>] )  
  
<propList> := ( <prop> [ <propList> ] )  
<prop>      := ( nil s_propName t_propValue s_propName
```

Virtuoso Schematic Editor User Guide

Customizing the Virtuoso Schematic Editor

```

                                t_propValue )
<pinList>      := ( <pin> [ <pinList> ] )
<pin>          := ( nil name "pinName" accessDir "pinAccessDir"
                    [prop <propList>] )
```

Example

```
schViewMasters = '(
  ("schematic"  "schSchemToPinList"    "schPinListToSchemGen")
  ("symbol"    "schSymbolToPinList"    "schPinListToSymbolGen")
  ("functional" "schVerilogToPinList"   "schPinListToVerilog")
  ("behavioral" "schVerilogToPinList"   "schPinListToVerilog")
  ("system"    "schVerilogToPinList"   "schPinListToVerilog")
  ("veriloga"  "schVerilogAToPinList"  "schPinListToVerilogA");
  ("layout"    "schLayoutToPinList"    nil)
)
```


schViewMastersOptionFunc

```
schViewMastersOptionFunc = list(  
    list( viewName optionsFunc )  
    ...  
)
```

Description

The text-to-symbol generator (TSG) uses this variable to determine whether the creation function of a particular destination cellview (in this case, *symbol*) has an options form.

For example, when you choose *Edit Options* on the Cellview From Cellview form and you want to create a symbol cellview, the Symbol Generation Options form appears.

To edit the `schViewMastersOptionFunc` variable, you need to open the configuration file.

Example

```
if( isCallable(`schPinListToVerilog) then  
    schViewMastersOptionFunc = list(  
        list("schematic"    nil)  
        list("symbol"       "schDisplaySymbolTemplateForm")  
        list("functional"   nil)  
        list("behavioral"   nil)  
        list("system"       nil)  
        list("veriloga"     nil)  
        list("abel"         nil)  
    )  
else  
    schViewMastersOptionFunc = list(  
        list("schematic"    nil)  
        list("symbol"       "schDisplaySymbolTemplateForm")  
        list("abel"         nil)  
    )  
)
```

schZoneFormatString

`schZoneFormatString =`

Description

Specifies the format of the pin cross-reference information, which specifies pin locations by sheet numbers, X and Y zones. `schZoneFormatString` is used when creating pin cross-references by the *Sheet – Cross Reference* command and when parsing pin cross-references by the *Sheet – Go To Pin* command.

To edit this variable, [open the configuration file](#). You can add or delete choices any time before you execute any schematic functionality in an interactive session.

`schZoneFormatString` specifies the formatting information for the sheet number, Y zone, and X zone. `schZoneFormatValue` specifies the order of the sheet number, Y zone, and X zone in the `schZoneFormatString` formatting string. For example, the 3B2 pin cross-reference specifies a reference to a pin located in zone B2 in sheet 3.

Example

```
schZoneFormatString = "%d%[?ABCDEFGHIJKLMNOPQRSTUVWXYZ]%d"
```

schZoneFormatValue

`schZoneFormatValue =`

Description

Specifies the format of the pin cross-reference information, which specifies pin locations by sheet numbers, X and Y zones. `schZoneFormatValue` is used when creating pin cross-references by the *Sheet – Cross Reference* command and when parsing pin cross-references by the *Sheet – Go To Pin* command.

To edit this variable, [open the configuration file](#). You can add or delete choices any time before you execute any schematic functionality in an interactive session.

`schZoneFormatString` specifies the formatting information for the sheet number, Y zone, and X zone. `schZoneFormatValue` specifies the order of the sheet number, Y zone, and X zone in the `schZoneFormatString` formatting string. For example, the 3B2 pin cross-reference specifies a reference to a pin located in zone B2 in sheet 3.

Example

```
schZoneFormatValue = list(nil 'sheet 1 'yZone 2 'xZone 3)
```

tsgConnectorMasters

```
tsgConnectorMasters = list(  
    list( pinType  
        list( terminalDirection pinMaster )  
        ...  
    )  
    ...  
)
```

Description

This IL structure describes the “indirection map” used by `tsg` to provide a simplified approach for selecting the pin connectors to be used during symbol generation.

Each entry contains a pin connector name followed by a cellview specification for each direction. The pin connector name is used on the [Symbol Generation Options](#) form. The cellview should contain the graphics for just the pin connector, not the pin stub or the clock and negative indicators.

Arguments

<i>pinType</i>	The pin type as it appeared on the Symbol Generation Options form; must be enclosed in quotation marks. Each pin type can have pins of different directions as defined by the lists that follow it.
<i>terminalDirection</i>	The terminal direction; must be enclosed in quotation marks.
<i>pinMaster</i>	A list of pin masters that look for the pin based on the signal type most suited. If a signal type is not specified for a master, that master can be used with any signal type. The elements of a pin master may take the form of <pre><i>library cell view signalType</i></pre> Note: Signal types are optional. Each element must be enclosed in quotation marks.

Example

```
tsgConnectorMasters = list(  
    list("square"
```

Virtuoso Schematic Editor User Guide

Customizing the Virtuoso Schematic Editor

```
list("input"
    list("basic" "sympin" "symbolNN")
    list("opticalLib" "optPinIn" "symbol" list("optical"
        "singleModeOptical" "multiModeOptical"))
    list("basic" "other" "symbol" ("power"))
)
list("output"
    list("basic" "sympin" "symbolNN")
)
list("inputOutput"
    list("basic" "sympin" "symbolNN")
)
list("switch"
    list("basic" "sympin" "symbolNN")
)
)
list("circle"
    list("input"
        list("basic" "circle" "symbol")
    )
    list("output"
        list("basic" "circle" "symbol")
    )
    list("inputOutput"
        list("basic" "circle" "symbol")
    )
    list("switch"
        list("basic" "circle" "symbol")
    )
)
)
```

tsgPinGraphicMasters

```
tsgPinGraphicMasters = list(  
    list( pinType  
        list( terminalDirection pinGraphic )  
        ...  
    )  
    ...  
)
```

Description

This IL structure describes the “indirection map” used by TSG to provide a simplified approach for selecting the pin graphics to be used during symbol generation.

If you use `tsgPinGraphicMasters`, other than with `actHi`, it adds an additional pin shape to the symbol for display purposes only. This information is not used by any other tool.

The `actHi` value (the type name for the pin) will match an entry in `schConfig.il` that maps to a corresponding cellview containing the pin graphics.

Note: You can edit the `tsgPinGraphicMasters` variable using the appropriate *Attributes* - *List* option in the Symbol Generation Options form.

Arguments

<i>pinType</i>	The pin graphic types as they appeared on the user interface form; must be enclosed in quotation marks. Each pin type can have pin graphics of different directions.
<i>terminalDirection</i>	The terminal direction; must be enclosed in quotation marks.
<i>pinGraphic</i>	A list of three elements that define the library, cell, and view names of a pin graphic master, such as a clock or negative indicator, in the format <pre>{ <i>library</i> <i>cell</i> <i>view</i> nil }</pre> <i>library</i>, <i>cell</i>, and <i>view</i> must each be enclosed in quotation marks; <code>nil</code> indicates there are no graphics for this type.

Virtuoso Schematic Editor User Guide

Customizing the Virtuoso Schematic Editor

Example

```
tsgPinGraphicMasters = list(  
  list("actHi"  
    list("input"      nil)  
    list("output"     nil)  
    list("inputOutput" nil)  
    list("switch"     nil))  
  list("actLo"  
    list("input"      list("basic" "tsgActLo" "symbol"))  
    list("output"     list("basic" "tsgActLo" "symbol"))
```

tsgTemplateMasters

```
tsgTemplateMasters = list(  
    list(tempMasterType (prependInstallPath installPath.tsg))  
    ...  
)
```

Description

This IL structure contains a list of `tsgType/tsgTemplateFile` pairs. Each `tsgTemplateFile` contains a different set of defaults for use by TSG when creating symbols. The environment variable `tsgTemplateType` is associated with the initialization of the default `tsgType`.

You can customize the four default TSG template masters in the `schConfig.il` file, or you can create a new TSG template file that contains the current settings within the Symbol Generation Options form.

Arguments

<i>tempMasterType</i>	The TSG template master you are redefining; must be enclosed in quotation marks. Valid Values: <code>digital</code> , <code>analog</code> , <code>PCB</code> , <code>metric</code>
<i>path</i>	The installation path to the new TSG template file you want to use as a default; must be enclosed in quotation marks.

Example

```
tsgTemplateMasters = list(  
    list("digital" (prependInstallPath "samples/symbolGen/default.tsg"))  
    list("analog" (prependInstallPath "samples/symbolGen/artist.tsg"))  
    list("PCB" (prependInstallPath "samples/symbolGen/package.tsg"))  
    list("metric" (prependInstallPath "samples/symbolGen/metric.tsg"))  
)
```


Customizing the Plot Configuration

Most of the choices for the cyclic fields on the Plot Options form are set in the display resource file (the `.cdsplotinit` file), which you can edit. This file is the foundation plotter model data base for Cadence Design Systems plot software. This file builds site specific `.cdsplotinit` files.



You must create a site-specific version of the `.cdsplotinit` file and locate it in your `_install_dir/tools/plot`, your working directory, or your home directory.

A read-only version of the `.cdsplotinit` file is located in `your_install_dir/tools/plot/etc/cdsplotinit`. You can copy this file to your working directory or your home directory and then edit it. A sample `.cdsplotinit` file is available in `your_install_dir/tools/plot/samples/cdsplotinit.sample`.

You can edit the following specifications:

- Type and name of plotter
- Paper size
- Page orientation
- Page size
- Resolution

Customizing Schematic Object Attributes



If you amend objects' layer purpose pair, it would impact the actual meaning of an object. For example, only a line on a wire layer, and drawing/flight, can determine connectivity. Likewise, only a label on a pin layer, with label purpose, would be treated as a pin name, and so on. Therefore, if you change the layer purpose pair (LPP) of an object, then the original intent of that object can be lost.

However, if you want to change settings in the display resource file, or the associated setting in virtual memory, there is a facility to do so.

To change the color, and other object attributes, you have to make amendments to *packet groups*. Each LPP has an associated packet that, in turn, has an associated color. You will need to change the packet color, of the relevant packet, associated with the LPP.

For more information on how to do this, see [Display Resource File Development and Editing, Reusing, and Merging Display Resources](#) in the *Virtuoso Technology Data User Guide*.

Note: Changing colors using the Display Resource Editor will cause all wires of that type to change color. If you want to only change the color of, for example, one or more selected wires, you can use the Edit Object Properties form (in this case see [Edit Object Properties – Wire Segment](#) on page 1419).

Schematic Objects and LPPs

The table below details the various schematic objects and their associated LPPs:

Object	LPPs
Pins	dbcPinLayer, dbcDrawingPurpose
Pin Names	dbcPinLayer, dbcLabelPurpose
Pin Net Expression	dbcPinLayer, dbcAnnotatePurpose
Differential Pin Names	dbcAnnotateLayer, dbcDrawingPurpose
Inst Pins	dbcPinLayer, dbcDrawingPurpose
Inst Pin Names	dbcPinLayer, dbcLabelPurpose
Wires	dbcWireLayer, dbcDrawingPurpose

Virtuoso Schematic Editor User Guide

Customizing the Virtuoso Schematic Editor

Flight Wires	<code>dbcWireLayer, dbcFlightPurpose</code>
Wire Names	<code>dbcWireLayer, dbcLabelPurpose</code>
Wire Net Expression	<code>dbcWireLayer, dbcLabelPurpose</code>
Instance Name	<code>dbcInstanceLayer, dbcLabelPurpose</code>
Instance Label (instance names)	<code>dbcInstanceLayer, dbcLabelPurpose</code>
Symbol Name	<code>dbcDeviceLayer, dbcLabelPurpose</code>
Symbol Shapes (line/rect etc)	<code>dbcDeviceLayer, dbcDrawingPurpose(outline)</code> <code>and</code> <code>dbcDeviceLayer, dbcDrawing1Purpose(filled)</code>
Device Annotate Label (ref Des)	<code>dbcDeviceLayer, dbcAnnotatePurpose</code>
Logical Label (Symbol/Part Names)	<code>dbcDeviceLayer, dbcLabelPurpose</code>
Physical Labels (phys part name etc)	<code>dbcAnnotateLayer, dbcDrawing2Purpose</code>
Pin Annotate (pinNet Expr etc)	<code>dbcPinLayer, dbcAnnotatePurpose</code>
Analog Instance Label (cdsName())	<code>dbcAnnoateLayer, dbcDrawing7Purpose</code>
Analog Pin Annotate (cdsTerm)	<code>dbcAnnoateLayer, dbcDrawing8Purpose</code>
Analog device Annotate (cdsParam())	<code>dbcAnnoateLayer, dbcDrawingPurpose</code>
Instance BBox	<code>dbcInstanceLayer, dbcDrawingPurpose</code>
solder dot	<code>dbcWireLayer, dbcDrawingPurpose</code>
"Index Schematic" Label	<code>dbcTextlayer, dbcDrawingPurpose</code>
Markers (CDB)	<code>dbcMarkerLayer, dbcErrorPurpose,</code> <code>dbcMarkerLayer, dbcWarningPurpose</code>
Note Label	<code>dbcTextLayer, dbcDrawingPurpose</code>
Note Shape	<code>dbcTextLayer, dbcDrawingPurpose(solid lines)</code> <code>dbcTextLayer, dbcDrawing1Purpose(dashed line)</code>
Wire Text (aliases)	<code>dbcAnnotateLayer, dbcDrawing3Purpose</code>

Virtuoso Schematic Editor User Guide

Customizing the Virtuoso Schematic Editor

Sheet Borders

`dbcBorderLayer, dbcDrawingPurpose`

The following table lists the various schematic objects and their corresponding LPPs which are located in the Display Resource Editor, along with details of where the exact packet can be located (in the Display Resource Editor).

Object	LPPs	Packet
Pins	pin, drawing	pin
Pin Names	pin, label	pinLbl
Pin Net Expression	pin, annotate	pinAnt
Differential Pin Names	annotate, drawing	annotate
Inst Pins	pin, drawing	pin
Inst Pin Names	pin, label	pinLbl
Wires	wire, drawing	wire
Flight Wires	wire, flight	wireFlt
Wire Names	wire, label	wireLbl
Wire Net Expression	wire, label	wireLbl
Instance Name	instance, label	instanceLbl
Instance Label (instance names)	instance, label	instanceLbl
Symbol Name	device, label	deviceLbl
Symbol Shapes (Outline) (line/rect etc.)	device, drawing	device
Symbol Shapes (Filled) (line/rect etc.)	device, drawing1	device1
Device Annotate Label (ref Des)	device, annotate	deviceAnt
Logical Label (Symbol/Part Names)	device, label	deviceLbl
Physical Labels (physical part name etc.)	annotate, drawing	annotate
Pin Annotate (pin Net Expressions etc.)	pin, annotate	pinAnt
Analog Instance Label (cdsName ())	annotate, drawing7	annotate7
Analog Pin Annotate (cdsTerm())	annotate, drawing8	annotate8

Virtuoso Schematic Editor User Guide

Customizing the Virtuoso Schematic Editor

Object	LPPs	Packet
Analog Device Annotate (cdsParam())	annotate, drawing	annotate
Instance BBox	instance, drawing	instance
Solder Dot	wire, drawing	wire
“Index Schematic” Label	text, drawing	text
Error Markers (CDB)	marker, error	markerErr
Warning Markers (CDB)	marker, warning	markerWarn
Note Label	text, drawing	text
Note Shape-solid Lines	text, drawing	text
Note Shape-dashed Lines	text, drawing1	text1
Wire Text (aliases)	annotate, drawing3	annotate3
Sheet Borders	border, drawing	border
Selection	hilite, drawing	hilite
Command-specific highlights	hilite, drawing1-drawing9	hilite1-hilite9

Virtuoso Schematic Editor User Guide

Customizing the Virtuoso Schematic Editor

The Search Assistant and Toolbar

This chapter contains the following topics on how to use *Search* functionality in the Virtuoso Schematic Editor L/XL:

- Introducing Search
 - Search Terminology
- Accessing the Search Assistant and Toolbar
- Hiding the Search Assistant and Toolbar
- The Search User Interface
 - The Search Toolbar
 - The Search Assistant Pane

Introducing Search

The *Search* assistant pane and *Search* toolbar provide a wide range of design search facilities, including dynamic context search categories, fast keyword directed searching with immediate feedback, and an intuitive user interface (UI) that makes design data searching both simple and productive.

Search uses Virtuoso platform data structures and commands, and presents them through an interactive UI allowing you to refine queries based on the results fed back. Precise searches within searches are then possible as design data is structured and contains unique identifiers for netlisting and backannotation purposes.

Note: The schematic editor also provides database (cellview) centric *Find* and *Replace* functionality via the *Edit – Find* and *Edit – Replace* options.

Some of the key features provided by *Search* include the ability to:

- Apply queries to specific categories of design data
- Perform dynamic context aware searching
- Utilize keyword searching with immediate feedback
- Refine search queries on initial search results
- Text auto-completion based on previous search history

Note: Auto-completion remains in sync between The Search Toolbar, The Search Assistant Pane and the Annotation Browser. If you choose to *Clear History* in the *Search* assistant, auto-completion will be wiped.

Search Terminology

To fully appreciate the extent of the functionality provided by Schematics *Search*, you should be familiar with some of the common terminology used:

Term	Description
<i>Category</i>	A search option that matches a specific type of content. Search results are collected in categories which provide a convenient means of characterizing and sorting <u>hits</u>. Categories can be specific to a particular session window, object, or stage in the design flow.
<i>Category Group</i>	A collection of <u>categories</u> that share common characteristics. In order to provide quick access to related objects, categories can also belong to one, or more, category groups that are always referenced by name. You can choose to search individual categories or across all categories simultaneously (for example, typing an instance name may generate search results across many categories). Therefore, multi-dimensional searches, and the ability to browse and sort <u>hits</u> across dimensions, can greatly speed up the browsing of design data. See also the <u>Search Categories Pull-Down</u> on page 797.
<i>Context Aware</i>	An application specific collection of search <u>categories</u>. As there is potential for a large amount of design data content, it is important that search space is minimized as much as possible. This helps focus the number of <u>hits</u>, as well as simplifying feedback and allowing quicker search performance. This search space reduction is achieved by limiting each search to data that is associated with the current view or application. For example, searching a schematic sheet will only return <u>hits</u> for the associated cellview. Descending into another cellview, or changing the active window, will change the search scope. Note: <u>Categories</u> are registered and unregistered dynamically so that each scope can prioritize <u>hits</u> according to the available context information. See also <u>Ranking Hits</u> on page 806.

Virtuoso Schematic Editor User Guide

The Search Assistant and Toolbar

History

Displays the most recent search criteria entered.

As search queries can be difficult to remember, *Search* functionality maintains a history of applied queries. These can be accessed through a pull-down menu.

See also [Search History/Results Pull-Down](#) on page 803.

Hit

A target object that matches a given query.

By instantly displaying search results, you can then refine your search criteria interactively. In schematic views, because many objects have unique identifiers, a single word search will often be sufficient, removing the need to specifically state the object type.

For example, it is common in schematics to use naming conventions to distinguish types of instances (the schematic editor's automatic naming functionality helps reinforce this). Resistors may be named `r1`, `r2`, `res5`, and capacitors named `cap1...cap5`, nets named `net1...net50`, and so on. In such cases, typing "r", "cap", or "net" will often be sufficient to list a small number of results to find the desired object.

Note: Design teams may adopt certain naming conventions which can be used to filter objects without the need to build these conventions into the search engine.

Hit Group

A collection of hits from the same category.

Virtuoso Schematic Editor User Guide

The Search Assistant and Toolbar

Keyword

A special query word that matches a category or a category group.

Each category or category group has one or more keywords that allow you to limit the scope of searches. This is particularly useful where specific identifiers are not known. For example, when presented with a new schematic, you may quickly navigate the design data using keywords such as *cells*, *pins*, *notes*, or *markers*.

Implicit keywords are also created for each category group so that a single keyword may match multiple categories.

As you will realize, keywords only become useful once they are known. To help with this a number of conventions and UI aids are provided:

- each category name is a keyword.
- each category group name is a keyword.
- each keyword is a plural noun which will make it less likely to clash with identifier substrings.
- keywords match text in existing menu entries and commands.
- keywords may end in an optional “:” character (although this is guaranteed to only match a category or category group).

Note: Keyword matches are always case insensitive, but may be treated as regular query words, which means that they may additionally be matched using the current search options.

Virtuoso Schematic Editor User Guide

The Search Assistant and Toolbar

<i>Query</i>	An expression that contains <u>words</u>, <u>keywords</u>, and operators. Each query is first split into multiple terms, each of which is categorized as being either a regular search <u>word</u> or <u>keyword</u>. Words are matched using logical operators specified through the following <i>Find With</i> options: ■ <i>All Of The Words</i> (AND) ■ <i>Any Of The Words</i> (OR) ■ <i>The Exact Phrase</i> (EXACTLY) ■ <i>None Of The Words</i> (NOT) Terms are delimited by whitespace, but the <i>Find With: The Exact Phrase</i> option can be used to define multiple word terms. The default search mode uses the AND operator so that results are refined as more terms are entered. See also Advanced Search on page 800.
<i>Result</i>	A collection of <u>hit groups</u> that are generated by the search engine.
<i>Scope</i>	Where the search result was found. This could be in the current tab, the current window, all windows, the current hierarchy, or the top-level hierarchy. See also Advanced Search on page 800 and Displaying The Search Scope on page 815.
<i>Target</i>	A <u>category</u> object that is matched using a search <u>query</u>.

Word

A regular query token that is matched with the content search data.

There are four types of word matching that are available:

- *Prefix*
- *Substring*
- *Exactly*
- *Suffix*

These *Match Word* options do not apply to keywords, but allow you to take advantage of naming conventions that you may have adopted in your design.

Note: All characters of a word will be matched.

See also Advanced Search on page 800.

Accessing the Search Assistant and Toolbar

- To access the *Search* assistant pane in Schematics either:
 - a. Select *Window – Assistants*, and then check the *Search* option.
 - b. Right-click in the main Schematics toolbar area and select *Search* from the upper section of the context-menu displayed.

Schematics will embed *Search* as a docked assistant pane within the current session window; by default, if available, on the right hand side of the canvas area.

- To access the *Search* toolbar in Schematics :
 - a. Select *Window – Toolbars*, and then check the *Search* option.
 - b. Right-click in the main Schematics toolbar area and select *Search* from the lower section of the context-menu displayed.
 - c. Select the *Explore* workspace (from either the *Window Configuration* toolbar pull-down or by selecting *Window – Workspaces – Explore*).

Hiding the Search Assistant and Toolbar

- To hide the *Search* assistant pane from the current session window in Schematics either:
 - a. Right-click in the main Schematics toolbar area and de-select *Search* from the upper section of the context-menu display.
 - b. Select *Window – Assistants*, and then uncheck the *Search* option.
 - c. Click the *Hide* button (X) in the *Search* assistant pane title bar.
- To hide the *Search* toolbar in Schematic :
 - a. Select *Window – Toolbars*, and then uncheck the *Search* option.
 - b. Right-click in the main Schematics toolbar area and de-select *Search* from the lower section of the context-menu displayed.
 - c. Change from the *Explore* workspace.

The Search User Interface

Search functionality in Schematics comprises of two main search entry mechanisms:

- [The Search Toolbar](#)
- [The Search Assistant Pane](#)

The Search Toolbar

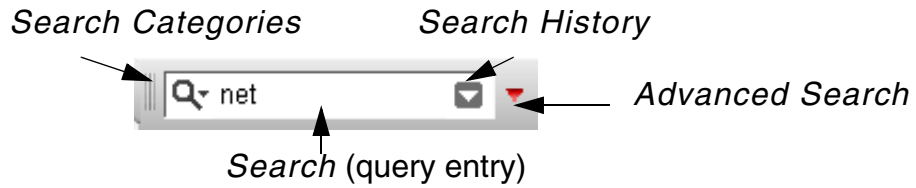


Figure 14-1 The Search Toolbar

The *Search* toolbar comprises of the following UI components:

- Search Query Entry
- Search Categories Pull-Down
- Advanced Search
- Search History/Results Pull-Down
- Search Toolbar Context-Menus

Search Query Entry

The *Search Query Entry* field is used to enter any search queries (searches) that you want to perform on the current design data.

When the *Search* toolbar is inactive, the word *Search* will be displayed in gray text in the entry field (alternatively, the current category will be displayed). As search queries are entered in the *Search Query Entry* field, results will begin to display immediately in the Search History/Results Pull-Down.

Note: If a query is taking some time to complete a progress bar will be displayed in the status bar area at the bottom right of the schematic's window. This search can be canceled (stopped) using the `Ctrl+C` keys.

Right-clicking over the *Search Query Entry* field will display a context-menu which will allow to perform a variety of text editing tasks including *Cut*, *Copy* and *Paste* (see also Search Toolbar Context-Menus).

Search Categories Pull-Down

The *Search Categories* (see category) pull down is used to refine the scope of your search. For example you can choose to only include *Instances* in your Search Query Entry.

If you want to perform multiple category group searches enter the category keywords in the Search Query Entry field. For example you could enter "Instances Nets Libraries".

From the pull-down you can also choose to:

- Check the *Selected* option, at the bottom of the pull-down list, to restrict a search query to only those objects that are selected in the current view.

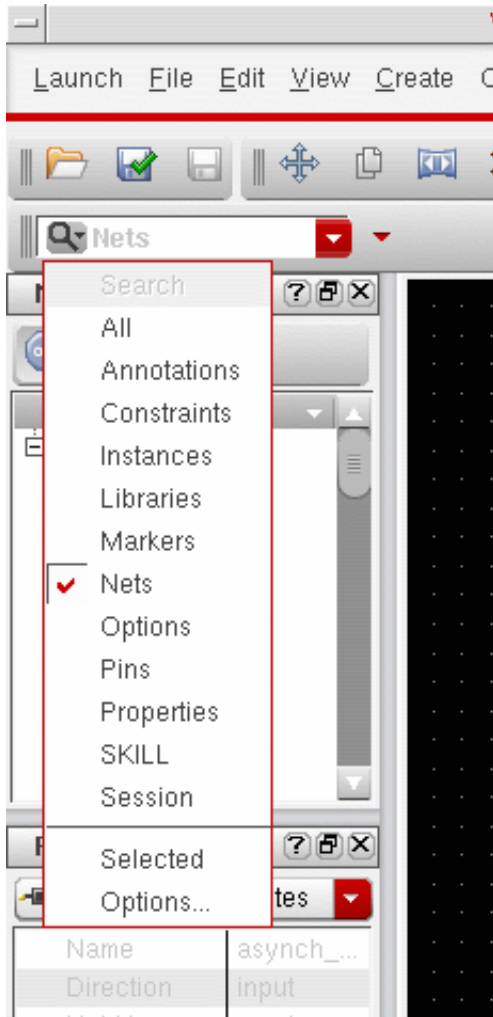


Figure 14-2 Search Categories list

- Click the *Options* entry to display the Search Options form. From here, you can select and deselect the categories of object that are to be displayed in the *Search Categories* pull-down. The search options listed here are also known as “search plug-ins”.

Virtuoso Schematic Editor User Guide

The Search Assistant and Toolbar

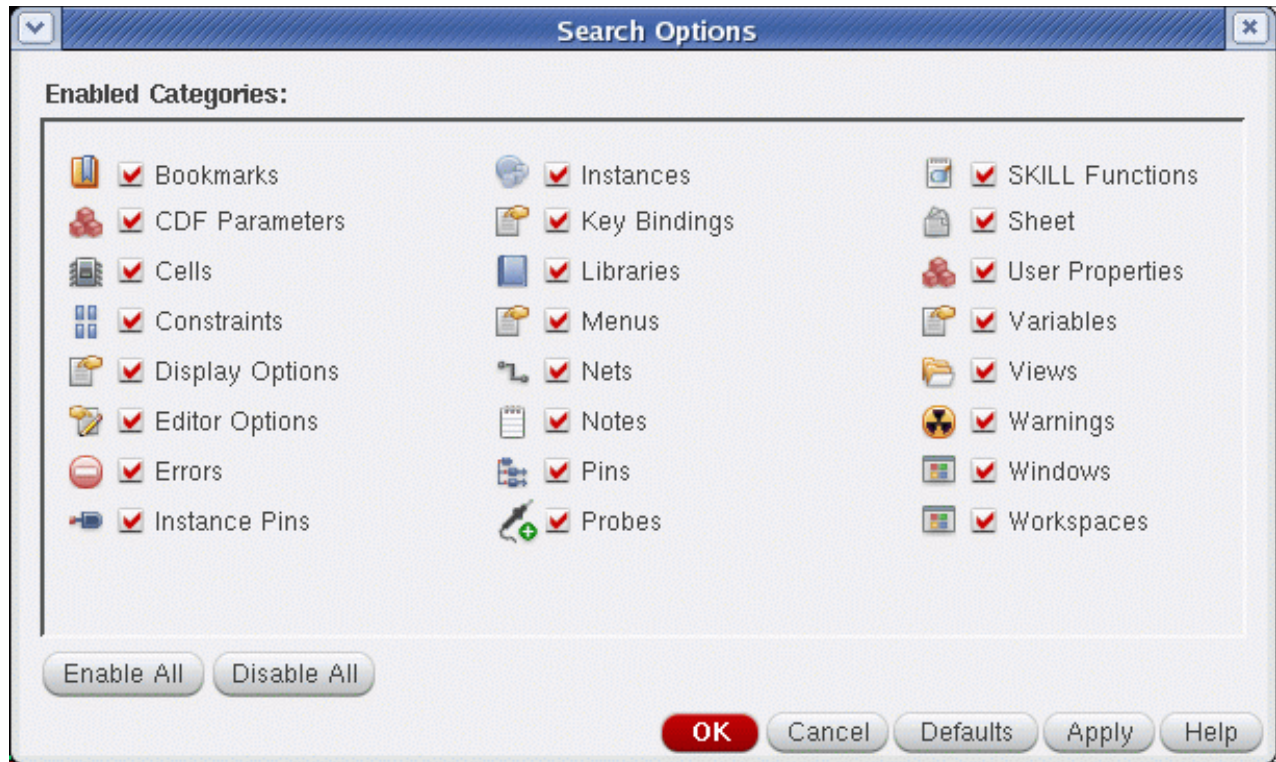


Figure 14-3 Full List of Search Options (Plug-Ins)

Note: You can choose to *Enable All* or *Disable All* search categories (plug-ins).

Advanced Search

The *Advanced Search* pull-down is located to the far right of [The Search Toolbar](#) and [The Search Assistant Pane](#), and provides the following commonly used search options to refine your search results.

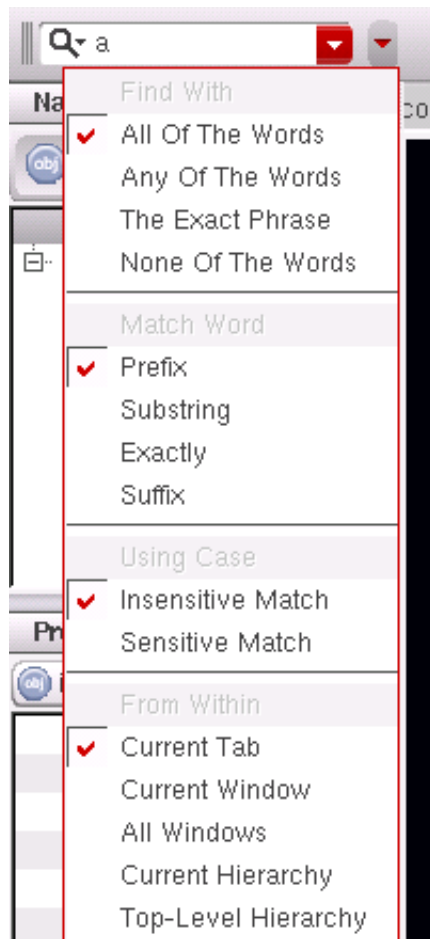


Figure 14-4 The Advanced Search Pull-Down

- The options under the *Find With* section are used to specify the default matching operators AND (*All Of The Words*), OR (*Any Of The Words*), EXACTLY (*The Exact Phrase*), and NOT (*None Of The Words*).

See also [Query](#) in the [Search Terminology](#) section.

- The options under the *Match Word* section let you specify a regular query token that must be matched in the current content search data to display a successful result. The options available here are *Prefix*, *Substring*, *Exactly*, or *Suffix*.

See also [Word](#) in the [Search Terminology](#) section.

- The options under *Using Case* let you determine whether search results must be case sensitive (*Sensitive Match*) or whether any text case is acceptable (*Insensitive Match*).

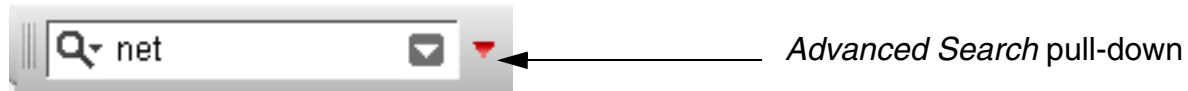
For example, you may only want to display search results that have “R1” included in the object name, but not “r1”. In this case, you should ensure the *Sensitive Match* option is selected.

- The options under *From Within* allow you to specify the search *scope*. By default, a search operation will be contained within the *Current Tab*, however you can extend the scope to cover the *Current window*, *All windows*, the *Current hierarchy*, or the *Top-level hierarchy*. The search scope options are listed in ascending order of the time anticipated to perform a search.

Note: If an application does not provide plug-ins that can run on multiple scopes, the additional scope options will not be available. For example, options to search the *Current Hierarchy* and the *Top-Level Hierarchy* are not currently available in the *Virtuoso Layout Suite*.

See also [Displaying The Search Scope](#) on page 815.

Advanced Search Use Model Example



To locate all devices in a design hierarchy click the *Advanced Search* pull-down, on the *Search* toolbar, and then select *Top-Level Hierarchy* from the *From Within* section.

The “cells” returned in the query results will detail all of the devices used in the design (sorted by name or library). Each device will also have a *Master Status* column which can be used to identify unbound devices.

Note: To locate all blocks which use a particular device, you can use a hierarchical instances query. For example, “`nmos instance`”.

Search History/Results Pull-Down

The pull-down option on the immediate right of the Search Query Entry field has a dual purpose.

- When there is **no** search query specified in the Search Query Entry field, clicking on the pull-down will display a recent *Search History* list where you can immediately recall the results of any recent searches and reinstate any search options selected when the search criteria was last applied.

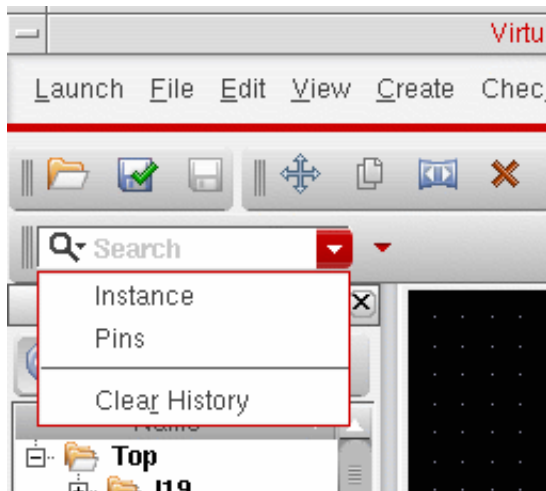


Figure 14-5 Search History showing previous searches on “Pins” and “Instance”

Note: Search query history is not maintained between sessions.

- Alternatively, as search queries are entered into the Search Query Entry field (or recalled from *Search History*), the pull-down becomes a *Search Results* pull-down list, displaying search results dynamically.

Note: The *Search History* option changes to a *Clear Query* option (X) after a search has been run.



Figure 14-6 Clear Query replaces Search History after a search has completed

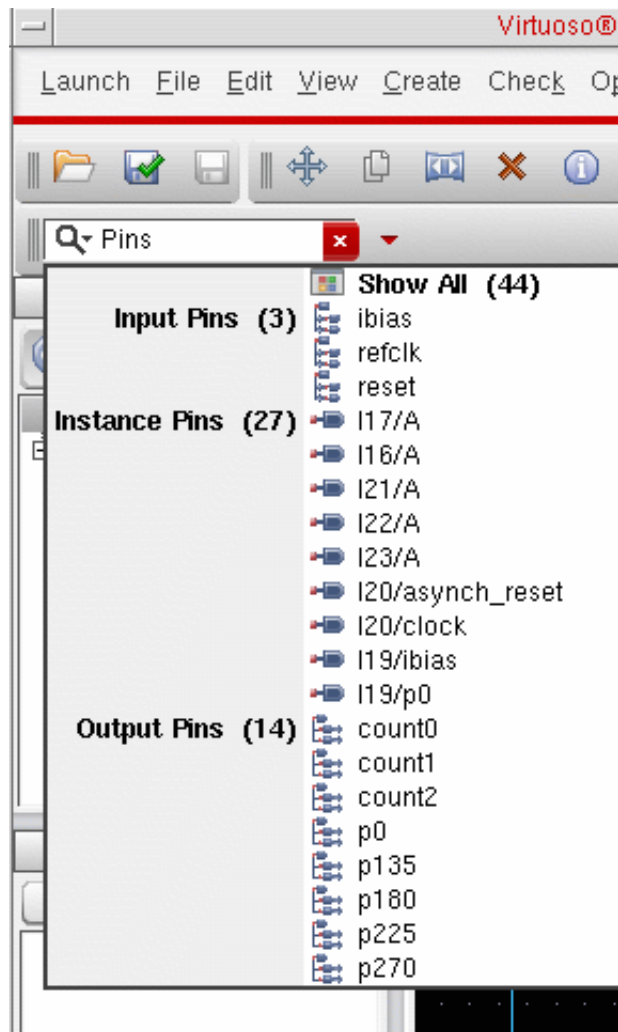


Figure 14-7 Search History results for a search on “Pins”

Note: The *Search Results* pull-down can display up to 30 hits, with at least one search result from each matched search category on display. See also [Ranking Hits](#) and [Displaying The Search Scope](#).

You can extend initial search results by performing one of the following:

- ☐ Display *full search results* by selecting the *Show All* option at the top of the results pull-down. This will display all search results in [The Search Assistant Pane](#) where you can investigate them further.
- ☐ Display category specific search results by clicking on a category heading in the search results, for example *Instances* or *Input Pins*. This will display the full search

Virtuoso Schematic Editor User Guide

The Search Assistant and Toolbar

results for the selected category only in The Search Assistant Pane where you can investigate them further.

You can dismiss search history/results by clicking elsewhere on the session window, or selecting the `ESC` key.

Note: Because search results only reflect a *snapshot*, object selection is one-directional. That is, cross-selection of results works only in the design canvas.

Ranking Hits

The category with the highest rank score will appear at the top of the results list. You should note however that ranking only influences the UI presentation, as all hits will remain accessible.

Search Results Pull-Down ToolTip Information

Floating your cursor over a specific result in the *Search Results* pull-down will raise a ToolTip which details full information on the current search result.

For example, if you have an *Instance Pin* highlighted, the ToolTip will display *Name*, *Instance*, *Net* and *Direction* property information.

Selecting a Search Result

Selection of an object listed in the displayed *Search Results* will cause that item to be selected in the current session window view, highlighted in the Navigator Assistant pane and it's properties also displayed in the Property Editor Assistant pane.

Searching Further on Search Results

A right-click over a search result allows you to perform a search on that particular result (for example, "*Search For: "vdd!"*"). Clicking on this option will perform a search within a search, and the search results list will update to only display those search results that include "*vdd!"*". All other search results will be dismissed.

Search Toolbar Context-Menus

Right-clicking over the following *Search* toolbar UI components will display a related context-menu.

- A right-click over the Search Query Entry field will display a context-menu which will enable you to perform a variety of text editing tasks including *Cut*, *Copy* and *Paste*.
- A right-click over any object listed in the search results will offer options to:
 - *Expand All* or *Collapse All* of the object category results currently listed.
 - *Open Owner in Current Tab*, *Open Owner in New Tab*, or *Open Owner in New Window* options, that allow you to quick access to the owner cellview.

Note: The above options will only be displayed for plug-ins that support search scopes.

- A right-click over a displayed search result will enable you to perform a further search on that particular result (for example, “*Search For: “vdd!”*”). Clicking on this will perform a search within a search and the search results list will update to only display search results that include “*vdd!*”. All other search results will be dismissed.
 - A right-click over a particular category result may also provide context-menu options related to that object selection. For example, a right-click over a *Libraries* or *Cells* category result will provide options to *Add Instance* or invoke the *Library Manager*, and a right-click over a *Note* category result will provide an *Edit Note* option.

Note: Other examples would be right-clicking over a SKILL function result and accessing the SKILL Finder Window, or right-clicking over a key binding and accessing the Bindkey Editor by selecting the *Edit Key Bindings* option.

Virtuoso Schematic Editor User Guide

The Search Assistant and Toolbar

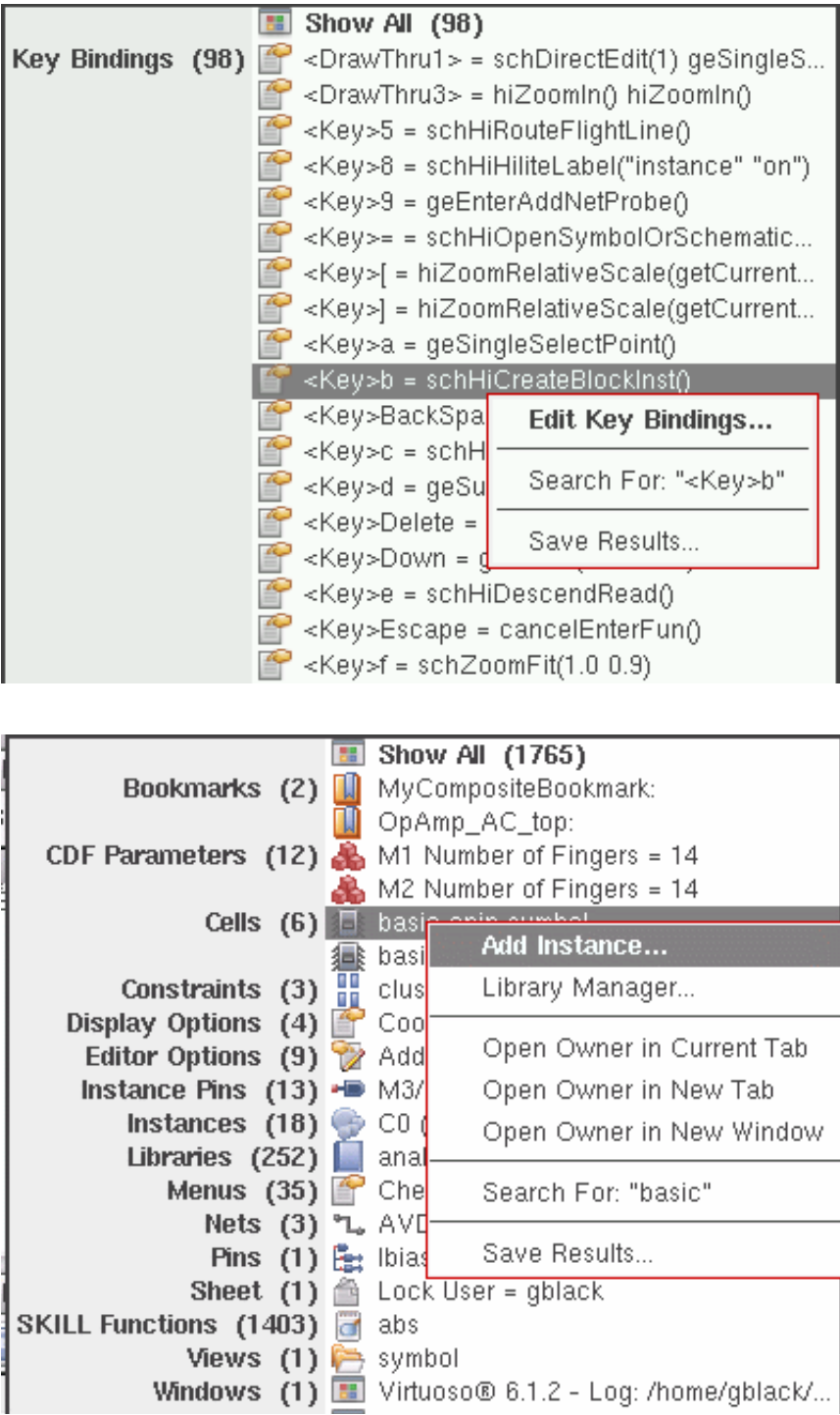


Figure 14-8 Context-menu Examples for Cells or Key Bindings Results

Saving Search Results

You can save search results by right-clicking over any displayed results and selecting the *Save Results* option from the search context-menu.

Selecting *Save Results* will display the Save Search Results Form from where you can save search results as a `.csv` file.

This file can then be re-opened in, for example, a text editor or a spreadsheet application.

Note: You can also save search results using the Search Assistant Pane Context-Menus.

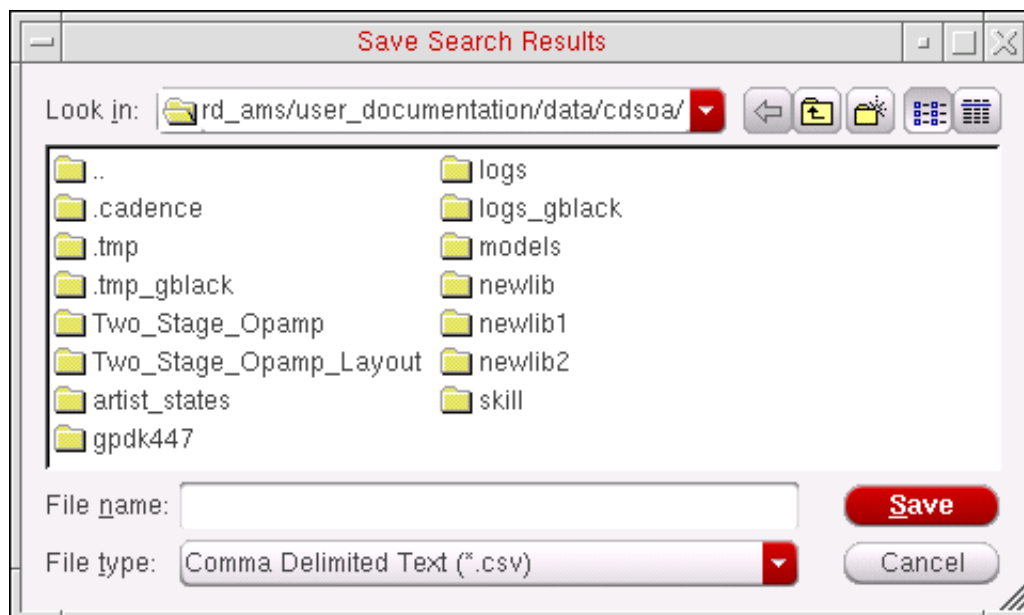


Figure 14-9 Save Search Results Form

The Search Assistant Pane

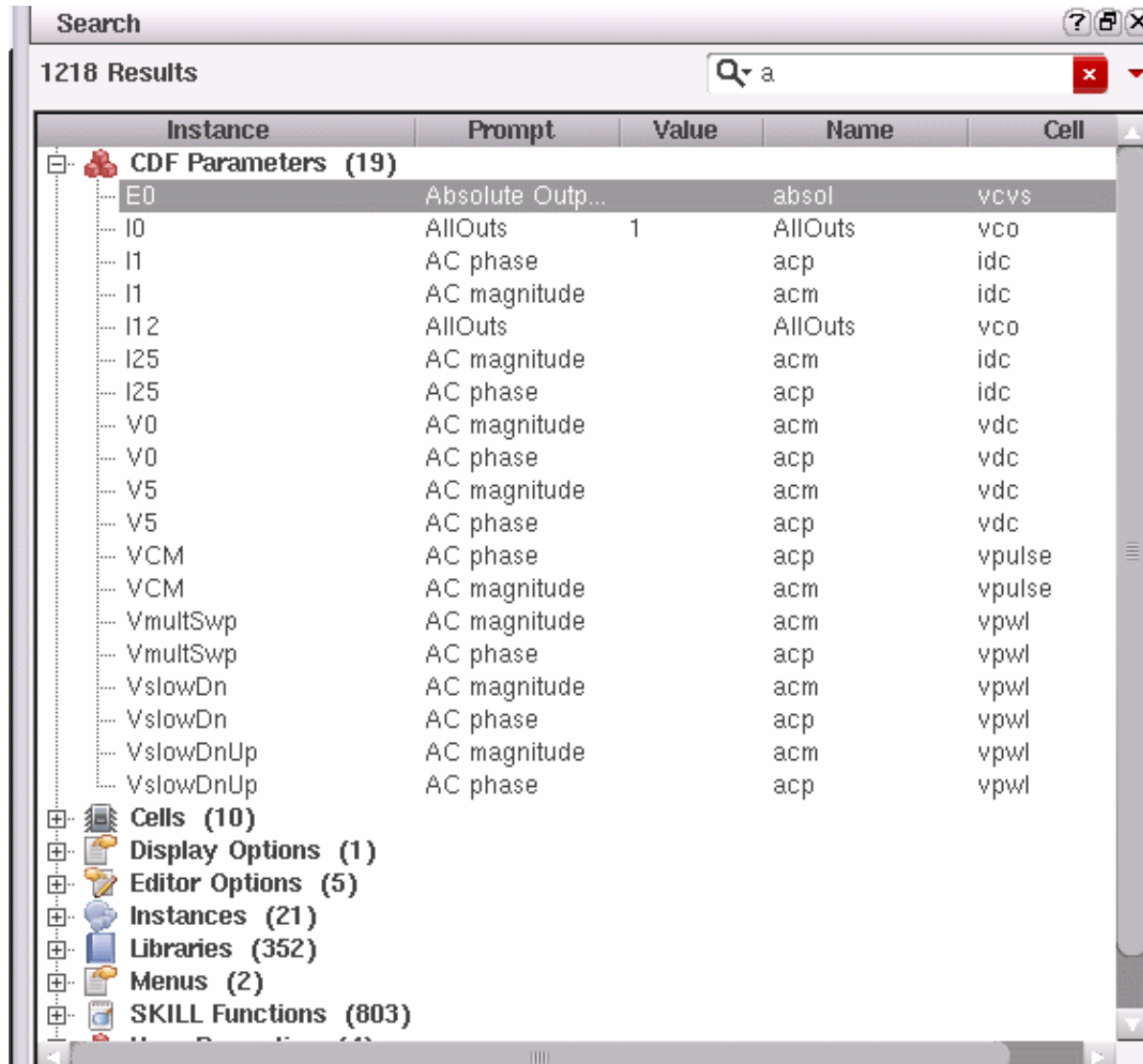


Figure 14-10 The Search assistant pane

The interactive *Search* assistant pane is a dockable/undockable pane that offers additional search viewing information to that offered by [The Search Toolbar](#).

The *Search* assistant pane becomes active in either of the following cases:

1. By choosing to use the *Search* assistant rather than [The Search Toolbar](#) to perform a design search, or

Virtuoso Schematic Editor User Guide

The Search Assistant and Toolbar

2. The *Search* assistant pane will also become active when a category title is selected in the *Search* toolbar search results or if you click the *Select All* option from the *Search Results* pull-down.

Note: The total number of search hits is displayed in the top-left corner of the *Search* assistant pane. Floating your cursor over this number will also display how many potential result targets there were and how long it took to perform the current search. For example, “238 hits from 1700 targets, found in 0.29 seconds”. A maximum of 1023 search hits can be displayed at any given time. If the number of search hits exceeds 1023, refine your search conditions to display all results.

The *Search* assistant pane comprises of the following UI components that are also available with The Search Toolbar:

- Search Query Entry
- Search Categories Pull-Down
- Advanced Search

The *Advanced Search* feature in the *Search* assistant pane displays available advanced search options using four pull-down menus (*Find With*, *Match Word*, *Using Case* and *From Within*), as opposed to the one pull-down used on The Search Toolbar.

Note: The *Advanced Search* pull-down menus are not mutually exclusive.

Clicking again on the *Advanced Search* pull-down will return you to *Basic Search* entry mode.

Virtuoso Schematic Editor User Guide

The Search Assistant and Toolbar

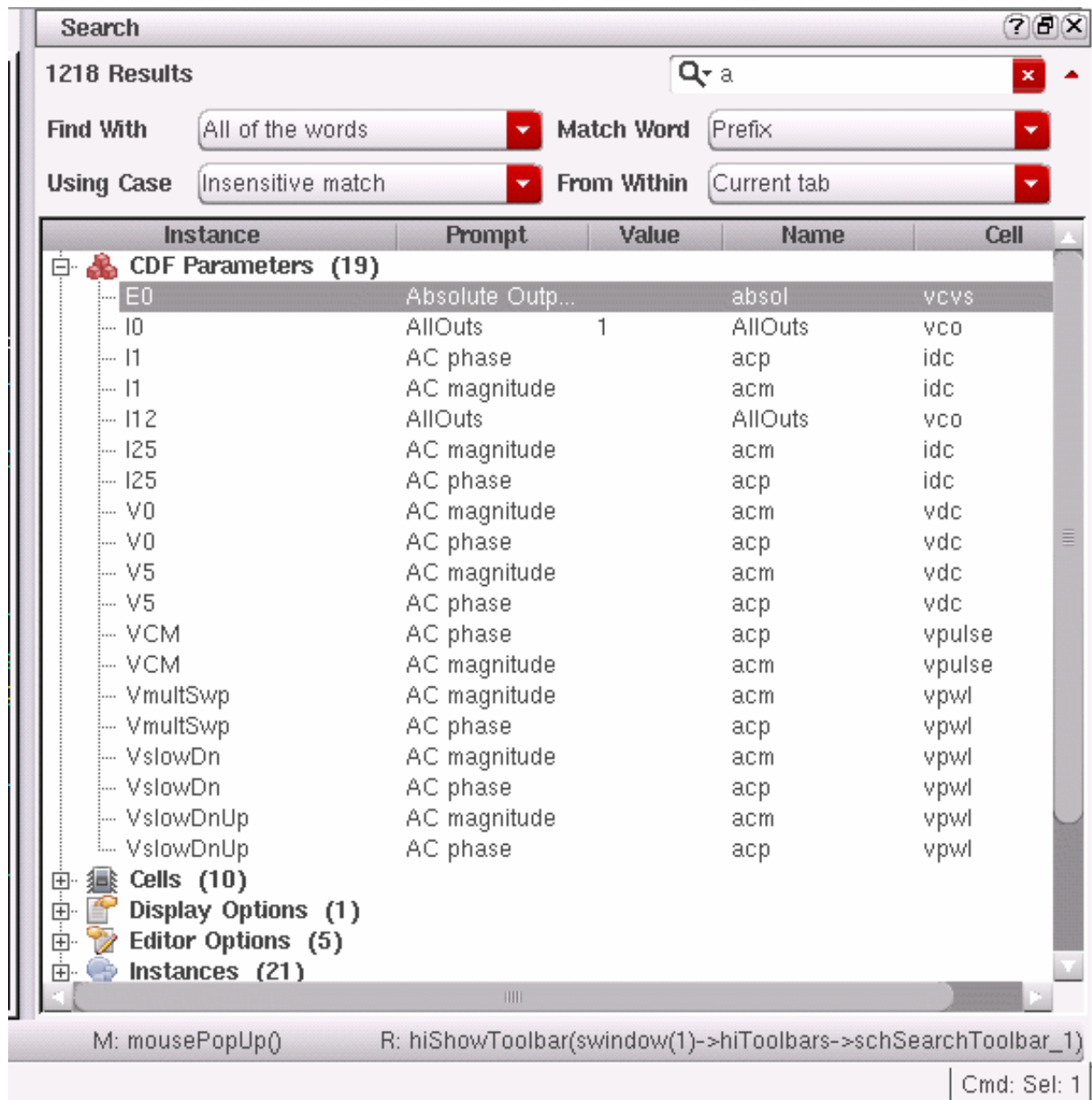


Figure 14-11 Search assistant pane with Advanced Search options displayed

The *Search* assistant pane also has the following unique UI components:

- Search Results Table
- Search Assistant Pane Context-Menus

Search Results Table

As mentioned, the *Search Results Table* will display search results if you choose to use [The Search Assistant Pane](#) rather than [The Search Toolbar](#) to perform a design search, or if you choose a [category](#) title or click the *Select All* option from the *Search Results* pull-down.

The *Search Results Table* is a fully populated version of the *Search Results* pull-down available from [The Search Toolbar](#). It contains a number of features that enable you to better interpret results.

Name	Library	Cell	View	Owner
Cells (2)				
basic	ipin	symbol	bound	opamp090.full_diff...
basic	opin	symbol	bound	opamp090.full_diff...
Editor Options (4)				
Instances (6)				
PIN0	basic	ipin	symbol	opamp090.full_diff...
PIN1	basic	ipin	symbol	opamp090.full_diff...
PIN3	basic	ipin	symbol	opamp090.full_diff...
PIN4	basic	opin	symbol	opamp090.full_diff...
PIN5	basic	opin	symbol	opamp090.full_diff...
PIN11	basic	ipin	symbol	opamp090.full_diff...
Libraries (40)				
Menus (6)				
SKILL Functions (195)				
User Properties (6)				

Figure 14-12 Search Results Table

- Table columns can be sorted alpha-numerically by clicking on the table header.
- Table columns can be resized by clicking and dragging the column separators.
- Single clicking on an object category will expand or collapse its contents.
- Context-menus are used to provide access to additional category specific options and additional search options (see also [Search Assistant Pane Context-Menus](#)).
- ToolTips are used to detail full information on the current search result (see also [Search Results Table ToolTip Information](#)).

Virtuoso Schematic Editor User Guide

The Search Assistant and Toolbar

Column heading titles are updated depending on the object category result currently selected. For example, if you have an *Instance Pin* result selected the column titles will display related property details on *Name*, *Instance*, *Net* and *Direction*. However, if you have a *Net* result selected the column titles will be labeled *No. Connections* and *Instance Pins*.

If currently available, object selection in the Search Results Table (and the *Search Results* pull-down) will also be reflected in the Navigator Assistant pane and the Property Editor Assistant pane, with the *Navigator* showing the selected object's location in the current hierarchy and the *Property Editor* detailing associated object properties.

Likewise, if you search on constraints using the *Search* assistant, selecting one or more constraint results in the *Search* assistant will cause the constraints to be automatically selected in the Constraint Manager. Double-clicking on these results will also run the default action of selecting all constraint members and having these selections also cross-selected in the *Navigator* and *Property Editor* assistants.

Note: As search results only reflect a *snapshot*, object selection is one-directional. That is, cross-selection of results works only in the design canvas.

Virtuoso Schematic Editor User Guide

The Search Assistant and Toolbar

Displaying The Search Scope

For any given search result, the *owner* is usually a cellview in the form `lib:cell:view`. As this information can be expansive it is detailed in a separate *Owner* column, displaying those results found in the current window with a white background and those found *out-of-context* in a gray background. The *Owner* column can then be sorted to group results by their source cellview rather than by (the default) object name.

Note: Owner information is also displayed in the ToolTip information.

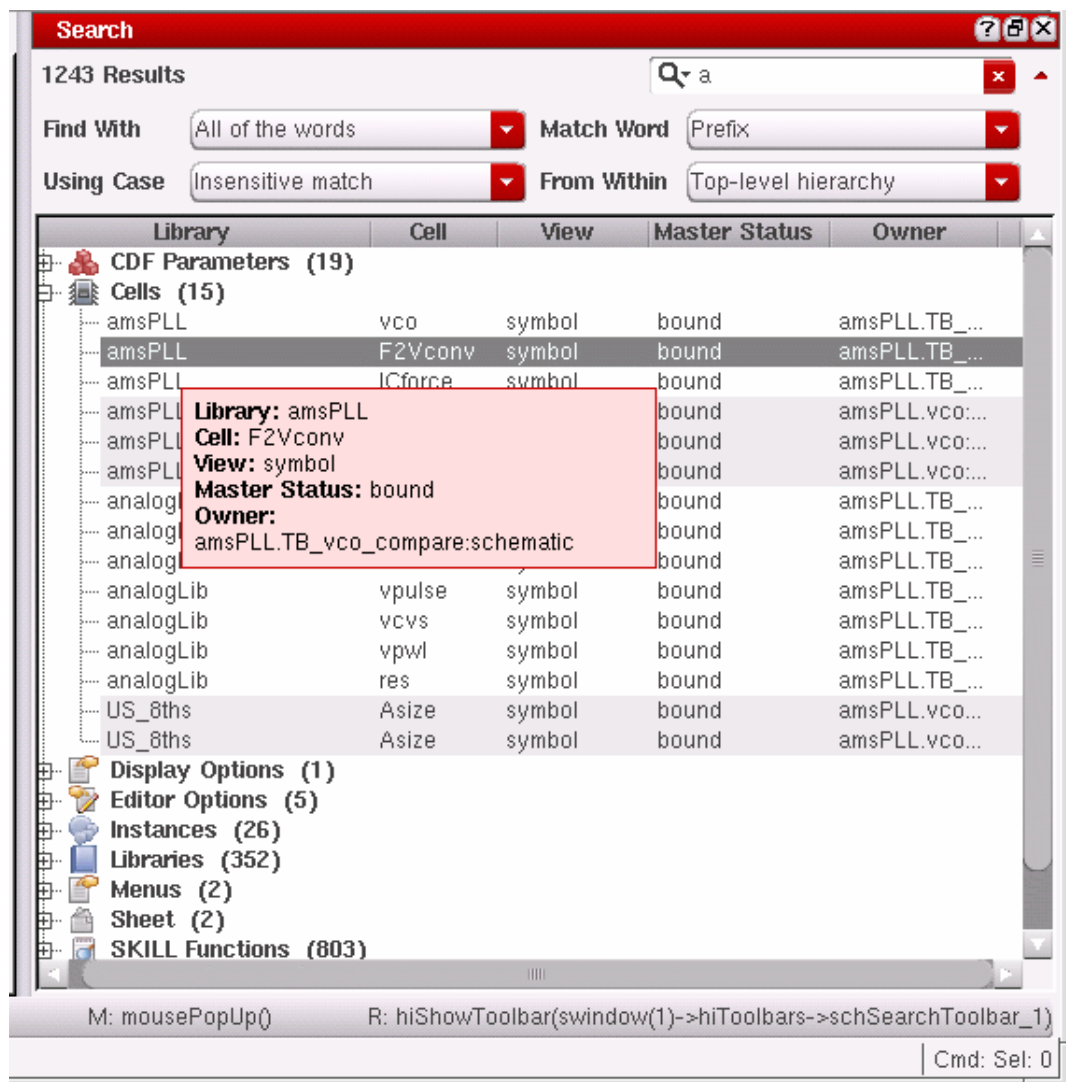


Figure 14-13 The Search Scope is shown in the Owner column

Search Assistant Pane Context-Menus

Right-clicking over the following *Search* assistant pane UI components will display a related context-menu.

- A right-click over the Search Query Entry field will display a context-menu which will enable you to perform a variety of text editing tasks including *Cut*, *Copy* and *Paste*.
- A right-click over any object listed in the Search Results Table will offer options to:
 - ❑ *Expand All* or *Collapse All* of the object category results currently listed in the table.
 - ❑ *Open Owner in Current Tab*, *Open Owner in New Tab*, or *Open Owner in New Window*, allowing you to quick access to the owner cellview.
 - ❑ *Select All Below* allows you to select the contents of the current search results directory, whether it is expanded or folded. When selected, the command will select all the search result rows (visible if the directory is expanded). Selected search results will then be reflected in the *Navigator*, *Property Editor*, and design canvas.

Note: Some of these options will only be displayed for plug-ins that support search scopes.

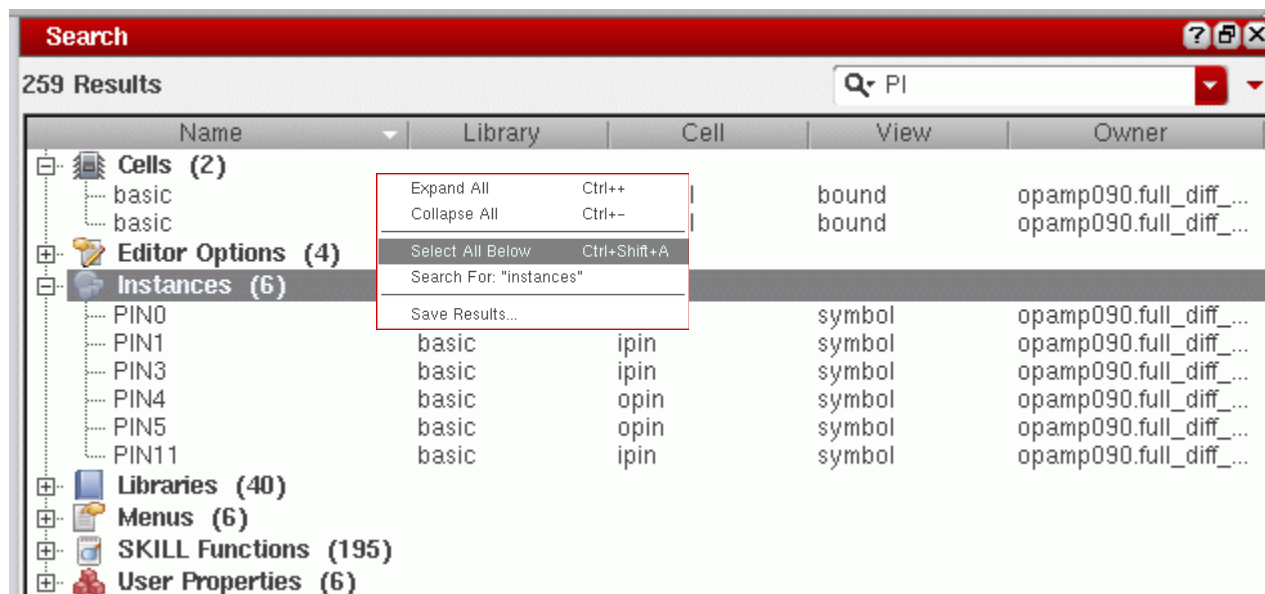


Figure 14-14 Search assistant pane context-menu

- ❑ The Search Results Table context-menu also enables you to perform a search on specific search results (for example, "*Search For: "instances"*"). Clicking on this

Virtuoso Schematic Editor User Guide

The Search Assistant and Toolbar

option will perform a search within a search, and the results table will update to only display those search results that include “*instances*”. All other search results will be dismissed.

- ❑ A right-click over a particular category result may also provide options related to that object selection. For example, a right-click over a *Library* category result will provide you with options to *Add Instance* or invoke the *Library Manager*, and a right-click over a *Note* category result will provide with an *Edit Note* option.
- Right-clicking over a column header in the Search Results Table allows you to show or hide any columns currently on display or hidden. Check or uncheck the column name as appropriate. This context-menu also provides *Expand All* or *Collapse All* object category options.
- Right-clicking over the Search Results Table at any time will display a *Save Results* option. For more information on saving search results see Saving Search Results.

Search Results Table ToolTip Information

Floating your cursor over a specific search result in the Search Results Table will raise a ToolTip which details full information on the current search result. This is especially useful when data is obscured.

For example, if you have an *Instance* highlighted, the ToolTip will display *Name*, *Library*, *Cell*, *View* and *Owner* information.

Virtuoso Schematic Editor User Guide

The Search Assistant and Toolbar

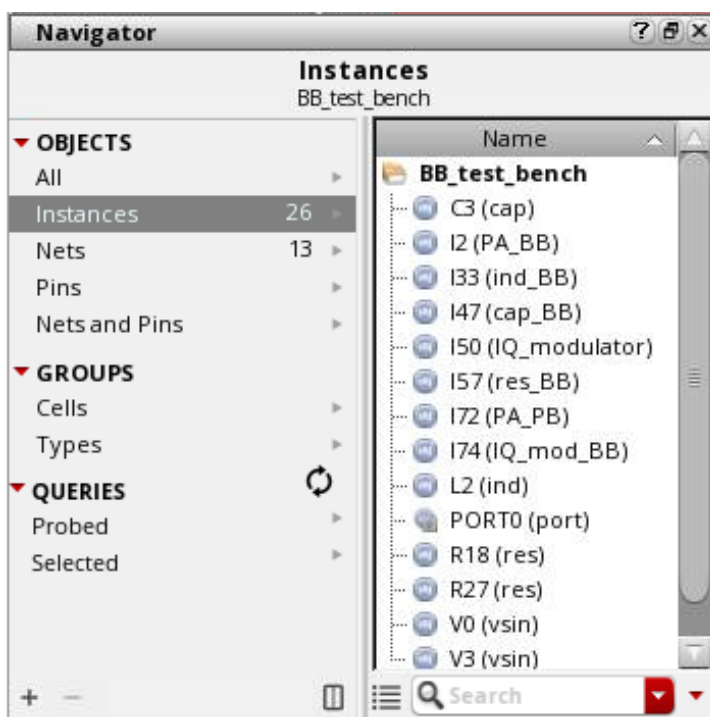
The Navigator Assistant

This chapter contains the following topics on the *Navigator*:

- [Introducing the Navigator](#)
- [Accessing the Navigator](#)
- [Hiding the Navigator](#)
- [Navigator Keyboard Shortcuts](#)
- [The Navigator User Interface](#)
- [Selection and Display in the Navigator](#)
- [The Navigator Options Form](#)

Introducing the Navigator

The *Navigator* is a dockable/undockable assistant pane that provides facilities to view objects across the design hierarchy using a tree representation.



The *Navigator Tree* is a hierarchical expandable/collapsible navigation system, which is used to display large datasets, while utilizing icons to depict status information.

The *Navigator* also provides options to selectively view instances, pins, nets, FigGroups and categories, and allows you to sort this information by columns, for example, by *Cell Name* or *View Name*.

The *Navigator* displays the information about the complete design in the *Summary* and *Details* panes. It provides various categories, such as *Objects*, *Groups*, *Design Intent*, *Queries*, *Miscellaneous*, and so on. The categories consist of predefined and user-defined sets.

Predefined sets include *Nets*, *Pins*, and so on. The arrangement of various objects in these sets allows you to view and search through them conveniently.

User-defined sets can be created, saved, and reused based on a specific requirement. For example, you can choose to display only the unrouted nets in the *Navigator*, which represent a significantly smaller number as opposed to the total nets in the design. This is more effective

Virtuoso Schematic Editor User Guide

The Navigator Assistant

than using search in the *Navigator* because selecting a specific number of nets from numerous nets in the design consumes considerable time and memory. The user-defined sets can be used in various ways, such as grouping the objects and performing multiple operations on them. Grouping the objects makes it convenient to use them repeatedly. It also helps in partitioning the design so that smaller parts can be searched faster, and grouped sets can also be refreshed together. You can use these sets while working on a small subset of the nets in the design containing a large number of nets. With the tighter integration of large digital designs, it is required to efficiently store, retrieve, and display the objects.

You can filter and control the type of object information that the *Navigator* displays in the *Navigator Tree*. For more information, see [Controlling Object Display in the Navigator with Search](#) and [Controlling Object Display in the Navigator with Sets](#).

Cross-selection (see also [Cross Selecting Between the Navigator and the Current Cellview](#)) is also available from/to the *Navigator*, where selecting, for example, an instance in the *Navigator* will cause the same instance to be highlighted in the active cellview tab and in other assistants, such as the [Property Editor Assistant](#) and the [Constraint Manager Assistant](#), if currently in use.

You can undock the *Navigator* from the main canvas area, and relocate it as appropriate. It also has resizable columns that allow you to make the best use of the available screen space.

You can only have one *Navigator* per session window, and this ties the *Navigator* to that session window. That is, you cannot open multiple session windows, undock the *Navigator*, and then be able to use it in all session windows.

In summary, the main purposes of the *Navigator* are:

1. To enable the display of a fully unfolded hierarchy (occurrence tree), which serves as a design navigator.
2. To display the sets and the number of objects within them.
3. To provide a convenient method of selecting sets, expanding, viewing, and interacting with the objects in it, such as instances, nets, and pins across a schematic hierarchy.
4. To provide commands and menu entries or controls for editing or updating the user-defined set and its contents.
5. To save the user-defined sets without the need to create full internal storage for the objects in the transient sets.



Video

For a video overview of the Navigator assistant, see [Using the Enhanced Navigator Assistant](#) on Cadence Online Support.

Accessing the Navigator

To access the *Navigator* in Schematic Editor, do one of the following:

- Select *Window – Assistants – Navigator*.
- Right-click the main Schematics menu or toolbar area, and select *Navigator*.
- Access one of the following Schematic workspaces by selecting *Window – Workspaces* or by choosing one of them from the *Workspace Configuration* toolbar:
 - ☐ *Basic*
 - ☐ *Constraints*
 - ☐ *Explore*

The Navigator is available as a default assistant in these Schematic workspaces.

Note: The *Navigator* is also available in the Virtuoso Layout Suite.

Once the *Navigator* is selected, the schematic editor embeds the *Navigator* as a docked assistant pane within the current session window. By default, the *Navigator* will be positioned in the upper left area of the session window.

For more information on workspaces in Schematics, see [Workspaces and Assistants in the Schematic Editor](#).

For more information on customizing your own workspaces, see [Working With Workspaces](#) in the *Virtuoso Design Environment User Guide*.

Hiding the Navigator

To hide the *Navigator* in the current session window:

- Right-click in the Schematics menu or toolbar area and deselect the *Navigator* option.
- Change to a workspace configuration that does not include the *Navigator* as part of its default configuration.
- Click the *Hide* button (X) in the *Navigator* assistant pane title bar.

Navigator Keyboard Shortcuts

You can use the following keyboard shortcuts to aid your interaction with the *Navigator*:

Keyboard Shortcut	Action
Up Arrow	Move up the tree by one row/object.
Down Arrow	Move down the tree by one row/object.
Left Arrow	Collapse current object and contents.
Right Arrow	Expand current object and contents.
Home/Page Up	Move to top of tree.
End/Page Down	Move to bottom of tree.

The following shortcut options are also available (the commands can be accessed through the *Navigator* context-menu):

Ctrl ++	Expand all objects below current object.
Ctrl +-	Collapse all objects below current object.

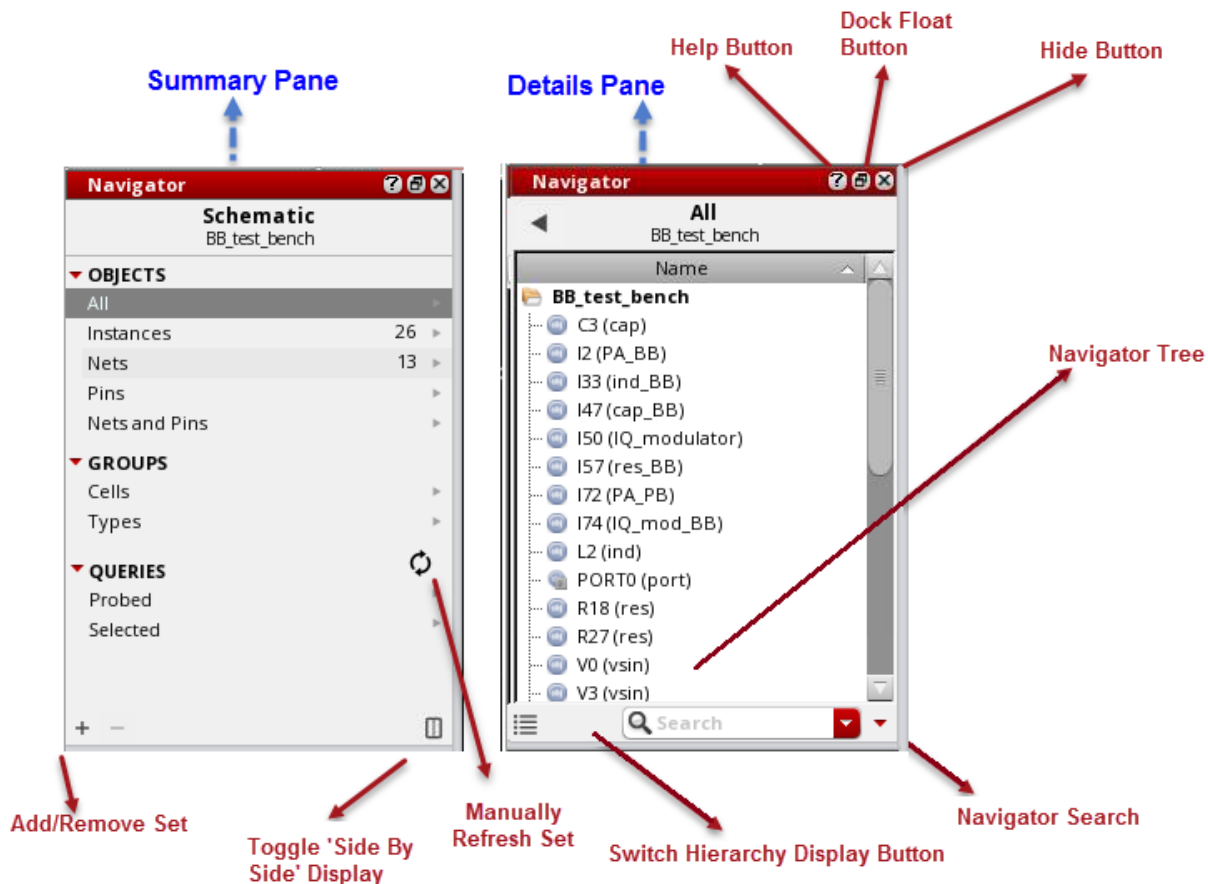
The Navigator User Interface

The Navigator GUI consists of a *Summary* pane and a *Details* pane.

The sets are displayed in the *Summary* pane under various categories. The set contents are displayed in the *Details* pane.

When a cellview is opened for the first time in a Virtuoso session, the Summary pane is displayed in the Navigator. You can control whether the Summary pane is displayed by default by using the `showSummaryPage` environment variable. When a cellview is subsequently opened, the Navigator displays the last set selected.

The Navigator comprises of the following user interface (UI) elements:



- The *Navigator* title bar buttons allow you to:
 - ❑ Obtain *Help* information on the *Navigator*
 - ❑ *Float* (undock) the *Navigator* (select again to *Dock* the *Navigator*).

Virtuoso Schematic Editor User Guide

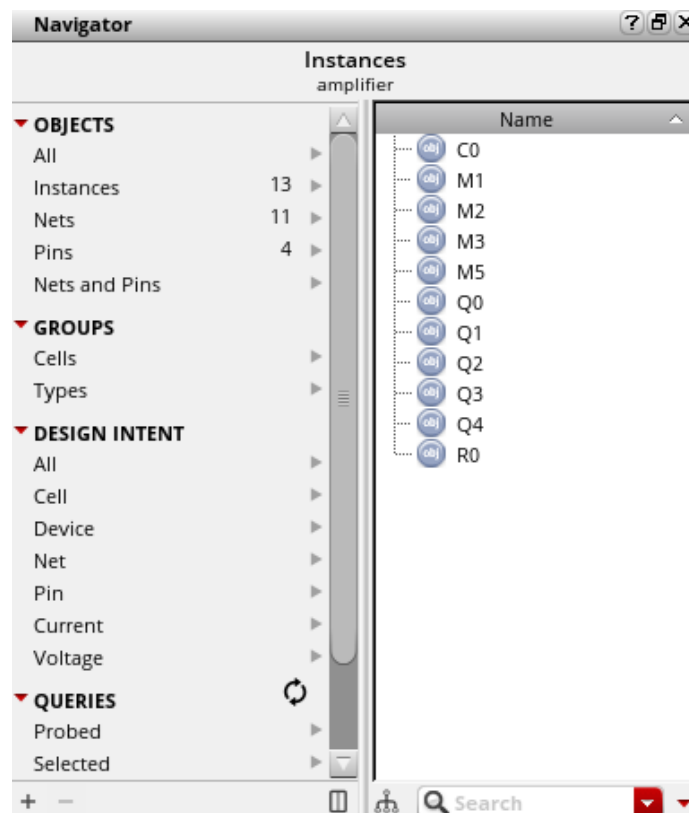
The Navigator Assistant

❑ *Hide the Navigator.*

- The *Summary* pane shows the name of each set and the number of objects within the set, if any. When you click a set, the other assistants and the canvas display the related information and the context menus display operations that can be applied on the set.
- The *Details* pane shows the contents of the selected set. The predefined sets are not cellview-specific but user-defined sets are cellview-specific; therefore, only the ones for the current cellview are shown. *All*, one of the sets in the *Summary* pane, shows all the object types supported in the *Navigator* and this set allows you to display the complete design. You can choose to display only the specific parts of the design using the user-defined sets. To select which columns are displayed, right-click the header and select the columns that you want to display. This setting is persistent in the session window and across Virtuoso sessions.
- The *Navigator Search* allows you to filter design objects displayed in the *Navigator Tree*.

For more information see [Controlling Object Display in the Navigator with Search](#).

- *Toggle 'Side By Side' Display* is used to arrange the sets and their contents sideways as shown here.



Virtuoso Schematic Editor User Guide

The Navigator Assistant

- *Switch Hierarchy Display* button is used to control whether to display the design hierarchy along with the members of the set. This setting is persistent across Virtuoso sessions. If the hierarchy is displayed, double-clicking an item descends in the hierarchy. Also, searching takes into consideration the hierarchical content of the design.
- *Add Set* or *Remove Set* enables you to add a new set or delete an existing set, respectively. Click *Add Set* to open the New set dialog box, and select the category under which the new user-defined set is to be created, as shown below.

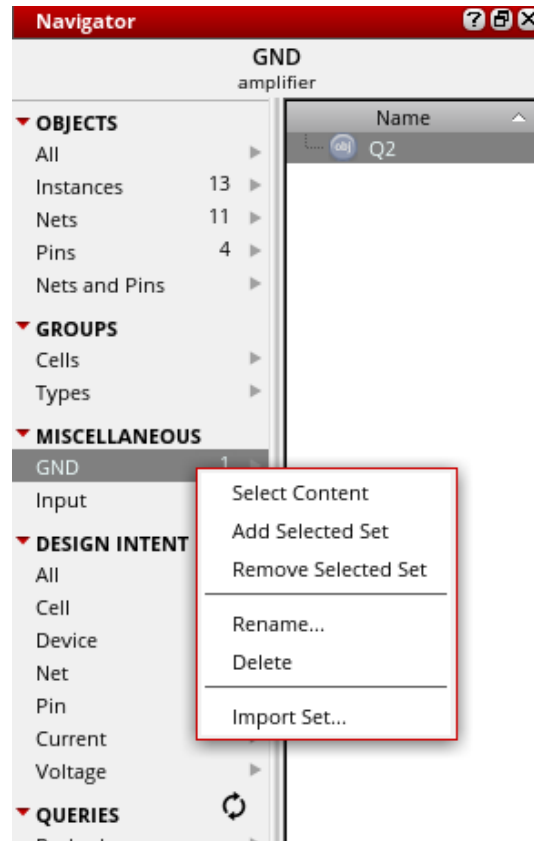


After the new set is created, you can use one of the following methods to add or remove the objects from the set:

Virtuoso Schematic Editor User Guide

The Navigator Assistant

- ❑ Right-click in the *Summary* pane to view the following context menu and select *Add Selected Set*.



- *Select Content*: Selects all the members in a set.
- *Add Selected Set*: Adds the objects/sets, selected in Navigator, Schematic Editor, or Property Editor, to the current user-defined set.
- *Remove Selected Set*: Removes the objects/sets, selected in Navigator, Schematic Editor, or Property Editor.
- *Rename*: Renames the set.



- *Delete*: Deletes the set.

Virtuoso Schematic Editor User Guide

The Navigator Assistant

- *Import Set*: Imports the set from another cellview. Imported sets get added to the same category as the category of the source set.

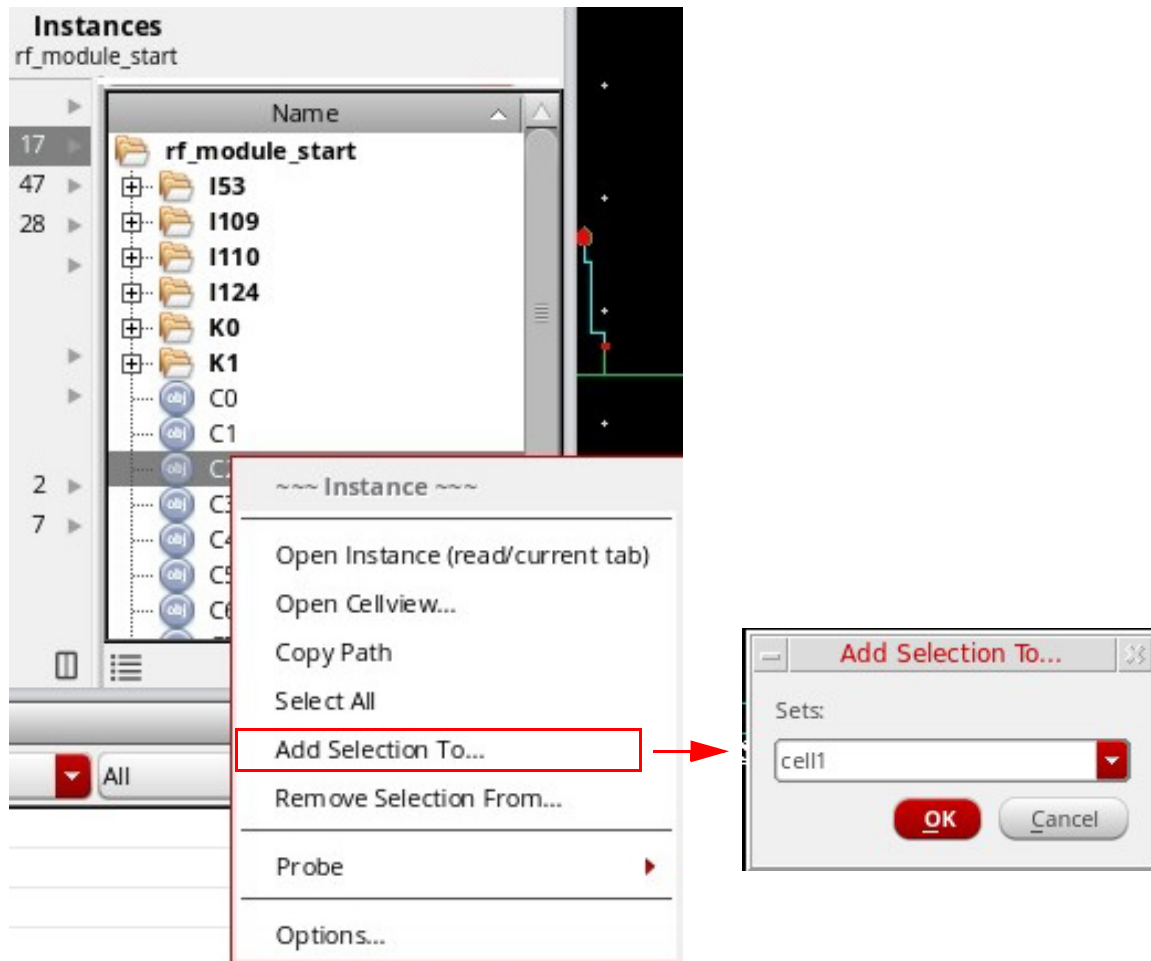


- Click *Add Selection To* in the context menu of the *Details* pane to view the following dialog box. You can choose from the existing user-defined sets in the *Sets*

Virtuoso Schematic Editor User Guide

The Navigator Assistant

drop-down list box or specify a new set in the text field where the selected objects are required to be added.



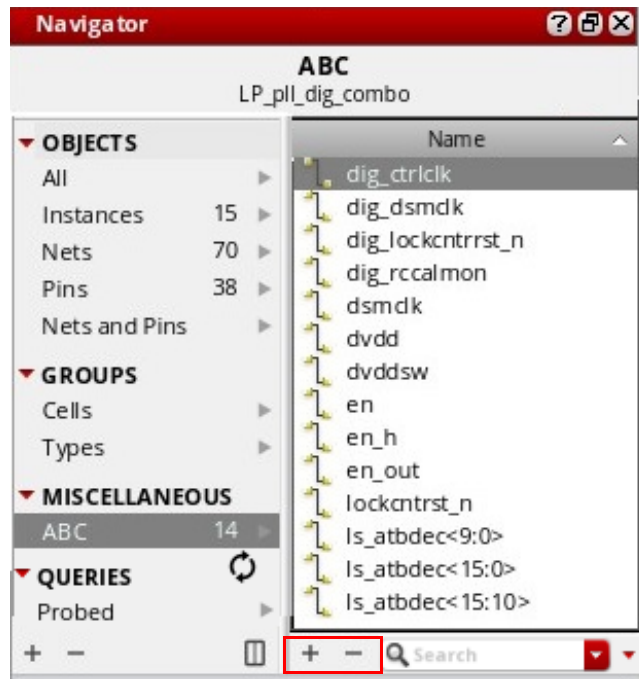
- ❑ Click *Remove Selection From* in the context menu of the Details pane to remove the selected objects from the existing user-defined sets.



Virtuoso Schematic Editor User Guide

The Navigator Assistant

- *Select All*: Selects all the members in a set.
- Select the objects and click + to add them to a set as shown below.



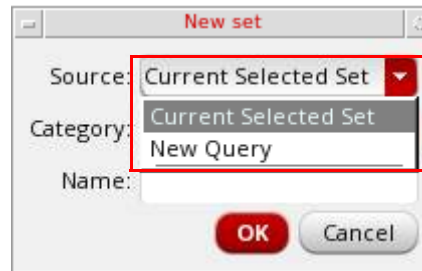
- *Add Set* or *Remove Set* enables you to add a new set or delete an existing set, respectively. Click *Add Set* to open the New set dialog box, and select the category under which the new user-defined set is to be created, as shown below.



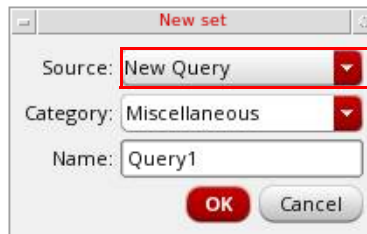
Virtuoso Schematic Editor User Guide

The Navigator Assistant

- *Add Set* and *Remove Set* enables you to add or delete a set or query, respectively. Use the *Add Set* to open the New set dialog box, and select the source as *Current Selected Set* or *New Query* to add queries using the New set form, as shown below.



To create a new query, select *New Query* as the source.

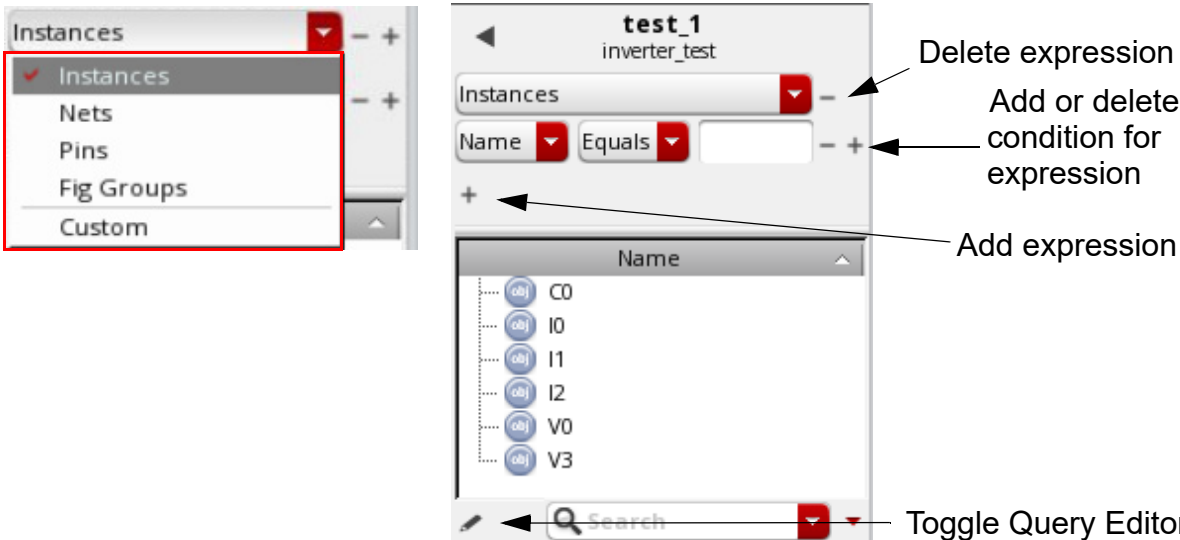


You can use queries in the Navigator assistant in many ways:

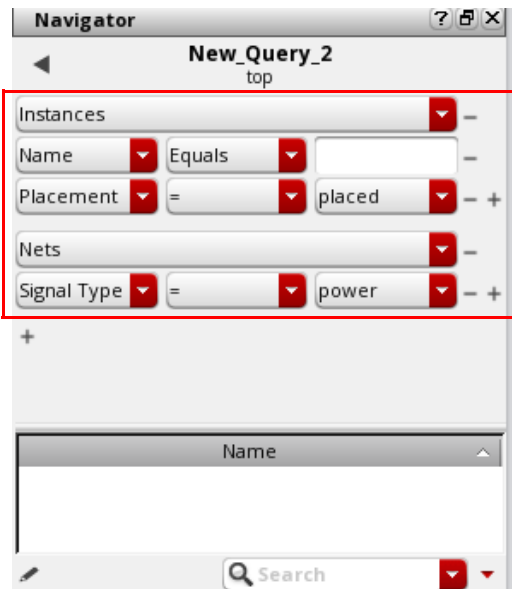
- ❑ To group design objects, such as nets to be routed together, into categories.
- ❑ To check design work stages, such as unplaced instances.
- ❑ To create a group of objects to be included in an operation, such as a set of clock nets when defining net priorities.
- ❑ To maintain 'To Do' lists of objects that require an operation to be carried out, such as clock nets without net priority when defining net priorities.

The Navigator Assistant

You can select one of the available data sources, *Instances*, *Nets*, or *Pins*, from the drop-down list.



You can customize the selection by specifying the appropriate filters in the query editor. The query consists of expressions. An expression always has a data source, such as *Instances*, *Nets*, or *Pins*. Expressions can also have some conditions, such as Name Equals "N" or Name Ends With "4, 6, 8". Results are filtered based on the expressions specified in the query editor.

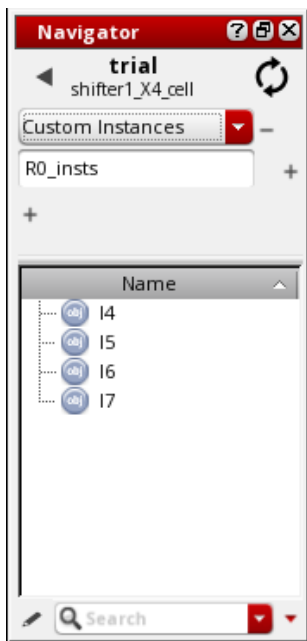


Virtuoso Schematic Editor User Guide

The Navigator Assistant

You can also specify a custom data source. For this, you need to specify the type of data source as *Instances*, *Nets*, or *Pins*. You must also specify the name of an existing SKILL function that is loaded in the `.cdsinit` file. This SKILL function must have a cellview as an argument and should return a list of objects of the specified type of data source.

Note: A pink border around the text field indicates either that the function name is incomplete or that the specified function does not evaluate correctly. When the typed content specifies a function that evaluates correctly, the pink border disappears.

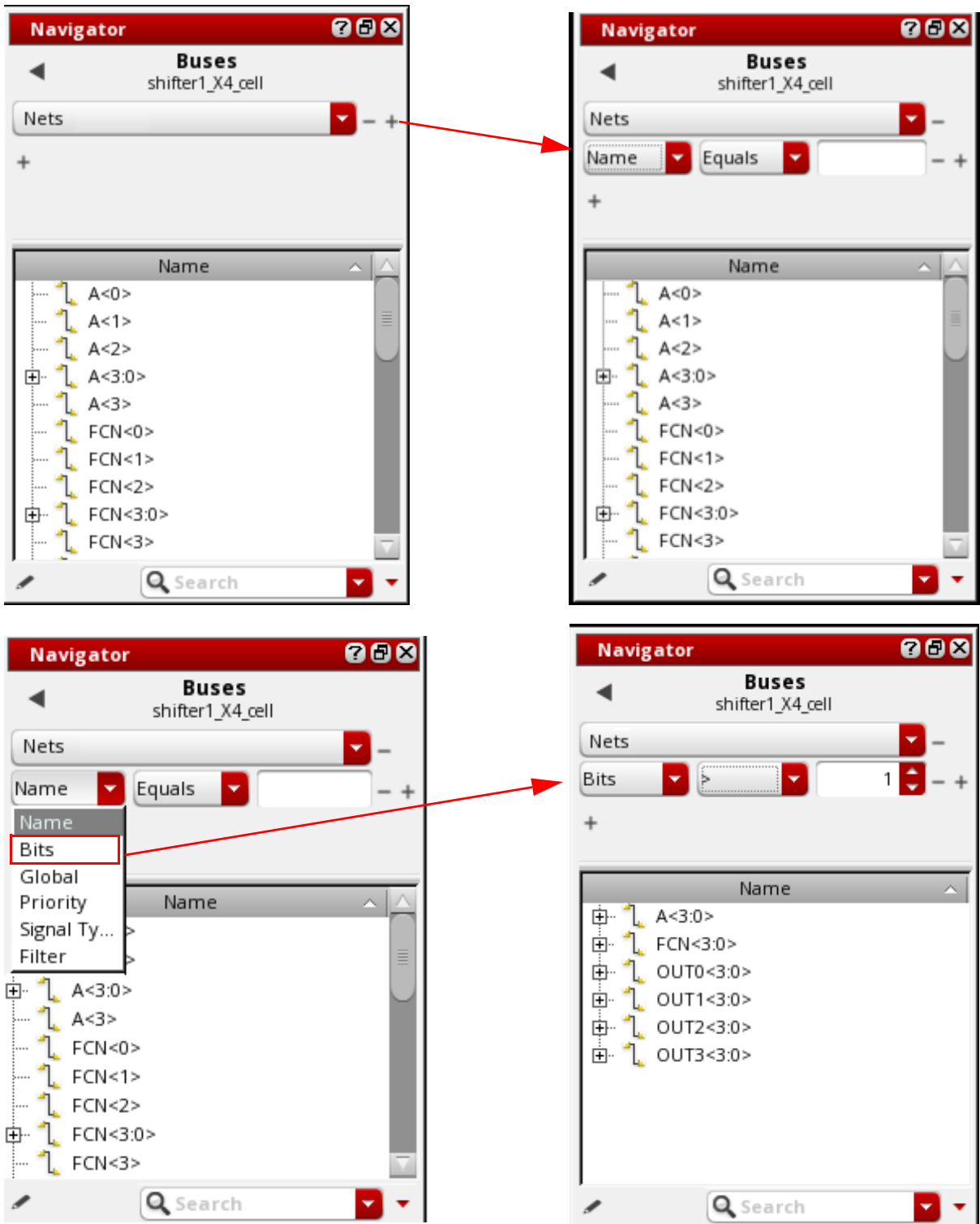


```
defun(R0_insts (cv)
  remove(nil mapcar(
    lambda( (x)
      when(x->orient == "R0" &&
        x->purpose != "pin" x))
      cv~>instances)
  )
)
```

Virtuoso Schematic Editor User Guide

The Navigator Assistant

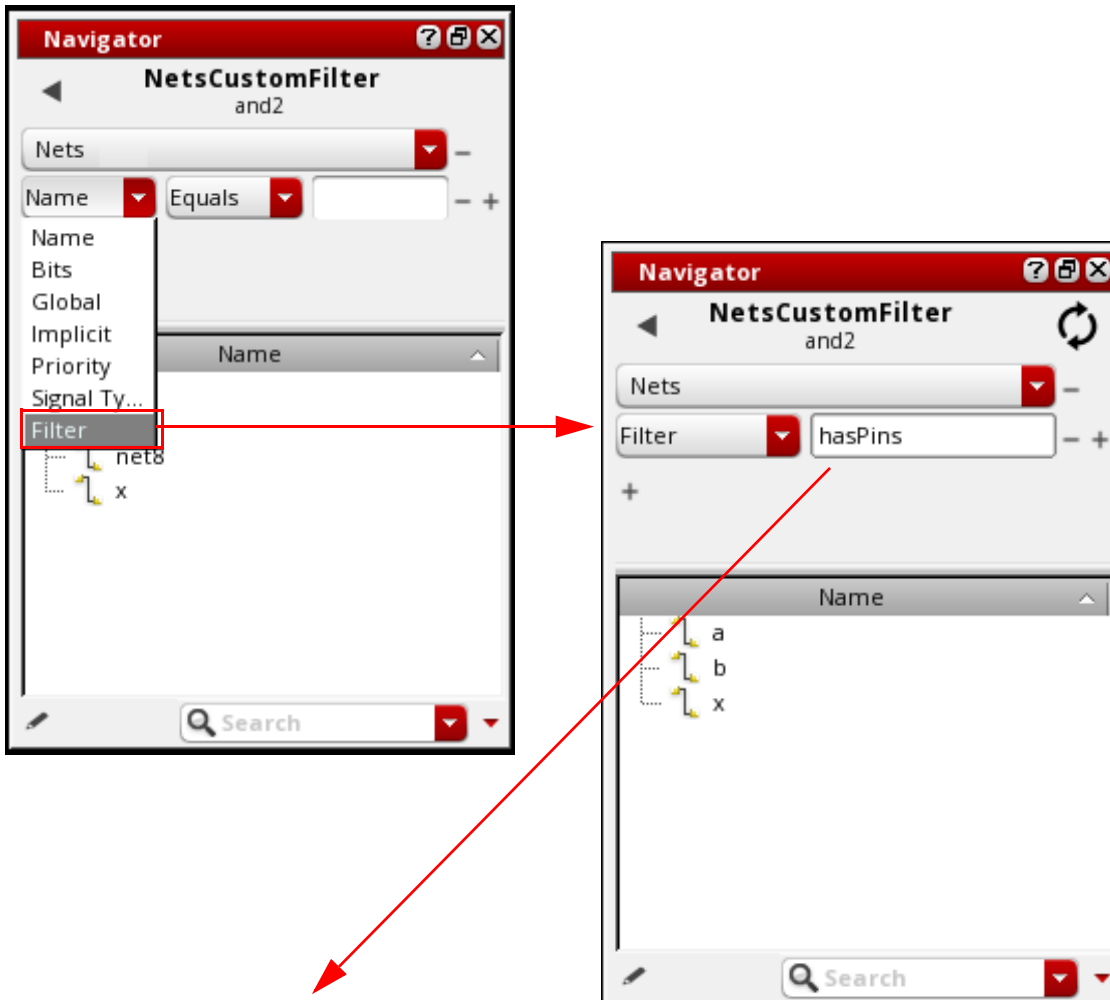
You can add conditions to the queries that you create.



Virtuoso Schematic Editor User Guide

The Navigator Assistant

You can add a filter in the query editor. To specify a filter, you need to specify the name of an existing SKILL function that has a design object as an argument. This function returns `t` if the object is to be retained in the query.

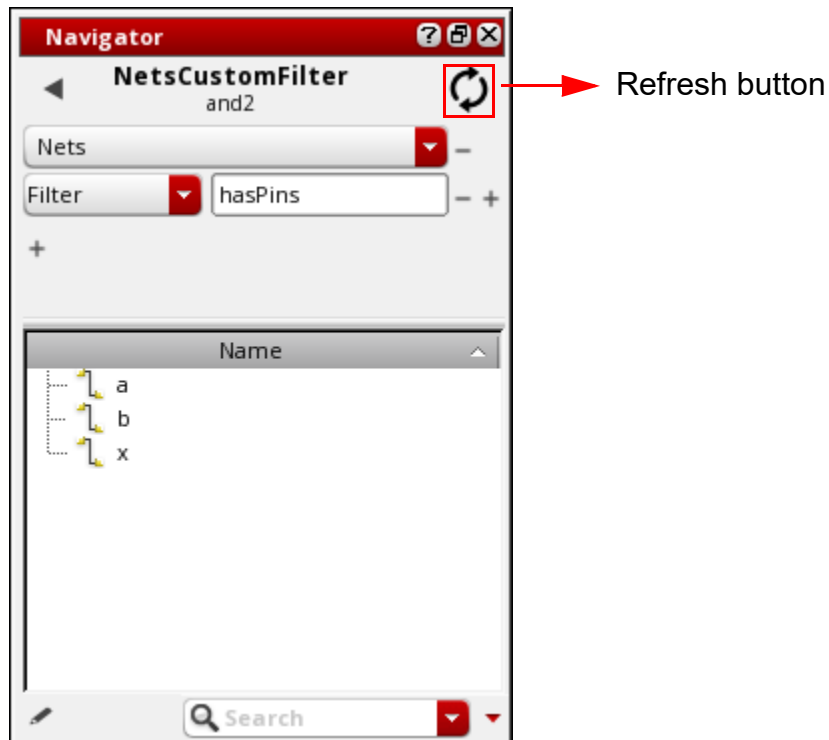


```
defun(hasPins (net) net~>pins != nil)
```

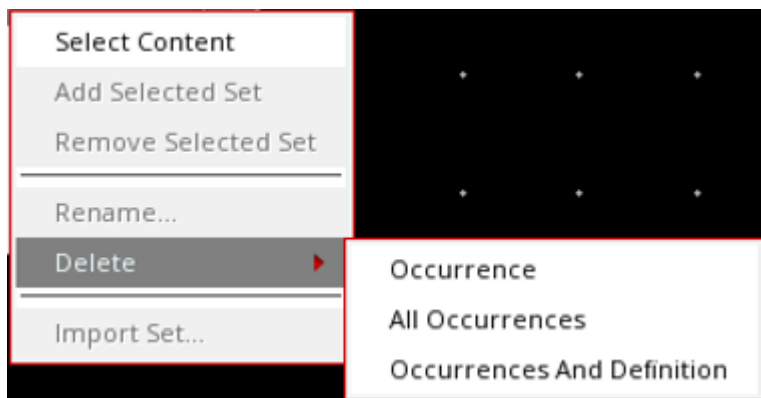

Virtuoso Schematic Editor User Guide

The Navigator Assistant

Sets or queries that do not have custom SKILL functions, custom sources or filter are updated automatically within the *Navigator* assistant. Sets that have custom SKILL functions have to be updated manually by pressing the refresh button.



Right-click the *Summary* pane and select one of the following options from the *Delete* sub-menu to delete the set or query:



Occurrence: Deletes only the selected set

All Occurrences: Deletes all sets generated from the same query as the selected set

Occurrences And Definition: Deletes all sets and the query used to generated them

Virtuoso Schematic Editor User Guide

The Navigator Assistant

- The *Navigator Tree* displays the *Name* of the instances, nets, and pins, in the current cellview tab, and allows you to scroll to and expand/collapse the objects that you are interested in viewing.

Note: To customize the information displayed in the *Name* column see [The Navigator Options Form](#).

- The *Navigator* has the following resizeable and sortable columns that are displayed by right-clicking over the column header area. The *Lib Name*, *Cell Name*, *View Name*, and *XL Status* columns are displayed for user-defined sets as well.

- ☐ *Lib Name*
- ☐ *Cell Name*
- ☐ *View Name*
- ☐ *Type*

Note: Information on the signal type (sigType) is important in the areas of design verification, consistency checking for mixed-signal designs, and to also set routing constraints. Selecting to have this column displayed will, for example, provide information on whether signals and nets have a sigType, such as `signal`, `analog`, `power`, `ground`, or any other type that is listed.

- ☐ *Lock Status*

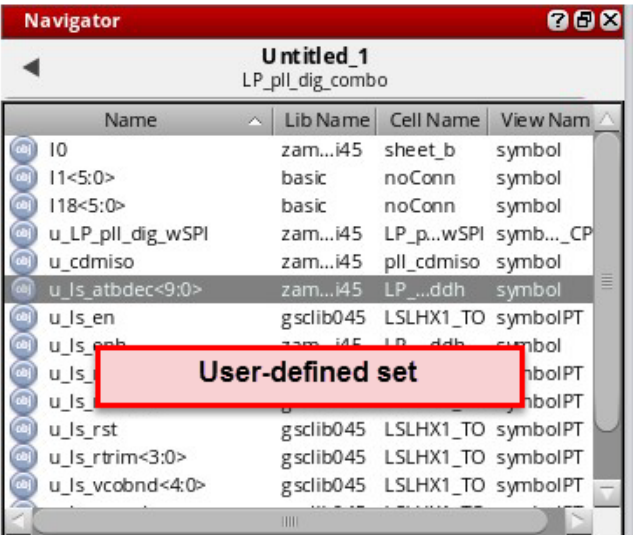
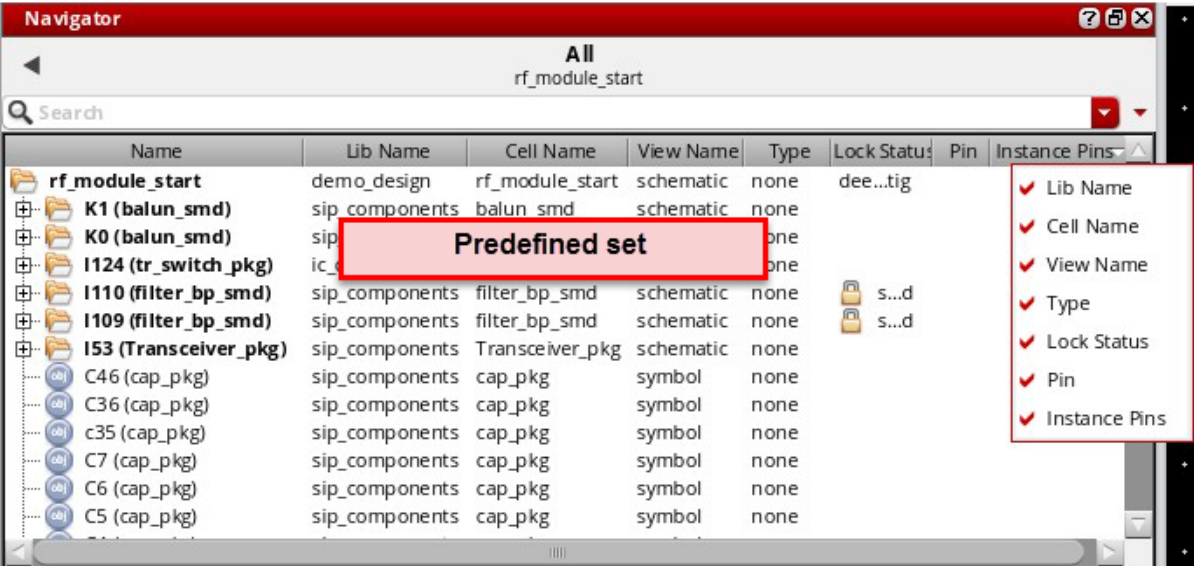
Note: The *Lock Status* column tooltip shows information about who (*User name*) is currently editing a particular cellview and the machine (*Host name*) where the editing is taking place. If you are not the current user a lock icon will be displayed in the column against the appropriate cellview components.

- ☐ *Pin*
- ☐ *Instance Pins*
- ☐ *Clone Family*
- ☐ *XL Status*
- ☐ *Abstract Status*

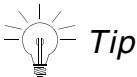
Virtuoso Schematic Editor User Guide

The Navigator Assistant

After displaying the *Column* context-menu, select or deselect the *Column* heading of choice to add or remove it from the *Navigator Tree*.



- The *Navigator* context-menu provides access to the Options form and alternative methods for opening a cellview, expanding and collapsing directory contents. It also provides options to add and remove probes from the selected set.



If the display (expanding of) directory content is taking longer than expected, use the CTRL+C keys to abort the operation and display a partial result.

Virtuoso Schematic Editor User Guide

The Navigator Assistant

For more information see [The Navigator Options Form](#), [Opening Instances and Cellviews from the Navigator](#) and [Adding and Removing Probes Using the Navigator](#) respectively.

Selection and Display in the Navigator

The *Navigator* can display the complete instance design hierarchy associated with the current tab in the current session window. You can also list nets, pins, FigGroups, and categories at any hierarchical level in the current design.

Note:

- If you are in the Virtuoso Layout Suite, the design hierarchy displayed in the *Navigator* (and the *Constraint Manager*) is controlled by the CPH ([Configure Physical Hierarchy](#)).

The *Navigator* is synchronized with the design that is in memory and is automatically updated whenever you, for example, add, delete, or rename instances. It also maintains the current context whenever you change the configuration of a design (see [Updating the Navigator Tree](#)).

- Iterated and array instances are displayed using a hierarchical placeholder which, when expanded, displays each instance and its associated children. When you select individual members of an iterated object in the *Navigator*, they are highlighted in the design canvas.
- Bundle, bus, and iterated nets are displayed using a hierarchical place holder, which can be expanded to view the individual members. If the multi-bit nets are present in the design canvas, then the objects representing all bits or the individual bits can be probed in the *Navigator*.
- All multiple-bit nets (bundles, <*>, buses) are shown as containers and their individual bits are listed as unique members in the Navigator assistant. If an explicit or implicit net is a member of a multi-bit net, its occurrence under the multi-bit net is shown as the copy of the original net. However, the individual bits under an implicit net (a bit with no wire shape on the canvas) can only be viewed by expanding the implicit net.

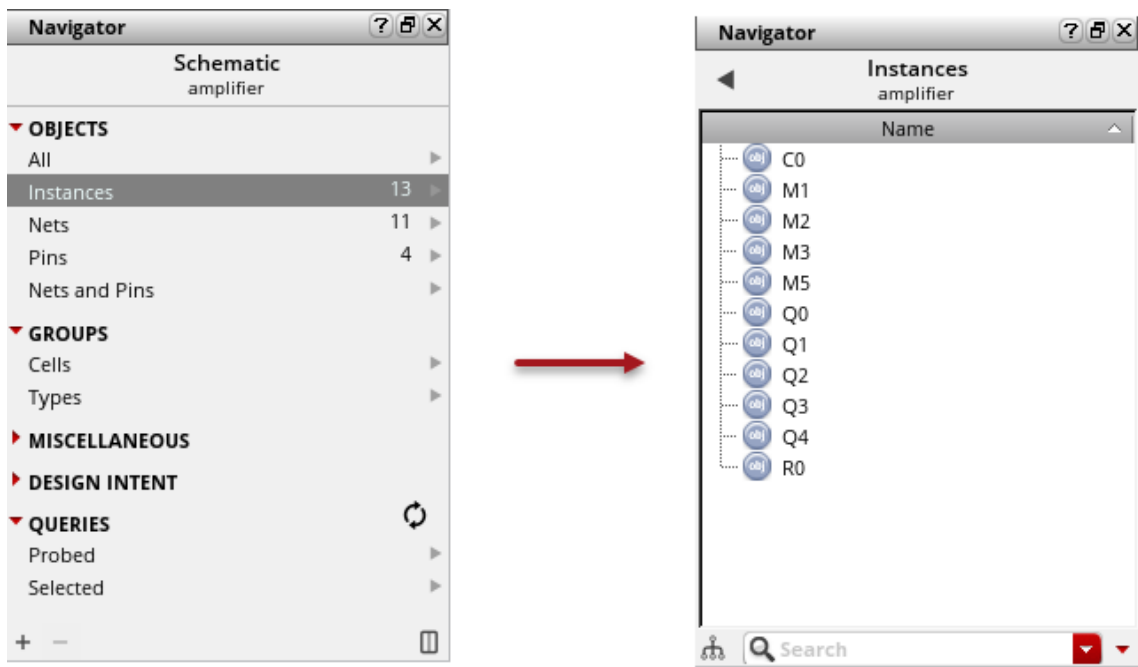
Navigator does not show the leaf-level ports and nets unless enabled through the *Elaborate Through Leaf Level Instances* option in the Options form.

Controlling Object Display in the Navigator with Sets

The *Navigator* displays the sets, for example, Instances, Nets, Pins, or Pcell, in the *Summary* pane. The *Details* pane shows only those object types that are associated with the selected set.

Virtuoso Schematic Editor User Guide

The Navigator Assistant



The following table lists the predefined sets in the Navigator. Not all of these sets are always visible. The visibility of sets depends on design data and other settings.

Sets	Description
All	Shows all the design hierarchy objects for the current session window
Buses	Shows each bus as a container, which, when expanded, lists the associated bits as unique members See compactBusDisplay for information on how to enable this set
Cells	Shows common cells in the design such that the cells of the same type are grouped together and highlighted in the schematic
Fig Groups	Shows objects belonging to specific groups (more applicable to VLE)
Instances	Shows only the design instances
Modules	Shows only the design modules

Virtuoso Schematic Editor User Guide

The Navigator Assistant

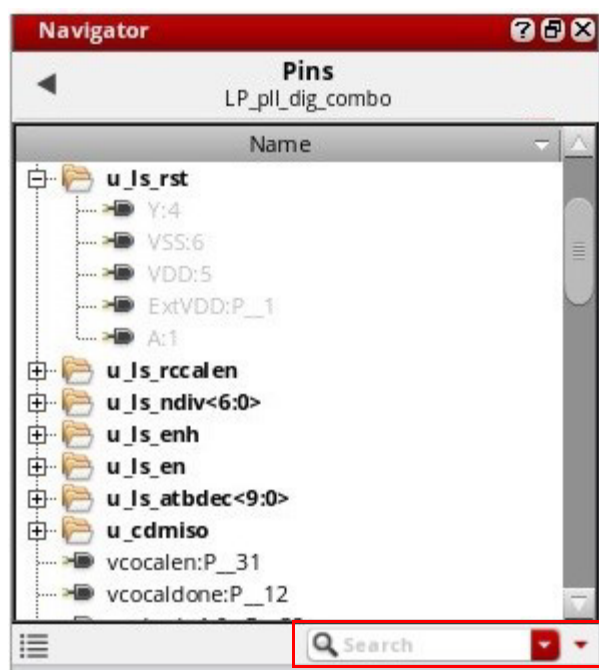
Sets	Description
Names	Shows instances, nets, and pins grouped by name (more applicable to VLE)
Nets	Shows only the design nets
Pins	Shows only the design pins
Types	Shows instances, nets, and pins grouped by type
Ungenerated	Shows instances and pins that have not been generated in the corresponding layout (more applicable to VLE)

You can also create and populate your own sets either by selecting and grouping objects directly from the assistant or by creating a custom SKILL query to extract and group the desired objects from the design. For more information, see [The Navigator User Interface](#). Custom sets created in these ways are persistent across Virtuoso sessions.

Controlling Object Display in the Navigator with Search

You can also control the display of objects in the *Navigator Tree* using *Navigator Search*.

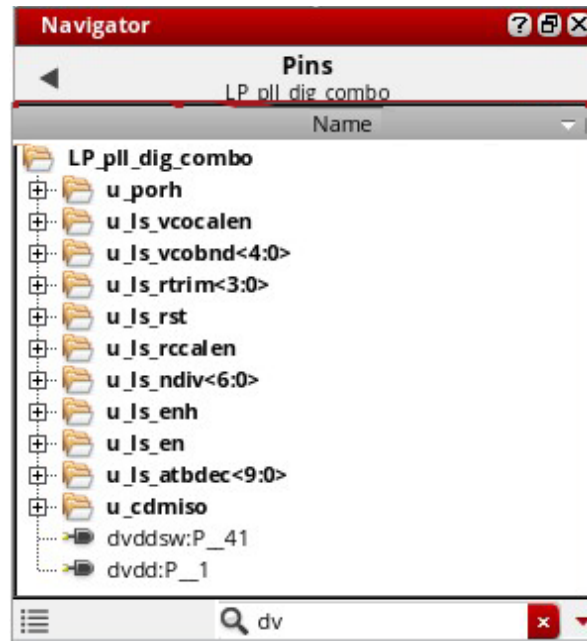
Note: Search facilities in the *Navigator* only operate on those sets that are currently visible in the *Navigator Tree*. This allows you to control the search scope by adding and removing sets as required. See also [The Navigator User Interface](#).



Virtuoso Schematic Editor User Guide

The Navigator Assistant

Once you begin to type your search criteria into the *Navigator Search*, the *Navigator Tree* automatically updates to display only the matching results below the top levels of the hierarchy.



When searching in user-defined sets, the Navigator Tree automatically updates to display only the matching results. The unrelated hierarchical information is removed.

To the right of the *Search* entry field, there are two drop-down list boxes: one for search history and the other for specifying search criteria. For more information on these, see the [Search Toolbar](#).

Updating the Navigator Tree

The *Navigator Tree* automatically updates its display if any of the following changes in the design environment occur:

- If you switch between cellview tabs in a session window.
- If you select specific design objects, for example, *Pins* or *Nets*.
- If another application (for example the Virtuoso® Layout Suite) is used to select an instance.
- If you ascend or descend through the design hierarchy.
- If you make any changes to the design hierarchy, for example deleting instances, or changing the configuration.

Being In-Context or Out-Of-Context

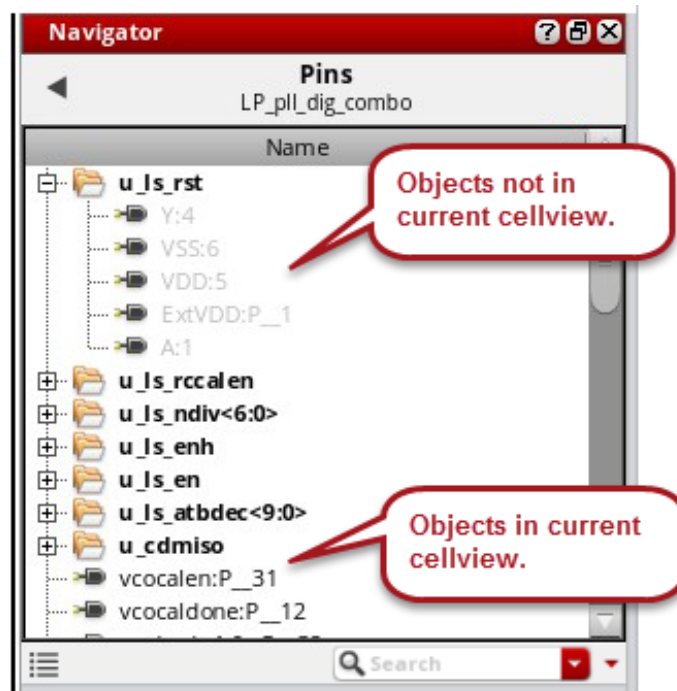
Being *in-context* refers to design information that is on display in the current view tab.

There are two main concepts to consider here:

- The current cellview (the view displayed in the current tab)
 - All active objects (instances/nets/pins) in the current cellview are displayed as black text on a white background.
 - All inactive objects (instances/nets/pins) not in the current cellview are displayed as gray text on a white background.

Virtuoso Schematic Editor User Guide

The Navigator Assistant



- Out of context
 - Going out of context is not currently supported. If you descend into a view that is different from the default view, the view that is current in the Descend form (displayed by selecting *Edit – Hierarchy – Descend Edit/Read*), the *Navigator* does not follow the descent.

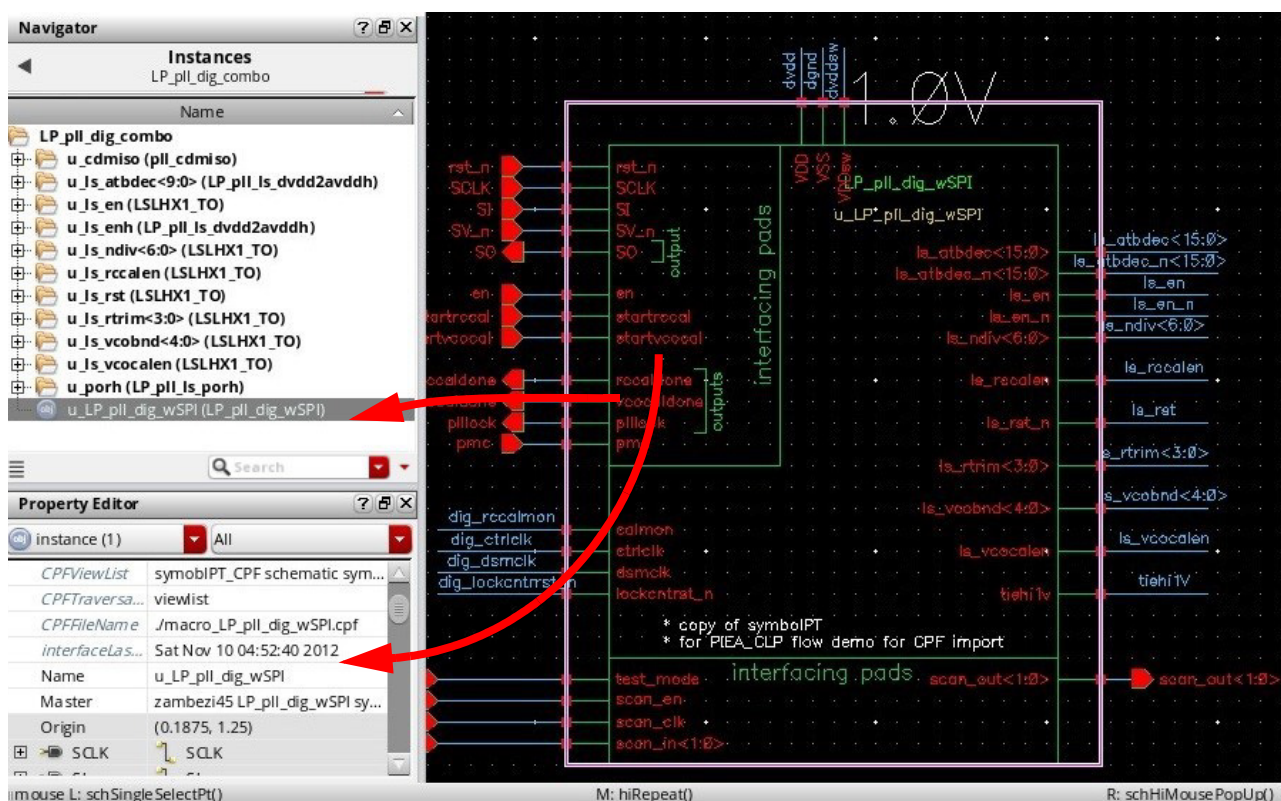
Cross Selecting Between the Navigator and the Current Cellview

If you select, for example, an instance, pin, or net from the current cellview, that selection is reflected in the *Navigator*.

If a selected (in the canvas) object is not initially visible in the *Navigator Tree* display, the *Navigator* automatically scrolls and expands to display its location in the tree. Likewise, an object selection made in the *Navigator Tree* automatically highlights the chosen instance in the current cellview.

Note: If *Dynamic Zooming* is checked in the *View* menu, VSE will also auto-pan to that selected object in the design canvas.

Selections made in the *Navigator* are reflected in the Property Editor Assistant and the Constraint Manager Assistant, and vice versa.

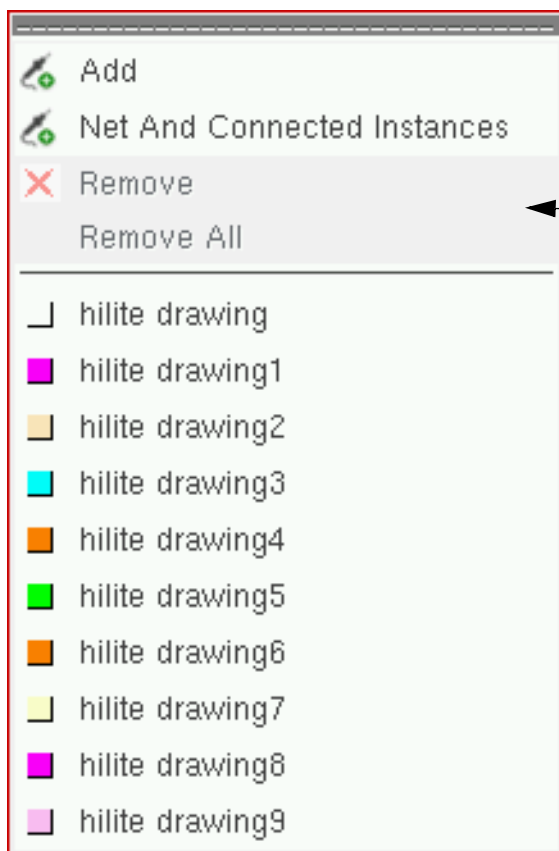


Adding and Removing Probes Using the Navigator

You can use the *Navigator* assistant to add and remove probes from one or more selected objects.

1. Select the object(s) that you want to probe (hilite) in the *Navigator* tree.
2. Right-click the *Navigator* tree and select the *Probe* option from the context menu displayed.

This displays a probes' sub-context menu.



You can also choose to remove the current probe or all probes

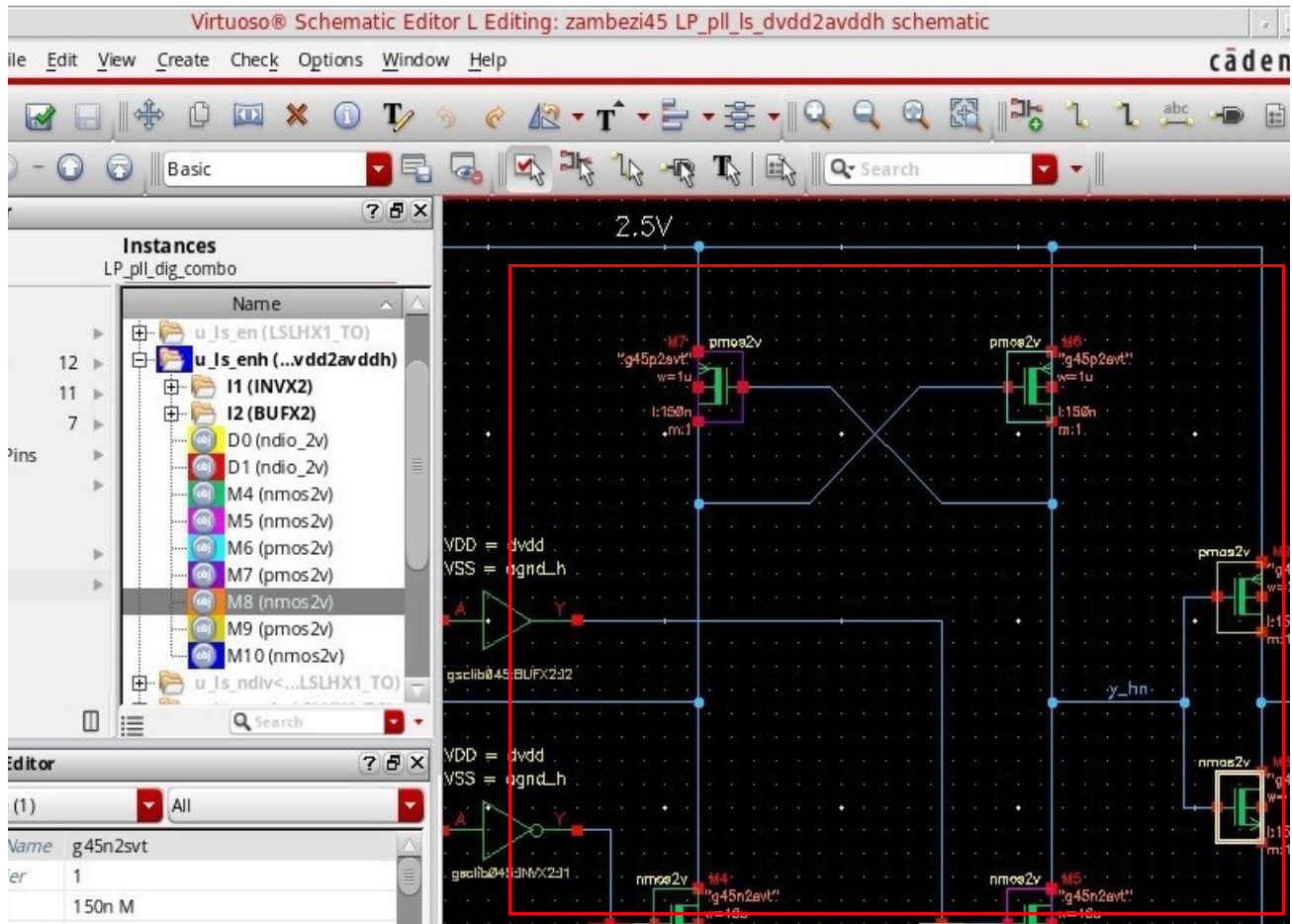
3. Select *Add* and the selected object(s) will be highlighted in probe color order.

Note: If the last color used to probe an object was *hilite drawing4*, then *hilite drawing5* will automatically be used next in line. You can however, alternatively, directly select the highlight color that you want to apply.

Virtuoso Schematic Editor User Guide

The Navigator Assistant

The probed objects are highlighted in the *Navigator* tree and shown on the design canvas in the selected highlighted color.



Tip

If the display of probe information in the *Navigator* tree is taking longer than expected, use the CTRL+C keys to abort the operation and display a partial result.

Net and Connected Instances Option

The *Probe* context-menu also provides a *Net and Connected Instances* option which can be used to probe a net, and all instances that are connected to that net, in a schematic hierarchy.

Virtuoso Schematic Editor User Guide

The Navigator Assistant



Tip

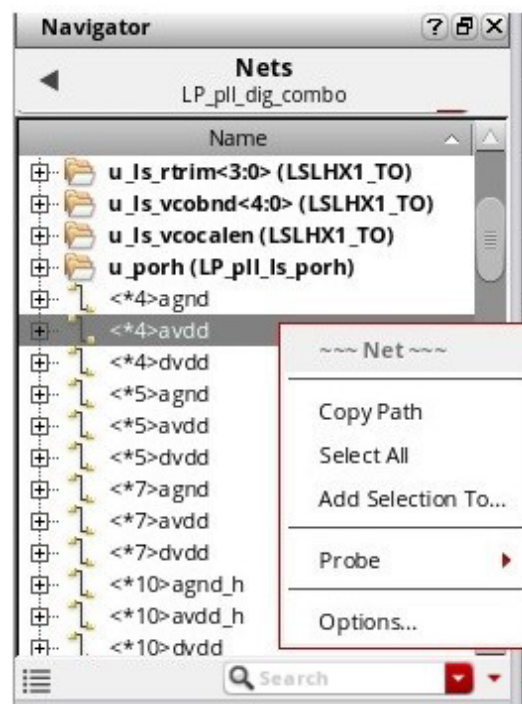
This option is only available if a net is selected.

Adding and Removing Probes from Nets on the Canvas

To add or remove probes from a net (wire) in the design canvas:

1. In the canvas, select the net that is to have a probe added or removed.
2. Right-click over the selected net.

The net (*Wire*) context-menu is displayed.



3. Select *Probe – Add*.

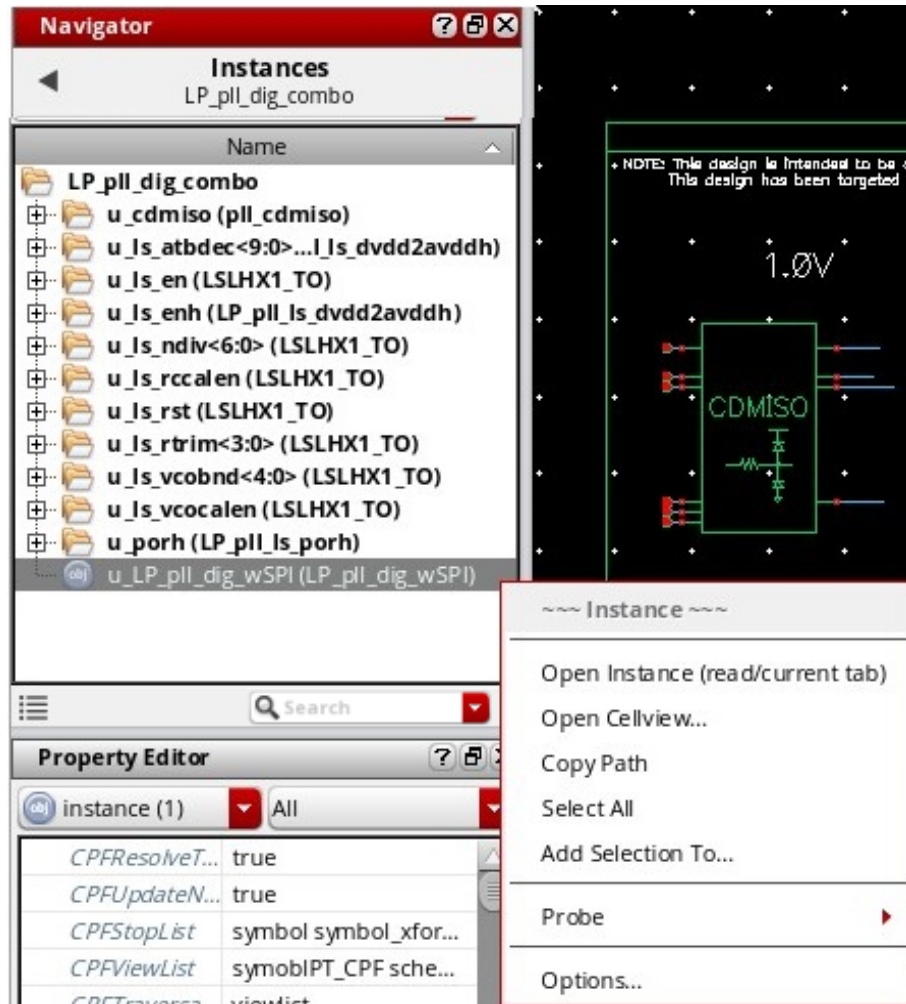
The wire connectivity probe will now be colored (displayed) on the canvas and the wire will be shown in the *Navigator* to have a probe currently applied.

4. Select *Probe – Remove* or *Remove All* to clear any probes from the current wire.

Note: Selecting *Remove Probe* will start an enter function that will allow you to remove net probes one by one.

Opening Instances and Cellviews from the Navigator

If you right-click a selected instance, the *Navigator* provides a number of options for opening an instance or cellview.



- **Open Instance (read/current tab)**

Opens the selected instance view in the current session window tab (in read mode).

Note: This is configurable using [The Navigator Options Form](#).

You should use this option if you are working on a *design* and want to switch between different levels of the hierarchy. This option keeps the design at the top and allows you to traverse the hierarchy.

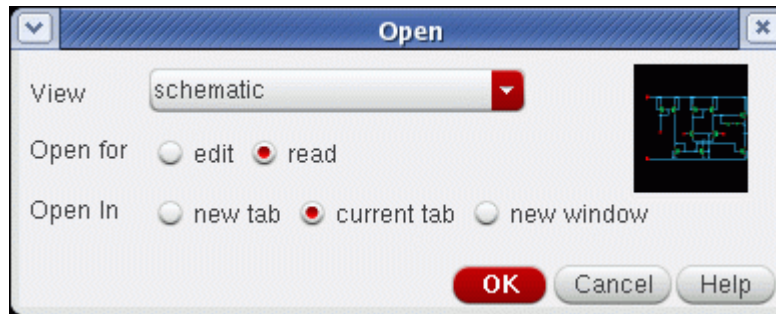
- **Open Cellview**

Virtuoso Schematic Editor User Guide

The Navigator Assistant

Displays the Open form from where you can select the open options of your choice.

Note: A thumbnail will also be displayed in the Open form if there is one available for the current cellview. For more information on the generation and use of thumbnails, see [Thumbnail View Functions](#) in the *Cadence User Interface SKILL Reference*.



- ❑ *View* provides the details of all the available views that can be opened.
- ❑ *Open for* lets you open the selected view in *edit* or *read* (the default) mode.
- ❑ *Open In* enables you to open the selected view in a *new tab*, the *current tab*, or in a *new window*.

You should use this option if you are working on a cellview. The options available in the Open form are used to re-route the *Navigator* to base the design in the selected cellview.

You can alter the default open mechanisms in [The Navigator Options Form](#) using the *Open in* option.

Note: For more information on tabs and session windows, see [Virtuoso Design Environment User Guide](#).

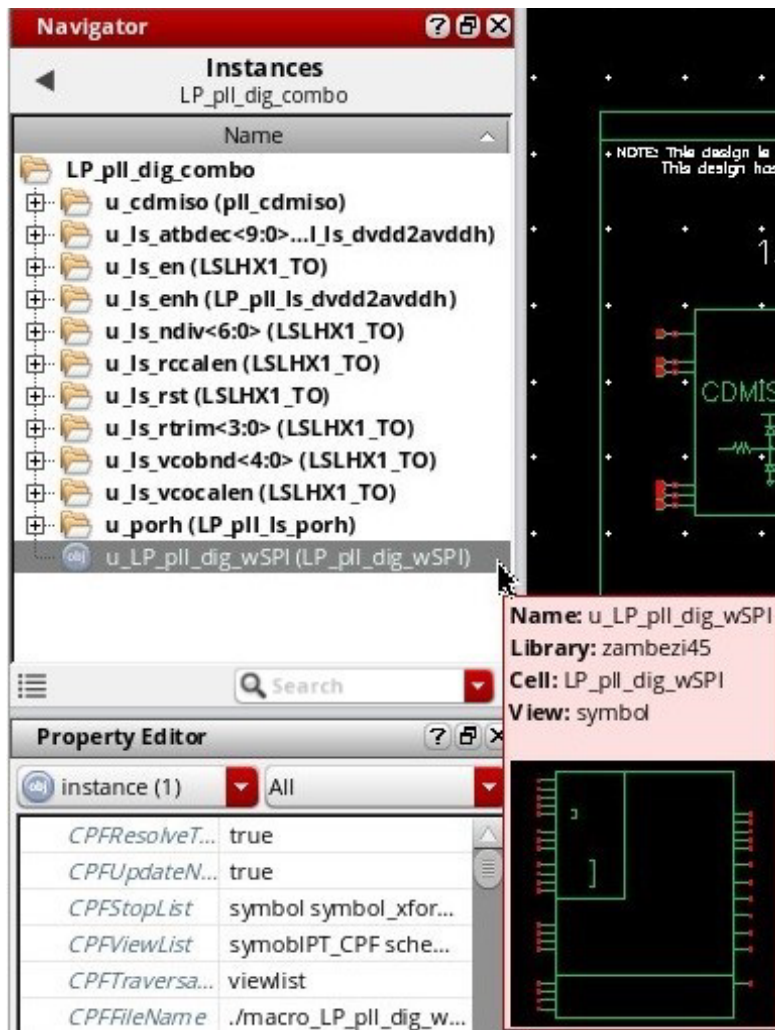
Virtuoso Schematic Editor User Guide

The Navigator Assistant



Tip

Thumbnails, where available, are also displayed in tooltips when you move cursor over such objects as *Instances* and *Nets*.



The Navigator Options Form

The *Navigator* Options form can be used to customize the information displayed in the "Name" column of the *Navigator Tree* and also for setting defaults for cellview opening.

You can access the Options form by selecting *Options* from the *Navigator* context-menu.

Options

"Name" Column

Show: Lib Name
Cell Name
View Name
Type
Lock Status
Pin
Instance Pins

Place: ☐ Before ☒ After

Separator: -

Delimiters: (and)

Sample Text:
Name(Cell Name)

Display Control

Order ☒ blocks first ☐ instances first ☐ nets first ☐ pins first

Suppress at	Instances (K)	Nets (K)	Pins (K)	Twigs (K)	Ungenerated Instances (K)	Ungenerated Pins (K)
	1	1	1	1	1	1

☒ Column Sorting
☐ Elaborate Layout
☐ Elaborate Through Leaf Level Instances

Default Open

Mode: ☐ edit ☒ read

Open in: ☐ new tab ☒ current tab ☐ new window

OK **Cancel** **Help**

Virtuoso Schematic Editor User Guide

The Navigator Assistant

The table below describes the field options available in the Options form:

Option	Description
<i>"Name" Column section</i>	Controls the information displayed in the <i>Name</i> column.
<i>Show</i>	Specifies what column information is to be shown when generating the display content of the <i>Name</i> column. For example, if you choose to include <i>Cell Name</i> column information, the cell name will also be included when displaying the instance name. For example: <pre>(vco2phase) I18.</pre> The information within parenthesis being the name of the cell where the instance can be found.
<i>Place</i>	Controls where the information from the <i>Show</i> selection is placed, either <i>Before</i> the object name or <i>After</i>. For example: <pre>(vco2phase) I18 or I18 (vcophase).</pre>
<i>Separator</i>	Specifies the separator string to be used between each element of the <i>Name</i> column information. For example: <pre>Name(Cell Name-Instance Pins)</pre> or <pre>Name(Cell Name, Instance Pins)</pre>
<i>Delimiters</i>	Specifies the beginning and end strings to be used when representing <i>Name</i> column information. For example: <pre>Name(Cell Name-Instance Pins)</pre> or <pre>Name[Cell Name-Instance Pins]</pre>

Virtuoso Schematic Editor User Guide

The Navigator Assistant

Sample Text

You can determine the order in which columns are reused using the *Up* and *Down* buttons that are adjacent to the *Add Columns* option. The positional order of the *Add Columns* list represents the order in which information will be used when generating the content of the *Name* column.

For example, if you select both *Cell Name* and *Instance Pins* from the *Add Columns* option, using **Ctrl+Shift**, the *Sample Text* option will display a summary of how this will be displayed in the *Name* column:

```
Name(Cell Name-Instance Pins)
```

If the column information is not available for a given object it will be omitted.

Note: The *Sample Text* field cannot be edited manually and it will only reflect selections made in the *Show*, *Separator*, and *Delimiters* fields.

Display Control section Manages the information content displayed in the *Navigator Tree*.

Order

Specifies which category of objects are to be shown first under the cell when it is expanded. Select one from: *blocks first*, *instances first*, *nets first*, *pins first*.

You can use this option in association with the *Navigator* toolbar to control the visibility of object types. For example, you can focus on the creation of a net or pin centric *Navigator Tree* display.

Virtuoso Schematic Editor User Guide

The Navigator Assistant

<i>Suppress at</i>	Controls the upper threshold number of instances, nets, pins, twigs, and ungenerated instances and pins (in thousands - <i>K</i>) after which the <i>Navigator</i> should create a placeholder for a given object type (rather than elaborating and representing all possible objects). If a placeholder is in place this will be represented on the <i>Navigator Tree</i>, for example as “100K Further Instances”. Note: Ungenerated instances and pins are the design objects that are not yet generated in the layout. You can un-suppress selected object type information by right-clicking over the, for example, “100K Further Instances” row and selecting the <i>Un-suppress</i> option from the context-menu displayed. To suppress object type information again, right click again in the <i>Navigator Tree</i> and select the <i>Suppress Leaves</i> option which will activate the <i>Suppress at</i> field option settings.
<i>Column Sorting</i>	Specifies whether alphabetical column sorting is performed by default. Note: Disabling this option can speed up the expansion of large cellviews.
<i>Elaborate Layout</i>	Layout cellviews will be automatically elaborated if this option is checked. Elaboration determines the parent-child relationship between instances and identifies blocks (hierarchical instances).
<i>Elaborate Through Leaf Level Instances</i>	Elaborates nets and pins of the leaf instances. By default, the option is turned off. Click this option to enable expansion of leaf instances.
<i>Default Open section</i>	Controls the default open mechanisms used in the Open form (see <u>Opening Instances and Cellviews from the Navigator</u>).
<i>Mode</i>	Choose to open, by default, a selected instance cellview in either <i>edit</i> or <i>read</i> mode (default). Environment variable: <u>defaultOpenMode</u>

Virtuoso Schematic Editor User Guide

The Navigator Assistant

Open in

Choose to open, by default, a cellview in a *new tab*, the *current tab* (default), or a *new window*.

Environment variable: defaultOpenIn

The Property Editor Assistant

This chapter contains the following topics on the *Property Editor* assistant pane (the *Property Editor*):

- [Introducing the Property Editor](#)
- [Accessing the Property Editor](#)
- [Hiding the Property Editor](#)
- [Property Editor Keyboard Shortcuts](#)
- [The Property Editor User Interface](#)
- [Editing Property Values](#)
- [Property Status Visualization](#)
- [The Property Editor and CDF Support](#)
- [Modgen Property Updates using the Property Editor Assistant](#)

Introducing the Property Editor

The *Property Editor* is a dockable assistant pane that provides a flexible framework for defining a collection of objects and their associated properties. An object represents a component of interest in an application, for example a wire segment in a schematic, a constraint, or an analysis to be passed to a simulator. Properties are the characteristics of these objects and can be viewed and edited as required.

Note: The Edit Object Properties form can still be used to view and edit properties and other tasks such as property creation.

Some of the key features provided by the *Property Editor* include the ability to:

1. Filter by object type and property categories
2. Visualize the current status of an object property, for example if the property value is non-default or read-only
3. Compare values across multiple selected objects and schematic windows
4. Directly edit individual property values on single and multiple objects.

Accessing the Property Editor

To access the *Property Editor* in Schematics:

- Select *Window – Assistants – Property Editor*.
- Right-click in the main Schematics menu or toolbar area and select *Property Editor*.
- The *Property Editor* is also available in the following default window workspaces (first select *Window – Workspaces – ...* , or select from the *Workspace* toolbar):
 - *Basic*
 - *Constraints*
 - *Explore*

You can access a workspace by either selecting *Window – Workspaces*, or by choosing a workspace from the *Workspace Configuration* toolbar pull-down.

Once selected, Schematics will embed the *Property Editor* as a docked assistant pane within the current session window. By default the *Property Editor* will be positioned in the lower left area of the session window.

For more information on workspaces see [Workspaces and Assistants in the Schematic Editor](#).

For more information on customizing your own workspaces see [Working With Workspaces in the Virtuoso Design Environment User Guide](#).

Hiding the Property Editor

To hide the *Property Editor* in the current session window:

- Right-click in the Schematics toolbar area and uncheck the *Property Editor* option.
- Change to a window workspace that does not include the *Property Editor* as part of its default configuration.
- Click the *Hide* button (X) in the *Property Editor* assistant pane title bar.

The Property Editor User Interface

The *Property Editor* comprises of the following user interface (UI) elements:

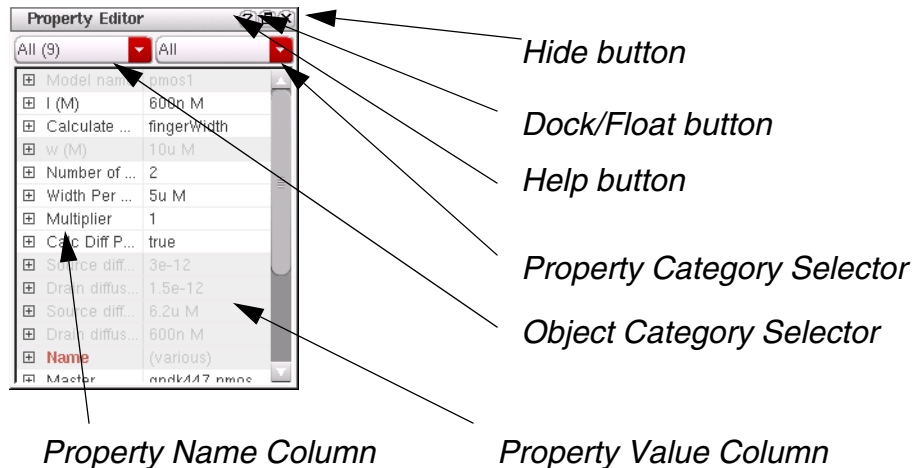


Figure 16-1 The Property Editor

- The *Property Editor* title bar buttons, which allow you to:
 - ❑ Obtain *Help* information on the *Property Editor*.
 - ❑ *Float* (undock) the *Property Editor* (select again to *Dock* the *Property Editor*).
 - ❑ *Hide* the *Property Editor*.
 - The *Property Editor* contains two drop-down selection categories:
 - ❑ Selecting Objects
 - ❑ Filtering by Property Type
 - The *Property Editor* contains two distinct columns:
 - ❑ On the left there is the *Property Name Column* (see Filtering by Object Type) which displays the name of a property.
 - ❑ On the right there is the *Property Value Column* (see Editing Property Values) which displays the current value of a property.
- Note:** Columns can be resized by clicking and dragging the column margin separator.
- The *Property Editor* context-menu (for related information see The Property Editor Context-Menu).

Virtuoso Schematic Editor User Guide

The Property Editor Assistant

- The *Property Editor* also utilizes *ToolTips* that can display fully expanded property names and values as well as provide property value status information (such as read-only properties and if the property value has been changed from a default value).

To obtain *ToolTip* information float your cursor over the property value or name. If you add or make changes to the description of the CDF parameters, the same information gets reflected in the tooltip in the Property Editor.

Selecting Objects

The *Property Editor* displays information for the currently selected set of objects. As you select or deselect objects in the editor window the *Property Editor* display is automatically updated. If no objects are selected in the design canvas, the *Property Editor* displays the properties of the current cellview.

Note: The *Property Editor* does not perform any automatic connectivity checks. Rather, for any selected wires, it displays the net currently stored in the design database. Any changes take effect the next time connectivity is updated.

If multiple objects have been selected in the design canvas, the *Property Editor* will highlight, in the canvas, the object whose properties are currently selected for edit in the *Property Editor*.

For example, in the figure below, instances M10 and M9 are both currently selected in the canvas, but it is a property of instance M10 (highlighted) that is currently selected for edit in the *Property Editor* assistant.

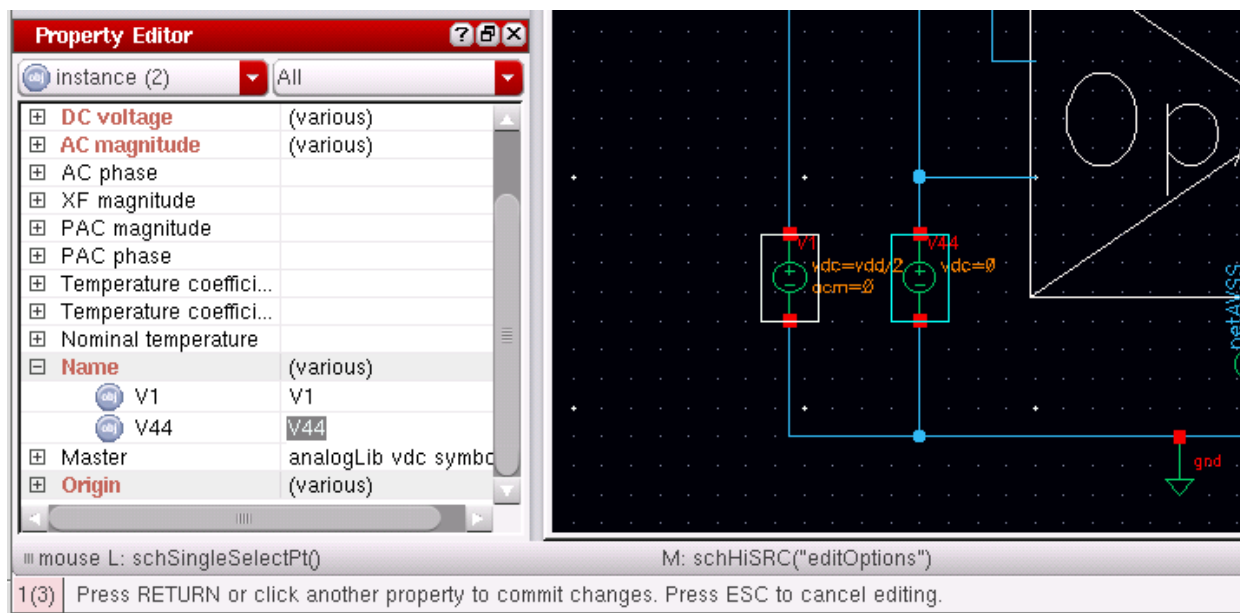


Figure 16-2 A Property of Instance V44 Currently Selected for Edit

If you are editing a property across several objects all of these objects will be highlighted in the canvas.

Virtuoso Schematic Editor User Guide

The Property Editor Assistant

The following object types are supported: instance, instance pin, pins, pin names, wire names, net expressions, property labels, instance labels, wire segments, note shapes, note text.

Note: When a wire segment is selected in the design canvas, the *Net Name* is displayed for information in the *Property Editor*, but it cannot be edited. To edit the *Net Name* in the *Property Editor*, the wire label must be selected and changes take effect the next time connectivity is updated.

You can specify minimum and maximum voltages for a net in the VSE property editor assistant. However, if you want to set the voltage for an individual bit, draw a wire representing that bit. For example, if there is a bus, BUS<0:7>, set the voltage on bit#1 and draw the wire, BUS<1>. Then, set the voltage on that wire.

Object Selection Thresholds

If more than a specified number of object selections are made in a single selection (either through manually selecting objects or using *Edit – Select – Select By Filter*), in the current design canvas, the *Property Editor* will enter *lazy* display mode. That is, rather than automatically attempt to display property information for all the currently selected objects, the *Property Editor* will inform you of the number of objects selected and provide an option to *Show* their associated values, if required.



Figure 16-3 Property Editor Threshold Exceeded

You can set *Property Editor* threshold values using SKILL variables that can be set in the `.cdsinit` file.

- `oiSelectionThreshold` sets the maximum number of objects the *Property Editor* displays before entering *lazy* mode. The default is 50.

For example: `oiSelectionThreshold = 50`

Note: If you have 10 objects initially selected and displayed in the *Property Editor*, then attempt to select a further 90 objects on top of this, the *Property Editor* will state that there are a *Further 90 objects* that you can choose to *Show* (while still displaying property information on the 10 initially selected).

Virtuoso Schematic Editor User Guide

The Property Editor Assistant

- `oiMaxSelections` sets the maximum number of objects displayed by the *Property Editor*.

For example: `oiMaxSelections = 1000`

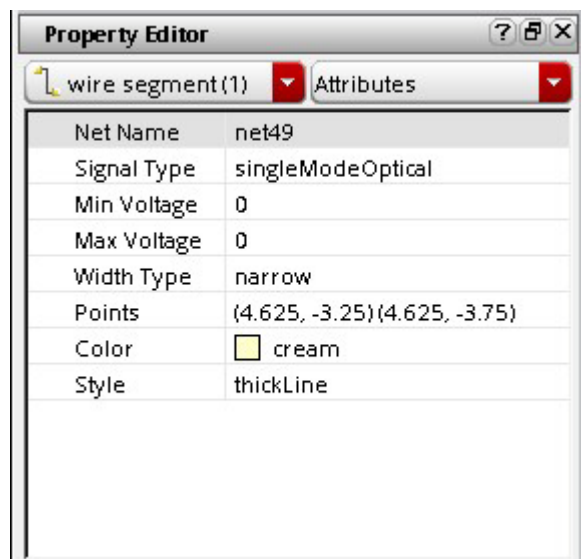
- `oiMaxInstTerms` sets the maximum number of instance pins displayed by the *Property Editor* for an instance. The default is 1000.

For example: `oiMaxInstTerms = 1000`

Editing the Optical Signal Types in Property Editor (Virtuoso Photonics Option)

Note: Virtuoso Photonics Solution - You need the *Virtuoso_Photonics_Option* license to use the functionality discussed in this section. For information about obtaining the required license, contact your local Cadence representative.

You can view or edit optical signal types on pins and wires in the Property Editor assistant.



Filtering by Object Type

The Object Category Selector is a drop-down list located at the top-left of the *Property Editor* assistant. It shows a count of the different types of objects selected.

If you want to display property information for a particular object category, select the object category that you are interested in viewing, for example *instance* or *pin*, from the *Object Category Selector* drop-down list.

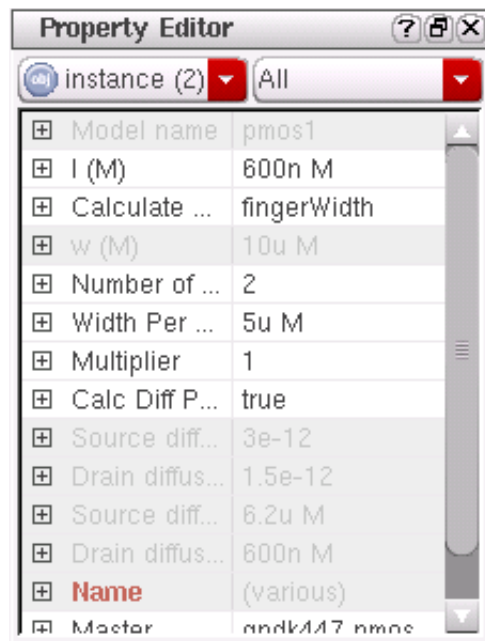


Figure 16-4 Instances selected as the object type

When an object category is selected, Filtering by Property Type drop-down list will automatically update to allow you to choose to display the property categories relevant to the visible objects.

In summary, to display an object category's associated properties and values in the *Property Editor*:

1. Select one or more objects from the current session window view.

Note: You can click and drag, or use `shift` and select, to make multiple object selections.

The currently selected object types will be displayed in the *Object Category Selector* drop-down.

Virtuoso Schematic Editor User Guide

The Property Editor Assistant

2. From the *Object Category Selector* drop-down, select the object category that you want to view properties for.

Only those properties that are associated with the selected object category will be displayed in the *Property Name Column*.

Filtering by Property Type drop-down updates to only list those property categories that are associated with the currently selected object category.

3. Choose from the *Property Category Selector* the property category that you want to view properties and values for the selected object category, for example *All* or *Parameters*.

Filtering by Property Type

The *Property Category Selector* is a drop-down list that displays the Property Categories associated with the current object category selection in Selecting Objects.

Selecting a property category from the *Property Category Selector* drop-down list will further filter the properties shown in the *Property Editor*.

You can choose to filter property display using the following property category options: *Attributes*, *Properties*, *CDF Parameters*, or *All* properties.

Property Categories

The *Property Category Selector* provides a range of property categories that can filter property data sources.

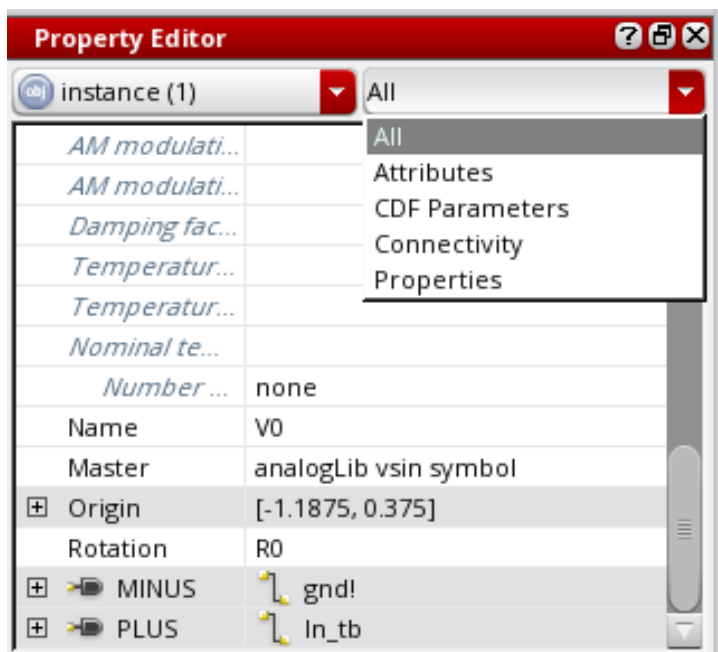


Figure 16-5 Filtering by property types

■ *Attributes*

Attributes are a combination of the basic and system property sections found in the Edit Object Properties form and represent the core characteristics of a selected object.

■ *Connectivity*

Virtuoso Schematic Editor User Guide

The Property Editor Assistant

This category is applicable only to pins and details such as *Pin Name*, *Signal Type*, and *Direction*.

■ *Properties*

The *Properties* category contains details of any user-defined properties for the current objects.

For more information on basic, system and user-defined properties, see the [Edit Object Properties](#) form in the *Virtuoso Schematic Editor L User Guide*.

■ *CDF Parameters*

The *Property Editor* lets you display details of Component Description Format (CDF) parameters. This enables you to view and edit them as you would database attributes and properties.

Note: The tooltip for a CDF parameter will display the parameter name.

For information on CDF parameters see the [Component Description Format User Guide](#)

For information on CDF parameter display in the *Property Editor* see [Property Status Visualization](#).

■ *All*

Select *All* property classes to display all the properties associated with all the object categories selected in the [Selecting Objects](#).

The Property Editor Context-Menu

The *Property Editor*'s context-menu provides options for arranging and navigating displayed properties. Right-click over the *Property Editor* to display the context-menu contents.

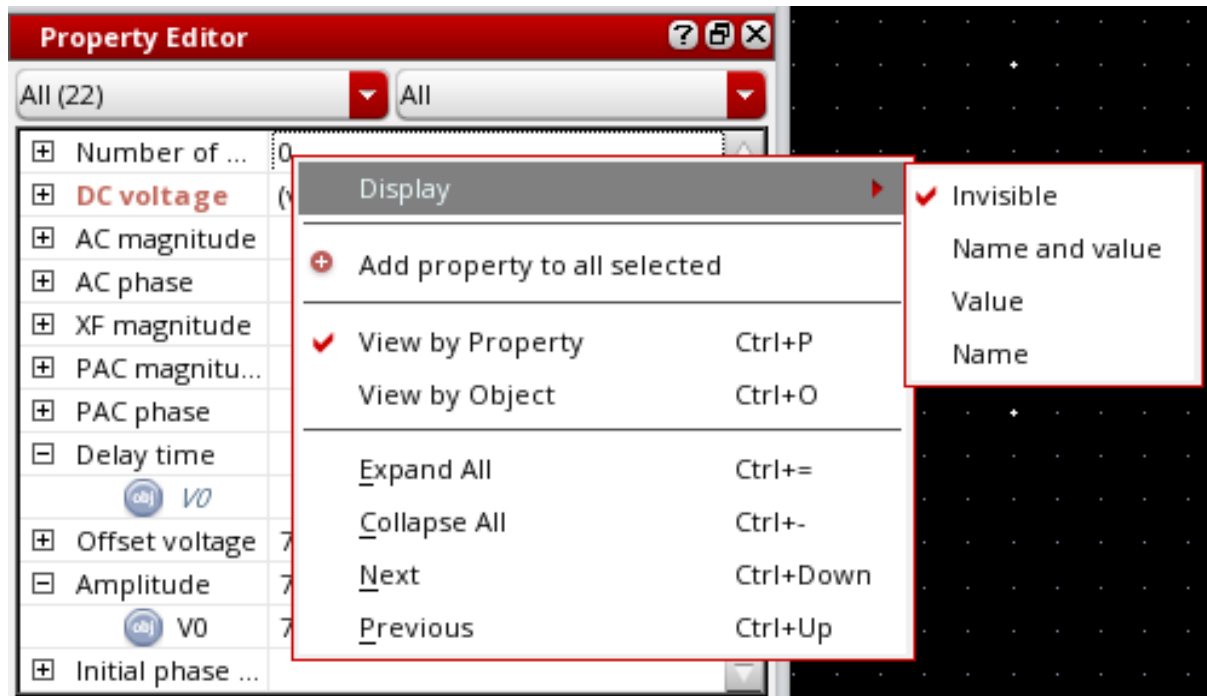


Figure 16-6 The Property Editor assistant context-menu

The *Property Editor* context-menu provide the following options:

- *Display* (see [Displaying Object Property Values on the Canvas](#))
- *Delete property* (see [Adding and Deleting Properties using the Property Editor](#))

Note: The *Delete property* option is only available for the deletion of user created properties.

- *Add property* ([Adding and Deleting Properties using the Property Editor](#))
- *View by Property*
- *View by Object*
- *Expand All*
- *Collapse All*

Virtuoso Schematic Editor User Guide

The Property Editor Assistant

- [Next](#)
- [Previous](#)

Note:

- When you are in property value edit mode (see [Editing Property Values](#)) the context-menu will display editing options such as *Cut*, *Copy*, *Paste*, and *Delete*.
- Where applicable, a *Revert to default* option will be displayed in the context-menu (see [Reverting to Default Values](#)).

Displaying Object Property Values on the Canvas

To use the *Property Editor* assistant to display (or hide) selected object property values on the canvas:

1. From the design canvas, or the *Navigator* assistant, select the object(s) whose property values you want to show or hide on the design canvas.
2. In the *Property Editor*, right-click over the object property whose value(s) you want to display on the design canvas.

The Property Editor assistant context-menu will be displayed.

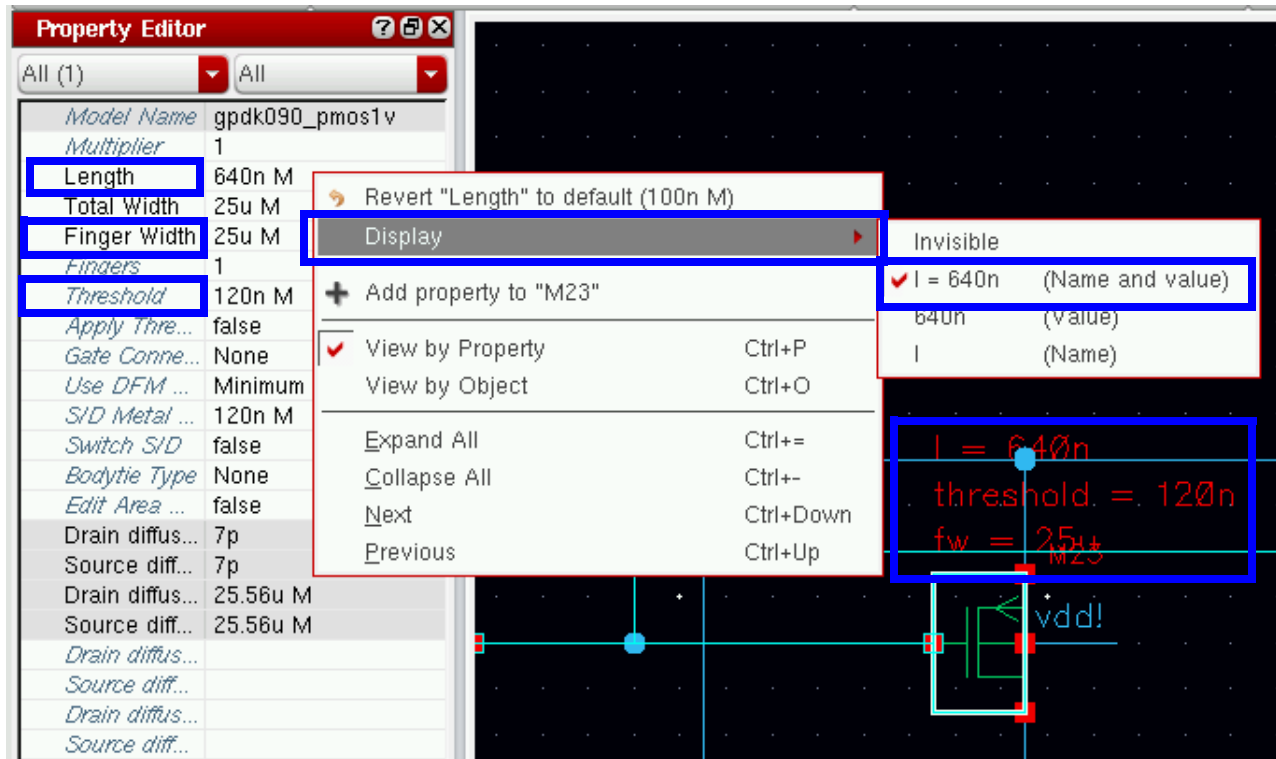
3. Select *Display* from the *Property Editor* assistant context-menu.
4. From the sub-menu now displayed, choose to either display the selected object property *Name and Value*, *Value* (only), *Name* (only), or to make these properties *Invisible* on the canvas.

The selected object will update to display the selected property values on the canvas.

5. Repeat process as required to add multiple property values to the object in the design canvas. Where multiple property values are chosen to be displayed, they will be “stacked” (see below).

Virtuoso Schematic Editor User Guide

The Property Editor Assistant



The *Length*, *Finger Width*, and *Threshold* object properties with their *Name* and *Value* settings selected for display on the canvas for instance *M23*.

Figure 16-7 Using Property Editor Assistant to Display Property Values on Canvas

Virtuoso Schematic Editor User Guide

The Property Editor Assistant

View by Property

Select *View by Property* to display all properties associated with the selected objects, grouped by property name.

Each object property is listed in a separate row providing a single or summary value for each object that has that property.

You can choose to view the information rolled-up (collapsed) or expanded by clicking on the +/- icon to the left of the *Property Name Column* row (see also [Expand All](#) and [Collapse All](#)).

If there are differences in property values across selected objects, the property name will be highlighted in red font and the property value summarized by the string (*various*). If there are no differences in property values across selected objects with the same property, a single value will be displayed (see also [Property Status Visualization](#)).

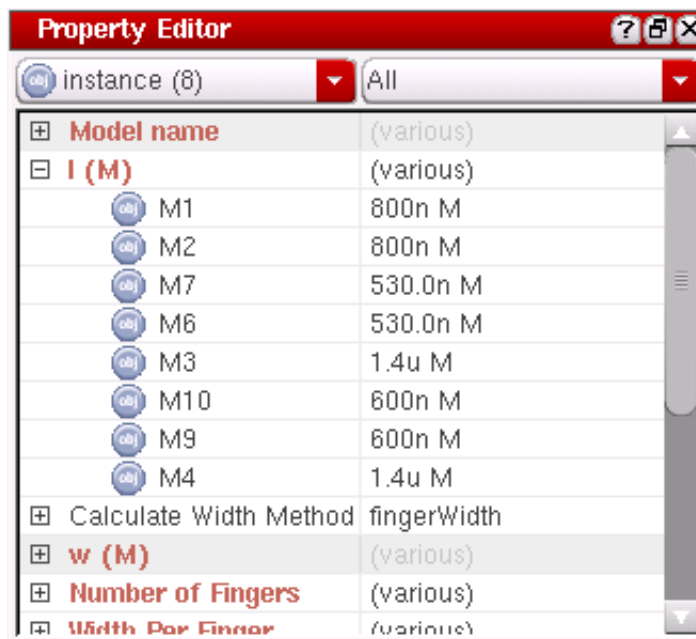


Figure 16-8 Red font property names indicate differences in property values

View by Object

Select *View by Object* to display information grouped by object rather by property. This mode gathers all the properties for an object together under a single entry.

You can choose to view information rolled-up (collapsed) or expanded by clicking on the +/- icon adjacent left (see also [Expand All](#) and [Collapse All](#)).

Note: Right-clicking again and selecting View by Property will restore the default viewing mode.

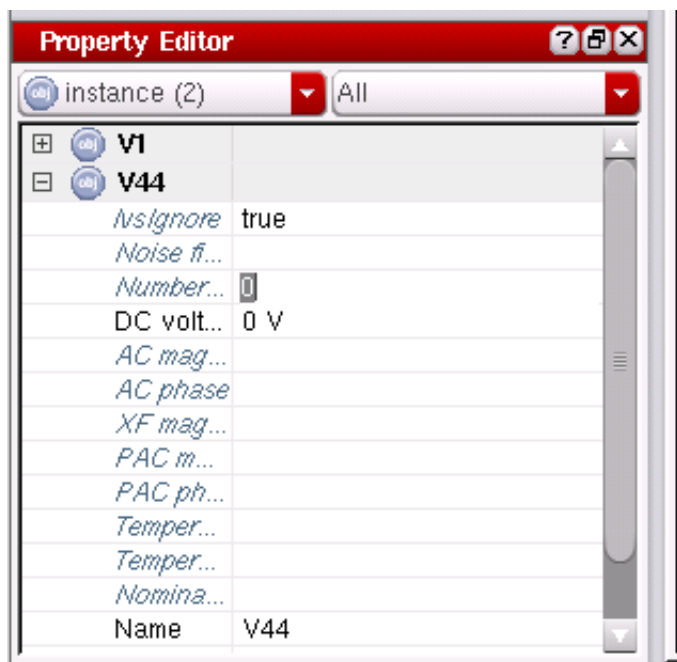


Figure 16-9 Viewing by object

Expand All

Select *Expand All* to expand all of the properties or objects currently listed in the *Property Name Column*.

This will display a row for every selected object and related property, with the current value reflected in the *Property Value Column* in an alphabetical order.

Note: You can expand on a single property basis by clicking on the adjacent “+” icon or double-clicking on the property you want to expand.

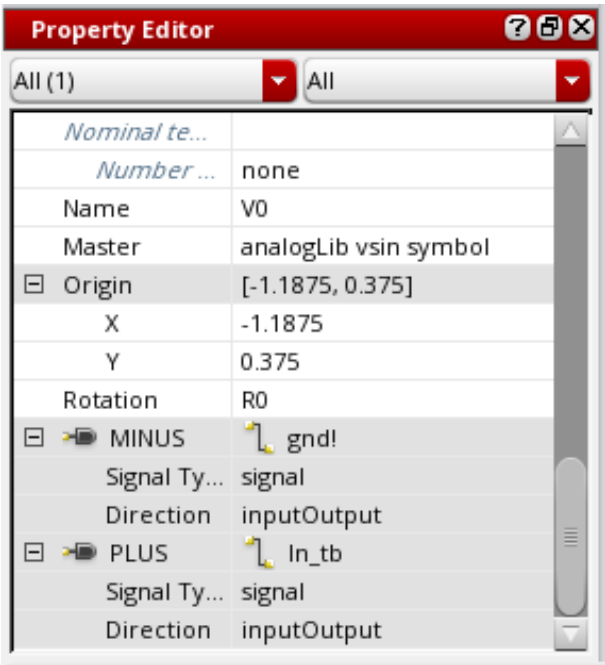


Figure 16-10 All properties expanded

Collapse All

Select *Collapse All* to collapse all of the properties or objects currently listed in the *Property Name Column*.

This will hide all the value rows for every object or property and display either a summary value (*various*), or a single value if the value is applicable to all collapsed objects or properties.

Note: You can collapse on a single property basis by clicking on the adjacent “-” icon or double-clicking on the property you want to collapse.

Next

Select *Next* from the *Property Editor* context-menu to move down a row to the next object or property row.

This action will automatically expand the selected row to display all object or property values associated with the current object or property.

Previous

Select *Previous* from the *Property Editor* context-menu to move back up a row to the previous object or property row.

This action will collapse the currently selected object or property and select and expand the object or property immediately above it.

Property Editor Keyboard Shortcuts

You can use the following keyboard shortcuts to assist interaction with the *Property Editor*:

Note: The keyboard shortcuts listed table below also have *Property Editor* context-menu equivalents. For more information see [The Property Editor Context-Menu](#).

Key Combination	Action
Ctrl and P	Switches to View by Property mode.
Ctrl and O	Switches to View by Object mode.
Ctrl and +	Expands all collapsed properties (see Expand All).
Ctrl and -	Collapses all expanded properties (see Collapse All) .
Ctrl and down arrow	Advances to the <i>next</i> top-level property and expands it while collapsing the previously selected property (see Next).
Ctrl and up arrow	Moves to the <i>previous</i> top-level node and expands it while collapsing the previously selected property (Previous).

Note: The Ctrl down/up arrows can be used to quickly expand/collapse through all table properties or objects.

Adding and Deleting Properties using the Property Editor

You can use The Property Editor Context-Menu to add or delete properties from one or more objects.

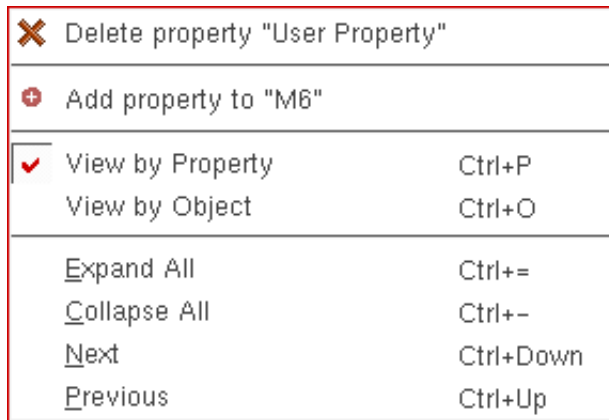


Figure 16-11 The Property Editor Context-Menu

Adding a Property Using the Property Editor

To add a property to one or more objects:

1. Select the object(s) in the design canvas that you want to add the property to.
2. Right-click over the *Property Editor*.

This will display The Property Editor Context-Menu.

3. Select *Add property*.

Note: If you are adding a property to multiple objects, the option will be titled *Add property to all selected*.

This will display the Add Property form.

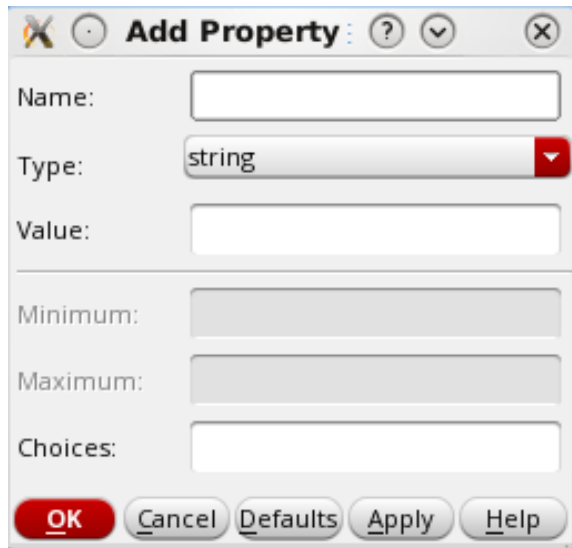


Figure 16-12 The Add Property Form

1. Complete the Add Property form as required:

- ☐ *Name* specifies a name for the new property.
- ☐ *Type* specifies a property type.
- ☐ *Value* specifies a value for the property.

Note: To define most properties, you need to specify only the *Name*, *Type*, and *Value* fields. However, when you specify *string*, *int*, or *float* as the *Type*, you will have to also set one or more of *Minimum*, *Maximum*, or *Choices*.

- ☐ *Choices* require to be set when *Type* is set to *string*. If the string can have more than one value, type the possible values. Entries in this field must match an entry in the *Value* field.
- ☐ *Minimum* and *Maximum* requires to be set when *Type* is set to *int* or *float*. Specify a range for the minimum and maximum range for the integer or floating-point value.

2. Click the *OK* button.

The new, user defined, property is added to the currently selected object(s) and displayed in the *Property Editor*.

Deleting a Property Using the Property Editor

Note: You can only delete user created properties using [The Property Editor Context-Menu](#).

To delete a property from one or more objects:

1. Select the object(s) that contains the property you want to delete.
2. In the *Property Editor*, select the user property that you want to delete.

Note: If you want to delete a property from multiple objects at the same time, choose *View by Object* from [The Property Editor Context-Menu](#). This will display all objects currently selected in the canvas in a list. Selecting *Delete property*, in step 4 below, will delete the property, if it exists, from all the selected objects.

3. Right-click over the *Property Editor*.

This will display the *Property Editor* context-menu.

4. Select the *Delete property* option.

You will not be asked to confirm the property deletion but if you require to restore the property select *Edit – Undo*.

The property will be removed from the current object(s) and will no longer be displayed in the *Property Editor*.



Tip

To quickly delete a property, select the user property to be deleted and click the red X icon to the right of the property row.



Figure 16-13 Quickly delete a property by selecting the red X icon

Editing Property Values

The *Property Editor* displays properties on a property-by-row basis inside two columns. The *Property Name Column* displays the current property names, while the *Property Value Column* displays the related property values.

You can edit property values for a single object or for multiple objects.

Note: Some properties, for example CDF parameters, can have read-only flags applied to them. See [Property Status Visualization](#).

■ See also [Compound Properties](#).

Editing Single Object Property Values

To edit properties using the *Property Editor*:

1. Select one or more objects, such as instances, wire names, wire segments, and so on, from the current session window view.

You can select multiple objects by clicking and dragging around your selected view area, or using `shift` and select. When you select an object in the current view it is highlighted in a white outline.

Once a selection has been made the *Property Editor* automatically updates to display the properties and property values of the currently selected objects.

Note: If the Navigator Assistant is on display it will also update to highlight the current objects.

Virtuoso Schematic Editor User Guide

The Property Editor Assistant

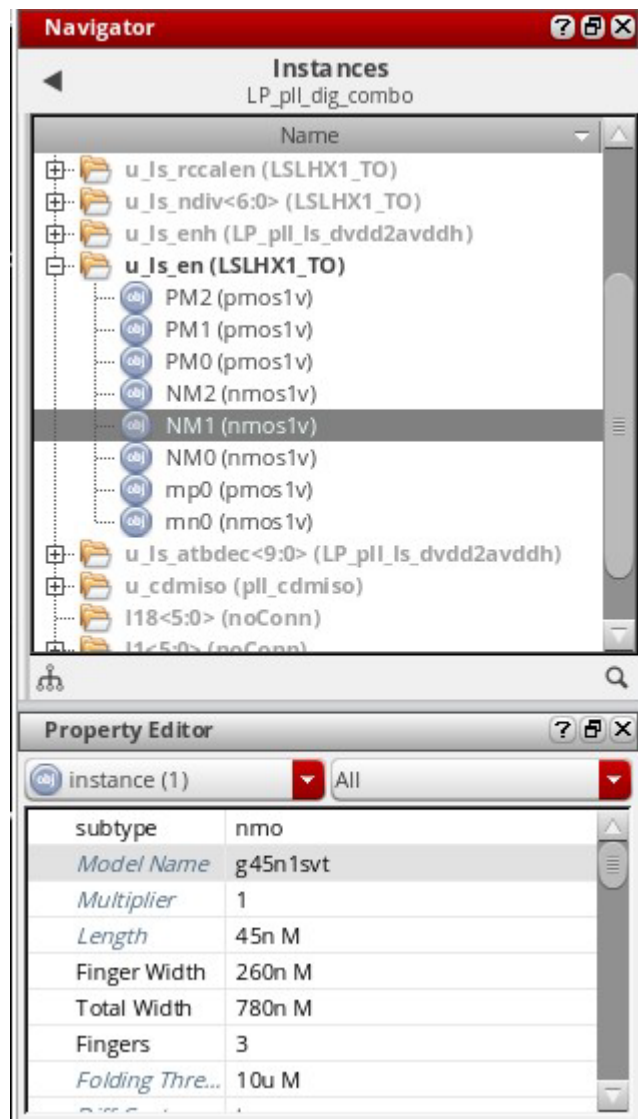


Figure 16-14 Cross-highlighting (NM1 instance) between Property Editor and Navigator

2. In the *Property Value Column* click the property value you want to edit.

Note: If a property or object has been collapsed you will have to expand it before selecting a property value to edit.

Depending on the property you want to edit, a specific editing mechanism becomes active when you click in the property value field. For example, a cyclical field, a pull-down menu, or direct text editing may be used (see also [Extended Editing Forms](#)).

Virtuoso Schematic Editor User Guide

The Property Editor Assistant

Note: Some property values, such as certain *CDF Parameter* values, cannot be edited.

Note: Set the `CDS_Netlisting_Mode` SHELL variable to *Analog* to ensure that AEL evaluator is able to find the CDF property in OA. Subsequently, set the `aelEvalCDFParam` environment variable. Property Editor displays the value of the expression, however, the tooltip displays the expression. The expressions must be valid Analog Expression Language (AEL) expressions.

When multiple objects of the same type are selected in the current view, any properties common to all of the selected objects will store all of their values in one property row. For example, if you have selected three instances all of their name values will be rolled up in the one summary *Name* row. Clicking on the “+” icon to the left of the *Property Name Column* row will expand the row to display all instance *Name* values.

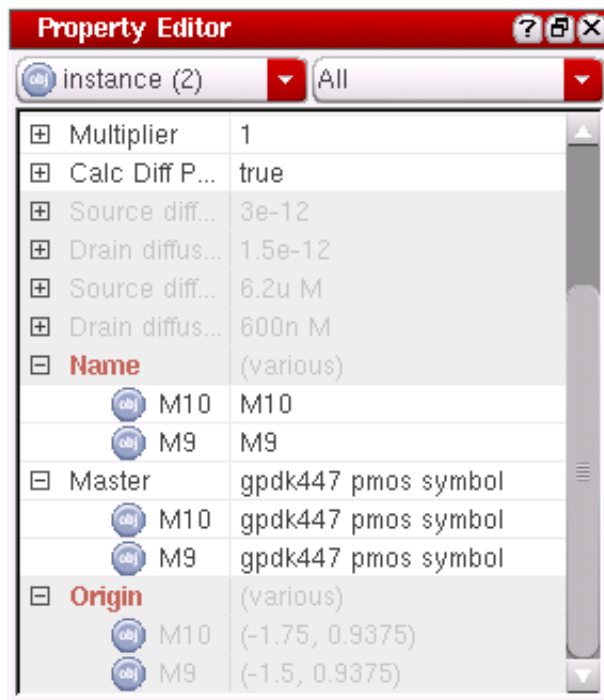


Figure 16-15 Multiple property values expanded

Where the values of selected objects differ, for example *Name* above, then a (*various*) value will be displayed in the adjacent property value field, and the property name will be displayed in red text. When editing such variant values, a pull-down menu will be available to enable you to choose what instance's property value you want to edit.

3. Edit the property value as required using the associated editing mechanism.

Note: When in edit mode you can right-click in the property value field to perform *Undo*,

Virtuoso Schematic Editor User Guide

The Property Editor Assistant

Redo, Clear, Cut, Copy, Paste and *Select All* actions from the property edit context-menu.

To cancel a property value change click another property value row or press the `Esc` key.

Note: This information is also displayed in the session window *Status Bar*.

4. To finish editing and commit your change, click anywhere in the canvas/toolbar/another window or press `Tab`. On pressing `Tab`, the control does not go to the next field but the value entered for the current field gets committed.

Editing Multiple Object Property Values

As well as editing individual property values you can also apply a single property value across multiple selected objects by editing the property value field of a property summary row.

Note: Certain property values cannot be edited across multiple objects. For example, when it is important to maintain unique instance names.

To edit multiple object property values:

1. Click in the property value field of a property summary row.

Note: The property value field will display as *(various)* if the property value differs across the selected objects, or as a single value if the current property value is shared across all selected objects.

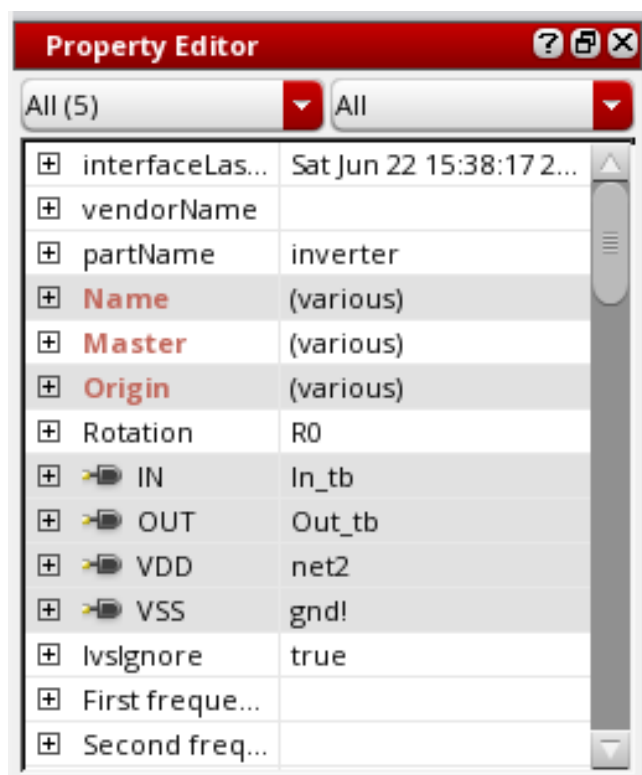


Figure 16-16 Multiple object property values indicated by “various” status

1. Optionally, expand the property summary row to display the property values for all the objects you want to edit.

Virtuoso Schematic Editor User Guide

The Property Editor Assistant

2. Edit the summary property value to be the value that you want to apply to all selected objects that share the same property.

Note: Depending on the type of property being edited the correct editing mechanism will be presented to perform the edit, for example a simple text entry or a cyclic entry field.

To cancel a value change click another property value row or press the `ESC` key.

Compound Properties

As well as being able to display “simple” properties and their values in the *Property Editor*, you can also view and edit compound (hierarchical or grouped properties) properties and their values.

Compound properties introduce another level of hierarchy in the *Property Editor* where the parent row shows a summary of the related sub-properties.

Note: Each sub-property can also have a default or master value and be set as read-only.

Editing Compound Property Values

1. Expand the compound property in the *Property Name Column*.
2. Edit the required compound sub-property value in the *Property Value Column*.

The compound property summary values will be updated to reflect property value changes made.

The *Property Editor* will check that the values entered are valid before applying them to the sub-properties.

Extended Editing Forms

For certain property value editing the structure of the *Property Editor* is insufficient to perform edits. In such circumstances extended editing forms are necessary.

For example, extended editing forms exist for the following properties:

- *Master* property (see [Editing Master Property Values](#))
- *Points* property (see [Editing Point Property Values](#))
- *Text* property (see [Editing Text Property Values](#))

Editing Master Property Values

Master property values exist, for example, for each instance *Master* property.

To edit a *Master* property value:

1. Click in the property value field for the *Master* property value you want to edit.

Note: You will need to expand the *Master* property if more than one *Master* property value exists.

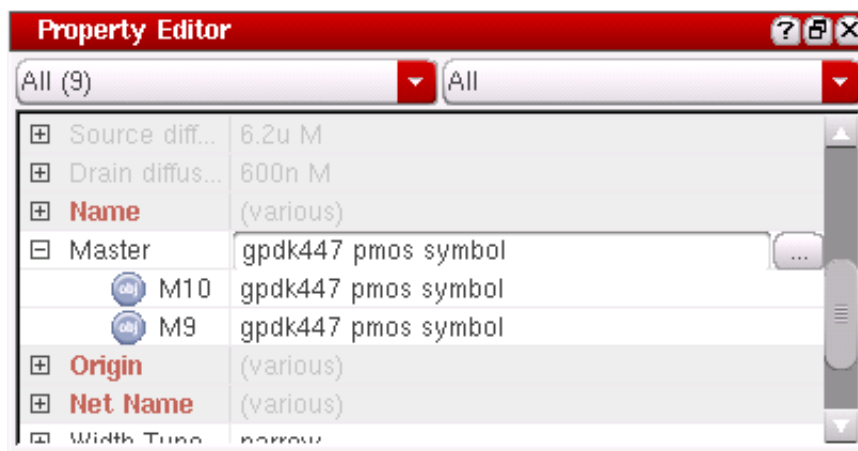


Figure 16-17 Click to expand editor represented by ellipses “...”

2. Select the *Click to expand editor* icon that is displayed to the right of the property value field.

This will display the Select Master form.

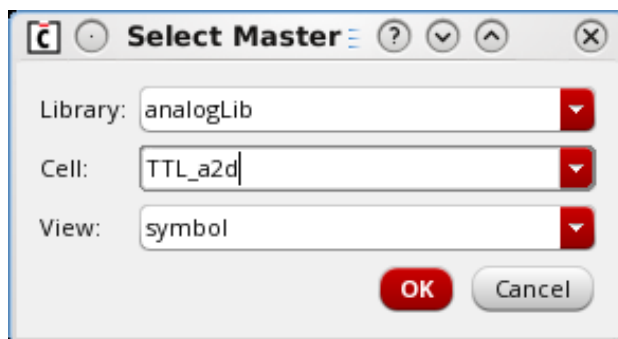


Figure 16-18 The Select Master form

Virtuoso Schematic Editor User Guide

The Property Editor Assistant

1. In the Select Master form edit the instance master *Library*, *Cell*, and *View* option settings as appropriate.

Note: You can either choose the required *Library*, *Cell*, and/or *View* name from the associated pull-down menus directly, or start to type in the required L/C/V name, and the Set Master form will start to auto-scroll to those entries that match what is being typed.

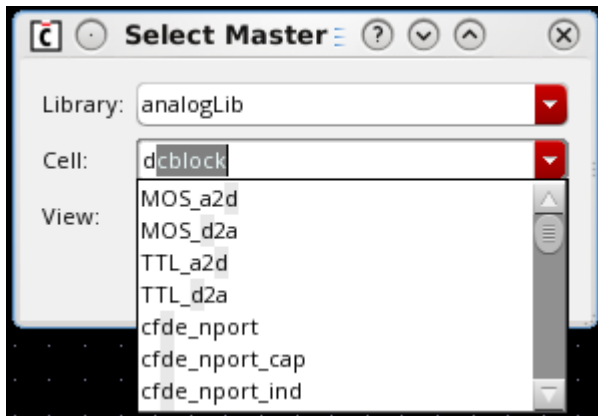


Figure 16-19 Auto-Scroll to Matching LCV Names

1. Click the *OK* button to apply the new *Master* property values.

Editing Point Property Values

Point property values exist, for example, for each wire segment *Point* property.

To edit a *Point* property value:

1. Click in the property value field for the *Point* property value you want to edit.
Note: You will need to expand the *Point* property if more than one *Point* property value exists.
2. Select the *Click to expand editor* icon that is displayed to the right of the property value field.

This will display the Edit Points form.

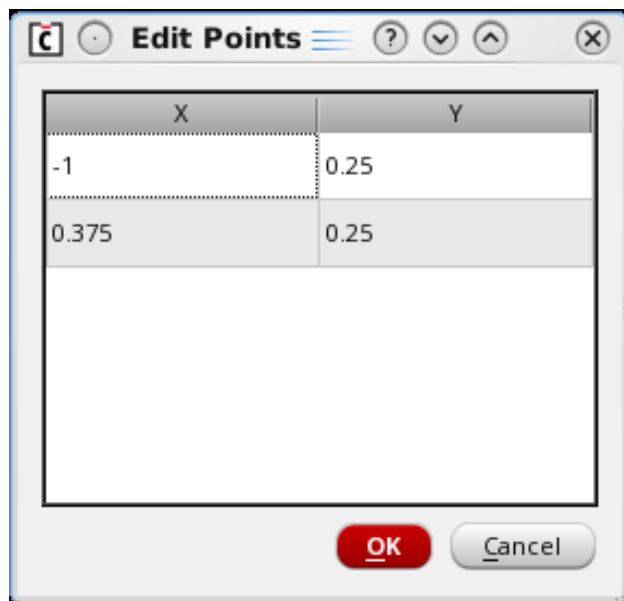


Figure 16-20 The Edit Points form

1. In the Edit Points form edit the *X* and *Y* point values as appropriate.
2. Click the *OK* button to apply the new *Point* property values.

Editing Text Property Values

Text property values exist, for example, for each note text *Text* property.

To edit a *Text* property value:

1. Click in the property value field for the *Text* property entry you want to edit.
Note: You will need to expand the *Text* property if more than one *Text* property value exists.
2. Select the *Click to expand editor* icon that is displayed.

This will display the Edit Text form.

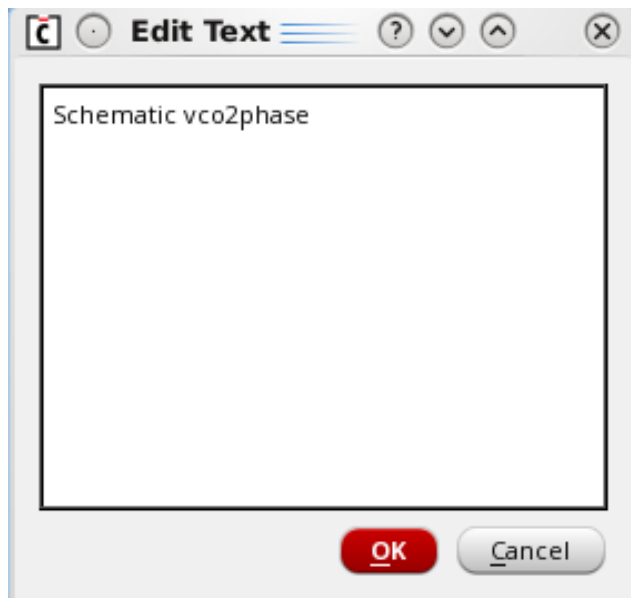


Figure 16-21 The Edit Text Form

3. In the Edit Text form edit the current note text entry as appropriate.

Note: You can also *Load* a text file to include as a note text.

4. Click the *OK* button to apply the new note text.

Property Status Visualization

The *Property Editor* provides a variety of visual mechanisms to display the current status of a property and its current value.

Default Values

Note: Some types of property, for example CDF Parameters, have a default value that is used until an alternative is provided.

Properties whose current values are the same as the property default value are displayed in *italic blue* text. This provides an indication of which properties have not yet been changed and those properties that have been manually altered.

Those property values that have been modified manually are displayed in standard black font.

Reverting to Default Values

If you have overridden a default value you can quickly revert back to the default:

1. Click in the property value field of the property you want to restore the default value for.
2. If the current value is different from the default value for the property a *Revert to default* button will be displayed to the right of the property value field.

The *ToolTip* for the button shows what the default value is.

Note: Where applicable, The Property Editor Context-Menu will also display a *Revert to default* option.

3. Click the *Revert to default* button to automatically change the value of the current property back to the default value for that property.

Virtuoso Schematic Editor User Guide

The Property Editor Assistant

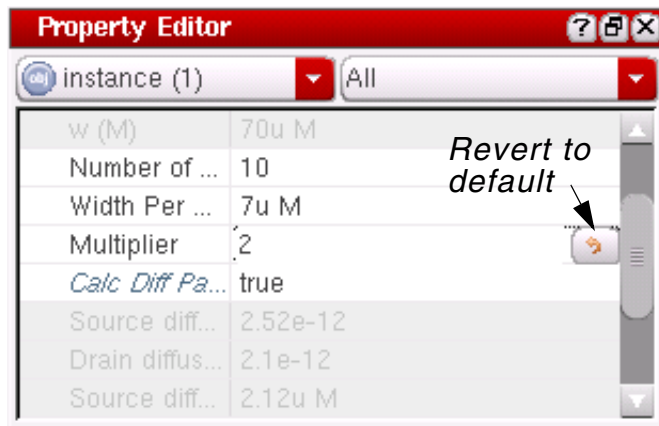


Figure 16-22 Option to revert to default property value

Read-Only Values

Property values that are set as read-only are displayed in gray font and cannot be edited.

Differences In Property Values Across Selected Objects

If there are differences in property values across selected objects, the property name will be highlighted in red font and the property value summarized by the string (*various*).

If there are no differences in property values across selected objects with the same property, a single value will be displayed.

The Property Editor and CDF Support

The *Property Editor* supports the viewing and editing of CDF (Component Description Format) parameters in much the same way as the [Edit Object Properties](#) form does. However, because the *Property Editor* is a dockable assistant, as opposed to a standard schematic editor form (such as the Edit Object Properties form), it is important to be aware of some differences in behavior.

This section summarizes the CDF features supported, while highlighting areas to be aware of.

Note: For a full discussion on CDF, see the [Component Description Format User Guide](#).

Callbacks

The *Property Editor* supports all CDF callbacks:

- `use`

Determines if the parameter is available to be displayed. If the expression evaluates to `t`, the parameter is available to the Property Editor and its display is controlled using the `display` callback.

If the expression evaluates to `nil` for a netSet, the schematic editor intentionally ignores this setting when the `CDS_Netlisting_Mode` is `Analog`. For more details see, [Effect of Netlisting Mode on netSets](#).

- `display`

Controls whether or not a parameter is to be shown in the *Property Editor*.

- `editable`

Specifies whether a parameter is editable or read-only.

- `callback`

Is evaluated whenever a parameter's value is edited in the *Property Editor*.

For more information on the above, see [Defining Parameters](#) in the [Component Description Format User Guide](#).

Virtuoso Schematic Editor User Guide

The Property Editor Assistant

In addition, the initialization and postprocessing procedures described in the [Component Description Format User Guide](#) are also supported.

■ `formInitProc`

Is executed prior to displaying CDF parameters in the *Property Editor*.



The *Property Editor* displays CDF data for selected instances. In order to display the correct data, the *Property Editor* must first call `formInitProc`. This means that this procedure has to be called any time you select an instance.

Note: Hiding the *Property Editor* assistant will cause it to stop tracking selections. This will also prevent `formInitProc` being called.

■ `doneProc`

Is executed after a change is made to any parameter.

The *Property Editor* operates in a similar manner to a spreadsheet application, in that individual parameter changes are applied immediately. When a parameter value is changed, the *Property Editor* performs the following operations:

- ☐ It will call `formInitProc` again to ensure that the `cdfgData` variable is correctly set up for use in any subsequent parameter callbacks.
- ☐ It applies the parameter change to the current instance.
- ☐ It calls the parameter callback.
- ☐ It calls `doneProc`.

Note: If a parameter value is changed across several devices, the above sequence will be performed for each object.

cdfgData

`cdfgData` is used within callback code to access CDF data for the current object.

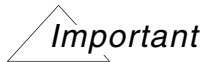
The *Property Editor* will set up this variable correctly even in cases where multiple objects are selected.

For more information on `cdfgData` see [CDF SKILL Summary](#) in *Virtuoso Analog Design Environment L SKILL Reference*.

cdfgForm

`cdfgForm` is the form on which CDF data is displayed, if there is one.

However, as already mentioned, the *Property Editor* is a dockable assistant, different from standard schematic editor forms. For this reason, `cdfgForm` is set to `nil` when using *Property Editor* to display CDF.



You should ensure that your callback code can handle a case where `cdfgForm` is `nil`.



As there is no form, the `cdfgForm -> cdfModified` flag is not supported in the *Property Editor*.

For more information on `cdfgForm` see [CDF SKILL Summary](#) in *Virtuoso Analog Design Environment L SKILL Reference*.

Emulating cdfgForm

It is possible to set up the *Property Editor* to emulate `cdfgForm`. This can be useful in cases where it is not possible to change the way `cdfgForm` is used in your CDF definition (and access to `cdfgForm` is limited to certain properties).

Setting the SKILL variable “`oiEmulateCdfgForm`” to “`t`” will cause the *Property Editor* to set `cdfgForm` to be effectively the same as [cdfgData](#). This means that all of `cdfgData`’s properties are accessible through `cdfgForm`.

With emulation enabled, the *Property Editor* will also display a warning any time a callback tries to access `cdfgForm`. This is done to alert you to the fact that the callback operation may not be working as intended.

Note: The warning is only emitted once per cell.

In cases where the only use of `cdfgForm` is to access parameter values, for example `cdfgForm -> parameter -> value`, then `cdfgForm` and `cdfgData` are interchangeable, and emulation will yield correct results. If this is the case, then the recommended course of action is to modify the callback so that it does not use `cdfgForm`.

Virtuoso Schematic Editor User Guide

The Property Editor Assistant

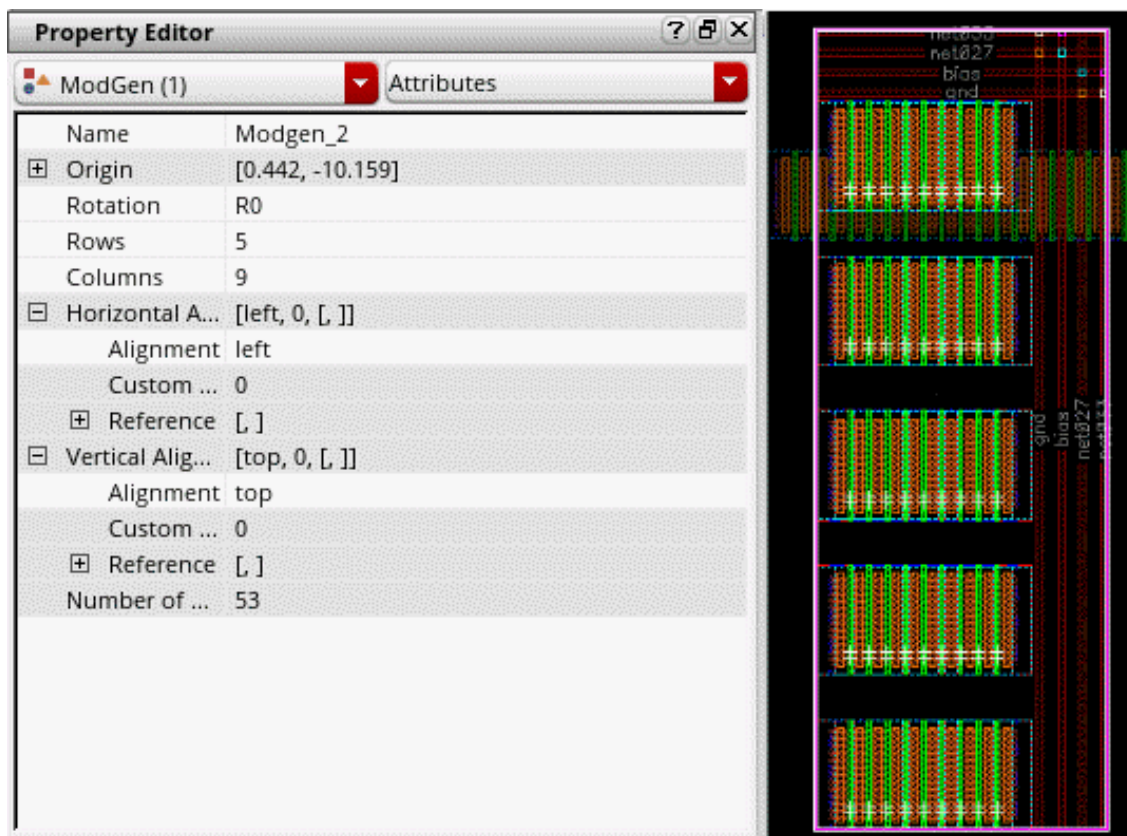
In cases where modifying the CDF definition is not possible, and you have verified that emulation is giving the intended results, the `cdfgData` warning message can be disabled by setting the SKILL variable “`oiSuppressCdfgFormMessages`” to “t”.

Important

The emulated `cdfgForm` is not a *genuine* form. If you are using this feature, be sure to examine your callback code to verify that it will operate correctly.

Modgen Property Updates using the Property Editor Assistant

You can use the Property Editor assistant to interactively update the properties of individual Modgen instances and Modgen arrays containing stack-type members. Any changes you make to the instance properties are immediately applied to the Modgen in the layout canvas, as displayed in the figure below.



You can also use the Array Assistant to edit Modgen properties.

Related Topics

[Module Generators \(Modgens\)](#)

[Specifying Modgen Parameters](#)

[Support for Stacks in Modgens](#)

Virtuoso Schematic Editor User Guide

The Property Editor Assistant

The Probes Assistant

This chapter describes the following topics:

- [Introducing the Probes Assistant](#)
- [Accessing the Probes Assistant](#)
- [Hiding the Probes Assistant](#)
- [The Probes Assistant Graphical User Interface \(GUI\)](#)
- [Displaying Probed Nets in the Probes Assistant](#)
- [Controlling the Display in the Probes Assistant](#)
- [Saving Probe Paths to a File](#)
- [Removing a Probe or All Probes](#)
- [Context-Menu Options](#)

Introducing the Probes Assistant

The Probes Assistant is a dockable assistant pane that provides facilities to view the probed nets across the design hierarchy.

When working on a design with probed nets, you may want to view the schematic hierarchy or the lib/cell/view of a specific probed net. For a simple design, you can effectively trace the probed net using the Navigator Assistant. However, with a large design containing multiple probed nets, it may become difficult to trace the probed net you are working on in the design canvas. For more details, see [Tracing the Hierarchy of a Probed Net Using the Probes Assistant](#).



Video

See related videos:

1. To find out how to use the Probes Assistant to identify connections that exist for a net and to save probe path information to a CSV file, see [Identifying Net Connections](#).
2. To find out how to use the SKILL [schTraceNet](#) to calculate the connected area of a net, see [Probing and Calculating the Area of a Net](#).

Benefits of using Probes Assistant

- View the hierarchy of the probed path and the library/cell information of the probes.
- Instances with a global signal are highlighted if they are probed at the top-level. The assistant traverses the hierarchical path, therefore, instances corresponding to the top-level instance are also highlighted down the hierarchy.
- The information that the assistant displays in the Display Area can be filtered using the Filter options and Search Widget.
- Cross-selection is available from/to the assistant, where selecting, for example, a probed net in the assistant will cause the same probed net to be highlighted in the design and in the Navigator Assistant.
- Each object on a Probe Path is easily identifiable because the background of the icon of each object is set to the color of the probe.
- Save the probe paths to an exportable file.

Accessing the Probes Assistant

To access the Probes Assistant in Schematics, do one of the following:

- Select *Window – Assistants – Probes*.
- Right-click in the main Schematics menu or toolbar area and select *Probes*.

Once selected, Schematics will embed the Probes as a docked assistant pane within the current session window. By default, the assistant will be positioned in the lower area of the session window.

Hiding the Probes Assistant

To hide the Probes Assistant in the current session window, do one of the following:

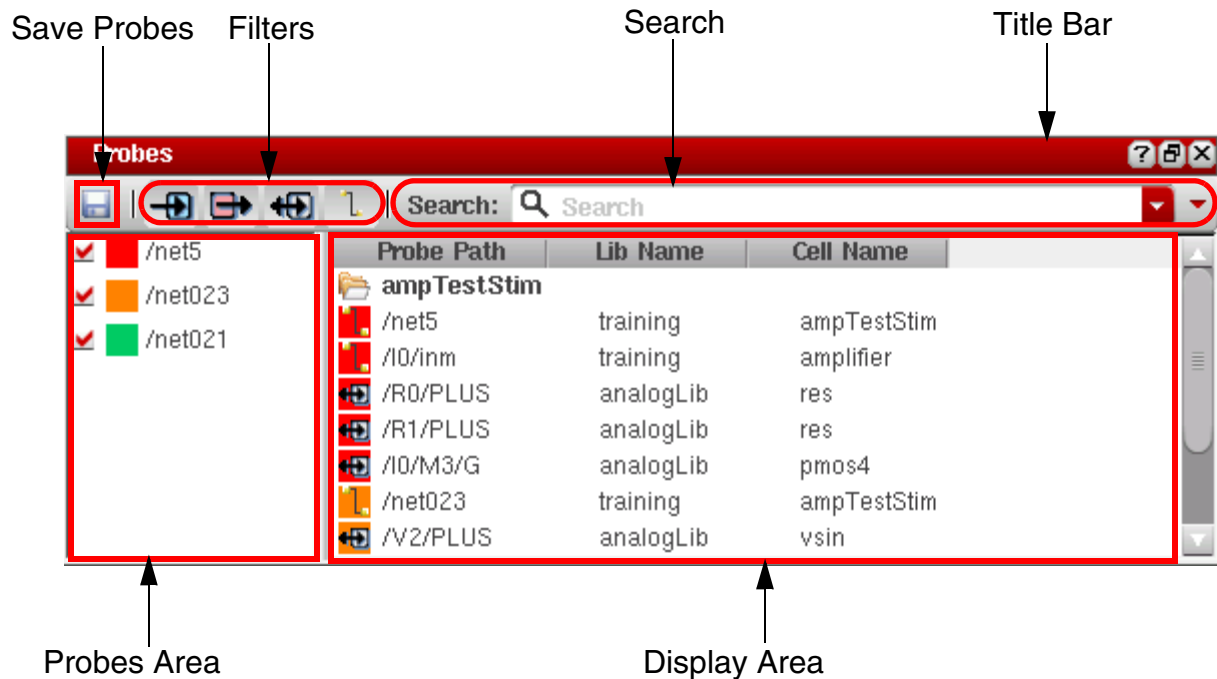
- Right-click in the Schematics menu or toolbar area and de-select the *Probes* option.
- Click the Hide button (X) in the Probes Assistant pane title bar.

The Probes Assistant Graphical User Interface (GUI)

The Probes Assistant includes the following GUI elements:

Virtuoso Schematic Editor User Guide

The Probes Assistant



- Save icon to save the probed paths to a file. For more information, see [Saving Probe Paths to a File](#).
- The Filters allow you to filter the objects in the Display Area. For more information, see [Using Probes Filters](#).
- The Search allows you to search objects in the Display Area. For more information, see [Using the Search Widget](#).
- The Probes Assistant is divided into two main sections:
 - On the left is the Probes Area which contains the list of the probed nets. Each probe has a check box which you can use to control the information being displayed. For more information, see [Using the Probes Area](#).
 - On the right is the Display Area which displays the names of the probed objects and their types. For more information, see [Displaying Probed Nets in the Probes Assistant](#).

Displaying Probed Nets in the Probes Assistant

By default, the Probes Assistant displays all the probed objects across the design hierarchy in the Display Area. It contains the following columns that can be resized and sorted in ascending or descending order:

Virtuoso Schematic Editor User Guide

The Probes Assistant

- *Probe Path* – Schematic hierarchy of the probed object.
- *Lib Name* – Name of the library where the probed object is located.
- *Cell Name* – Name of the cell where the probed object is located.

The Display Area also contains a number of features that enable you to better interpret results:

- Each object on a Probe Path is easily identifiable because the background of the icon of each object is set to the color of the probe.
- Clicking an object on the Probes Assistant, causes the same object to be highlighted in the Navigator Assistant and double-clicking causes the same object to update the design canvas.
- When an instTerm is selected in the Probes assistant, a highlight is added around the parent instance of the instTerm in the canvas.
- Right-click menu options provide access to additional probe options. For more information see [Context-Menu Options](#).
- Hovering the mouse cursor on an object displays a ToolTip detailing full information on the current object.
- Improve readability of the results by changing the path separator using the environment variable [alternateHierSeparator](#).

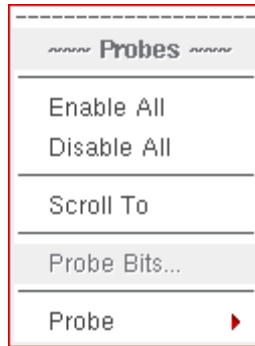
Context-Menu Options

You can use the context-menu options in the Probes Assistant to manage the probed nets.

- [Using Context-Menu Options in Probes Area](#)
- [Using Context-Menu Options in Display Area](#)

Using Context-Menu Options in Probes Area

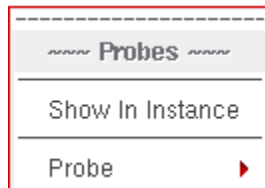
Right-click and choose from one of the following, as per requirements:



<i>Enable All</i>	Enables all the probes to display objects in the Display Area.
<i>Disable All</i>	Disables all the probes.
<i>Scroll To</i>	Highlights and scrolls to the selected in the Display Area.
<i>Probe</i>	Provides access to probes options. For more information, see Using Probes Sub-Context Menu .

Using Context-Menu Options in Display Area

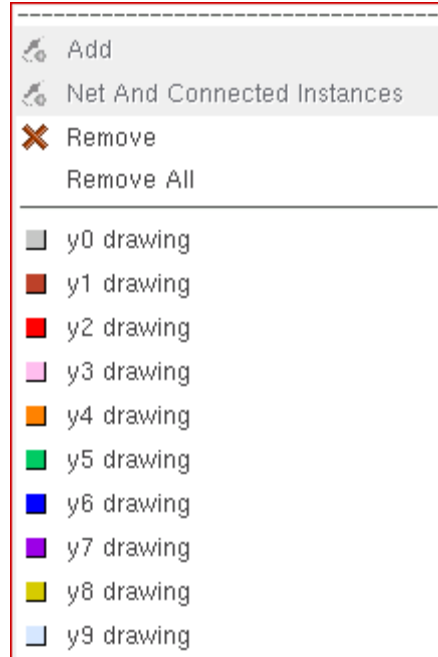
Right-click and choose from one of the following, as per the requirements:



<i>Show In Instance</i>	Displays the selected instance in the design canvas. For example, for a /I1/VSS net, it will open I1 and display VSS.
<i>Probe</i>	Provides access to probe options. For more information, see Using Probes Sub-Context Menu .

Using Probes Sub-Context Menu

You can choose from one of the following, as per the requirements:



Note: *Add* and *Net and Connected Instances* options are disabled for a selection containing already probed objects.

Remove Removes the selected probe.

Remove All Removes all the probes from the design.

Color Range A color to distinguish the probe path from others can be applied.

Controlling the Display in the Probes Assistant

By default, the Probes Assistant displays all the probes across the design hierarchy. You can control the information being displayed as follows:

- [Using Probes Filters](#)
- [Using the Search Widget](#)
- [Using the Probes Area](#)
- [Changing the Color of the Probe Path](#)

Using Probes Filters

All types of objects in the probed path are displayed in the Display Area. Use the Probes Filters to remove/display particular types of design objects that you want to display in the Display Area. For example, you may choose to only view nets.

Note: By default, The Input, Output, and InputOutput pins connected to the probed net have been classified as Loads, Drivers, and InputOutput respectively.

To remove/display the different types of probe, toggle the following buttons as required:

 Loads

 Drivers

 InputOutput pins

 Nets

Using the Search Widget

Control the display of objects using the Search Widget.

Note: Search facilities in the Probes only operate on those columns that are currently visible in the Display Area.

Once you begin to type your search criteria into the Search, the Display Area will automatically update to reflect this. For example, if you want to display objects that include v, enter v in the search field, the Display Area will be updated as shown in the figure below:



Note: The search criteria is saved in the history. You can delete this history by clicking *Clear History* in the *Search* drop-down list box.

Using the Probes Area

In the Probes Area, you can select/deselect the check boxes of the probed nets in the probes list as per your requirements. The selected probes will be displayed in the Display Area. Note in the figure below, when you select the check box for `/net023`, the Display Area only displays the probes for the selected net.



Changing the Color of the Probe Path

Each object on a Probe Path is easily identifiable because the background of the icon of each object is set to the color of the probe.

A color is automatically selected. If the last color used to probe an object was *y3 drawing*, then *y4 drawing* will automatically be used next in line.

To change the color of a probed net:

1. Right-click in the Display or Probes Area.
2. From the context and sub-context menus select *Probes - Probe* and select the color required.

Alternatively, while selecting objects to probe, press 9 to rotate through the different color options for that probe path.

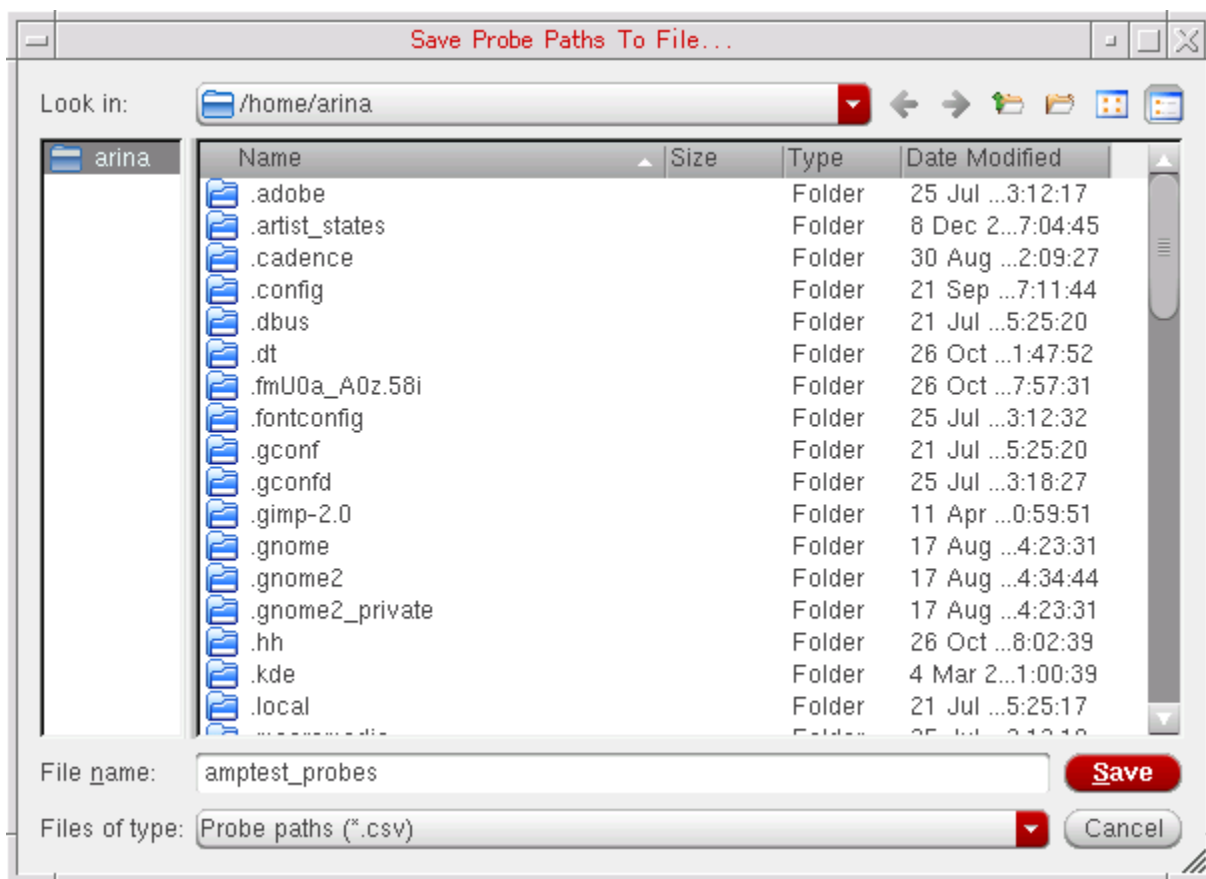
Saving Probe Paths to a File

The probe path information displayed in the Display Area can be saved and exported into a .csv file.

To save probe paths to a file, do the following:

1. Click .

The *Save Probe Paths To File* form appears.

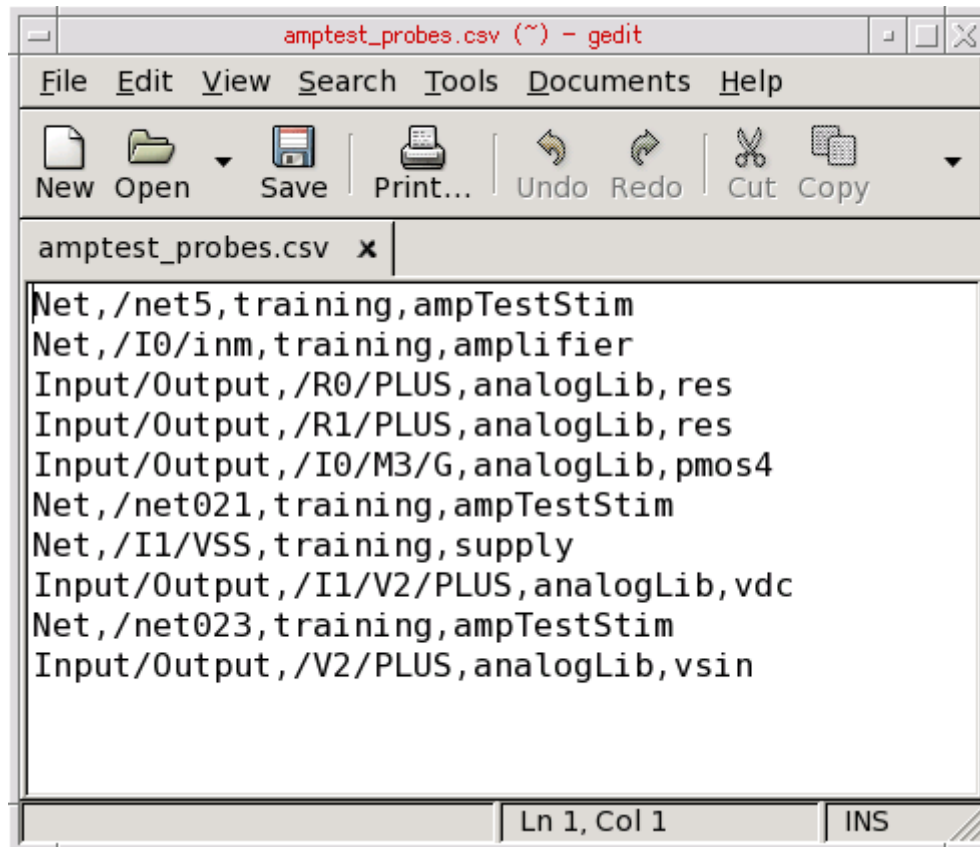


2. In the *Files of Type* list, select *Probe paths (*.csv)*.
3. In the *File name* field, specify a name for the file to which the probe paths will be saved.
4. Click *Save*.

Virtuoso Schematic Editor User Guide

The Probes Assistant

The contents of file are as shown in the figure below:



Note: This does not save the probes on your design. To save the probes, you save them to a file so they can be reattached next time you open your design. See the section [Saving Probes](#) for more details.

Removing a Probe or All Probes

To remove a single probed net from a design:

1. Right-click on the selected probe in the Display or Probes Area.
2. From the context and sub-context menus select *Probes - Probe - Remove*.

To remove all the probed nets from a design:

1. Right-click in the Display or Probes Area.
2. From the context and sub-context menus select *Probes - Probe - Remove All*.

Virtuoso Schematic Editor User Guide

The Probes Assistant

The World View Assistant

The *World View* assistant displays a panoramic view of an entire schematic, indicating which portion of it is currently visible in the editor window. This is especially useful navigation aid when working at high zoom levels.

On launch, the *World View* is positioned in the lower left-hand corner of the Schematics session window. It can then be undocked and re-positioned as required.

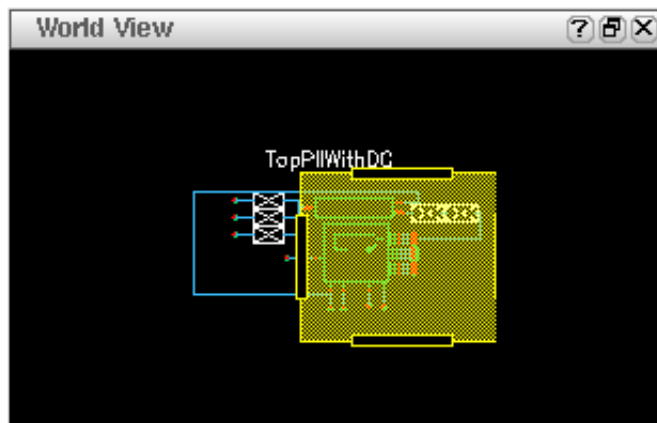
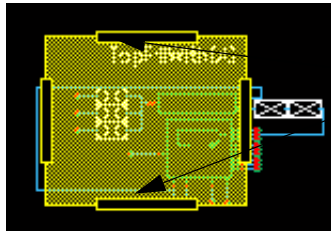


Figure 18-1 The World View assistant pane

The current design area on display in the canvas is shown as a yellow rectangle in the *World View*. You can resize this rectangle to amend the design display area in the canvas.

- Drag the box around to pan the display without affecting the zoom level.

- Drag a handle on the edge of the box to show more or less of the design (that is, decrease or increase the zoom level)



Drag here to change zoom levels

Figure 18-2 World View handle being dragged to change zoom level

To add the *World View* assistant to your current session window either:

- Select *Windows – Assistants – World View*.
- or
- Right-click in Schematics menu or toolbar area and select *World View*.

Note: The *World View* is not included in any of the default workspace configurations. You will have to actively add it to your current session window.

World View Context-Menu

Right-click over the *World View* assistant to display a number of related operations that can be performed directly from the *World View* context-menu.

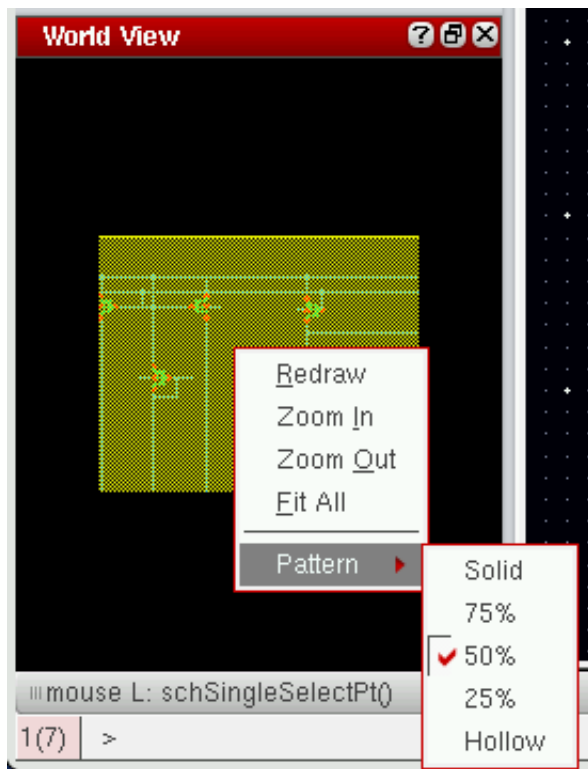


Figure 18-3 World View Assistant Context-Menu

From here, you can refresh the current window (*Redraw*), *Zoom In* and *Zoom Out* of the current view, auto-fit the design to the *World View* assistant window (*Fit All*), or select the background *Pattern* of the design display (yellow rectangle).

Virtuoso Schematic Editor User Guide

The World View Assistant

Exporting a Schematic Hierarchy to HTML



Exporting a schematic hierarchy to HTML is only available in *Virtuoso Schematic Editor XL*.

Schematic HTML Publisher Overview

You can export a schematic design hierarchy to HTML by selecting *File – Export HTML* from Virtuoso Schematic Editor XL.

The schematic design hierarchy is published in a format that can be rendered and displayed in a web browser. Once in HTML, various aspects of the design can then be traversed and navigated as part of a multi-frame display format.

Use Scenarios

The anticipated scenarios, and use models, where it is expected that benefits will arise from the ability to export a schematic hierarchy to HTML include the following:

- Design review situations where access to a company's intranet is not possible, but schematic hierarchy information is still required to be accessed and analyzed.
- Design leads who may be working offline, without access to Virtuoso, but can still require to review schematic hierarchy structures and details.
- Design publishing situations where elements of design collateral can be exported and distributed to customers who may not, for example, have access to Virtuoso.

Supported Browsers

Cadence recommends the use of Mozilla Firefox or Google Chrome for best results using the HTML schematic hierarchy export facility. However, performance on both Opera and Safari is considered good, while Internet Explorer (unsupported) works with limited functionality.

Important

You need to use the `--allow-file-access-from-files` option when launching `Chrome.exe` on a file. This option is not needed when accessing the HTML report via HTTP.

```
chrome.exe --allow-file-access-from-files <local path>\temp\temp\index.html
```


Exporting a Design Hierarchy to HTML (non-configured schematic)

Important

Exporting design content from a configured schematic largely follows the same process as doing so for a non-configured schematic. However, for additional information on exporting the design hierarchy for a configured schematic, see [Exporting a Design Hierarchy to HTML \(configured schematic\)](#).

To export your schematic design hierarchy to HTML:

1. In Virtuoso Schematic Editor XL, open the design with the hierarchy you want to export to HTML.
2. Select *File – Export HTML*.

The [Export Schematic Hierarchy](#) form is displayed.



The dialog box titled "Export Schematic Hierarchy" contains the following fields and options:

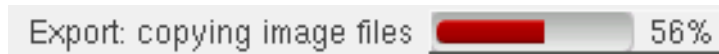
- Starting cell:** Radio buttons for ☒ current and ☐ top.
- View list:** Text field containing "schematic symbol".
- Stop list:** Text field containing "symbol".
- Export to Directory:** Text field containing "/tmp/" with a browse button (...).
- Image Color Options:** Radio buttons for ☒ fullcolor, ☐ grayscale on white, and ☐ black on white.
- Open in Browser?:** Checkmark icon.
- Buttons:** OK (red), Cancel, Defaults, Apply, and Help.

Virtuoso Schematic Editor User Guide

Exporting a Schematic Hierarchy to HTML

3. Specify your design export options.
4. Click the *OK* button to commence the schematic design hierarchy export process.

A progress bar will be displayed in the status bar, giving step-by-step export updates.



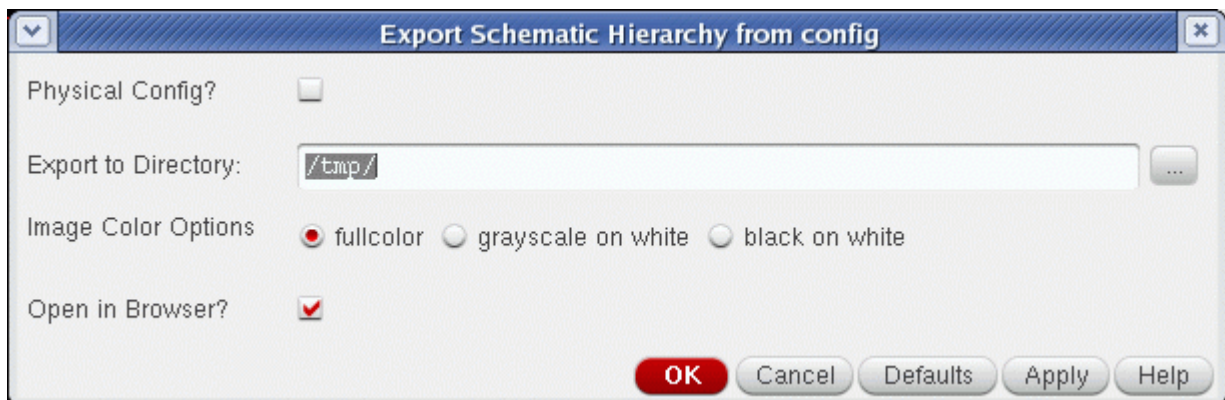
On completion, the exported schematic design will be displayed in your browser (see [Navigating Exported HTML in a Web Browser](#)).

Exporting a Design Hierarchy to HTML (configured schematic)

Exporting design hierarchy content in HTML format from a configured schematic largely follows the same process as doing so for a non-configured schematic.

Note: For information on exporting the design hierarchy for a non-configured schematic, see [Exporting a Design Hierarchy to HTML \(non-configured schematic\)](#).

However, in a configured schematic, you do not specify the *Starting Cell*, *View List* or *Stop List* options (as they are already specified within the configuration itself), in the [Export Schematic Hierarchy](#) form, rather you are required to select whether or not the *Physical Config* is to be used to perform hierarchy traversal. If you do not choose to use the physical configuration in this way, standard traversal, as performed by the Cadence Hierarchy Editor, will be used.



Navigating Exported HTML in a Web Browser

After exporting your design hierarchy to html your web browser will display your design information in a multi-framed format.

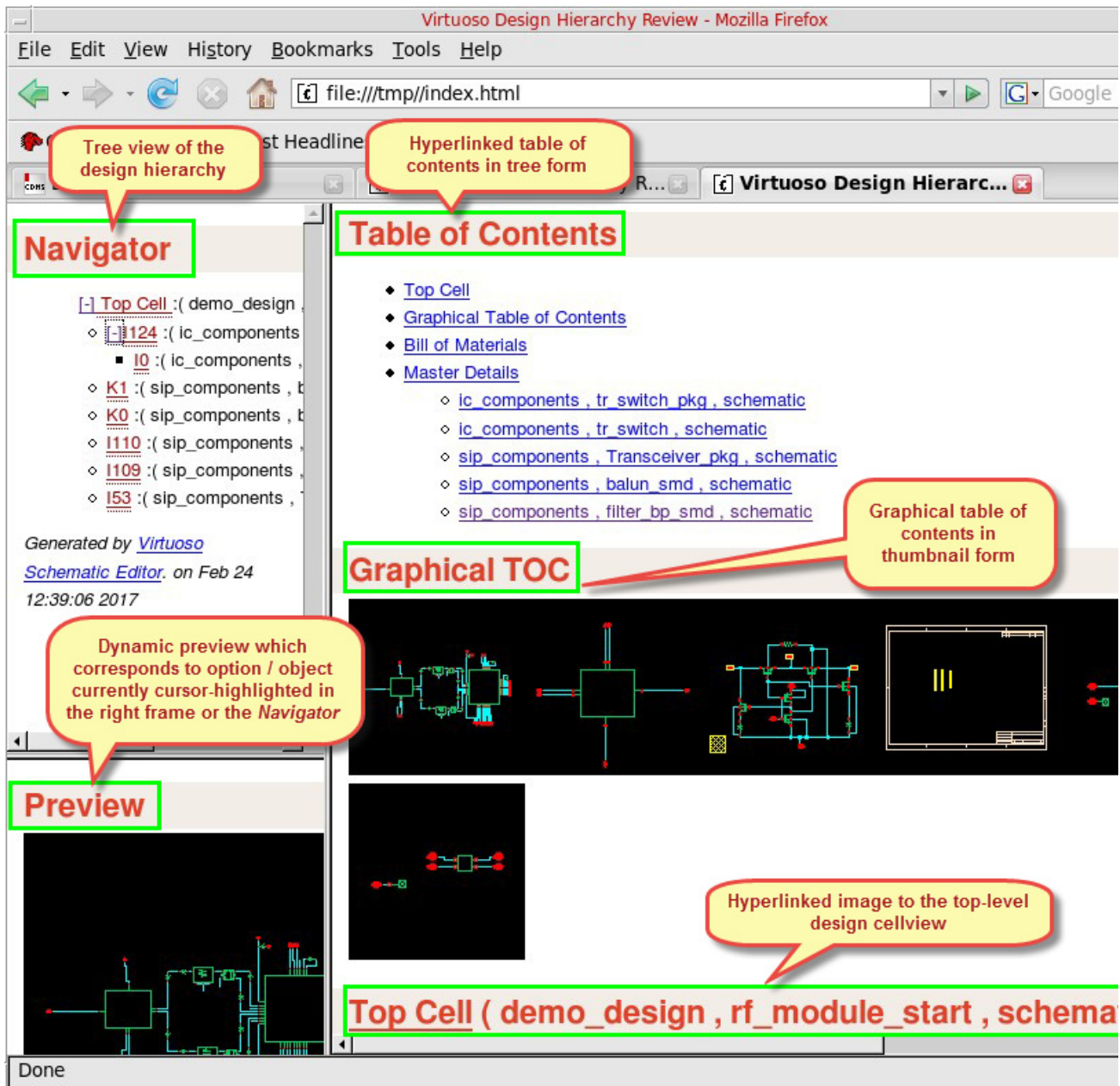


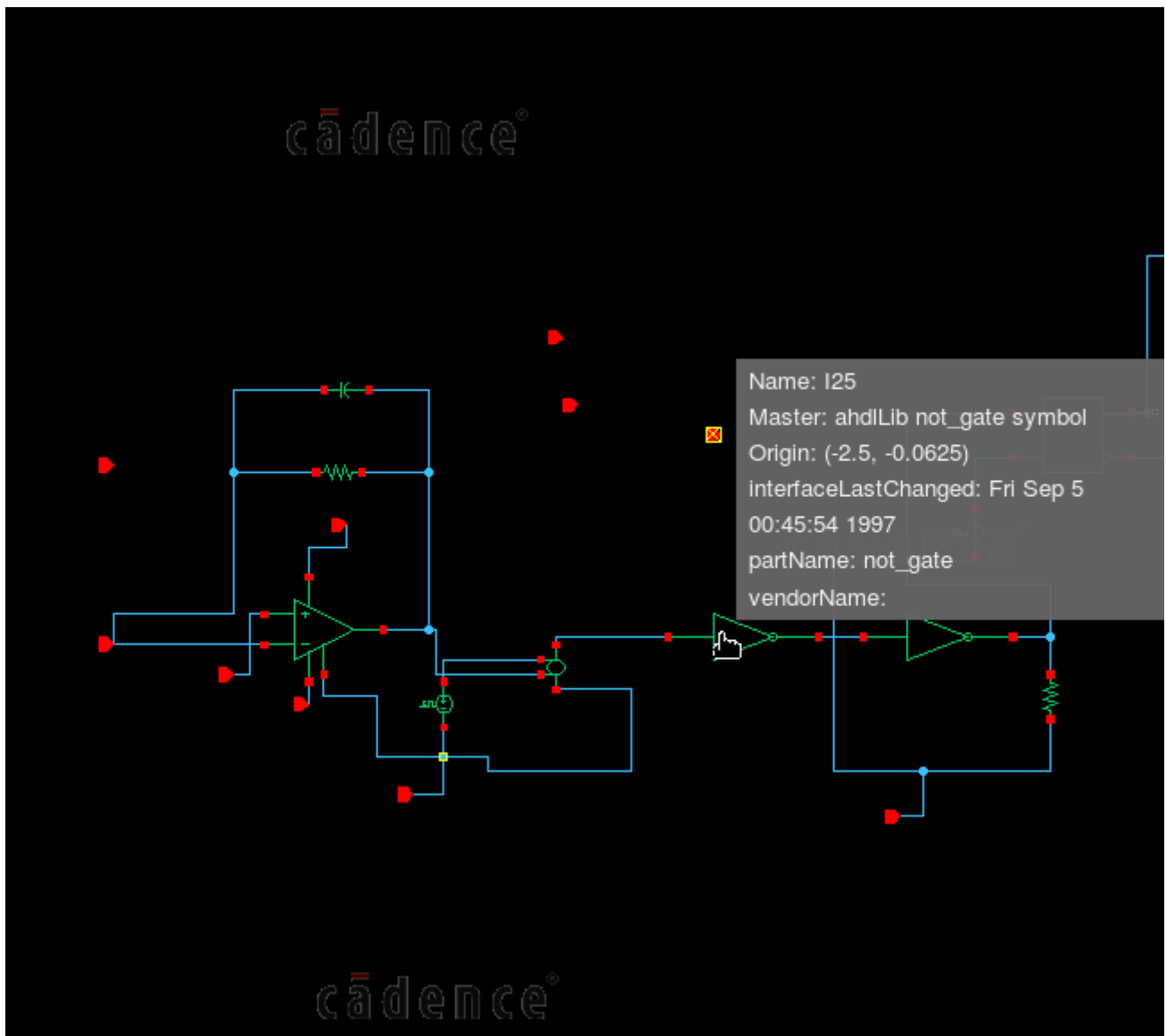
Figure 19-1 A Published Schematic Design in HTML Format in Mozilla Firefox

Virtuoso Schematic Editor User Guide

Exporting a Schematic Hierarchy to HTML

You can also view the property and attribute information in a tooltip when you move the mouse over an instance, an instance pin, or a pin.

- The tooltip of an instance displays the name, master library, origin, property names, and property values.
- The tooltip of an instance pin displays the name, instance name, and connected net name.
- The tooltip of an instance pin displays the terminal name, direction, and pin master.



Additionally, as you scroll down the exported design hierarchy, there are further sections covering a *Bill of Materials* (which lists the non-leaf cells in the design hierarchy), and

Master Details for the design, where there are separate images and sections for each non-leaf cell in the design.

Note: All corresponding thumbnail images are hyperlinked for ease of traversal.

The Dynamic Preview Pane

The *Preview* pane's content is dynamically updated dependent upon the cursor position in the main frame. By default, when the cursor is not over an image, the top level cell will be displayed.

Note: As well as being sensitive to images that are cursor-highlighted in the main frame, the *Preview* pane will also reflect text hyperlinks in the *Navigator* pane, entries in the *Bill of Materials*, and any graphical or textual element in the *Table of Contents*.

Additionally, if the cursor is positioned over a non-leaf symbol instance (in any relevant image in the main frame), and a corresponding sub-schematic exists for that instance, then the sub-schematic will be shown in the pane.

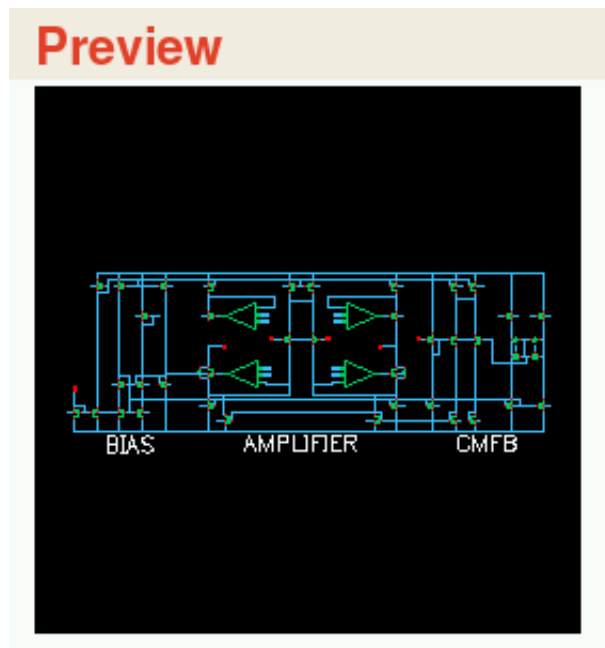


Figure 19-2 Preview Pane

Virtuoso Schematic Editor User Guide

Exporting a Schematic Hierarchy to HTML

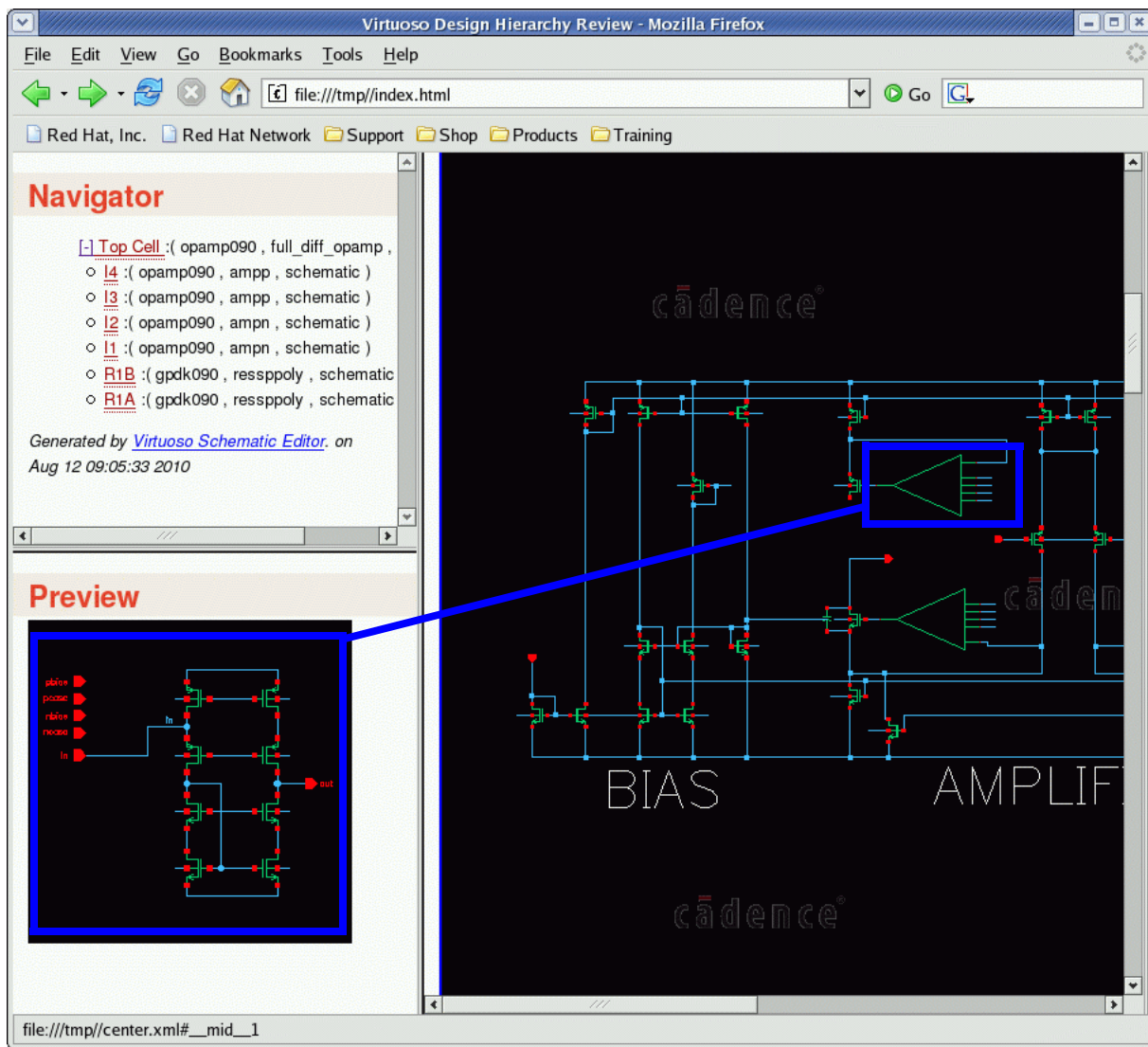


Figure 19-3 Sub-Schematic Shown in Preview Pane for Cursor-Highlighted Instance

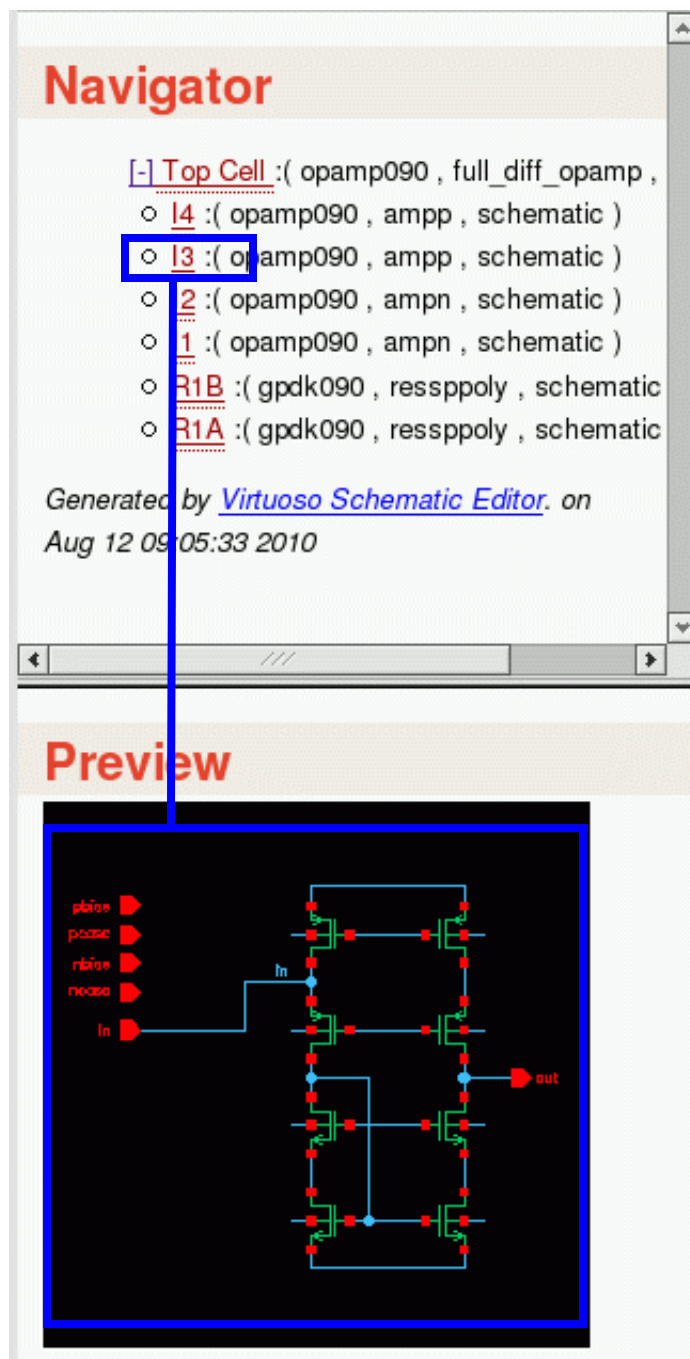


Figure 19-4 Instance I3 Cursor-Highlighted in Navigator Pane and Reflected in the Preview Pane

The Navigator Pane

The *Navigator* pane can be used to explore the design hierarchy.

Initially displayed in a collapsed format, the *Navigator* pane can be expanded to display the design hierarchy in a tree view structure.

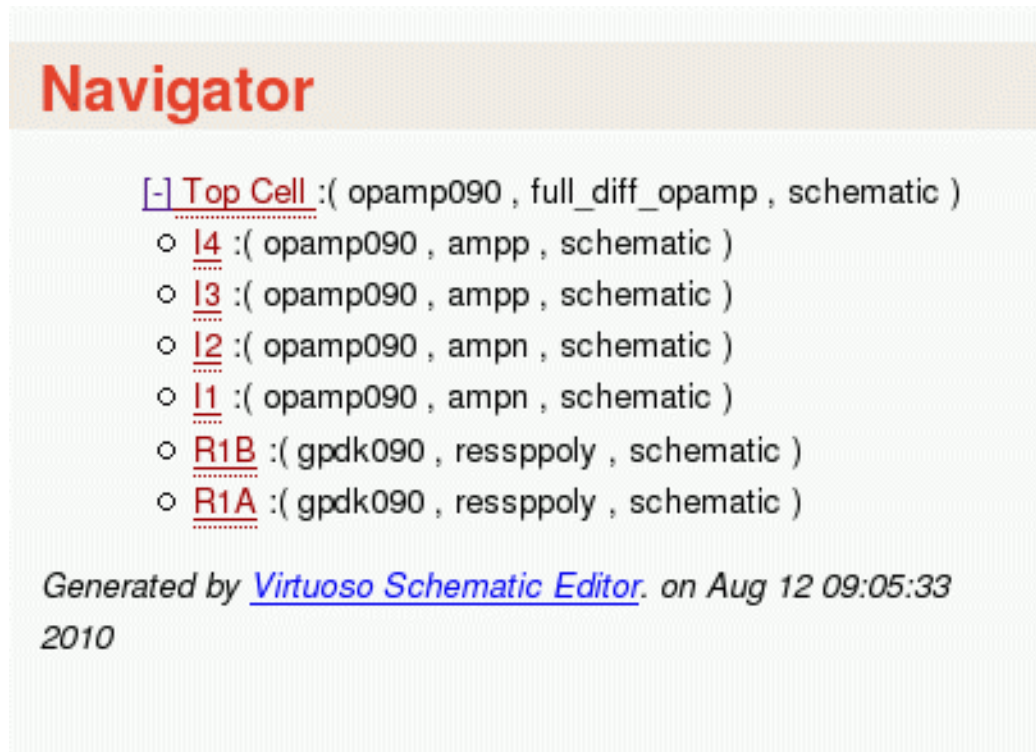


Figure 19-5 The Navigator Pane with the Top Cell Expanded

- Placing your cursor over any of the red-colored hyperlinks in the *Navigator* pane, will update the *Preview* pane to display the corresponding, underlying, object schematic (assuming that it is not a leaf-level object).
- Clicking on any of the red-colored hyperlinks will jump to the corresponding object image in the *Master Details* section in the main frame (compare this action with a schematic descend operation) to quickly assist review and exploration.

Note: In Firefox, for example, you can also use the *Go back one page* and *Go forward one page* toolbar options to perform ascend and descend operations, as if you were navigating a number of hyperlinked web pages.

Additional Navigation Within the Main Frame

You can further navigate the design hierarchy in the main frame by performing any of the following actions:

- Clicking on a (hyperlinked) instance.
- Clicking on an entry in the *Table of Contents* section.
- Clicking on an entry in the *Bill of Materials* section.

By performing any of the above actions, the main frame will adjust its content to display the corresponding object's *Master Details*.

Note: Again, these selections can be compared to performing a descend operation, but with the advantage of the descent being performed with a single mouse-click.

The Bill of Materials

The *Bill of Materials* section, in an exported design hierarchy, can be located by scrolling down through the main frame display, or by selecting the *Bill of Materials* hyperlink in the *Table of Contents* at the top of the page.

The purpose of the *Bill of Materials* is to list any non-leaf cells used in a design, in tabular format.

Bill of Materials					
Count	Library	Cell	View	Click to Visit	Hover to Preview
2	opamp090	ampp	schematic	link	preview
2	opamp090	ampn	schematic	link	preview
2	gpdtk090	ressppoly	schematic	link	preview

Figure 19-6 The Bill of Materials Table

- The table is initially sorted to show all of those non-leaf cells (sorted by the secondary order of an increasing number of instances count) which belong in the same library as the top level design. This display order is then followed by all remaining non-leaf cells listed by increasing order of instance count.
- Clicking on a table header (for example, *Library* or *Cell*) will sort the table based on the table header criteria, in either ascending or descending order.
- Selecting a *link* hyperlink will jump to that lib/cell/view in the *Master Details* section.
- Hovering your cursor over a *preview* option will display a (ToolTip) thumbnail image of the corresponding object.

Printing an Exported Design



Tip

As mentioned, you can choose to export your schematic design hierarchy to HTML in either *full color*, *grayscale on white*, or *black on white*. Whatever selection you make should take into consideration any future intention to print design data.

Note: Any design images displayed in the exported, or printed, design data will have Cadence watermarks. To know how to change this watermark, refer to [Customizing the HTML Files](#).

To print exported design data:

1. Select the *Printer Friendly Version* option at the top of the main frame.
2. Select *Print this Frame*.

The Print window will be displayed.

3. Select *Print*.

The printed document will include the main frame contents along with the fully expanded contents of the *Navigator* pane; appended to the bottom of the printed document.

Note: The printed document will resize images as appropriate, and create page breaks where necessary. This will ensure that each primary schematic/sub-schematic image will be printed on separate pages.

Customizing the HTML Files

The exported HTML pages are formatted using a set of predefined formatting instructions contained in a set of `.xsl` and `.css` files that are exported along with the schematic design hierarchy data. The design hierarchy data is in the XML format. The web browser uses these formatting instructions to create the HTML frames and pages and renders the design hierarchy data in the HTML format.

By default, to render the design data in the HTML format, Schematic Editor uses the `.xsl` and `.css` files provided at the following default location:

```
<CDS_INST_DIR>/share/cdssetup/dfII/publisher/html/
```

where, `<CDS_INST_DIR>` is the directory in which Virtuoso is installed.

If required, you can edit the `.xsl` and `.css` files to customize the formatting instructions contained in these files. For example, you can change the font type, add or remove contents displayed in the HTML files, or change the logo displayed with the design.

To customize the default format of the HTML files:

1. Create the `/dfII/publisher` directory structure in the `.cadence` directory in your UNIX home.

```
mkdirhier ~/.cadence/dfII/publisher
```

2. Copy the default `html` directory to the directory that you created in Step 1.

```
cp -r <CDS_INST_DIR>/share/cdssetup/dfII/publisher/html ~/.cadence/dfII/publisher
```

The copied files include the default `.xsl` and `.css` files.

3. Edit the formatting instructions contained in the `.xsl` and `.css` files that you copied to the `~/.cadence/dfII/publisher/html` directory.

When you next export a design hierarchy to HTML, the `.xsl` and `.css` files from the `.cadence` directory in your UNIX home are used instead of the files from the default location.



Tip

In addition to the existing set of files, you can reference some additional `.html`, `.xsl`, or image files in the `.xml` or `.css` files. For this, use the `CDS_EXPORT_CUSTOM_FILES` shell environment variable to specify a space-separated list of paths to these files. The files specified with this variable are exported along with the schematic design hierarchy.

Virtuoso Schematic Editor User Guide

Exporting a Schematic Hierarchy to HTML

RF Transmission Lines

(Virtuoso RF Solution Option)

This chapter provides information about the options available from the *RF-Module* menu in Schematics XL.

The following sections are included:

- [RF Workspace and toolbar](#)
- [Inflate](#)
- [Create Net Topology](#)
- [Create Stackup](#)

Related topics

[*Virtuoso RF Solution Guide*](#)

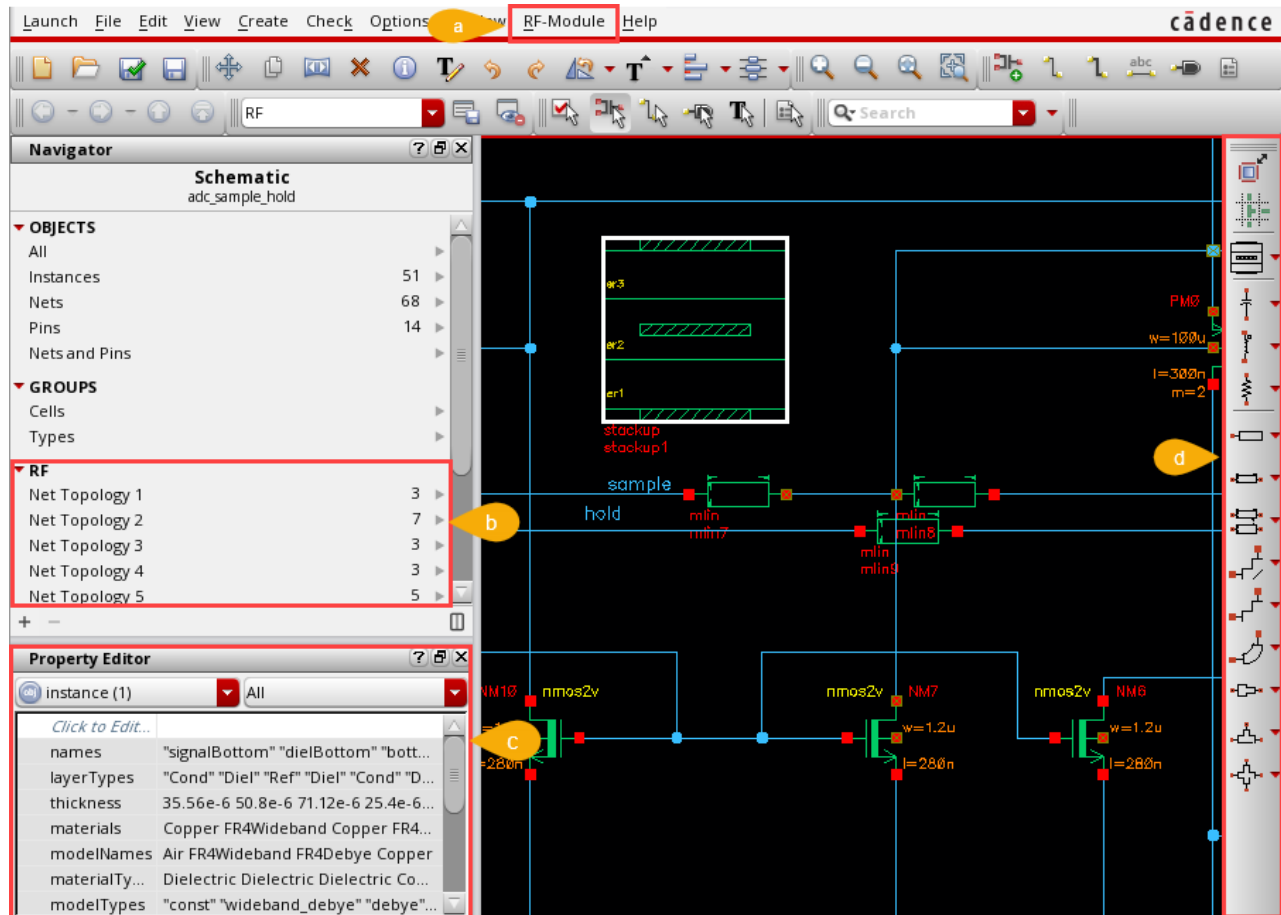
[*Spectre Circuit Simulator RF Analysis Library Reference*](#)

Virtuoso Schematic Editor User Guide

RF Transmission Lines

RF Workspace and toolbar

The RF workspace is designed to help you create transmission lines on the canvas. The workspace comprises the following elements.

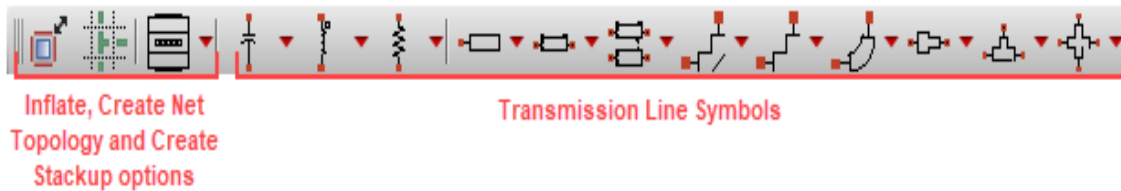


a	<u><i>RF-Module Menu</i></u>	Contains a list of RF features available in Schematics XL, including the <u><i>Transmission Lines</i></u> submenu.
b	Navigator assistant	RF sets are automatically created for each RF net topology the next time connectivity is updated. The <i>Details</i> pane shows only those object types that are associated with the selected set.
c	Property Editor assistant	Allows you to view and edit the properties of the selected transmission lines and their cells.
d	RF Toolbar	Commands to assist you in creating topologies and in adding transmission lines directly onto the schematic.

Virtuoso Schematic Editor User Guide

RF Transmission Lines

The RF toolbar commands contains the following icons:



The options to Inflate, Create Net Topology and Create Stackup are to assist in creating transmission lines and are also available from the *Transmission Lines* submenu on the *RF-Module* menu. The remaining toolbar commands are shortcuts to placing transmission cells, allowing you to quickly place instances onto the schematic. Each symbol on the toolbar represents command for a different transmission line cell.

Note: The command options available is determined by the symbols available in the selected libraries.

On the *RF-Module* menu, the *Transmission Lines* submenu contains the following commands:

- Inflate
- Create Net Topology
- Create Stackup
- Transmission Cells

For details on the other menu options, see the [Virtuoso RF Solution Guide](#).

Inflate

The *Inflate* command stretches the existing schematic, preserving the design to accommodate space for transmission line insertion. This is useful when inserting net topologies into existing designs.

This command is available from *RF-Module – Transmission Lines – Inflate* or by using the *Inflate*  button on the *RF* toolbar.

Each click on the *Inflate*  button increases the schematic size by a fixed increment.

To see the results, select *Zoom to Fit* from the *View* menu.

Create Net Topology

A net topology contains all schematic instances and nets associated with a single RF net implementation. The *Create Net Topology* command, provides a fast error-free method of creating and editing net topologies.

The command is available from *RF-Module – Transmission Lines – Create Net Topology* or by using the *Create Net Topology*  button on the *RF* toolbar. It allows you to create a design by building transmission lines and transmission cells on a grid mode. A catalog of transmission cells is available. Each transmission cell is provided with preset metrics already defined which are stored in the `RFtLineLib`. The metrics control the behavior and ensure that the placement of a cell is correctly constructed in relation to those adjacent to where it is being placed. For example, a cell with a curved bend cannot be placed next to another curved bend.

The `rfTlineLib` discontinuities (tees, bends, curves, and others) automatically collect the adjacent lines' connectivity properties from the circuit's topology.

There are two types of components:

microstrip	A microstrip consists of a conducting strip that is separated from a ground plate by a substrate. The dielectric on the other side of the conducting strip is typically air.
stripline	A stripline consists of a conducting strip that is sandwiched between two ground planes but separated from each of them by a substrate layer.

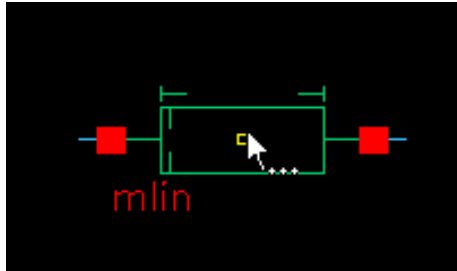
The *Create Net Topology* command proposes connected transmission lines and discontinuities to ensure a net topology is correctly constructed. Where there is existing wiring it proposes transmission lines and discontinuities based on the existing wire topology.

Using the Create Net Topology command on a blank canvas

The *Create Net Topology* command proposes connected transmission lines and discontinuities to ensure a net topology is correctly constructed.

1. Open the *RF* workspace.
2. Select *RF-Module - Transmission Lines – Create Net Topology* or click the *Create Net Topology*  button on the *RF* toolbar.

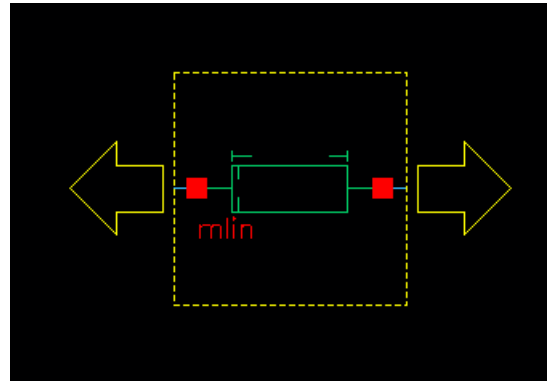
Identify the location on the design that requires transmission lines to be added.



3. The cursor is initially loaded with a *mlin* transmission cell. Hover the transmission cell over the required position and **Tab** to cycle / **SHIFT+Tab** to reverse cycle through the available options for that location.

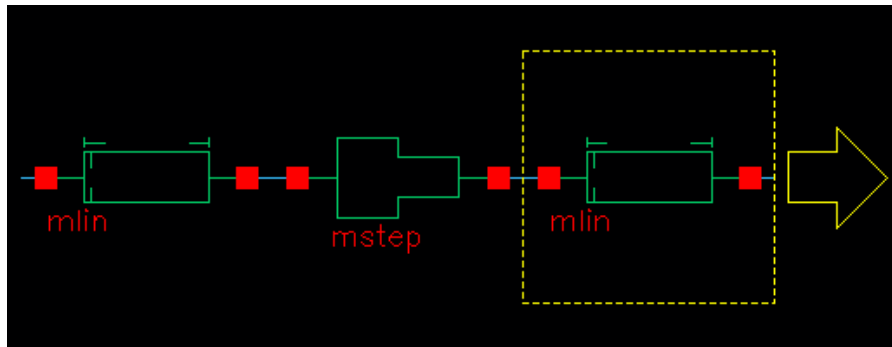
4. Once the required cell type is in the correct location, click on the canvas to fix the position. As soon as you click, the next proposed transmission cell is displayed.

The type of transmission cell proposed is based on the current topology. The yellow dotted lines and the arrows are the grid-based placer which guides you in the available directions for placing the next transmission cell.



Note: Pressing **Esc** cancels the *Create Net Topology* command and the entire net topology is deleted.

5. Move in the required direction and click to position the next cell.



Repeat until the required topology is complete.

Note: While the grid-based placer is active, you can continue to **Tab** to cycle / **SHIFT+Tab** to reverse cycle through the available options for that location.

6. Press **Enter** or double-click to build the topology. (Pressing **Esc** cancels the *Create Net Topology* command.)

Once built, the transmission lines and cells can be repositioned as required.

RF sets are automatically created for each RF net topology in the *Navigator* assistant the next time connectivity is updated.

To select all contents of a Net Topology, right-click on an RF set and select *Select Content* from the context menu. To copy, move, or delete an entire net topology, select a single wire and triple-click. All instances and wires in the same net topology are selected.

Using the Create Net Topology command on an existing canvas

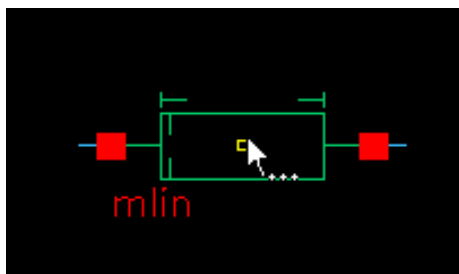
On a canvas where there is existing wiring, the *Create Net Topology* command proposes transmission lines and discontinuities based on the existing wire topology.

1. Open the *RF* workspace.

Identify the location on the design that requires transmission lines or cells to be added.

Before proceeding, ensure there is enough space to accommodate the additions that require to be made. You can use the *Inflate* option to expand the space between devices that are connected while maintaining the existing topology.

2. Select *RF-Module – Transmission Lines – Create Net Topology* or click the *Create Net Topology*  button on the *RF* toolbar.



3. The cursor is initially loaded with a *m1n* transmission cell. As you move the focal point of the transmission cell over existing wires, it snaps onto the wires and as you move over vertical and horizontal wires, it changes orientation. The type of transmission cell being proposed also varies as you move over the different junctions that are already in place.

Hover the transmission cell over the required position and *Tab* to cycle / *SHIFT+Tab* to reverse cycle through the available options for that location.

4. Once the required cell type is in the correct location, click on the canvas to fix the position. As soon as you click, the transmission cell splices into the existing wiring and the next proposed transmission cell is displayed.

The type of transmission cell proposed is based on the current topology.

Note: Pressing *ESC* cancels the *Create Net Topology* command. To remove a placed transmission cell, choose *Edit – Undo*.

5. Identify the next location in the design that requires transmission lines or cells to be added and continue.
6. Press `ESC` to cancel the *Create Net Topology* command.

Once built, the transmission lines and cells can be repositioned as required.

RF sets are automatically created for each RF Net Topology in the *Navigator* assistant the next time connectivity is updated.

To select all contents of a Net Topology, right-click on an RF set and select *Select Content* from the context menu. To copy, move, or delete an entire net topology, select a single wire and triple-click. All instances and wires in the same net topology are selected.

Probing a net topology

To probe a full Net Topology, right-click on a net in the Navigator and use the Probe menu entry. All schematic nets in a Net Topology are highlighted in the Navigator and on the canvas in the same probe color. For more details, see [The Probes Assistant](#).

Net Topologies on the canvas can also be visualized by selecting Net Highlighting in the *View* menu. For more details, see [Dynamic Highlighting](#).

Create Stackup

A stackup instance represents a partial slice through the technology stack and stores material properties and cross section information. Multiple stackup objects can be added to a schematic.

To create a stackup,

1. Open the *RF* workspace.
2. Select *RF-Module - Transmission Lines – Create Stackup* or click the *Create Stackup*  button on the *RF* toolbar.

3. Identify the location on the design that requires a stackup to be added.

The stackup can be edited before being placed on the canvas, if required, by pressing F3 to toggle the Add Instance form and selecting the Click to Edit Models button.

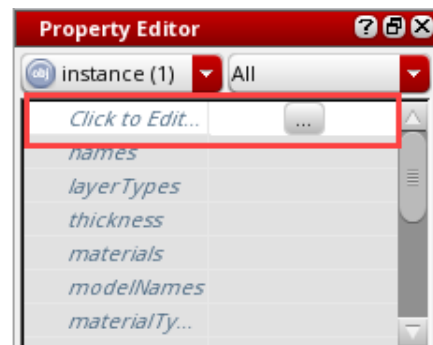
4. Once the stackup is in the correct location, click on the canvas to fix the position. As you click, another proposed stackup is displayed.

Note: Pressing `Esc` cancels the *Create Stackup* command.

5. Edit the positioned stackup by selecting the ... button that is displayed alongside the *Click to Edit* option on the Property Editor.

The Stackup Editor form is displayed which allows you to store substrate geometry and material properties.

For details on how to define material properties and set default materials, see the Quick-Start Guide in the Spectre Circuit Simulator RF Analysis Library Reference.



For more details on the RFTlineLib library, see the Spectre Circuit Simulator RF Analysis Library Reference.

Transmission Cells

The transmission cell commands displayed on the RF toolbar are controlled by the library specified. The default library is `RFTlineLib`. For details on customizing the RF toolbar, see [Customizing the RF Toolbar](#).

If a command has an arrow it allows you to select additional transmission cells that are currently not visible on the toolbar. The last used transmission cell on the toolbar, becomes the top entry. The command for transmission cell runs the Add Instance command with the form hidden to allow you to quickly place instances onto the canvas. F3 toggles the visibility of the [Add Instance](#) form.

Default Transmission Line Cells provided on the RF Toolbar

The following are a sample of the different types of transmission cell available:

Library:	analogLib		
	Capacitor	<u>cap</u>	capq
	Inductor	<u>ind</u>	indq
	Resistor	<u>res</u>	<u>presistor</u>

Library:	RFTlineLib		
		<u>Microstrip</u>	<u>Stripline</u>
	single-conductor	<u>m</u> lin	<u>s</u> lin
	mitered arbitrary angle	<u>m</u> bend	<u>s</u> bend
	bend		
	mitered 90 degree bend	<u>m</u> bend2	<u>s</u> bend2
	corner	<u>m</u> corn	N/A
	curve	<u>m</u> curve	<u>s</u> curve
	cross	<u>m</u> cross	<u>s</u> cross
	T-junction	<u>m</u> tee	<u>s</u> tee
	step	<u>m</u> step	<u>s</u> step
	open-circuit effect	<u>m</u> loceff	<u>s</u> loceff

Customizing the RF Toolbar

You can customize the RF toolbar to provide shortcuts to your own transmission line cells. A selection of default toolbar commands are shipped in `$CDS_INST_DIR/share/cdssetup/dfII/sch/RF/toolbar.cfg`. This file can be edited by your Virtuoso system administrator to add new commands or change the order of commands on the toolbar to meet the specific requirements of an individual or team.

This is additional customization to that provided by the [Toolbar Manager](#). See [Virtuoso Design Environment User Guide](#) for more details.

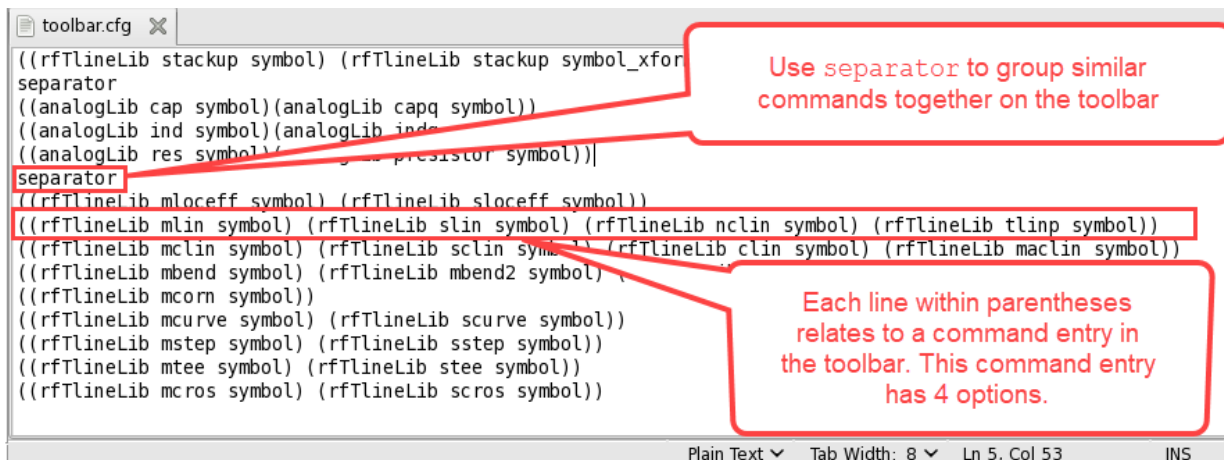
To update the RF toolbar file:

1. Save a copy of the supplied toolbar file `$CDS_INST_DIR/share/cdssetup/dfII/sch/RF/toolbar.cfg` into a search path that is defined by the Cadence Setup Search File mechanism (CSF) (`setup.loc`).

This is required in order for it be found through CSF. For more information on CSF, see [Specifying CSF Search for Additional Files](#) in the [Cadence Application Infrastructure User Guide](#).

2. Open the new toolbar file.

The `toolbar.cfg` file format is a simple text file.



3. Modify the file to include the commands you require.

Each line within parentheses relates to a command entry in the toolbar.

Examples of a command entry in the toolbar:

The format of each line is:

```
( (libName cellName viewName) [(libName cellName ViewName)]... )
```

Virtuoso Schematic Editor User Guide

RF Transmission Lines

A single command entry:

```
((analogLib res symbol))
```

A command entry with multiple options:

```
((analogLib res symbol)(analogLib presistor symbol))
```

Use `separator` to display a small dividing line between the toolbar buttons on the toolbar, useful to split commands and group similar commands together:

```
((analogLib res symbol)(analogLib presistor symbol))  
separator  
((rfTlineLib mloceff symbol) (rfTlineLib sloceff symbol))
```

4. Comment out or remove each line that is not to be displayed on the toolbar.
5. Save your amended version.
6. Restart Virtuoso.

The new toolbar is now available.

Using Split Symbols

Large symbols that have hundreds of pins are difficult to manage within a schematic design. The Split Symbols feature in Virtuoso Schematic Editor lets you split a large symbol into multiple partial symbols. Splitting such a large symbol master allows the schematic to be more readable and have a legible presentation of the pin symbols. This feature is most useful for PCB and package schematics.

A split symbol for a component can be created manually using Virtuoso Symbol Editor. The Symbol Editor then automatically generates a large split-primary symbol, which has a combination of all the pins from the splits created for the component.

You can add a split symbol to a schematic by using the *Create – Instance* command. You can add split symbol instances only for scalar instances. Vector instances are not supported.

When you add a split symbol instance, a hidden split-primary instance is also automatically added to the schematic. Although the split-primary is invisible in the schematic canvas, it is the split-primary instance that represents the component in the other design tools, such as simulation and layout. The splits are ignored instances, and therefore are avoided by netlisters.

Every split component has an attribute, `Split Primary`, which specifies the name of the primary instance they are merged into. The *Check – Current Cellview* command ensures that the signals on the split components are carried through to the component pins of the invisible split-primary. The properties and CDF parameters for the component are stored on the split-primary symbol.

The tool automatically maintains correct connectivity between the split-primary and the individual splits. Where there is ambiguity that cannot be resolved, for example if there are different nets attached to the same pin or terminal in splits of the same primary, an error marker with an appropriate message is placed in the canvas.

This chapter covers the following topics for the split symbols feature in Virtuoso Schematic Editor.

- [Creating Split Symbols](#)
- [Adding Split Instances to a Schematic](#)

- [Viewing the Splits in a Design](#)
- [Editing Properties of Split Instances](#)

Creating Split Symbols

Using the Virtuoso Symbol Editor, you can create split symbols like regular symbols. However, when creating a split symbol, ensure that the cell has no views with the name `symbol`. The view name `symbol` is reserved for the automatically generated split-primary symbol. The split view can have any other name, for example, `s1`, `s2`, `s3` or `a1`, `b1`, `c1`, and so on.

Related topics

- [Creating New Symbols](#)
- [Creating Symbol Cellviews Using Generators](#)
- [Automatically Creating a Cellview from a Pin List](#)

Adding a Split Annotate Label

To help identify the split you are currently working with, add a `split annotate IL` label to the symbol.

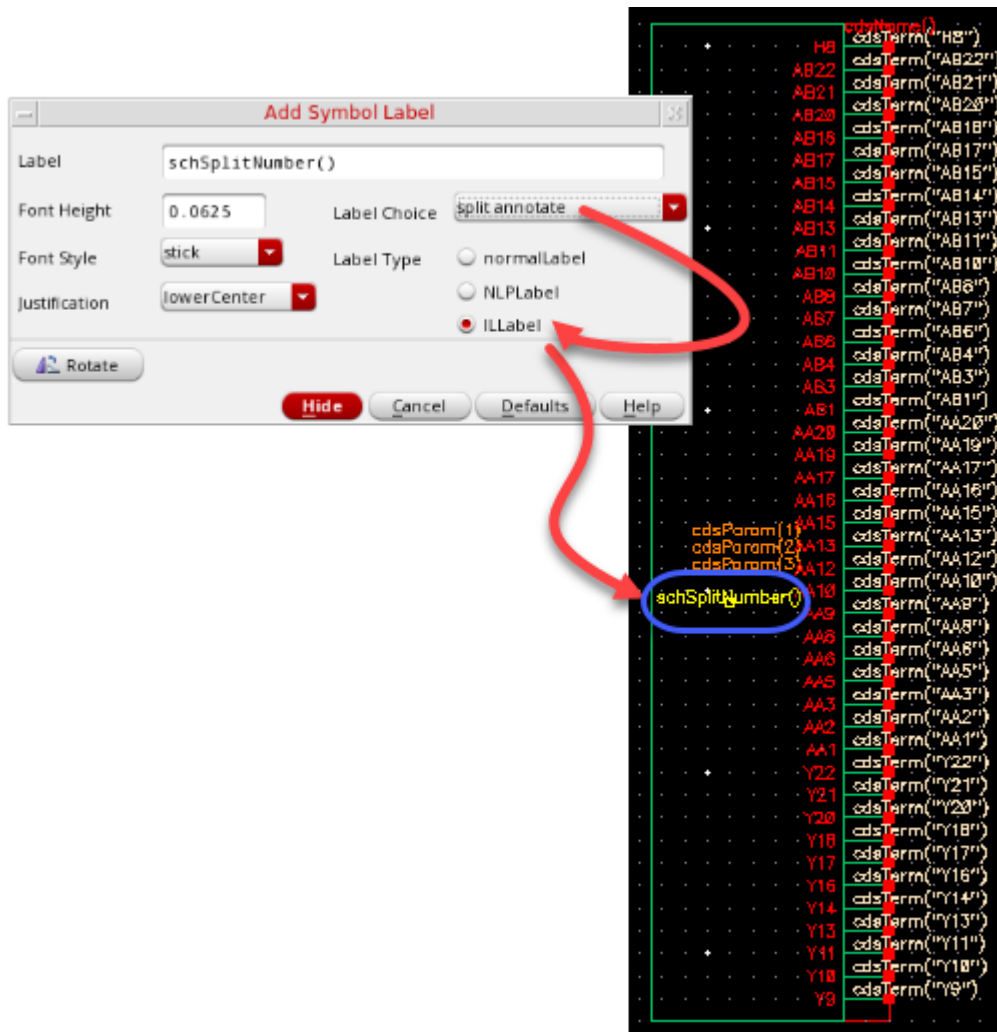
To do this:

1. Choose *Create – Label*. The Add Symbol Label form appears.
2. Set the *Label Choice* cyclic field to *split annotate*.
3. In the *Label Type* field, select *ILLabel*.

Virtuoso Schematic Editor User Guide

Using Split Symbols

4. Move the mouse cursor to the cellview and click to place the label. This label helps you understand which split you are viewing in your schematic by providing indication, such as 1 of 5, 2 of 5, on the split master symbol.



Once all the shapes, pins, and labels are added to the symbol and it is ready to be saved, the symbol needs to be marked as a split.

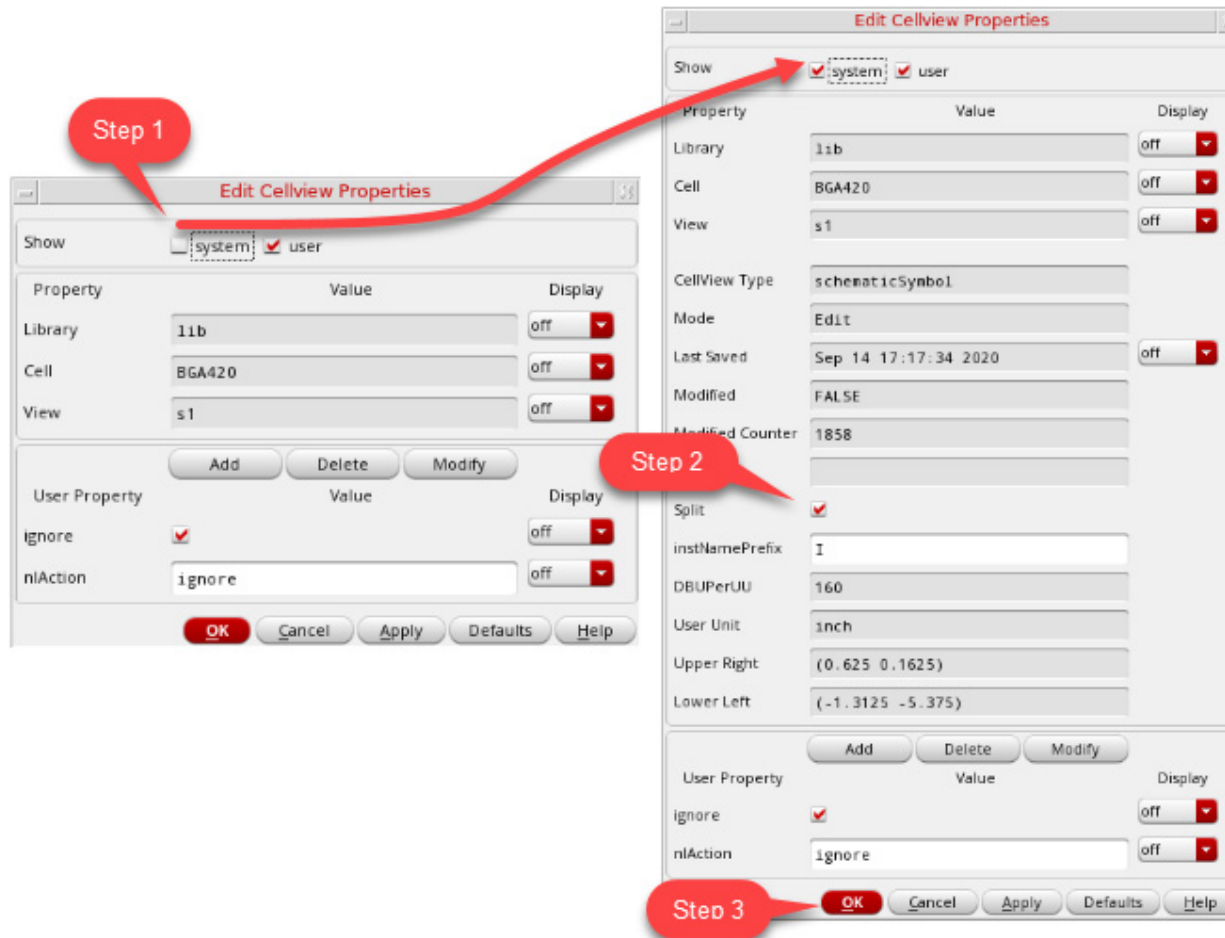
Marking a Symbol as a Split Symbol

1. Choose *Edit – Properties – Cellview*. The Edit Cellview Property form appears.
2. Select the *System* option in the *Show* group box to show the system properties for the form.
3. Select the *Split* check box in the system view.

Virtuoso Schematic Editor User Guide

Using Split Symbols

4. Click *OK* and save the cellview.

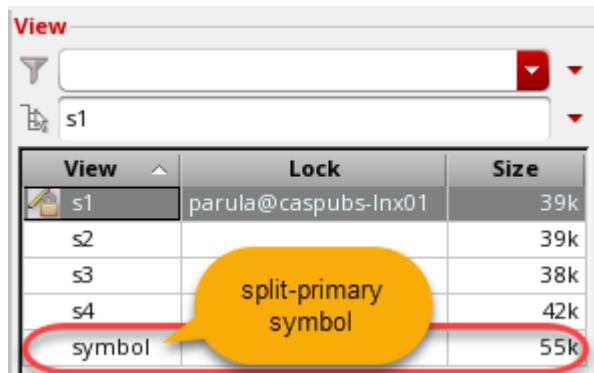


5. After all the splits are created, click *Check and Save* one of the split symbol, to check and save split symbols.

Virtuoso Schematic Editor User Guide

Using Split Symbols

This automatically generates the split-primary symbol. The split-primary symbol consists of terminals from all split symbols.



Important

Edits are not expected to be made to the split-primary symbol. The split-primary is re-generated every time a split symbol is checked.

Adding Split Instances to a Schematic

To add split instances to a schematic:

1. Choose *Create – Instance*. The Add Instance form displays.

Adding the first split instance automatically creates a hidden split-primary instance. The split-primary is invisible in the Virtuoso Schematic Editor canvas. However, it can be viewed in the *Objects* category in the Navigator Assistant. For example, as soon as a split instance `I0_a` is added, a split-primary `I0` is automatically created for it.

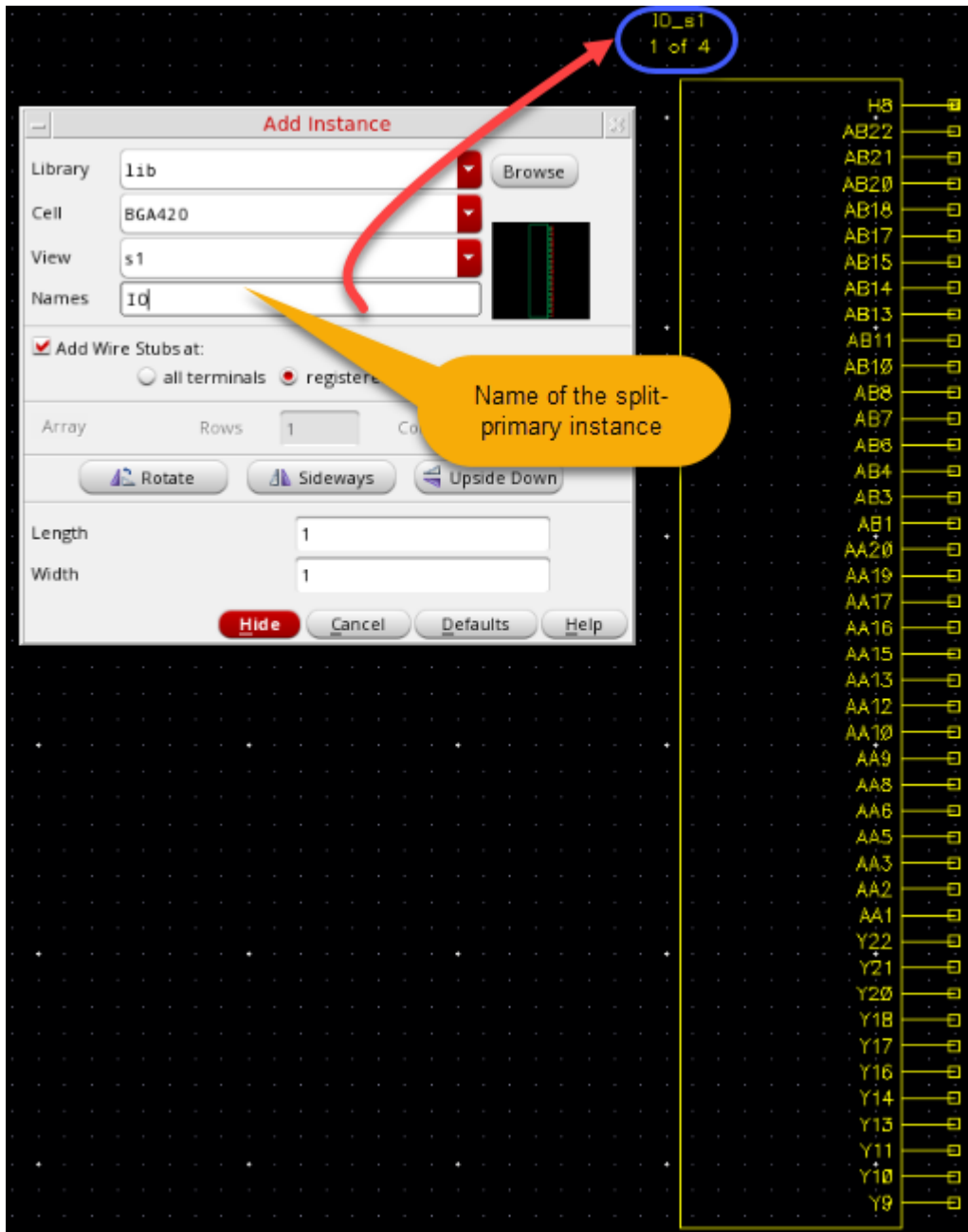
2. Specify the library, cell, and view name that you want to instantiate in the *Library*, *Cell*, and *View* name fields.
3. Place the selected instance in the schematic.

Similarly, use the Add Instance form to place the remaining splits in the schematic.

Virtuoso Schematic Editor User Guide

Using Split Symbols

When adding the remaining splits of a component, specify the name of the split-primary instance of which the splits are a part of. The name can be changed later by editing the Split Primary attribute in the Property Assistant.



You can also add new split instances by copying existing split instances in the design. When you copy a split instance, the relevant split-primary instance is copied as well, and the new split is correctly linked to the new split primary.

Note: This applies only when copying splits in the graphical user interface. When using the `schCopy` SKILL API, you must explicitly copy the split instance and the split-primary and then link the two.

Viewing the Splits in a Design

When a split instance is first added to a schematic, a split-primary instance is automatically created for it. This split-primary instance can be seen in the *Instances* set under the *Objects* category of the Navigator assistant. The split instances do not appear in the *Instances* set because their status is set to ignored.

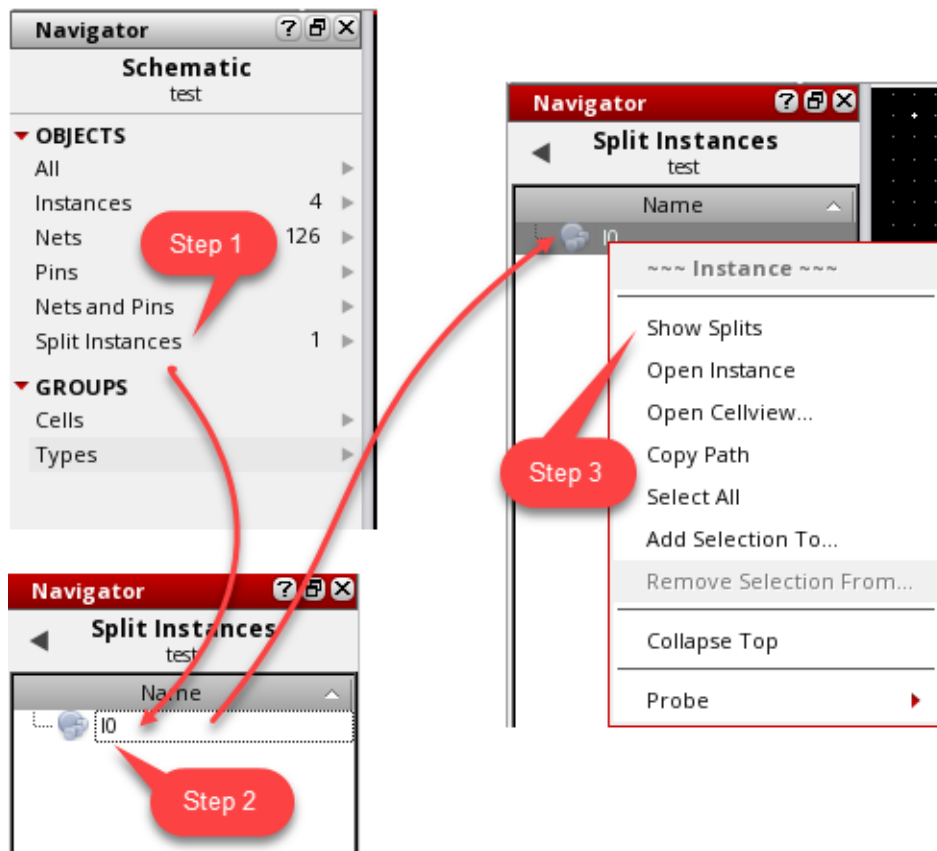
To view the split instances in the Navigator Assistant:

1. Click the *Split Instances* set under the *Objects* category in the Navigator Assistant. This set lists all the split-primary instances in the design.
2. Select the split-primary instance. When the split-primary instance is selected in the Navigator Assistant, all the associated split instances are automatically selected in the schematic editor canvas.

Virtuoso Schematic Editor User Guide

Using Split Symbols

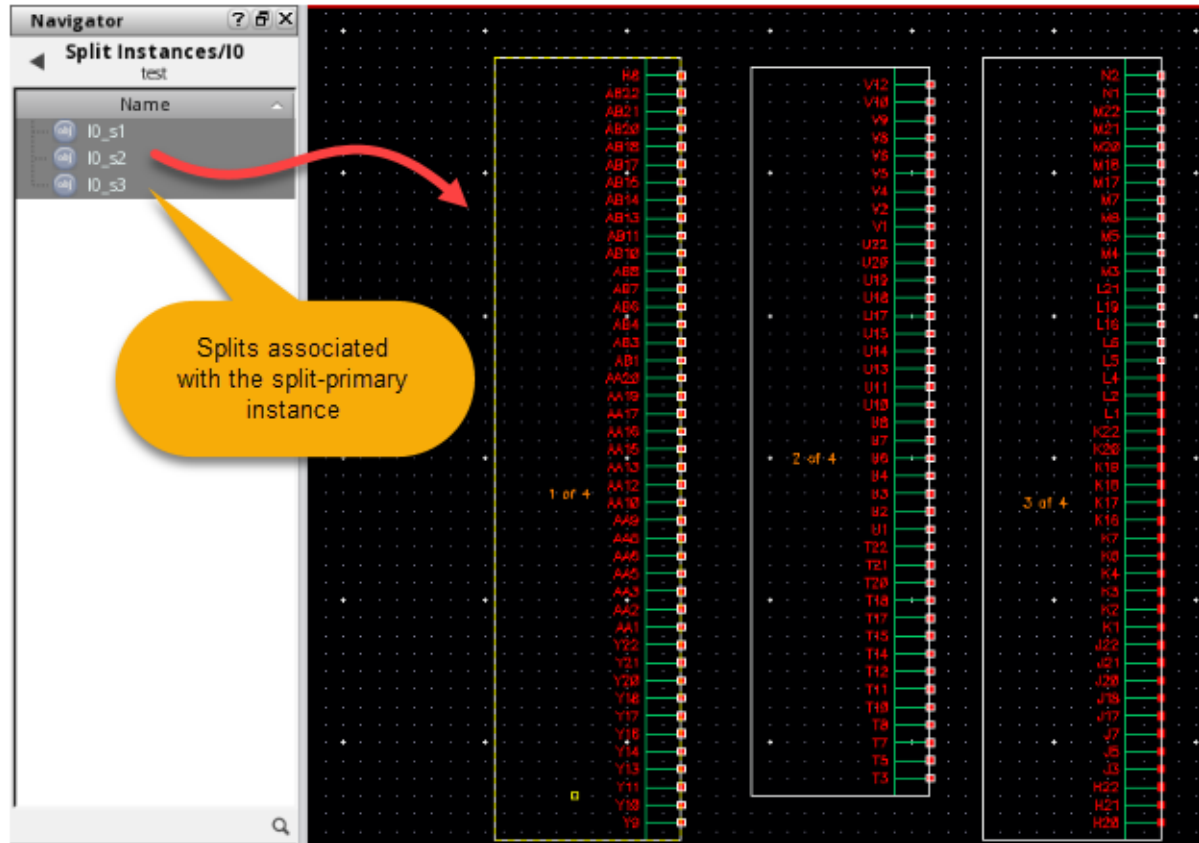
3. Right-click a split-primary instance in the Navigator Assistant and click *Show Splits*.



Virtuoso Schematic Editor User Guide

Using Split Symbols

The Navigator assistant displays the splits associated with the split-primary instance.



Editing Properties of Split Instances

The split-primary instance represents the splits in netlists and layouts. Therefore, instance CDFs, if any, are stored on the hidden split-primary instance and not the splits. However, it is possible to view and edit these properties by selecting any one of the split instances.

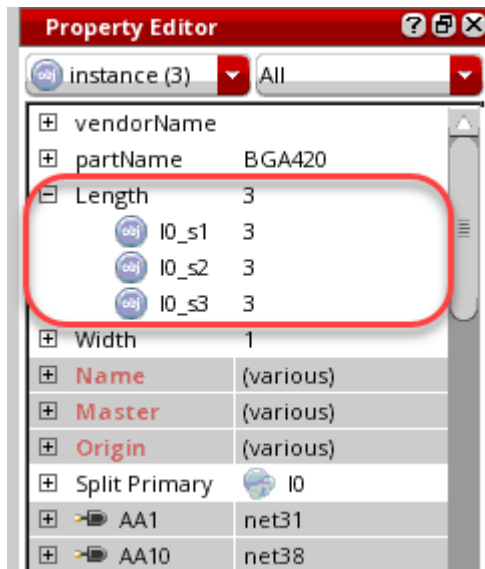
For example, to adjust the *Length* attribute for the BGA device:

1. Select a split symbol
2. Modify the attribute value using the Property Assistant or the Edit Object Property form.

Virtuoso Schematic Editor User Guide

Using Split Symbols

The modified value of the attribute is replicated on all the other splits too. This is because the modified attribute is saved to the split-primary.



Environment Variables

This appendix describes, in alphabetical order, the *schematic* environment variables that control the characteristics of the schematic editor and the behavior of many schematic editor commands. You can customize the behavior of the schematic editor by changing the value of a particular environment variable.

The [Navigator Environmental Variables](#) define the behavior of navigator commands. These are listed at the end of this appendix.

For information on *layout* and *graphic* environment variables, that control how various layout editor and window display commands work respectively, see [Environment Variables](#) in the *Virtuoso Layout Editor User Guide*.

Changing the Default Environment Variables

To change a default variable or setting,

- From the view, choose *Options – Display*, *Options – Editor*, or *Check – Options*.
For more information, see [Virtuoso Design Environment User Guide](#).

Copying Environment Variables from the Samples Directory to ~/.cdsenv

To change environment variables in the `.cdsenv` file,

1. Copy the variables you want to change from the default `.cdsenv` file to the `.cdsenv` file in your home directory.

The default file is located at

`your_install_dir/tools/dfII/samples/.cdsenv`

2. Edit the default values in your home directory.

The `.cdsenv` file in your home directory is loaded last; any values in your home directory override those previously loaded.

Checking the Value of an Environment Variable

To determine the current value of an environment variable,

- In the CIW, type

```
envGetVal("view" "envVar")
```

The default setting for the environment variable `browserType` is `library`.

Setting Environment Variables in the CIW

To set the environment variable in the CIW,

- Type the following SKILL function:

```
envSetVal("editorName" "env_var_name" type_of_value value)
```

For example:

```
envSetVal("schematic" "browserType" 'string "component")
```

Using this procedure, you specify the specific environment variable as an argument for the `schGetEnv` and `schSetEnv` SKILL procedural interface functions.

For information on `envGetVal` and `envSetVal`, refer to [Cadence User Interface SKILL Reference](#).

Changing Graphic Environment Variables That Are Cellview Properties

Some graphic environment variables are cellview properties. To make the changes to the environment variable visible in the cellview window immediately,

- In the CIW, type

```
window = hiGetCurrentWindow()  
window~>environmentVariable = value
```

For example:

```
window = hiGetCurrentWindow()  
window~>autoDot = t
```

This changes the environment variable for the current window only, and you must reset the variable if you open a new window. The change is effective for the current session only.

Retaining the Current Settings for Future Sessions

To retain the current settings for future sessions,

- From the view, choose *Options – Save Defaults*.

The system writes to the `.cdsenv` file automatically whenever you choose *Options – Save Defaults*. If the file already exists, the system prompts you to overwrite the file or save it using another name. For details, see [Saving Form Field Default Settings](#).

addStubs

```
schematic addStubs boolean { t | nil }
```

Description

Sets the *Add Wire Stubs* at option on (`t`) or off (`nil`), in the Add Instance form, to create wire stubs during instantiation. The default is `t`.

GUI Equivalent

Command	<i>Create – Instance</i>
Form Field	<i>Add Wire Stubs at</i>

Related topics

[Add Instance](#)

aeIEvalCDFParam

```
schematic aeIEvalCDFParam boolean { t | nil }
```

Description

Displays the evaluated results of the AEL expressions for CDF properties in the Property Editor assistant. The default is `t`. The CDF properties of an instance could contain expressions for its parameter values. These expressions use AEL and can include mathematical operations and functions that reference other property values. When the environment variable is set to `t`, it displays the evaluated result of these AEL expressions. Only the CDF properties that have `parseAsCEL` and `parseAsNumber` enabled are evaluated. Set the `CDS_Netlisting_Mode` SHELL variable to *Analog* to ensure that the AEL evaluator is able to find the CDF property in OA. See also [Effect of Netlisting Mode On Labels](#).

For details, refer to [Component Description Format \(CDF\)](#).

GUI Equivalent

None

allTerms

```
schematic allTerms boolean { t | nil }
```

Description

Sets the *all terminals* option on (`t`) or off (`nil`) in the Add Instance form to create wire stubs for all terminals during instantiation. When `nil`, the wire stubs and the wire name are created for only the registered nets. The default is `nil`.

GUI Equivalent

Command	<i>Create – Instance</i>
Form Field	<i>all terminals</i>

Related topics

[Add Instance](#)

annotationBalloonPinningStatus

schematic annotationBalloonPinningStatus boolean { t | nil }

Description

Enables you to pin the annotation balloons on the schematic canvas. If this variable is set, the Pin and Delete buttons are added to the top-right corner of the annotation balloons.

Variable Type	boolean
Default Value	t
Acceptable Values	{nil, t}

annotationSetupFileList

```
auCore.misc annotationSetupFileList string { "<pathToMyAnnotationSetup>.as" |  
      "<pathToMyOtherAnnotationSetup>.as" }
```

Description

Loads the specified annotation setup files while the schematic window opens. For details, refer to [Loading the Shared Annotation Settings Manually](#).

Variable Type	string
Default Value	" "
Acceptable Values	Any string value with full path

autoDot

```
schematic autoDot boolean { t | nil }
```

Description

Specifies whether a solder dot is automatically generated for three-wire T-intersections or intersections that include either angled wires or flight lines (t). The default is t.

For details, see [Disabling Automatic Solder Dot Creation](#). See also [Editor Options](#).

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Automatic Solder Dots (Commands tab)</i>

Related topics

[Editor Options](#)

autoEquivalentConnectivityTime

```
schematic autoEquivalentConnectivityTime boolean { t | nil }
```

Description

Specifies whether cosmetic symbol changes can be automatically signed off as having no impact on connectivity. The default value is `nil` in which case such changes are not automatically signed off. For details, see [Automatically Signing Off Cosmetic Changes to Symbols](#).

GUI Equivalent

N/A

Example

1. Set the `autoEquivalentConnectivityTime` variable in the `.cdsinit` file:

```
schSetEnv("autoEquivalentConnectivityTime" t)
```

2. Open a Virtuoso session and add a cosmetic change to a symbol instantiated within the schematic, such as a label.
3. Save the symbol.

When the schematic is opened, no warning is generated by this change.

autoLoadBindKeys

```
schematic autoLoadBindKeys boolean { t | nil }
```

Description

Specifies whether or not Cadence provided default bindkeys should be loaded.

See also [Bindkey Storage](#).

GUI Equivalent

None

autoWireStubBusTerms

```
schematic autoWireStubBusTerms cyclic { wide | narrow }
```

Description

Sets the default size for creating wire stubs for bus terminals; *wide* or *narrow*. Wire stubs require a dedicated wire width, either wide or narrow. When they are wide they share the same width as other wide wires. Bus terminals take wire stubs the same as other terminals. .

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Auto Wire Stub for Bus Terminals (Commands tab)</i>

Related topics

[Editor Options](#)

autoZoomMode

```
schematic autoZoomMode string { "Pan To Selected" | "Zoom To Selected" | "Zoom To Fit" }
```

Description

Specifies the auto zoom mode setting to be used with *View – Dynamic Zooming*. The default is “Pan To Selected”.

For example:

```
envSetVal("schematic" "autoZoomMode" `string "Zoom To Fit")
```

See also [Dynamic Zoom](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Mode</i>

Related topics

[Display Options](#)

autoZoomPan

```
schematic autoZoomPan boolean { t | nil }
```

Description

Specifies the setting for the facility to automatically pan to an object in the design canvas. The default is `t`.

The `autoZoomPan` global option can be set for any new session window, while any change made using the *View – Pan to Selected* menu option will be active for only the current session window.

A change to the setting of the `autoZoomPan` option will enable/disable the *View – Pan To Selected* option accordingly. However, a change made to *View – Pan To Selected* will not impact the current setting of `autoZoomPan`.

To switch off *Auto Pan to Selected*, either set the environment variable to `nil` in the `.cdsenv` file or enter the following into the CIW:

```
envSetVal( "schematic" "autoZoomPan" 'boolean nil )
```

Note: This option is also available in VSE L/XL from the *Navigator* and *Search* assistants.

See also [Dynamic Zoom](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Pan To Selected (Zoom / Pan Controls Mode option)</i>

Related topics

[Display Options](#)

autoZoomScale

```
schematic autoZoomScale int { 90 | nil }
```

Description

Specifies an integer value indicating the percentage of visible canvas area that should display the selected set when the Dynamic Zooming mode is set to Zoom To Selected. The default value is 90(auto zoom 1 (far) to 100 (near)).

See also [Dynamic Zoom](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Zoom Scale (%)</i>

Related topics

[Display Options](#)

blockDirRules

```
schematic blockDirRules cyclic { "edge" | "order of entry" }
```

Description

Specifies the rule for setting the direction of the new pins that the system creates automatically when you edit a block. The default is `edge`.

For details, see [Editor Options](#). See also [Changing the Block Pin Direction Rule](#).

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Block Pin Direction Rule (Commands tab)</i>

Related topics

[Editor Options](#)

broadcast

```
schematic broadcast boolean { t | nil }
```

Description

Specifies whether cross-selection is broadcast (t) or not (nil). The default is nil.

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Cross Section (Commands tab)</i>

Related topics

[Editor Options](#)

browserShowSchViews

`schematic browserShowSchViews boolean { t | nil }`

Description

Shows schematic views in the Library Browser (t) if the *Add Instance Browser Type* field in the *Commands* tab of the Editor Options form is set to *library*. It also controls the types of views available in the Create a Mapping Schematic form. If the variable is `nil`, only `schematicSymbol` views are shown in the form. When the value is `t`, both `schematicSymbol` and `schematic` views are shown in the form.

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Show Schematic Views in Browser (Commands tab)</i>

Related topics

[Editor Options](#)

browserType

```
schematic browserType string { "library" | "component" }
```

Description

Specifies which browser you access when the *Add Instance* command is active. The default is `library`.

For details, see [Editor Options](#).

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Add Instance Browser Type (Commands tab)</i>

Related topics

[Editor Options](#)

bundleDisplayMode

```
schematic bundleDisplayMode cyclic { "vertical" | "horizontal" }
```

Description

Specifies whether each member of a bundle net name is stacked vertically or horizontally when a wire label containing a comma separated bundle name is added. The default is horizontal.

GUI Equivalent

Command	<i>Create – Wire Name</i>
Form Field	<i>Display bundles vertically</i>

Related topics

[Create Wire Name](#)

checkAlignOverlaps

```
schematic checkAlignOverlaps boolean { t | nil }
```

Description

Issues warnings for any overlap in a given cellview. The default value is `nil`. When set to `t`, a CIW log is created to report any overlapping instances after alignment (pre-selected or post selected mode) in the cellview. Format of the generated report is controlled by the *Overlapping Instances* and *Percent Overlap Allowed* options on the *Physical* tab of the *Schematic Rules Checks Setup* form.

However, in the post-selected mode, you can also use the *Align* dialog box to report any overlapping instances. This dialog box can be displayed by pressing F3. It contains the *Check overlapping instances* check box.



GUI Equivalent

Command	<i>Press F3 in the post-selected mode.</i>
Form Field	<i>Check overlapping instances</i>

checkAlways

```
schematic checkAlways boolean { t | nil }
```

Description

Specifies whether the schematic editor checks all schematics encountered during traversal (`t`) or only those that have been modified (`nil`). The default is `nil`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Hierarchy</i>
	<i>Options – Check</i>
Form Field	<i>Process</i>
	<i>From a Multisheet Index, Check</i>

Related topics

[Check Hierarchy](#)

[Schematic Check Options](#)

checkBeepOn

```
schematic checkBeepOn boolean { t | nil }
```

Description

Specifies whether the system beeps after every save and check in. If set to `t`, the system beeps even if there is no error or warning message. The benefit of turning this option on is to let you know when the check is complete. If set to `nil`, the system restores the behavior to release 4.3.x (the system beeps only when an error or warning is found). The default is `t`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Options – Check</i>
Form Field	<i>Beep When 0 Errors/Warnings</i>

Related topics

[Schematic Check Options](#)

checkConstraints

```
schematic checkConstraints boolean { t | nil }
```

Description

Performs the Constraints checks when set to `t`. The default is `nil`.

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Mismatched Parameters (Constraints tab)</i>

Related topics

[Schematic Rules Checks Setup](#)

checkHierOpenMode

```
schematic checkHierOpenMode cyclic { "auto" | "edit" | "read" }
```

Description

Specifies the mode for opening cellviews during the hierarchy checks. The default value is `auto`. For details, refer to [Checking Read-Only Cellviews](#).

GUI Equivalent

Command	<i>Check – Hierarchy</i>
Form Field	<i>Open for</i>

Related topics

[Check Hierarchy](#)

checkHierSave

```
schematic checkHierSave boolean { t | nil }
```

Description

Specifies whether the system saves schematics as they are processed during a hierarchical check. When `nil`, schematics are not saved during processing, which allows you to save or discard the edits manually. When `t`, schematics containing zero errors are automatically saved. For those schematics that do contain errors, whether they are saved is based on the value of the `saveAction` variable. The default is `t`.

GUI Equivalent

Command	<i>Check – Hierarchy</i>
Form Field	<i>Save Schematics</i>

Related topics

[Check Hierarchy](#)

checkSymTime

```
schematic checkSymTime int { 0 | 1 | 2 | 3 }
```

Description

Specifies whether a consistency check of the symbol masters is run and specifies the level of the check. The default is 1.

When set to 0, no consistency check is run. When you specify a level of 1 to 3, the schematic editor attempts to run the specified check during loading. When you specify a level of 2, the schematic editor attempts to run the level 1 check first. When you specify a level of 3, the schematic editor attempts to run both the level 1 and level 2 checks first.

When set to 1, a warning is generated if the value of the modification counter changes for any symbol.

When set to 2, a warning is generated when the value of the UNIX file modification time stamp for a symbol is newer than that of the schematic. This check does not run if the level 1 check fails.

When set to 3, a warning is generated when the value of the check-in time stamp of a symbol is newer than the value of the UNIX file modification time stamp of the schematic. This check does not run if either the level 1 or level 2 check fails.

GUI Equivalent

None

comanageConstraintView

This function has been included within this appendix for completeness. For details on its description, type, and value refer to the [comanageConstraintView](#) function in the *Virtuoso Unified Custom Constraints User Guide*.

comanageFollowModeOnDescend

This function has been included within this appendix for completeness. For details on its description, type, and value refer to the [comanageFollowModeOnDescend](#) function in the *Virtuoso Unified Custom Constraints User Guide*.

comanageFollowModeOnOpen

This function has been included within this appendix for completeness. For details on its description, type, and value refer to the [comanageFollowModeOnOpen](#) function in the *Virtuoso Unified Custom Constraints User Guide*.

compBackTrace

```
schematic compBackTrace boolean { t | nil }
```

Description

Specifies whether routing should backtrace to components when you stretch an object (*t*) or not (*nil*). The default is *t*.

GUI Equivalent

Command	<i>Edit – Stretch</i>
Form Field	<i>Reroute Wire Segments Connected To</i>

Related topics

[Stretch – Schematic View](#)

[Stretch – Symbol View](#)

condOp

```
schematic condOp cyclic { "==" | "!=" | "<" | ">" | "<=" | ">=" }
```

Description

Specifies the conditional operator for the *Find* and *Replace* commands. The default is "==".

GUI Equivalent

Command	<i>Edit – Find</i>
	<i>Edit – Replace</i>
Form Field	<i>Search For/Replace With</i>

Related topics

[Schematic Find](#)

[Schematic Replace](#)

copyDesignIntent

```
schematic copyDesignIntent boolean { t | nil }
```

Description

Specifies whether the design intent associated with an object is included when using the *Copy* command to copy an object. The default is `nil`.

GUI Equivalent

Command	<i>Edit – Copy</i>
Form Field	<i>Include Design Intent</i>

Related topics

[Copy](#)

createCvOptions

```
schematic createCvOptions boolean { t | nil }
```

Description

Specifies whether the *Create Cellview* commands display an options form that is specific to the destination cellview (*t*) or not (*nil*). The default is *t*.

GUI Equivalent

Command	<i>Create – Cellview – From Cellview</i> <i>Create – Cellview – From Pin List</i>
Form Field	<i>Edit Options</i>

Related topics

[Cellview From Cellview](#)

[Cellview From Pin List](#)

createInstResetCDF

```
schematic createInstResetCDF boolean { t | nil }
```

Description

Specifies whether the *Create Instance* command retains the current effective CDF values for a component when the same component is placed at a later time. If set to `t`, the schematic editor CDF resets the CDF values to their default settings when you start the *Create Instance* command. The default is `t`.

GUI Equivalent

Command	<i>Create – Instance</i>
Form Field	N/A

createInstShowCDF

```
schematic createInstShowCDF boolean { t | nil }
```

Description

Specifies whether the Add Instance form appears when the component you want to add contains CDF parameters (*t*) or not (*nil*). The default is *t*.

GUI Equivalent

Command	<i>Create – Instance</i>
Form Field	N/A

createLabelAutoCenterSize

schematic createLabelAutoCenterSize float *size_fraction*

Description

Specifies the size of the central region of a wire as a fraction (0-1) of the wire length. The default is 0.4.

The “end” regions equally share any remaining space. For example, setting this variable to 1.0 would mean that every wire is assumed to be in the central region. Therefore a label that is placed near the middle of a wire is moved to be centralized.

GUI Equivalent

Command	<i>Create – Wire Name</i>
Form Field	<i>Justification</i>

Related topics

[Create Wire Name](#)

createLabelAutoJustify

```
schematic createLabelAutoJustify boolean { t | nil }
```

Description

Specifies whether the placement of a new wire label snaps into position automatically or requires to be manually positioned using the cursor.

When set to `t`, the justification of the wire name is automatically selected (`lowerCenter`, `upperLeft`, and so on) to achieve the best position relative to the current placement of the wire.

By default, the value is `nil`, which indicates that the wire name requires to be positioned manually using the cursor.

GUI Equivalent

Command	<i>Create – Wire Name</i>
Form Field	<i>Justification – Automatic (<u>Create Wire Name</u> form)</i>

Example

```
envGetVal("schematic" "createLabelAutoJustify")
envSetVal("schematic" "createLabelAutoJustify" 'boolean t)
envSetVal("schematic" "createLabelAutoJustify" 'boolean nil)
```

Related Topics

[Automatic Justification of Wire Label Names](#)

[Create Wire Name](#)

createLabelAutoOffset

schematic createLabelAutoOffset float *distance*

Description

Specifies the distance that auto-justified labels should be placed from their parent wires, in user units. The default is 0.045.

GUI Equivalent

Command	<i>Create – Wire Name</i>
Form Field	<i>Justification</i> (<u>Create Wire Name</u> form)

Related Topics

Create Wire Name

createLabelAutoSpacer

schematic createLabelAutoSpacer float *spacing*

Description

Specifies the space left between pins and the labels placed next to them. For example, if there is a pin at the chosen wire end, the label is adjusted to be placed on the wire as close to the pin as possible. The default is 0 . 0.

GUI Equivalent

Command	<i>Create – Wire Name</i>
Form Field	<i>Justification</i> (<u>Create Wire Name</u> form)

Related Topics

Create Wire Name

createLabelFontHeight

schematic createLabelFontHeight float *font_height*

Description

Specifies a value (in user units) for the font height for the wire names that you add with the *Create Wire Name* command. The default is 0.0625 (inches).

GUI Equivalent

Command	<i>Create – Wire Name</i>
Form Field	<i>Font Height</i> (<u>Create Wire Name</u> form)

Related Topics

Create Wire Name

createLabelFontStyle

```
schematic createLabelFontStyle cyclic { "euroStyle" | "gothic" | "math" | "roman"  
    | "script" | "stick" | "fixed" | "swedish" | "milSpec" }
```

Description

Specifies a font for the wire names that you add with the *Create Wire Name* command. The default is `stick`.

GUI Equivalent

Command	<i>Create – Wire Name</i>
Form Field	<i>Font Style (Create Wire Name form)</i>

Related Topics

[Create Wire Name](#)

createLabelHorzOffset

schematic createLabelHorzOffset float *offset_distance*

Description

Specifies the offset distance of the wire name text in relation to a horizontal wire. The default is 0.045 (inches).

GUI Equivalent

Command	<i>Create – Wire Name</i>
Form Field	<i>Horizontal/Text Offset (<u>Create Wire Name</u> form)</i>

Related Topics

Create Wire Name

Virtuoso Schematic Editor User Guide

Environment Variables

createLabelHorzPosition

```
schematic createLabelHorzPosition cyclic { "above" | "below" }
```

Description

Specifies the position of the wire name text in relation to a horizontal wire. The default is above.

GUI Equivalent

Command	<i>Create – Wire Name</i>
Form Field	<i>Horizontal/Text Position (<u>Create Wire Name</u> form)</i>

Related Topics

Create Wire Name

createLabelHorzRotation

```
schematic createLabelHorzRotation cyclic { "R0" | "R90" }
```

Description

Specifies the orientation of the wire name text in relation to a horizontal wire. The default is R0.

GUI Equivalent

Command	<i>Create – Wire Name</i>
Form Field	<i>Horizontal/Text Rotation (<u>Create Wire Name</u> form)</i>

Related Topics

Create Wire Name

createLabelJustify

```
schematic createLabelJustify cyclic { "upperLeft" | "centerLeft" | "lowerLeft" |  
    "upperCenter" | "centerCenter" | "lowerCenter" | "upperRight" | "centerRight"  
    | "lowerRight" }
```

Description

Specifies the justification for the wire name text. The default is `lowerCenter`.

GUI Equivalent

Command	<i>Create – Wire Name</i>
Form Field	<i>Justification (Create Wire Name form)</i>

Related Topics

Create Wire Name

createLabelOffsetMode

```
schematic createLabelOffsetMode boolean { t | nil }
```

Description

Specifies the wire name *Entry Style* mode. When set to `t`, the mode is *fixed offset*. When set to `nil`, the mode is *manual*. The default is `t`.

GUI Equivalent

Command	<i>Create – Wire Name</i>
Form Field	<i>Entry Style</i> (<u>Create Wire Name</u> form)

Related Topics

Create Wire Name

createLabelVertOffset

schematic createLabelVertOffset float *text_offset*

Description

Specifies the location of the wire name text in relation to a vertical wire. The default is 0.045 (inches).

GUI Equivalent

Command	<i>Create – Wire Name</i>
Form Field	<i>Vertical/Text Offset</i> (<u>Create Wire Name</u> form)

Related Topics

Create Wire Name

Virtuoso Schematic Editor User Guide

Environment Variables

createLabelVertPosition

```
schematic createLabelVertPosition cyclic { "left" | "right" }
```

Description

Specifies the position of the wire name text in relation to a vertical wire. The default is `left`.

GUI Equivalent

Command	<i>Create – Wire Name</i>
Form Field	<i>Vertical/Text Position (<u>Create Wire Name</u> form)</i>

Related Topics

Create Wire Name

Virtuoso Schematic Editor User Guide

Environment Variables

createLabelVertRotation

```
schematic createLabelVertRotation cyclic { "R0" | "R90" }
```

Description

Specifies the orientation of the wire name text in relation to a vertical wire. The default is R90.

GUI Equivalent

Command	<i>Create – Wire Name</i>
Form Field	<i>Vertical/Text Rotation (<u>Create Wire Name</u> form)</i>

Related Topics

Create Wire Name

createPinAutoNameFromNamedNet

```
schematic createPinAutoNameFromNamedNet boolean { t | nil }
```

Description

Enables the Create Pin command for suggesting pin names for the named nets in the design when set to `t`. The default value is `nil`. The connectivity needs to be up-to-date to be able to accurately suggest pin names. For details, refer to [Adding Pins](#).

GUI Equivalent

None

cvDisplay

```
schematic cvDisplay boolean { t | nil }
```

Description

Specifies whether the system displays a new cellview in a window when you complete the *Create – Cellview* command (t) or not (nil). The default is t.

GUI Equivalent

Command	<i>Create – Cellview – From Cellview</i> <i>Create – Cellview – Pin List</i>
Form Field	<i>Display Cellview</i>

Related Topics

[Cellview From Cellview](#)

[Cellview From Pin List](#)

Virtuoso Schematic Editor User Guide

Environment Variables

defaultsFile

```
schematic defaultsFile string "filename"
```

Description

Specifies which file is accessed when loading or dumping the schematic environment variables. The default is `.cdsenv`.

GUI Equivalent

Command	<i>Options – Load Defaults</i>
	<i>Options – Save Defaults</i>
Form Field	<i>File Name</i>

Related Topics

[Load Schematic Defaults](#)

[Save Schematic Defaults](#)

defWideWireWidth

schematic defWideWireWidth float *width*

Description

Specifies a value for the width of wires and flight lines that you create with the *Create Wire* command. The value must be greater than 0.0 and specified in user units. The default is 0.0625 (inches).

GUI Equivalent

Command	<i>Create – Wire (narrow)</i> <i>Create – Wire (wide)</i>
Form Field	<i>Width</i>

Related Topics

[Add Wire](#)

deleteInstPinsWire

```
schematic deleteInstPinsWire boolean { t | nil }
```

Description

Specifies whether to delete instance pins wires (`t`) or not (`nil`). The default is `nil`.

GUI Equivalent

None

descendCanvasType

```
schematic descendCanvasType string { "new tab" | "current tab" | "new window" }
```

Description

Specifies the type of canvas to descend into. The default value is *current tab*.

The `descendCanvasType` (schematic) environment variable works with the `setWinTraversal` environment variable, where:

- if `setWinTraversal` is set to true, `setWinTraversal` will be honored and `descendCanvasType` will not be checked.

The Descend Form *Open in* option, in this scenario, will show *new window* as the default descend selection.

- If `setWinTraversal` is not set, then the value set for `descendCanvasType` will be checked.

The Descend Form *Open in* option, in this scenario, will show whatever option has been set for `descendCanvasType`, the default being *current tab*.

Note: It is recommended that you use `descendCanvasType` as it is a schematic environment variable in preference to using `setWinTraversal` which is not a schematic environment variable.

GUI Equivalent

Command	<i>Edit – Hierarchy – Descend Edit</i>
	<i>Edit – Hierarchy – Descend Read</i>
Form Field	<i>Open in</i>

Related Topics

[Descend Edit/Read](#)

descendDemoteViewNameList

```
schematic descendDemoteViewNameList string { "view_to_be_filtered" }
```

Description

Specifies a space-separated list of view names that appear at the end of the descend choices. Definition of this environment variable prevents you from descending into a view that is not useful for editing or interaction. This environment variable affects only the VSE *Descend* command. For example, in the parasitic flow, the estimated view is bound by the configuration. Therefore, it appears as the first choice in the *viewNameList* when descending into a view. However, you can define this environment variable as follows to place the *estimated* view at the end of the descend choices:

```
envSetVal("schematic" "descendDemoteViewNameList" 'string "estimated")
```

The default behavior of the *Descend* command is that the descend view choices are available in a sequence as specified by the configuration or the *viewNameList*, followed by any other views of the cell in an alphabetical order. Therefore, the default value for this environment variable is "" (an empty string), which means there is no change to the default behavior of the *Descend* command.

GUI Equivalent

None

descendSheet

```
schematic descendSheet string { "sheet" | "index" }
```

Description

Specifies the type of sheet to descend into when descending into a level that expands to a multisheet schematic. The default is `sheet`.

GUI Equivalent

Command	<i>Edit – Hierarchy – Descend Edit</i>
Form Field	N/A

descendTarget

```
schematic descendTarget string { "query user" | "use viewNameList" }
```

Description

Specifies whether the schematic editor descends automatically when you start a *Descend* command. The default is `query user`.

If `use viewNameList` is specified, the descend command uses the first view found in the `viewNameList`. The View Name List is specified in the Editor Options, or the `config` or `physConfig` views.

Note: When using the schematic view with:

- Layout XL, the `viewNameList` is specified by the `physConfig`. For details of how to change the `physConfig`, see Configure Physical Hierarchy.
- the Hierarchy Editor, the `viewNameList` is specified by the `config`. For details of how to change the `config`, see Hierarchy Editor.

To avoid descending into views such as extracted views, use the environment variable descendDemoteViewNameList.

GUI Equivalent

Command	<i>Edit – Hierarchy – Descend Edit</i>
	<i>Edit – Hierarchy – Descend Read</i>
Form Field	<i>Target Cellview</i>

Related Topics

Descending Using the Descend Command

Descend Edit/Read

differential

```
schematic differential boolean { t | nil }
```

Description

Used by the `schHiCreatePin()` command. When when set to `t` will add a *Differential Expansion* field to the Create Pin form. The default is `nil`.

The differential variable is used to allow you to use the *Virtuoso Schematic Editor* for differential circuit design.

The use model is as follows:

Firstly, set the global variable `differential` to `t`. This can be done by calling `schSetEnv("differential" t)` in your `.cdsinit` file.

Secondly, modify the differential name expansion algorithm as necessary and load from your `.cdsinit` file.

The current differential name expansion algorithm has the following feature:

1. Simple, vector and bundle names can be expanded.
2. A simple name is always expanded by creating a vector of 2 bits wide.

For example, `A` expands to `A<0:1>`, `B<2>` expands to `B<4:5>`

3. A vector name is always expanded by calculating the proper starting index and then doubling the width of the given vector.

For example, `NAME<2:5>` expands to `NAME<4:11>`, `NAME<3>` expands to `NAME<6:7>`

The starting part is calculated by $s = v1 * 2$

The ending part is calculated by $e = v2 * 2 + 1$

4. A bundle name is handled by breaking the bundle name into simple or vector names and applies the above rules.

For example, `A, B` expands to `A<0:1>`, `B<0:1>`, `D<1,3>` expands to `D<2:3>`, `D<6:7>`

GUI Equivalent

None

dimming

```
schematic dimming boolean { t | nil }
```

Description

Allows you to set object dimming *on* (`t`) or *off* (`nil`) for selected, highlighted, or probed objects on the design canvas.

For more information, see [Display Dimming](#) and the [Display Options](#) form.

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Display Dimming</i>

Related Topics

[Display Options](#)

dimmingIntensity

schematic dimmingIntensity int *intensity_level*

Description

Allows you to set the intensity (brightness) display of the dimmed objects, where 0 equals the dimmed object display being the same as the selected object, right up to 100 which will cause the dimmed objects to be “invisible” on the canvas. The default setting is 50.

For more information, see [Display Dimming](#) and the [Display Options](#) form.

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Intensity</i>

Related Topics

[Display Options](#)

dimmingMode

```
schematic dimmingMode string { "Automatic" | "Always" }
```

Description

Allows you to choose whether dimming be `Automatic` (the default); where dimming will only be enabled when at least one object is currently selected, highlighted, or probed. Or, choose to set dimming to `Always` which will enable dimming permanently (until the option setting is changed).

For more information, see [Display Dimming](#) and the [Display Options](#) form.

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Mode (Display Controls)</i>

Related Topics

[Display Options](#)

directEditOnlySelected

```
schematic directEditOnlySelected boolean { t | nil }
```

Description

When set to `t`, the actions of the `schDirectEdit()` command apply only to objects that are already selected. When set to the default value `nil`, no pre-selection is required for edits.

GUI Equivalent

None

disablePortOrderPopup

```
schematic disablePortOrderPopup boolean { t | nil }
```

Description

Fixes the port mismatch issues whenever possible without displaying the Port Mismatch form. The default is `nil`.

For details, see [Resolving Pin Mismatch](#).

GUI Equivalent

None

dotOverlap

schematic dotOverlap float *overlap*

Description

Specifies the value for automatically generated solder dots that overlap wires at a wire intersection. The value must be specified in user units. The default is 0.03125 (inches).

For details, see [Editor Options](#).

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Solder Dot Wire Overlap (Commands tab)</i>

Related Topics

[Editor Options](#)

Virtuoso Schematic Editor User Guide

Environment Variables

drawMode

```
schematic drawMode string { "anyAngle" | "190X" | "190Y" | "145Long" | "145Angle" }
```

Description

Specifies the drawing mode for the *Shape* commands. The default is `anyAngle`.

GUI Equivalent

Command	<i>Create – Note – Shape (in the symbol editor)</i>
	<i>Create – Shape – Line (in the symbol editor)</i>
	<i>Create – Shape – Polygon (in the symbol editor)</i>
Form Field	<i>Draw Mode</i>

Related Topics

[Create Note Shape](#)

[Add Symbol Shape](#)

drawModeRoute

```
schematic drawModeRoute string { "route" | "anyAngle" | "190X" | "190Y" | "145Long"
    | "145Angle" }
```

Description

Specifies the drawing mode for the *Create Wire* command. The difference between this and *drawMode* is the route choice, which invokes the wire router when adding wires to a schematic. The default is `route`.

For a sample drawing of the values, see [Add Wire](#).

GUI Equivalent

Command	<i>Create – Wire (narrow)</i> <i>Create – Wire (wide)</i>
Form Field	<i>Draw Mode</i>

Related Topics

[Add Wire](#)

useDynamicHilightPacket

```
schematic useDynamicHilightPacket boolean { t | nil }
```

Description

Creates the dynamic highlight display packet and uses it to draw dynamic highlights in the design. The default is `t`.

When set to `nil`, the dynamic highlight is drawn with a non-configurable thin dashed outline.

When set to `t`, and you do not have the `dynamicHilight` display packet defined in the `display.drf` file, then the packet is created internally on the first dynamically highlighted figure. However, this packet is not saved automatically in `display.drf`. Therefore, if you close Virtuoso at this stage, `display.drf` is not modified. The next time you open Virtuoso and then DRE, the `dynamicHilight` display packet is not displayed.

However, if you open DRE, save `display.drf`, and then, close Virtuoso, the next time you open Virtuoso and DRE, the `dynamicHilight` display packet is present, even if the environment variable is set to `nil`.

GUI Equivalent

None

Virtuoso Schematic Editor User Guide

Environment Variables

echoToCIW

```
schematic echoToCIW boolean { t | nil }
```

Description

Prints a summary of error and warning messages to the CIW (t) when you check your design. This is the same summary displayed in the dialog box when warning and error conditions are encountered during a check of your design. The default is t.

GUI Equivalent

Command	<i>Check</i>
Form Field	N/A

editPropChangeAllConfirm

```
schematic editPropChangeAllConfirm string { "instance" | "pin" | "pin inherited" |  
      "symbol pin" | "symbol pin inherited" }
```

Description

Specifies the object type groups that require confirmation before applying the same property modification to multiple objects of the type selected on the Edit Object Properties – Basic form. This applies only when the *Apply To* field is set to *all selected* or *all*. The object type groups must each be separated by a comma and the list enclosed in quotation marks.

The default is "instance,pin,pin inherited,symbol pin,symbol pin inherited".

GUI Equivalent

None

Example

```
envGetVal("schematic" "editPropChangeAllConfirm")  
envSetVal("schematic" "editPropChangeAllConfirm" 'string "instance, pin, pin  
inherited, symbol pin, symbol pin inherited")  
envSetVal("schematic" "editPropChangeAllConfirm" 'string "instance, pin")
```

editPropProtectCDF

schematic editPropProtectCDF boolean t

Description

Controls whether the *Edit Object Properties* form uses its own distinct copy of cdfgData, or the same cdfgData as other forms in Virtuoso. Set the variable to `t` in order to prevent other forms, for example *VLS Edit Properties*, from unexpectedly altering the CDF parameters shown in the *VSE Edit Object Properties* form.

However, if you want share the CDF parameters shown in the *Edit Object Properties* form with other forms, set `editPropProtectCDF` to `nil` to allow the secondary form to share the same cdfgData as the main *Edit Object Properties* form. For example, you might have a button CDF parameter that pops up a secondary property editor form, the secondary form will share the data of the primary *Edit Object Properties* form.

The default value is `t`.

GUI Equivalent

Command	<i>Edit – Properties – Objects</i> and <i>Show CDF</i> selected
Form Field	All CDF parameter fields

Related Topics

[Edit Object Properties – Basic](#)

editPropReadOnlyFields

```
schematic editPropReadOnlyFields boolean { t | nil }
```

Description

When `t`, the fields on the form are not editable for read-only cellviews. When `nil`, the fields on [Edit Object Properties – Basic](#) form are editable for read-only cellviews.

Note:

- If you try to apply the edits made to the fields on the form, a warning message will be displayed saying that the edits cannot be applied.
- You might want to use this environment variable if you have parameter fields that are used for browsing but not editing.

GUI Equivalent

None

Example

```
envGetVal("schematic" "editPropReadOnlyFields")
envSetVal("schematic" "editPropReadOnlyFields" 'boolean t)
envSetVal("schematic" "editPropReadOnlyFields" 'boolean nil)
```

editPropReadOnlyNetName

```
schematic editPropReadOnlyNetName boolean { t | nil }
```

Description

Controls whether the Edit Object Properties – Wire Name form allows you to edit the net name of a selected wire for read-only cellviews. Editing a net name changes all wire labels associated with the net across the entire schematic; the actual net is changed only when you save a design using *Check and Save*.

When set to `t`, the default value, net names are uneditable on the Edit Object Properties form for read-only cellviews. When not set, net names are editable on the Edit Object Properties form for read-only cellviews.

GUI Equivalent

None

Example

```
envGetVal("schematic" "editPropReadOnlyNetName")
envSetVal("schematic" "editPropReadOnlyNetName" 'boolean t)
envSetVal("schematic" "editPropReadOnlyNetName" 'boolean nil)
```

editPropShowCDF

```
schematic editPropShowCDF boolean { t | nil }
```

Description

Specifies whether the *CDF* options field appears in the *Show* region of the Edit Object Properties form (*t*) or not (*nil*). The default is *t*.

GUI Equivalent

Command	<i>Edit – Properties – Objects</i> and <i>Show CDF</i> selected
Form Field	<i>Show CDF</i>

Related Topics

[Edit Object Properties – Basic](#)

editPropShowSystem

```
schematic editPropShowSystem boolean { t | nil }
```

Description

Specifies whether the *System* options field appears in the *Show* region of the Edit Object Properties form (t) or not (nil). The default is nil.

GUI Equivalent

Command	<i>Edit – Properties – Objects</i>
	<i>Edit – Properties – Cellview</i>
Form Field	<i>Show System</i>

Related Topics

[Edit Object Properties – Basic](#)

[Edit Cellview Properties](#)

editPropShowUser

```
schematic editPropShowUser boolean { t | nil }
```

Description

Specifies whether the *User* check box appears in the *Show* region of the Edit Object Properties form (t) or not (nil). The default is t.

GUI Equivalent

Command	<i>Edit – Properties – Objects</i> <i>Edit – Properties – Cellview</i>
Form Field	<i>Show User</i>

Related Topics

[Edit Object Properties – Basic](#)

[Edit Cellview Properties](#)

findAccessMode

```
schematic findAccessMode string { "edit" | "read only" }
```

Description

Specifies which mode you are in. The default is `edit`.

GUI Equivalent

Command	<i>Edit – Properties – Cellview</i>
Form Field	<i>Mode</i>

Related Topics

[Edit Cellview Properties](#)

findZoomButton

```
schematic findZoomButton boolean { t | nil }
```

Description

Specifies whether the schematic editor zooms into an object that matches the pattern specified in the *Find* command (*t*) or not (*nil*). The default is *nil*.

GUI Equivalent

Command	<i>Edit – Find</i>
Form Field	<i>Zoom To Object</i>

Related Topics

[Schematic Find](#)

flattenConnectMode

```
schematic flattenConnectMode cyclic { "by-name" | "flight" | "wire" }
```

Description

Sets the default connect mode for the Flatten command.

- `by-name` connects the flattened content to the existing schematic using connect-by-name.
- `flight` connects the flattened content to the existing schematic using flight lines.
- `wire` connects the flattened content to the existing schematic using routed wires.

The default is `by-name`.

GUI Equivalent

Command	<i>Edit – Hierarchy – Flatten</i>
Form Field	<i>Connect Mode</i>

Related topics

[Flatten](#)

flightWidth

schematic flightWidth float *line_width*

Description

Specifies the width of the dashed, straight line between two points. The default is 0 . 0.

GUI Equivalent

Command	<i>Create – Wire</i>
Form Field	<i>flight</i>

Related Topics

[Add Wire](#)

gravityOn

```
schematic gravityOn boolean { t | nil }
```

Description

Specifies whether the cursor snaps to instance pins when you use the *Create Wire* command (t) or not (nil). The default is t.

For details, see [Editor Options](#).

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Gravity (Commands tab)</i>

Related Topics

[Editor Options](#)

ignoreHighlightColor

```
schematic ignoreHighlightColor string "red"
```

Description

Specifies the colour of the cross displayed on the ignored instances in the schematic when the `ignoreHighlightOn` value is set to `t`. The default value is `red`.

For details, see [Ignoring Instances](#).

GUI Equivalent

None

Example

```
envGetVal ("schematic" "ignoreHighlightColor")  
envSetVal ("schematic" "ignoreHighlightColor" 'string "red")  
envSetVal ("schematic" "ignoreHighlightColor" 'string "blue")
```

ignoreHighlightOn

```
schematic ignoreHighlightOn boolean { t | nil }
```

Description

Highlights the ignored instances in the schematic by displaying a cross on the instances (t) or removing cross (nil). The default value is t.

For details, see [Ignoring Instances](#).

GUI Equivalent

Command	<i>Options – Display Options</i>
Form Field	<i>Ignored Instances</i>

Example

```
envGetVal ("schematic" "ignoreHighlightOn")
envSetVal ("schematic" "ignoreHighlightOn" 'boolean t)
envSetVal ("schematic" "ignoreHighlightOn" 'boolean nil)
```

Related Topics

[Display Options](#)

ignoreShowMessage

```
schematic ignoreShowMessage boolean { t | nil }
```

Description

With the *Ignored Instances* option on the [Display Options](#) form disabled when you select an instance and choose *Edit – Ignore Instances* the View Ignored Instances dialog box is displayed.

If the `ignoreShowMessage` environment variable is set to `t`, the View Ignored Instances dialog box does not appear and changes take place without the warning.

For details, see [Ignoring Instances](#).

GUI Equivalent

Command	<i>Edit – Ignore Instances</i> (With the <i>Ignored Instances</i> option on the Display Options form disabled)
Form Field	<i>Do not show this dialog again</i> (View Ignored Instances dialog box)

Example

```
envGetVal ("schematic" "ignoreShowMessage")  
envSetVal ("schematic" "ignoreShowMessage" 'boolean t)  
envSetVal ("schematic" "ignoreShowMessage" 'boolean nil)
```

Related Topics

[Display Options](#)

incrementalConn

```
schematic incrementalConn boolean { t | nil }
```

Description

Controls the incremental connectivity mode used to update the connectivity in a design while you are editing.

- The default is `t`, which means that connectivity is automatically checked and updated while you edit.
- When set to `nil`, batch connectivity mode is used to update connectivity. This requires an explicit update request using the *File – Check and Save*, *Check – Current Cellview*, or *Check – Hierarchy* commands to update the connectivity in a design.

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Incrementally update connectivity while editing (Connectivity tab)</i>

Example

```
schSetEnv("incrementalConn" t)  
envSetVal("schematic" "incrementalConn" 'boolean t)  
envSetVal("schematic" "incrementalConn" 'boolean nil)
```

Related Topics

[Updating Connectivity](#)

[Editor Options](#)

incrementalConnTimeout

schematic incrementalConnTimeout float *timeout*

Description

Specifies the maximum time in seconds permitted for an incremental connectivity extraction operation. If an operation takes longer than the specified time, incremental connectivity extraction is automatically paused. The default is 0.1s.

GUI Equivalent

None

Example

```
schSetEnv("incrementalConnTimeout" 0.200)
envSetVal("schematic" "incrementalConnTimeout" 'float 0.200)
```

Related Topics

[Incremental Connectivity Extraction Status](#)

[Resuming a Paused Incremental Connectivity Extraction](#)

[Updating Connectivity](#)

inhConnDisableDefNetField

```
schematic inhConnDisableDefNetField boolean { t | nil }
```

Description

Disables editing of the *Default Net Name* field on the Create Wire Name and Create Net Expression forms to prevent modification of generated values. The default is `nil`.

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Disable editing of default net name field (Inherited Conn tab)</i>

Related Topics

Editor Options

inhConnDisablePinFields

```
schematic inhConnDisablePinFields boolean { t | nil }
```

Description

Disables the net expression editing fields on the Create Pin form to prevent modification of the generated values. The default is `nil`.

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Disable editing of property and default net name fields (Inherited Conn tab)</i>

Related Topics

[Editor Options](#)

inhConnGenNetExprDefNet

```
schematic inhConnGenNetExprDefNet boolean { t | nil }
```

Description

Enables the *Default Net Name* field in the Create Net Expression form, to be automatically seeded based on the *Property Name* specified. This value will continue to update if multiple pins are added simultaneously. The default is `nil`.

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Derive default net name from property name (Inherited Conn tab)</i>

Related Topics

[Editor Options](#)

inhConnGenPinNetExpr

```
schematic inhConnGenPinNetExpr boolean { t | nil }
```

Description

Enables the *Property Name* and *Default Net Name* fields on the Create Pin form to be automatically seeded with values derived from the *Pin Name* currently being added. These values will continue to update if multiple pins are added simultaneously. The default is `nil`.

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Derive property name and default net name from pin name (Inherited Conn tab)</i>

Related Topics

[Editor Options](#)

instObjectSelectionMode

```
schematic instObjectSelectionMode cyclic { "point" | "any" }
```

Description

Specifies the selection mode for instance objects. If `point` is specified, you can select instance objects only by clicking on them. If `any` is selected, you can select instance objects using area/line or select all queries. The default is `point`.

GUI Equivalent

Command	<i>Options – Select Filter</i>
Form Field	<i>Scope</i>

Example

```
schSetEnv("instObjectSelectionMode" "point")  
envSetVal("schematic" "instObjectSelectionMode" 'cyclic "point")  
envSetVal("schematic" "instObjectSelectionMode" 'cyclic "any")
```

Related Topics

[Schematic Selection Filter](#)

Virtuoso Schematic Editor User Guide

Environment Variables

invokeKanjiEditor

```
schematic invokeKanjiEditor string { "kterm -km euc -e vi" }
```

Description

Starts the Kanji Editor.

GUI Equivalent

None

labelsVisibleDuringDrag

```
schematic labelsVisibleDuringDrag boolean { t | nil }
```

Description

Specifies whether instance labels should be rendered during Add Instance, Move, Stretch, or Copy commands (t). This option only applies to labels with a LabelType of *normalLabel*. The default is t.

GUI Equivalent

None

Virtuoso Schematic Editor User Guide

Environment Variables

libList

```
schematic libList string "list_of_libraries"
```

Description

Specifies a list of libraries the schematic editor processes during hierarchical checking when the environment variable `spanLib` is set to `check`. The default is an empty string (" ").

GUI Equivalent

Command	<i>Check – Hierarchy</i>
Form Field	<i>Reference Libraries</i>

Related Topics

[Check Hierarchy](#)

magAnchorPosition

This function has been included within this appendix for completeness. For details on its description, type, and value refer to the [magAnchorPosition](#) function in the *Virtuoso Layout Suite XL: Basic Editing User Guide*.

Note: When entering this function in *Schematics L/XL*, you should specify that the application is `schematic` instead of `layout`.

magConfined

This function has been included within this appendix for completeness. For details on its description, type, and value refer to the [magConfined](#) function in the *Virtuoso Layout Suite XL: Basic Editing User Guide*.

Note: When entering this function in *Schematics L/XL*, you should specify that the application is `schematic` instead of `layout`.

magCursorOffset

This function has been included within this appendix for completeness. For details on its description, type, and value refer to the [magCursorOffset](#) function in the *Virtuoso Layout Suite XL: Basic Editing User Guide*.

Note: When entering this function in *Schematics L/XL*, you should specify that the application is `schematic` instead of `layout`.

magDisplayMode

This function has been included within this appendix for completeness. For details on its description, type, and value refer to the [magDisplayMode](#) function in the *Virtuoso Layout Suite XL: Basic Editing User Guide*.

Note: When entering this function in *Schematics L/XL*, you should specify that the application is `schematic` instead of `layout`.

magFloatPosition

This function has been included within this appendix for completeness. For details on its description, type, and value refer to the [magFloatPosition](#) function in the *Virtuoso Layout Suite XL: Basic Editing User Guide*.

Note: When entering this function in *Schematics L/XL*, you should specify that the application is `schematic` instead of `layout`.

magOffsetCorner

This function has been included within this appendix for completeness. For details on its description, type, and value refer to the [magOffsetCorner](#) function in the *Virtuoso Layout Suite XL: Basic Editing User Guide*.

Note: When entering this function in *Schematics L/XL*, you should specify that the application is `schematic` instead of `layout`.

magScale

This function has been included within this appendix for completeness. For details on its description, type, and value refer to the [magScale](#) function in the *Virtuoso Layout Suite XL: Basic Editing User Guide*.

Note: When entering this function in *Schematics L/XL*, you should specify that the application is `schematic` instead of `layout`.

magSize

This function has been included within this appendix for completeness. For details on its description, type, and value refer to the [magSize](#) function in the *Virtuoso Layout Suite XL: Basic Editing User Guide*.

Note: When entering this function in *Schematics L/XL*, you should specify that the application is `schematic` instead of `layout`.

makeCellFactorize

```
schematic makeCellFactorize boolean { t | nil }
```

Description

Specifies whether the *Simplify iterated names* check box is selected by default. If the check box is selected, any instances and wire labels in the selection with a common iteration factor have this common factor removed, and an iterated instance of the new scalar cell is created. The default is `t`.

GUI Equivalent

Command	<i>Edit – Hierarchy – Make Cell</i>
Form Field	<i>Simplify iterated names</i>

Related Topics

[Make Cell](#)

makeCellPlaceSymbol

```
schematic makeCellPlaceSymbol boolean { t | nil }
```

Description

Specifies whether the *Place symbol* check box is selected by default. If the check box is selected, a symbol is automatically generated for the newly created cell. The default is `t`.

GUI Equivalent

Command	<i>Edit – Hierarchy – Make Cell</i>
Form Field	<i>Place symbol</i>

Related Topics

[Make Cell](#)

mapCvMappingSchViewName

```
schematic mapCvMappingSchViewName string "schematic"
```

Description

Specifies the mapping schematic view name. The default is `schematic` and it can be changed as required.

GUI Equivalent

Command	<i>Create – Mapping Schematic</i>
Form Field	<i>View Name</i>

Related Topics

[Create a Mapping Schematic Step 1 of 2](#)

mapCvTermDirectionCheck

```
schematic mapCvTermDirectionCheck cyclic { "unknown" | "ignored" | "warning" |  
    "error" }
```

Description

Controls how direction mismatches of mapped nets are reported when creating a mapping schematic on the [Create a Mapping Schematic](#) form. This lets you control the emphasis of the direction mismatch required between upper and lower level terminal direction. Messages will be displayed in the CIW. The default is `error`.

GUI Equivalent

None

Related topics

[Creating a Mapping Schematic](#)

markerSeverity

This function has been included within this appendix for completeness. For details on its description, type, and value refer to the [markerSeverity](#) function in the *Virtuoso Layout Suite XL: Basic Editing User Guide*.

Note: When entering this function in *Schematics L/XL*, you should specify that the application is `schematic` instead of `graphic`.

markerShowIgnore

```
schematic markerShowIgnore boolean { t | nil }
```

Description

Specifies whether the Find Marker form displays markers that have the *ignore* property set (t) or not (nil). The default is t.

GUI Equivalent

Command	<i>Check – Find Marker</i>
Form Field	<i>Show Ignored</i>

Related Topics

[Find Marker](#)

markerZoom

```
schematic markerZoom boolean { t | nil }
```

Description

Specifies whether the *Find Marker* command zooms to the current marker (t) or not (nil). The default is nil.

GUI Equivalent

Command	<i>Check – Find Marker</i>
Form Field	<i>Zoom to Markers</i>

Related Topics

[Find Marker](#)

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*.

Note: When entering this function in *Schematics L/XL*, you should specify that the application is `schematic` instead of `constraint`.

maxLabelOffsetUU

schematic maxLabelOffsetUU float *max_distance*

Description

Specifies the maximum distance between a label and a wire. The default is 0.045 (inches).

During schematic loading and connectivity extraction, any unglued wire names whose control point is within this distance of a wire is automatically glued to the wire. During schematic rules checking, this value is used in the *maxLabelOffset* check to flag any wire name that is farther than this distance from any wire whether or not the name is glued to the wire.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Maximum Offset From Wire (Physical tab)</i>

Related Topics

[Schematic Rules Checks Setup](#)

minMarkerSize

schematic minMarkerSize float *min_marker_size*

Description

Specifies a value for the minimum marker size. The value is specified in user units. The default is 0.05 (inches).

For details, see [Editor Options](#).

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Minimum Marker Size (Commands tab)</i>

Related Topics

[Editor Options](#)

modalCommands

```
schematic modalCommands boolean { t | nil }
```

Description

Specifies whether commands stay active until explicitly canceled by pressing the `Esc` key (`t`) or by starting another command (`nil`). The default is `t`.

For details, see [Canceling the Repeat Command Option](#).

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Repeat Commands (Commands tab)</i>

Related Topics

[Editor Options](#)

netExprDisplay

```
schematic netExprDisplay cyclic { "value only" | "value (default)" | "value  
[property]" | "value [expression]" | "expression only" }
```

Description

Specifies how to display inherited connections values. The value is what the connection resolves to; if it is overridden at a higher level in the hierarchy, the full path to a local signal might be displayed. You must type the square brackets as part of `value [property]` or `value [expression]`. The default is `value only`.

For details, see [Changing Display Options](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Net Expression Display</i>

Related Topics

[Display Options](#)

netNameMethod

```
schematic netNameMethod cyclic { "prefix" | "derived" }
```

Description

Specifies which naming mechanism is used for system-generated net names during connectivity extraction. The default is `prefix`.

The naming mechanisms work as follows:

- `prefix` generates a unique name for nets that you do not explicitly label or connect to a pin. The unique name consists of the prefix that you specify in the *Net Name Prefix* field followed by a number.
- `derived` generates a net name from the driving instance name and the driving terminal name. The derived net name is of the form *instName_pinName*.

Note: Instance pins named with bundle names, for example, A, B, C, are not used to derive system-generated names.

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Net Name Method (Connectivity tab)</i>

Examples

```
envGetVal("schematic" "netNameMethod")  
envSetVal("schematic" "netNameMethod" 'cyclic "derived")
```

Related topics

[Editor Options](#)

netNamePrefix

```
schematic netNamePrefix string "prefix_string"
```

Description

Specifies the prefix string for the net names generated during connectivity extraction. The default is `net`.

For details, see [Editor Options](#).

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Net Name Prefix (Connectivity tab)</i>

Related topics

[Editor Options](#)

netNameRangeAuto

```
schematic netNameRangeAuto boolean { t | nil }
```

Description

Auto-detects the range order of the object for which a bus has to be generated and assigns the same range order to the bus. If the range order cannot be decided, it uses the value of *Bus Range Order* on the Editor Options form. For example, if the cellview bus range order is *descending* and instance is A<1:3> and instance pin is B<1:3>, the environment variable when set to *t* enables generating net<0:8> with an *ascending* order. The default is *nil*.

For details, see [Editor Options](#). Also, refer to [netNameRangeDescend](#).

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Bus Range Order (Connectivity tab)</i>

Related topics

[netNameRangeDescend](#)

[Editor Options](#)

netNameRangeDescend

```
schematic netNameRangeDescend boolean { t | nil }
```

Description

Specifies whether the range specification in system-generated buses is ascending (`nil`) or descending (`t`). The default is `nil`.

For details, see [Editor Options](#).

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Bus Range Order (Connectivity tab)</i>

Related topics

[Editor Options](#)

netNameSeparator

schematic netNameSeparator string "name_separator"

Description

Specifies the name separator for net names generated during connectivity extraction if the value of netNameMethod is derived. The default is "_".

For details, see [Editor Options](#).

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Net Name Separator (Connectivity tab)</i>

Related topics

[Editor Options](#)

noteFontHeight

schematic noteFontHeight float *font_height*

Description

Specifies a font height for the text you add to your design using the *Create – Note – Text* command. The default is 0.0625 (inches).

GUI Equivalent

Command	<i>Create – Note – Text</i>
Form Field	<i>Font Height</i> (<u>Create Note Text</u> form)

Related topics

Create Note Text

noteFontStyle

```
schematic noteFontStyle cyclic { "euroStyle" | "gothic" | "math" | "roman" |  
    "script" | "stick" | "fixed" | "swedish" | "milSpec" }
```

Description

Specifies a font for the text you add to your design using the *Create – Note – Text* command. The default is `stick`.

GUI Equivalent

Command	<i>Create – Note – Text</i>
Form Field	<i>Font Style</i>

Related topics

[Create Note Text](#)

Virtuoso Schematic Editor User Guide

Environment Variables

noteJustify

```
schematic noteJustify cyclic { "upperLeft" | "centerLeft" | "lowerLeft" |  
    "upperCenter" | "centerCenter" | "lowerCenter" | "upperRight" | "centerRight"  
    | "lowerRight" }
```

Description

Specifies the justification for the text you add to your design using the *Create – Note – Text* command. The default is `lowerCenter`.

GUI Equivalent

Command	<i>Create – Note – Text</i>
Form Field	<i>Justification</i>

Related topics

[Create Note Text](#)

noteShape

```
schematic noteShape cyclic { "line" | "rectangle" | "polygon" | "circle" | "ellipse"  
    | "arc" }
```

Description

Specifies the shape for a note, which you add to your design using the *Create – Note – Shape* command. The default is `line`.

GUI Equivalent

Command	<i>Create – Note – Shape</i>
Form Field	<i>Shape</i>

Related topics

[Create Note Shape](#)

noteShapeLineStyle

```
schematic noteShapeLineStyle cyclic { "solid" | "dashed" }
```

Description

Specifies the line style used to draw a note shape. The default is `solid`.

GUI Equivalent

Command	<i>Create – Note – Shape</i>
Form Field	<i>Line Style</i>

Related topics

[Create Note Shape](#)

noteType

```
schematic noteType cyclic { "normalLabel" | "NLPLabel" | "ILLabel" }
```

Description

Specifies the note type. The default is `normalLabel`.

GUI Equivalent

None

objectInfoInStatusBar

```
schematic objectInfoInStatusBar boolean { t | nil }
```

Description

Displays the information about the selected objects on the status bar. The default is `t`.

GUI Equivalent

None

pinBackTrace

```
schematic pinBackTrace boolean { t | nil }
```

Description

Specifies whether the schematic editor stretches the wires connected to a pin when you stretch a pin (*t*) or not (*nil*). The default is *t*.

GUI Equivalent

Command	<i>Edit – Stretch</i>
Form Field	<i>Reroute Wire Segments Connected To</i>

Related topics

[Stretch – Schematic View](#)

[Stretch – Symbol View](#)

pinLabelDrafting

```
schematic pinLabelDrafting boolean { t | nil }
```

Description

Specifies that labels are never drawn upside-down or mirrored, even when rotated, therefore making them easier to read. The default is `t`.

Note: This variable is only effective if the pinLabelSetDrafting variable is first set to `t`.

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Drafting (Labels tab)</i>

Related topics

Editor Options

pinLabelFontHeight

schematic pinLabelFontHeight float *font_height*

Description

Defines the font height of pin labels. The default is 0.0625.

Note: This variable is only effective if the pinLabelSetFontHeight variable is first set to t.

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Font height (Labels tab)</i>

Related topics

[Editor Options](#)

pinLabelFontStyle

```
schematic pinLabelFontStyle cyclic { "euroStyle" | "gothic" | "math" | "roman" |  
    "script" | "stick" | "fixed" | "swedish" | "milSpec" }
```

Description

Specifies the font style for the pin label. The default is `stick`.

Note: This variable is only effective if the [pinLabelSetFontStyle](#) variable is first set to `t`.

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Font style (Labels tab)</i>

Related topics

[Editor Options](#)

pinLabelJustify

```
schematic pinLabelJustify cyclic { "upperLeft" | "centerLeft" | "lowerLeft" |  
    "upperCenter" | "centerCenter" | "lowerCenter" | "upperRight" | "centerRight"  
    | "lowerRight" }
```

Description

Specifies where the label appears in relation to the pin's origin (placement location). The default is `lowerCenter`.

Note: This variable is only effective if the pinLabelSetJustify variable is first set to `t`.

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Justification (Labels tab)</i>

Related topics

Editor Options

pinLabelOverbar

```
schematic pinLabelOverbar boolean { t | nil }
```

Description

Specifies whether overbars are displayed over pin labels. When this option is disabled (default), the software displays underscore characters (`_`) in the pin label as part of the label text string. When this option is enabled, the software interprets underscore characters (`_`) in the label as toggle switches that control where overbars begin and end. Overbars appear above the pin label.

Note: This variable is only effective if the [pinLabelSetOverbar](#) variable is first set to `t`.

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Overbar (Labels tab)</i>

Related topics

[Editor Options](#)

pinLabelSetDrafting

```
schematic pinLabelSetDrafting boolean { t | nil }
```

Description

Controls whether the default setting for the variable `pinLabelDrafting` can be changed. The default is `t`.

Note: If set to `t`, the variable `pinLabelDrafting` also needs to be defined.

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Override drafting (Labels tab)</i>

Related topics

Editor Options

pinLabelSetFontHeight

```
schematic pinLabelSetFontHeight boolean { t | nil }
```

Description

Controls whether the default setting for the variable `pinLabelFontHeight` can be changed. The default is `nil`.

Note: If this variable is set to `t`, a font height must be defined for the variable `pinLabelFontHeight`.

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Override font height (Labels tab)</i>

Related topics

[Editor Options](#)

pinLabelSetFontStyle

```
schematic pinLabelSetFontStyle boolean { t | nil }
```

Description

Controls whether the default setting for the variable `pinLabelFontStyle` can be changed. The default is `nil`.

Note: If this variable is set to `t`, a font style must be defined for the variable `pinLabelFontStyle`.

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Override font style (Labels tab)</i>

Related topics

[Editor Options](#)

pinLabelSetJustify

```
schematic pinLabelSetJustify boolean { t | nil }
```

Description

Controls whether the default setting for the variable `pinLabelJustify` can be changed. The default is `nil`.

Note: If this variable is set to `t`, the positioning of labels must be defined for the variable `pinLabelJustify`.

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Override justification (Labels tab)</i>

Related topics

[Editor Options](#)

pinLabelSetOverbar

```
schematic pinLabelSetOverbar boolean { t | nil }
```

Description

Controls whether the default setting for the variable `pinLabelOverbar` can be changed. The default is `nil`.

Note: If this variable is set to `t`, the display of overbars on pin labels must be defined for the variable `pinLabelOverbar`.

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Override overbar (Labels tab)</i>

Related topics

[Editor Options](#)

pinNameSeed

```
schematic pinNameSeed string "pin_name_prefix"
```

Description

Specifies the pin name prefix used when automatically creating new pins for blocks you add to your design. The default is `pin`.

GUI Equivalent

Command	<i>Create – Block</i>
Form Field	<i>Pin Name Prefix</i>

Related topics

[Create Block](#)

pinRefDuplicates

```
schematic pinRefDuplicates boolean { t | nil }
```

Description

Specifies whether the editor reports duplicate pin references found within one zone of a multisheet schematic (`t`) or not (`nil`). The default is `t`.

GUI Equivalent

Command	<i>Sheet – Cross Reference</i>
Form Field	<i>Show Duplicate Zones</i>

Related topics

[Cross Reference Options](#)

pinRefFormat

```
schematic pinRefFormat string { "<sheetNumber><zone><referenceName>" }
```

Description

Specifies the format for displaying the pin reference information that appears next to each pin when you use the *Cross Reference* command. The default is `<sheetnumber><zone><referenceName>`.

GUI Equivalent

Command	<i>Sheet – Cross Reference</i>
Form Field	<i>Format String</i>

Related topics

[Cross Reference Options](#)

pinRefSorting

```
schematic pinRefSorting string { "Direction" | "Sheet Number" }
```

Description

Specifies how pin references are sorted when you use the *Cross Reference* command. The default is `Sheet Number`.

GUI Equivalent

Command	<i>Sheet – Cross Reference</i>
Form Field	<i>Sort References By</i>

Related topics

[Cross Reference Options](#)

postProcNetSigTypes

```
schematic postProcNetSigTypes boolean { t | nil }
```

Description

Sets the signal type for global or registered power and ground nets, while saving a design using *Check and Save*. The schematic editor automatically identifies all of the global or registered power or ground nets and sets their signal type accordingly. The default is `nil`.

GUI Equivalent

Command	<i>Options – Check</i>
Form Field	<i>Set Signal Type from Net Name and Type Registration</i>

Related topics

[Schematic Check Options](#)

probeConnectedInstances

```
schematic probeConnectedInstances boolean { t | nil }
```

Description

Highlights all the instances that are connected to the net (except for the feedthrough connections) if that net is probed. The default is `nil`.

GUI Equivalent

None

probeCycleBool

```
graphic probeCycleBool boolean { t | nil }
```

Description

Specifies whether to cycle the probe layer-purpose pairs. If cycling, the current layer-purpose pair for the probe is picked from probeHiliteLppList. The default is `t`.

GUI Equivalent

None

probeDirection

```
schematic probeDirection cyclic { "input" | "output" | "inputOutput" }
```

Description

Specifies the direction of probe. The default is `input`. For details, refer to [Adding Probes to a Net, Pin, or Instance](#).

GUI Equivalent

Command	<i>Applications to Cross Probe form (accessed by pressing the F3 key on the Add Instance form)</i>
Form Field	<i>Probe Direction</i>

Related topics

[Add Instance](#)

probeHiliteLppList

schematic probeHiliteLppList string *lppList*

Description

Enables you to specify the layer-purpose list to be used for probe. The default value is y0 drawing y1 drawing y2 drawing y3 drawing y4 drawing y5 drawing y6 drawing y7 drawing y8 drawing y9 drawing.

GUI Equivalent

None

probeHiliteLppString

schematic probeHiliteLppString string *lpp*

Description

Specifies the layer-purpose pair to be used by probe if probeCycleBool is set to `nil`. If probeCycleBool is `t`, `probeHiliteLppString` stores the current layer-purpose pair used by the probe. The default is `y0 drawing`.

GUI Equivalent

None

probeTraverseFeedThrough

graphic probeTraverseFeedThrough boolean t

Description

Enables you to probe a feedthrough net in a design. For more information about the feature, see [Probing a Feedthrough Net](#).

GUI Equivalent

None

Related topics

[Probing a Feedthrough Net](#)

probeTraverseHierarchy

graphic probeTraverseHierarchy boolean t

Description

Enables the probing to traverse through design hierarchy levels. For more information about the feature, see [Probing Traverses Design Hierarchy](#).

GUI Equivalent

None

Related topics

[Probing a Feedthrough Net](#)

probeNetDisplayHierarchyLevel

graphic probeNetDisplayHierarchyLevel int hierarchy_level

Description

Specifies the hierarchy level in which the probing is to be followed. The value for the hierarchy level should be greater than 2. The default is 0.

GUI Equivalent

None

Related topics

[Probing a Feedthrough Net](#)

propEditShowEOPSections

schematic propEditShowEOPSections boolean { t | nil }

Description

Specifies whether the Property Editor Assistant displays the attributes, system properties, user properties, CDF parameters in the same order as in the Edit Object Properties Form (t) or not (nil). The default is nil. To show/hide individual fields, you can set editPropShowCDF, editPropShowSystem, and editPropShowCDF.

GUI Equivalent

Command	<i>Window – Assistants – Property Editor</i>
Form Field	N/A

recursionCheck

This function has been included within this appendix for completeness. For details on its description, type, and value refer to the [recursionCheck](#) function in the *Virtuoso Layout Suite XL: Basic Editing User Guide*.

Note: When entering this function in *Schematics L/XL*, you should specify that the application is `schematic` instead of `layout`.

replMasterSymPinName

schematic replMasterSymPinName boolean { t | nil }

Description

Specifies whether the Replace option can rename pins in the instance master when matching an instance pin. It can be used to update a pin name in schematic and symbol views across a hierarchy simultaneously. The default value of the variable is `nil`.

Note: The replace command through this variable can be applied only if the *Pin* option is selected in the Schematic Replace form. In addition, ensure that the master view is editable.

GUI Equivalent

Command	<i>Window – Assistants – Property Editor</i>
Form Field	N/A

Virtuoso Schematic Editor User Guide

Environment Variables

replSaveChanges

```
schematic replSaveChanges boolean { t | nil }
```

Description

Specifies whether changes are saved (`t`) when you complete the *Replace* command and your design is in multiple-cellview mode (for example, hierarchy or library) or not (`nil`). The default is `nil`.

GUI Equivalent

Command	<i>Edit – Replace</i>
Form Field	<i>Save Changes</i> (<u>Schematic Replace</u> form)

runSRC

```
schematic runSRC boolean { t | nil }
```

Description

Specifies whether to run the schematic rules checker (SRC) during schematic checking (t) or not (nil). The default is t.

Normally, during a check, connectivity is updated, the schematic rules checker is run, and the cross-view checker (VIC) is run. A value of nil disables the schematic rules checker.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Options – Check</i>
Form Field	<i>Run Schematic Rules Check (Schematic Check Options form)</i>

runVIC

schematic runVIC boolean { t | nil }

Description

Specifies whether to run the cross-view checker during schematic checking (t) or not (nil). The default is t.

Normally, during a check, connectivity is updated, the schematic rules checker is run, and the cross-view checker is run. A value of nil disables the cross-view checker.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Options – Check</i>
Form Field	<i>Cross View Check Severity (Schematic Check Options form)</i>

saveAction

```
schematic saveAction cyclic { "Save" | "No Save" | "Ask Me" }
```

Description

Specifies what to do during hierarchical checking when `checkHierSave` is `t` and schematics containing errors are encountered. The default is `Ask Me`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Options – Check</i>
Form Field	<i>Check And Save Action (Schematic Check Options form)</i>

saveDefaultPortOrder

```
schematic saveDefaultPortOrder boolean { t | nil }
```

Description

Controls whether the *Save if default pin order* check box is selected by default. If the check box is selected, the pin order is saved even if it is the same as the default pin order. The default port order is output input inputOutput, and the pins in each type are sorted alphanumerically.

GUI Equivalent

Command	<i>Edit – Properties – Pin Order</i>
Form Field	<i>Save if default pin order</i> (<u>E</u> dit Pin Order form)

schCoalesceLimit

schematic schCoalesceLimit int *coalesce_limit*

Description

Specifies the coalesce limit. The default is 2.

GUI Equivalent

None

schCursorShape

schematic schCursorShape boolean { t | nil }

Description

Controls the display of the cursor in all schematic editor windows, toggling on and off a cross-hair display.

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Cross Cursor</i> (<u>Display Options</u> form)

schDisplayDepth

schematic schDisplayDepth int *display_depth*

Description

Specifies the display depth for the graphics editor. A value of 0 specifies that none of the graphics are to be rendered. Instead, a bounding box is drawn with the name of the symbol. Valid Values: 0 through 20. The default is 3.

GUI Equivalent

None

schDrawAxesOn

schematic schDrawAxesOn boolean { t | nil }

Description

Specifies whether the schematic editor displays the X and Y axes when you view or edit a schematic view (t) or not (nil). The default is nil.

For details, see [Changing Display Options](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	Axes (Display Options form)

schDynamicHilightOn

schematic schDynamicHilightOn boolean { t | nil }

Description

Specifies whether the schematic editor dynamically highlights the items that it will select when you click them. When `t`, items with a dashed outline are highlighted as you move the cursor over objects in your design. The default is `t`.

For more information see:

- [Dynamic Net Highlighting](#)
- [Changing Display Options](#)

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Dynamic Highlighting (Display Options form)</i>

schDynamicNetHighlightColor

schematic schDynamicNetHighlightColor string *highlight_color*

Description

Specifies the color for the objects on the same net as the preselected object when the connectivity is up-to-date. The default is `magenta`. The color setting can be amended to be any color that is included in the display resource file (DRF).

For more information see also:

- [schDynamicNetHighlightColorAlt](#)
- [schDynamicNetHighlightColorPre](#)
- [schDynamicNetHighlightOn](#)
- [Dynamic Net Highlighting](#)
- [Changing Display Options](#)
- [Display Resource File Development](#) in the *Virtuoso Technology Data User Guide*.

GUI Equivalent

None

schDynamicNetHighlightColorAlt

schematic schDynamicNetHighlightColorAlt string *highlight_color*

Description

Specifies the color for the objects on the same net as the preselected object when the connectivity is *not* up-to-date. The default is `orange`. The color setting can be amended to be any color that is included in the display resource file (DRF).

For more information see also:

- [schDynamicNetHighlightColor](#) on page 1125
- [schDynamicNetHighlightColorPre](#) on page 1128
- [schDynamicNetHighlightOn](#) on page 1131
- [Dynamic Net Highlighting](#)
- [Changing Display Options](#)
- [Display Resource File Development](#) in the *Virtuoso Technology Data User Guide*.

GUI Equivalent

None

schDynamicNetHighlightColorMemNet

schematic schDynamicNetHighlightColorMemNet string *highlight_color*

Description

Specifies the color for the objects on the member net of the preselected object. It enables highlighting member nets such as, the net across a patch cord or other items in the bus. The default value is `cadetBlue`, but the color setting can be amended to be any color that is included in the display resource file (DRF).

For more information see also:

- [schDynamicNetHighlightColor](#) on page 1125
- [schDynamicNetHighlightColorAlt](#) on page 1126
- [schDynamicNetHighlightOn](#) on page 1131
- [Dynamic Net Highlighting](#)
- [Changing Display Options](#)
- [Display Resource File Development](#) in the *Virtuoso Technology Data User Guide*.

GUI Equivalent

None

schDynamicNetHighlightColorPre

schematic schDynamicNetHighlightColorPre string *highlight_color*

Description

Specifies the color for the preselected highlighted object that is under the cursor. If left empty, there will be no highlighting of the objects, otherwise it will highlight the objects in the given color. By default this will be `magenta`, but the color setting can be amended to be any color that is included in the display resource file (DRF).

For more information see also:

- [schDynamicNetHighlightColor](#)
- [schDynamicNetHighlightColorAlt](#)
- [schDynamicNetHighlightOn](#)
- [Dynamic Net Highlighting](#)
- [Changing Display Options](#)
- [Display Resource File Development](#) in the *Virtuoso Technology Data User Guide*.

GUI Equivalent

None

schDynamicNetHighlightIslands

```
schematic schDynamicNetHighlightIslands boolean { t | nil }
```

Description

Specifies whether the schematic editor draws arcs between islands of connectivity to show the connection between wires of the same net. The default is `t`.

For more information see:

- [Dynamic Net Highlighting](#)
- [Changing Display Options](#)

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Connect By Name Highlighting (Display Options form)</i>

schDynamicNetHighlightMemNetIslands

```
schematic schDynamicNetHighlightMemNetIslands boolean { t | nil }
```

Description

Specifies whether arcs are drawn for the highlighted nets and member nets. This is useful for buses and bundles where the net contains multiple nets. For bus $A<0:1>$ it will also highlight $A<0>$ and $A<1>$. The default is t .

For more information see [Dynamic Net Highlighting](#)

GUI Equivalent

None

schDynamicNetHighlightOn

```
schematic schDynamicNetHighlightOn boolean { t | nil }
```

Description

Specifies whether the schematic editor dynamically highlights all objects of the same net when the mouse pointer is placed over objects. When set to `t`, the schematic editor adds a halo to the other objects connected to the current wire segment under the mouse pointer. The color of the halo depends on the status of the connectivity and can be set using the [schDynamicNetHighlightColor](#) and [schDynamicNetHighlightColorAlt](#) environment variables. The default is `nil`.

GUI Equivalent

Command	<i>View – Net Highlighting</i>
Form Field	N/A

schFilterSize

schematic schFilterSize float *filter_size*

Description

Specifies the filter size. The default is 0.3.

GUI Equivalent

None

schGridMultiple

schematic schGridMultiple int *number_of_grids*

Description

When the grid type is either *dotted* or *line*, this value specifies the number of grids between major grids. The default is 8.

For details, see [Changing Display Options](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Multiple (Display Options form)</i>

schGridSpacing

schematic schGridSpacing float *dist_between_grids*

Description

Specifies the distance between grids. The default is 0.125 (inches).

For details, see [Changing Display Options](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Spacing</i> (Display Options form)

schGridType

schematic schGridType cyclic { "none" | "dotted" | "line" }

Description

Specifies the type of grid to display when you view or edit schematics. The default is dotted.
For details, see [Changing Display Options](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	Type (Display Options form)

schMaxDragFig

schematic schMaxDragFig int *number_of_figures*

Description

Specifies the maximum number of figures that will be rendered during dragging. If dragging objects would result in rendering more than this number of objects, a bounding box is dragged instead. The default is 100.

For details, see [Changing Display Options](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Drag Limit</i> (Display Options form)

Virtuoso Schematic Editor User Guide

Environment Variables

schVerboseFind

```
schematic schVerboseFind boolean { t | nil }
```

Description

If set to `nil`, specifies, when using `schHiFind`, that no “Found 0 objects matching...” messages will be displayed in the CIW, or listed in the `CDS.log` file, if the *find* criteria has not been met. A message window will instead be displayed informing that no objects matched the *find* criteria. This environment variable is set to `t` by default.

GUI Equivalent

None

schPathType

```
schematic schPathType cyclic { "full" | "outline" | "centerline" }
```

Description

Specifies how the schematic editor renders wide wires. `full` shows a solid wide line; `outline` shows the two outer edges (or lines) of the wide line; `centerline` shows only an imaginary line in the center of the wide line. The default is `full`.

GUI Equivalent

None

schShowDirectConnects

```
schematic schShowDirectConnects boolean { t | nil }
```

Description

Specifies whether the schematic editor highlights the direct connections from pin to pin (*t*) or not (*nil*). The default is *nil*.

For details, see [Changing Display Options](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Show Direct Connections</i> (Display Options form)

Virtuoso Schematic Editor User Guide

Environment Variables

schShowLabelOrigin

```
schematic schShowLabelOrigin boolean { t | nil }
```

Description

Specifies whether the origin of text in the schematic editor (wire names, pin names, note text, and so forth) is highlighted by a small crosshair (`t`) or not (`nil`). The default is `nil`.

This is useful for precise locating of names. For details, see [Changing Display Options](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Show Text Justification</i> (Display Options form)

schSnapSpacing

schematic schSnapSpacing float *snap_spacing*

Description

Specifies the minimum distance that the cursor moves. The default distance (0 . 0625 inches) is equal to half the default grid spacing distance (0 . 125).

For details, see [Changing Display Options](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Snap Spacing</i> (Display Options form)

Virtuoso Schematic Editor User Guide

Environment Variables

schWindowBBox

```
schematic schWindowBBox string { "((xCoordLL yCoordLL) (xCoordUR yCoordUR))" }
```

Description

Specifies the window coordinates that the schematic editor uses when creating a window. The lower left and upper right coordinates are specified in screen pixels. The default is (120 185) (1020 875).

GUI Equivalent

None

schematicAreaPartialSelect

```
schematic schematicAreaPartialSelect boolean { t | nil }
```

Description

Specifies whether the system selects partial objects during area selection (t) or not (nil). The default is nil.

For more details on the *Select Filter* command, see [Editing Properties](#).

GUI Equivalent

Command	<i>Edit – Select – Filter</i>
Form Field	<i>Area Partial Selection</i> (Schematic Selection Filter form)

schematicSelectAllDefault

```
schematic schematicSelectAllDefault string [ "wire" ] [ "pin" ] [ "instance" ] [
    "noteShape" ] [ "wireName" ] [ "marker" ] [ "pinName" ] [ "noteText" ] [
    "flight" ] [ "label" ] [ "allSchObj" ] [ "instanceName" ] [ "instancePin" ] [
    "instancePinName" ] [ "instanceLabel" ] [ "allInstObj" ] }
```

Description

Controls default value of the Schematic Select By Filter form. The defaults are "allSchObj " (for schematic objects) and "instancePin" for instance objects.

GUI Equivalent

Command	<i>Edit – Select – Select By Filter</i>
Form Field	N/A

schematicSelectFilter

```
schematic schematicSelectFilter string [ "wire" ] [ "pin" ] [ "instance" ] [
    "noteShape" ] [ "wireName" ] [ "marker" ] [ "pinName" ] [ "noteText" ] [
    "flight" ] [ "label" ] [ "allSchObj" ] [ "instanceName" ] [ "instancePin" ] [
    "instancePinName" ] [ "instanceLabel" ] [ "allInstObj" ] }
```

Description

Specifies the selection filter setting for schematic objects. The default is `allSchObj` `instancePin`.

For more details on the *Select Filter* command, see [Editing Properties](#).

GUI Equivalent

Command	<i>Edit – Select – Filter</i>
Form Field	<i>Schematic Objects</i> (Schematic Selection Filter form)

sensitiveMenu

schematic sensitiveMenu boolean { t | nil }

Description

Specifies whether the middle mouse button uses the object-sensitive menus (*nil*) or the pop-up menu (*t*). The default is *t*.

For details, see [Changing Display Options](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Object-Sensitive Pop-Up Menus</i> (Display Options form)

shapeLineType

```
schematic shapeLineType cyclic { "narrow" | "wide" }
```

Description

Specifies the line style that the schematic editor uses to draw a shape. The default is `narrow`.

GUI Equivalent

Command	<i>Create – Shape – Line</i> (in the symbol editor)
Form Field	<i>Line Style</i> (<u>Add Symbol Shape</u> form)

shapeLineWidth

schematic shapeLineWidth float *line_width*

Description

Specifies a width for the line that the schematic editor uses to draw a note shape. The default is 0.0625 (inches).

GUI Equivalent

Command	<i>Create – Note – Shape</i> <i>Create – Shape – Line</i> (in the symbol editor)
Form Field	<i>Line Width</i> (<u>Create Note Shape</u> form or <u>Add Symbol Shape</u> form)

snapAlignDistribute

```
schematic snapAlignDistribute boolean { t | nil }
```

Description

Controls the behavior of schAlign, which, by default, snaps objects to the grid for alignment. When set to the default value `t`, it allows snapping of the objects during alignment. When it is set to `nil`, snapping is not performed. For details, refer to [Aligning](#) and [Distributing](#).

GUI Equivalent

None

Virtuoso Schematic Editor User Guide

Environment Variables

snapEnabled

```
schematic snapEnabled boolean { t | nil }
```

Description

Specifies whether the *Create Wire* command uses *snap* mode to complete a wire (*t*) or not (*nil*). The default is *t*.

For details, see [Editor Options](#).

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Wire Snapping</i> (<i>Commands</i> tab) (Editor Options form)

Virtuoso Schematic Editor User Guide

Environment Variables

spanLib

```
schematic spanLib string { "do not check" | "check" }
```

Description

Specifies whether the schematic editor spans reference libraries when you use the *Check – Hierarchy* command. You use the `libList` environment variable to specify the list of reference libraries. The default is `do not check`.

GUI Equivalent

Command	<i>Check – Hierarchy</i>
Form Field	<i>Referenced Libraries</i> (<u>Check Hierarchy</u> form)

srcCheckFloatingSignals

```
schematic srcCheckFloatingSignals boolean { t | nil }
```

Description

Specifies a check for floating nets, checking for signals that are neither global, nor associated with any terminal or instance terminal. Such signals, that only have a net associated with them and no pin/terminal, will be marked as “floating”.

The default setting is `nil`. If you want to run this check, set the `srcCheckFloatingSignals` value to `t` in the `.cdsenv` file or in the CIW.

GUI Equivalent

None

srcConnByName

```
schematic srcConnByName cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies a check for naming conventions between wires and pins connected by name. This check verifies identical names that exist in more than one physically disjoint group of wires or pins. A user might have inadvertently connected objects when using a name that was already used elsewhere in the schematic. The default is `ignored`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Connection By Name (Name tab) (Schematic Rules Checks Setup form)</i>

srcFixedGlobalsCheck

```
schematic srcFixedGlobalsCheck cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) Plain Globals check. Controls the check for any globals that are not the default for a net expression within a cellview. The default setting is `ignored`.

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Plain Globals (Logical tab) (Schematic Rules Checks Setup form)</i>

srcFloatingBIDIR

```
schematic srcFloatingBIDIR cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) signal check for floating bidirectional pins. Controls the check for component or schematic bidirectional pins whose signals are not driven by component output pins, schematic input pins, or any other schematic I/O pins. The default is `warning`.

Note: In the Schematic Rules Checker form, a check on *Floating Switch Pins* will also check *jumper* and *tristate*.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Floating Switch Pins</i> (<i>Logical</i> tab) (Schematic Rules Checks Setup form)

srcFloatingInput

```
schematic srcFloatingInput cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) signal check for floating inputs. Controls the check for component input pins and schematic output pins with signals that are not driven by component output pins, schematic input pins, or any schematic I/O pins. The default is `warning`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Floating Input Pins (Logical tab) (Schematic Rules Checks Setup form)</i>

srcFloatingIO

```
schematic srcFloatingIO cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) signal check for floating I/Os. Controls the check for component or schematic input/output pins with signals that are not connected to component output pins, schematic input pins, or any other schematic or symbol I/O pins. The default is `warning`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Floating I/O Pins (Logical tab) (Schematic Rules Checks Setup form)</i>

srcFloatingIterTermStub

schematic srcFloatingIterTermStub boolean t nil

Description

Controls the behavior of the floating net checker while checking the designs. By default, a floating net marker is added for stubs on the iterated instTerms connected to scalar nets. If the environment variable is set to `nil`, the design check continues without adding the floating net marker.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

None

srcFloatingNets

```
schematic srcFloatingNets cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) signal check for floating nets. Controls the check for wires that are not physically connected to schematic or component pins. It does not check for wires connected by name to schematic or instance pins. The default is `warning`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Floating Nets (Logical tab) (Schematic Rules Checks Setup form)</i>

srcFloatingOutput

```
schematic srcFloatingOutput cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) signal check for floating outputs. Controls the check for component output pins and schematic input pins with signals that are not driving any component input pins, schematic output pins, or any schematic I/O pins. The default is `warning`.

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Floating Output Pins (Logical tab) (<u>Schematic Rules Checks Setup</u> form)</i>

srcInhImplicitTerm

```
schematic srcInhImplicitTerm cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) Implicit Inherited Terminals check. Controls the check for inherited signals' connections. The default setting is `ignored`.

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Implicit Inherited Terminals (Inherited Connection tab)</i> (<u>Schematic Rules Checks Setup</u> form)

srcInhInstPinPinConn

```
schematic srcInhInstPinPinConn cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) Inherited Instance Pin/Pin Connections check. Controls the check for inherited instance terminals with connections to nets with terminals. The default for this option is `ignored`.

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Inherited Instance Pin/Pin Connections (Inherited Connection tab) (<u>Schematic Rules Checks Setup</u> form)</i>

srcInhPinNetCollision

```
schematic srcInhPinNetCollision cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) check for inherited pins that are connected to nets whose names do not match the pin.

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Inherited Pin/Net Name Collision (Inherited Conn. tab)</i> (<u>Schematic Rules Checks Setup</u> form)

srcInheritedConn

```
schematic srcInheritedConn cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) inherited connections check. Controls the check for inherited connections. The default is `ignored`.

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Inherited Pin/Net Connections (Inherited Conn. tab)</i> <i>(<u>Schematic Rules Checks Setup</u> form)</i>

srcInstNameExpr

schematic srcInstNameExpr string "*expression*"

Description

Specifies the regular expression to use when the schematic rules checker (SRC) instance name check is enabled. Specifies the regular expression used in the instance name check. The default is " ".

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Instance Name Expression (Name tab) (Schematic Rules Checks Setup form)</i>

srcInstNameSyntax

```
schematic srcInstNameSyntax cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) instance name check. Controls name-syntax checking for instance names. The default is `ignored`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Instance Name Syntax (Name tab) (Schematic Rules Checks Setup form)</i>

Virtuoso Schematic Editor User Guide

Environment Variables

srcInstOverlap

```
schematic srcInstOverlap cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) instance overlap check. Controls the check for instances that overlap other instances. The default is `warning`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Overlapping Instances (Physical tab) (Schematic Rules Checks Setup form)</i>

srcInstOverlapValue

```
schematic srcInstOverlapValue int overlap_percent
```

Description

Specifies the overlap percentage to use when the schematic rules checker (SRC) instance overlap check is enabled. When enabled, the variable specifies the amount (as a percentage of the smaller instance bounding box) that two instances can overlap. The default is 10 (per cent).

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Percent Overlap Allowed (Physical tab) (Schematic Rules Checks Setup form)</i>

srcIteratedInsts

```
schematic srcIteratedInsts cyclic { "ignored" | "warning" | "error" }
```

Description

Controls the iterated instance check for verilog language (only). Default setting is `warning`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

None

srcLogWarnLimit

```
schematic srcLogWarnLimit int warning_limit
```

Description

Controls the output of warning messages in the CIW by limiting the number of messages displayed, for specific schematic rules checks, that have been set to warning.

The default value is 100. To log all warning messages in the CIW, set the variable to 0.

There are four warning counters that are controlled by `schSRCLogWarnLimit`:

1. A first counter tracks all floating and shorted output warnings on instance pins.
2. A second counter tracks all floating and shorted output warnings on pins.
3. A third counter tracks all floating net and signal warnings in the design.
4. A fourth counter tracks all connect by name warnings generated in the design.

If a specific counter reaches the maximum limit, as specified using `srcLogWarnLimit`, then no further warnings will be generated by that particular warning counter.

Note: Markers will only be generated in a design for those corresponding warnings displayed in the CIW. For example, if a limit of 50 warnings has been set, then only 50 corresponding markers will be generated in a design. This limit also curtails the creation of OA markers.

Note: If both warnings and error severities are used within one category, set a higher limit to reduce the chance of errors being suppressed or 0 to ensure all errors and warnings are displayed.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

None

srcMatchedParameters

```
schematic srcMatchedParameters cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) for the Mismatched Parameters check. Controls the check of mismatched parameter values for devices that are members of the Matched Parameters constraint. The default setting is `warning`.

Note: This is available in Schematics XL only.

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Mismatched Parameters (Constraints tab) (<u>Schematic Rules Checks Setup</u> form)</i>

srcMaxLabelOffset

```
schematic srcMaxLabelOffset cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) maximum label offset check. Controls the check for those labels that are beyond a given distance from a wire. The default is `ignored`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Maximum Label Offset (Physical tab) (Schematic Rules Checks Setup form)</i>

srcNameCollision

```
schematic srcNameCollision cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) instance and net name collision check. Controls the check for instance and net name collisions within the schematic. The default is `warning`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Instance/Net Name Collision (Name tab) (Schematic Rules Checks Setup form)</i>

srcNetNameExpr

```
schematic srcNetNameExpr string "expression"
```

Description

Specifies the regular expression to use when the schematic rules checker (SRC) net name check is enabled. The default is " ".

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Net Name Expression (Name tab) (Schematic Rules Checks Setup form)</i>

srcNetNameSyntax

```
schematic srcNetNameSyntax cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) net name check. Controls name-syntax checking for net names. The default is `ignored`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Net Name Syntax (Name tab) (Schematic Rules Checks Setup form)</i>

srcNoOverrideNet

```
schematic srcNoOverrideNet cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) check for missing override nets. Controls the check for override nets when a schematic contains override properties. If the override property is a net expression, the SRC will try to validate the property by ensuring the default name specifies a scalar, global name. If the override property is a literal string, the SRC will first validate the property by ensuring it is a legal, scalar name. In both cases, the SRC will then check if a net by that name exists in the cellview. The default is `warning`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Missing Override Nets (Inherited Conn. tab) (Schematic Rules Checks Setup form)</i>

srcOffsheetConnCheck

```
schematic srcOffsheetConnCheck cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) check for offsheet connectors. Controls the check for offsheet connectors used on only one sheet in a multisheet design. It also checks for nets used locally in more than one sheet that are not connected across the sheets by an offsheet connector. The default is `ignored`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Offsheet Connectors (Logical tab) (Schematic Rules Checks Setup form)</i>

srcPinDirConflictCheck

```
schematic scrPinDirConflictCheck cyclic { "ignored" | "warning" | "error" }
```

Description

Checks for any inconsistencies in the direction of the bit terminals of a multi-bit terminal. For example, if set to *warning/error*, it detects scenarios such as where terminal A<3:0> with output direction and terminal A<0> with input-output directions, both co-exist in the schematic. The default setting is *ignored*.

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Pin Direction Collision (Logical tab) (<u>Schematic Rules Checks Setup</u> form)</i>

Example

```
schSetEnv("srcPinDirConflictCheck" "error") -> t  
schGetEnv("srcPinDirConflictCheck") -> "error"
```

srcPinNetCollision

```
schematic srcPinNetCollision cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) check for pin/net name collisions. Controls the check for pins that are connected to nets whose wire names do not match the pin. The default is `warning`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Pin/Net Name Collision (Name tab) (Schematic Rules Checks Setup form)</i>

srcReportIncompleteChecks

```
schematic srcReportIncompleteChecks boolean { t | nil }
```

Description

Displays a message in the Schematic Check dialog box and CDS log file when internal checks are not run due to schematic extractor errors or incomplete connectivity. The default is `t`.

GUI Equivalent

Command	<i>Options – Check</i>
Form Field	<i>Extract connectivity during schematic checks (<u>Schematic Check Options</u> form)</i>

srcSamePortCheck

```
schematic srcSamePortCheck boolean { t | nil }
```

Description

Specifies that those wires connected to the same instance, through a common instance terminal, be considered as physically connected, and not logically (that is, not to be considered as connected by name), and that they would also be considered as floating. The default is `t`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

None

srcShortedOutputGlobal

```
schematic srcShortedGlobal boolean { t | nil }
```

Description

Works in tangent with [srcShortedOutputs](#) where, if set to `t`, a shorted output warning will be flagged if two ports are connected with a wire that has a net expression/global signal. The default is `nil`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Shorted Output Pins (Logical tab) (Schematic Rules Checks Setup form)</i>

srcShortedOutputs

```
schematic srcShortedOutputs cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) shorted outputs check. Controls the check for component output or schematic input pins that are connected to any other component output or schematic input pins. The default is `warning`.

See also [srcShortedOutputGlobal](#).

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Shorted Output Pins (Logical tab) (Schematic Rules Checks Setup form)</i>

srcSigTypeIntegrity

```
schematic srcSigTypeIntegrity boolean { t | nil }
```

Description

Specifies whether or not signal type integrity checks are performed. The default is `nil`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Signal Type Checks (Signal Type tab) (Schematic Rules Checks Setup form)</i>

Virtuoso Schematic Editor User Guide

Environment Variables

srcSolderOnCrossover

```
schematic srcSolderOnCrossover cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) check for solder dots on crossovers (four or more wires at an intersection). The default is `warning`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Solder on CrossOver (Physical tab) (Schematic Rules Checks Setup form)</i>

srcSparseIndexBus

```
schematic srcSparseIndexBus cyclic { "ignored" | "warning" | "error" }
```

Description

By default, this checks that all bits of a vectored net in the range are used. The default is `ignored`. To exclude nets that are not connected to terminals from the check, set [srcSparseIndexBusOnlyTerms](#).

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Sparse Index Bus (Name tab) (<u>Schematic Rules Checks Setup</u> form)</i>

srcSparseIndexBusOnlyTerms

```
schematic srcSparseIndexBusOnlyTerms boolean { t | nil }
```

Description

Filters out sparse nets which are not attached to a term from a Sparse Bus Index Check configured using srcSparseIndexBus. The default is `nil`.

For more details on setting schematic check options and rules, see Checking Designs.

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Sparse Index Bus Only Terms (Name tab) (<u>Schematic Rules Checks Setup</u> form)</i>

srcSysVerilogSyntax

```
schematic srcSysVerilogSyntax cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) System Verilog syntax name check. Controls the check for System Verilog HDL syntax across all instance and signal names. This check verifies that names are valid System Verilog HDL identifiers and are not Verilog HDL reserved words. The default is `ignored`.

When System Verilog HDL syntax is set to `warning` or `error` the following checks are made:

- *Iterated Instance Check*

The use of iterated instances is currently flagged as an error/warning in the System Verilog Language Check because it is not supported in certain versions of System Verilog.

- *Reserved Word Check*

The base names of all instances/signals are checked to see if they conflict with reserved words in Verilog. If they do, an error/warning is flagged.

- *Bad Identifier Check*

The base names of all instances/signals are checked to determine if they are legal identifiers that could be used in the scope of Verilog. If they are not, an error/warning is flagged.

- *Sparse Bus Index Check*

The sparse bus index check checks that all bits of a vectored net in the range are used. If they are not, this will be flagged as an error.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>System Verilog HDL Syntax (Name tab) (Schematic Rules Checks Setup form)</i>

srcTermNameExpr

schematic srcTermNameExpr string "*expression*"

Description

Specifies the regular expression to use when the schematic rules checker (SRC) terminal name check is enabled. The default is " ".

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Pin Name Expression</i> (Name tab) (Schematic Rules Checks Setup form)

srcTermNameSyntax

```
schematic srcTermNameSyntax cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) terminal name check. Controls name-syntax checking for net names. The default is `ignored`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Pin Name Syntax (Name tab) (Schematic Rules Checks Setup form)</i>

srcUnconnectedWires

```
schematic srcUnconnectedWires cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) unconnected wires check. Controls the check for wire segments whose endpoints do not physically connect to schematic or component pins, labels, or other wire segments. The default is `ignored`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Unconnected Wires (Physical tab) (Schematic Rules Checks Setup form)</i>

srcUnnamedConnection

```
schematic srcUnnamedConnection cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) unnamed connection check, which checks for nets with no user-generated names. The default is `ignored`.

The checker flags nets without user-generated names when one of the following conditions is met:

- The net has fewer than two attached instance terminals.
- The net has a system-generated name.
- The net has wire segments.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Unnamed Connection (Name tab) (Schematic Rules Checks Setup form)</i>

srcVHDLSyntax

```
schematic srcVHDLSyntax cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) VHDL syntax name check. Controls the check for VHDL syntax across all instance and signal names. This check verifies that names are valid VHDL identifiers and are not VHDL reserved words. The default is `ignored`.

When VHDL syntax is set to `warning` or `error` the following checks are made:

- *Reserved Word Check*

The base names of all scalar instances and signals are checked to see if they conflict with reserved words in VHDL. If they do conflict an error/warning is flagged.

- *Bad Identifier Check*

The base names of all instances and signals are checked to see if they are legal identifiers that could be used in the scope of VHDL. If they are not this will be flagged as an error/warning.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>VHDL Syntax (Name tab)</i> (Schematic Rules Checks Setup form)

srcVerilogSyntax

```
schematic srcVerilogSyntax cyclic { "ignored" | "warning" | "error" }
```

Description

Specifies the severity of the schematic rules checker (SRC) Verilog syntax name check. Controls the check for Verilog HDL syntax across all instance and signal names. This check verifies that names are valid Verilog HDL identifiers and are not Verilog HDL reserved words. The default is `ignored`.

When Verilog HDL syntax is set to `warning` or `error` the following checks are made:

- *Iterated Instance Check*

The use of iterated instances is currently flagged as an error/warning in the Verilog Language Check as it is not supported in certain versions of Verilog.

- *Reserved Word Check*

The base names of all instances/signals are checked to see if they conflict with reserved words in Verilog. If they do an error/warning is flagged.

- *Bad Identifier Check*

The base names of all instances/signals are checked to determine if they are legal identifiers that could be used in the scope of Verilog. If they are not an error/warning is flagged.

- *Sparse Bus Index Check*

The sparse bus index check checks that all bits of a vectored net in the range are used. If they are not this will be flagged as an error.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Verilog HDL Syntax (Name tab) (Schematic Rules Checks Setup form)</i>

ssgSortPins

```
schematic ssgSortPins cyclic { "alphanumeric" | "geometric" }
```

Description

Specifies how the text-to-symbol generator sorts the symbol pins for the new symbol. The default is `alphanumeric`.

For details, see [Editor Options](#).

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Symbol Generator Pin Sorting (Commands tab) (Editor Options form)</i>

statusBanner

```
schematic statusBanner boolean { t | nil }
```

Description

Specifies whether the status banner appears on all windows that contain schematic or symbol data (`t`) or not (`nil`). The default is `t`.

For details, see [Changing Display Options](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Display Status Banner</i> (Display Options form)

statusBannerXY

schematic statusBannerXY boolean { t | nil }

Description

Specifies whether the schematic editor displays the X and Y coordinates in the status banner when the status banner is displayed (t) or not (nil). The default is nil.

For details, see [Changing Display Options](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Coordinates on Status Banner</i> (Display Options form)

stickyWireLabelRange

schematic stickyWireLabelRange float *range*

Description

Specifies the range between the origin of a wire name and a wire segment vertex. When you reroute the wire segment using the *Stretch* command, the wire name remains in a relative position to the vertex if the wire name is within the specified range. The default is 0.3125 (inches).

GUI Equivalent

Command	<i>Edit – Stretch</i>
Form Field	N/A

stubLength

schematic stubLength int *length*

Description

Sets the length - in visible grid points of the minor grid - of the wire stubs created automatically. The default is 2.

GUI Equivalent

None

symCursorShape

```
schematic symCursorShape boolean { t | nil }
```

Description

Controls the display of the cursor in all symbol editor windows, toggling on and off a cross-hair display.

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Cross Cursor</i> (<u>Display Options</u> form)

symDisplayDepth

schematic symDisplayDepth int *display_depth*

Description

Specifies the display depth for the graphics editor. A value of 0 specifies that none of the graphics are to be rendered. Instead, a bounding box is drawn with the name of the symbol. The default is 3.

GUI Equivalent

None

symDrawAxesOn

```
schematic symDrawAxesOn boolean { t | nil }
```

Description

Specifies whether the symbol editor displays the X and Y axes when you view or edit a symbol view (*t*) or not (*nil*). The default is *nil*.

For details, see [Changing Display Options](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	Axes (Display Options form)

symDynamicHilightOn

schematic symDynamicHilightOn boolean { t | nil }

Description

Specifies whether the symbol editor dynamically highlights the items that it will select when you click the left mouse button. When `t`, the items with a dashed outline are highlighted as you move the cursor over objects in your design. The default is `t`.

For details, see [Changing Display Options](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Dynamic Highlighting (Display Options form)</i>

symGridMultiple

schematic symGridMultiple int *number_of_grids*

Description

When the grid type is either *dotted* or *line*, this value specifies the number of grids between major grids. The default is 8 (inches).

For details, see [Changing Display Options](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Multiple</i> (Display Options form)

symGridSpacing

schematic symGridSpacing float *dist_between_grids*

Description

Specifies the distance between minor grids. The default is 0.125 (inches).

For details, see [Changing Display Options](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Spacing</i> (Display Options form)

symGridType

```
schematic symGridType cyclic { "none" | "dotted" | "line" }
```

Description

Specifies the type of grid to display when you view or edit symbols. The default is `dotted`.

For details, see [Changing Display Options](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	Type (Display Options form)

symMaxDragFig

schematic symMaxDragFig int *max_number_of_figures*

Description

Specifies the maximum number of figures that will be rendered during dragging. If dragging objects would result in rendering more than this number of objects, a bounding box is dragged instead. The default is 100.

For details, see [Changing Display Options](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Drag Limit</i> (Display Options form)

symPathType

```
schematic symPathType cyclic { "full" | "outline" | "centerline" }
```

Description

Specifies how the schematic editor renders wide wires. "full" shows a solid wide line; "outline" shows the two outer edges (or lines) of the wide line; "centerline" shows only an imaginary line in the center of the wide line. The default is full.

GUI Equivalent

None

symShowDirectConnects

```
schematic symShowDirectConnects boolean { t | nil }
```

Description

When `t`, the symbol editor highlights the direct connections from pin to pin. The default is `nil`.

For more details, see [Changing Display Options](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Show Direct Connections</i> (Display Options form)

Virtuoso Schematic Editor User Guide

Environment Variables

symShowLabelOrigin

```
schematic symShowLabelOrigin boolean { t | nil }
```

Description

Specifies whether the origin of text in the schematic editor (wire names, pin names, note text, and so forth) is highlighted by a small crosshair (`t`) or not (`nil`). The default is `nil`.

This is useful for precise locating of names. For details, see [Changing Display Options](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Show Text Justification</i> (Display Options form)

symSnapSpacing

schematic symSnapSpacing float *snap_distance*

Description

Specifies the minimum distance that the cursor moves. The default (0.0625 inches) is equal to half the default grid spacing distance (0.125).

For details, see [Changing Display Options](#).

GUI Equivalent

Command	<i>Options – Display</i>
Form Field	<i>Snap Spacing</i> (Display Options form)

Virtuoso Schematic Editor User Guide

Environment Variables

symWindowBBox

```
schematic symWindowBBox string { "((xCoordLL yCoordLL) (xCoordUR yCoordUR))" }
```

Description

Specifies the window coordinates that the schematic editor uses when creating a window. The lower left and upper right coordinates are specified in screen pixels. The default is (120 185) (1020 875).

GUI Equivalent

None

symbolAreaPartialSelect

```
schematic symbolAreaPartialSelect boolean { t | nil }
```

Description

Specifies whether partially selected objects are included in an area selection action (`t`) or not (`nil`). The default is `nil`.

GUI Equivalent

Command	<i>Edit – Select – Filter</i>
Form Field	<i>Area Partial Selection (<u>Schematic Selection Filter</u> form)</i>

symbolLabelChoice

```
schematic symbolLabelChoice cyclic { "instance label" | "device annotate" |  
    "logical label" | "physical label" | "pin name" | "pin annotate" | "analog  
    instance label" | "analog pin annotate" | "analog device annotate" }
```

Description

Changes the type of label for a symbol (which can change the values of the *Label* and *Label Type* fields).

GUI Equivalent

None

symbolLabelChoiceText

```
schematic symbolLabelChoiceText string { "component label" | "device annotate" |  
    "logical label" | "physical label" | "pin label" | "pin" | "annotate" }
```

Description

Specifies the purpose of a symbol label. The default is " ".

GUI Equivalent

Command	<i>Create – Label</i> (in the symbol editor)
Form Field	<i>Label Choice</i> (<u>Add Symbol Label</u> form)

symbolLabelFontHeight

schematic symbolLabelFontHeight float *font_height*

Description

Specifies a value (in user units) for the font height for a label that you add with the *Create – Label* command. The default is 0.0625 (inches).

GUI Equivalent

Command	<i>Create – Label</i> (in the symbol editor)
Form Field	<i>Font Height</i> (<u>Add Symbol Label</u> form)

symbolLabelFontStyle

```
schematic symbolLabelFontStyle cyclic { "euroStyle" | "gothic" | "math" | "roman"  
    | "script" | "stick" | "fixed" | "swedish" | "milSpec" }
```

Description

Specifies a font for a label that you add with the *Create – Label* command. The default is *stick*.

GUI Equivalent

Command	<i>Create – Label</i> (in the symbol editor)
Form Field	<i>Font Style</i> (<u>Add Symbol Label</u> form)

symbolLabelJustify

```
schematic symbolLabelJustify cyclic { "upperLeft" | "centerLeft" | "lowerLeft" |  
    "upperCenter" | "centerCenter" | "lowerCenter" | "upperRight" | "centerRight"  
    | "lowerRight" }
```

Description

Specifies the position of the cursor with respect to the label that you add with the *Create – Label* command. The default is `lowerCenter`.

GUI Equivalent

Command	<i>Create – Label</i> (in the symbol editor)
Form Field	<i>Justification</i> (<u>Add Symbol Label</u> form)

symbolLabelType

```
schematic symbolLabelType cyclic { "normalLabel" | "NLPLabel" | "ILLabel" }
```

Description

Specifies how a symbol label is displayed. The default is `NLPLabel`.

GUI Equivalent

Command	<i>Create – Label</i>
Form Field	<i>Label Type</i> (<u>Add Symbol Label</u> form)

symbolPinFlatten

```
schematic symbolPinFlatten boolean { t | nil }
```

Description

Specifies whether the symbol editor flattens pins that you add to your design. When `nil`, the pins that you add to your design are flattened. This corresponds to the *on* option in the *Commands* tab of the Editor Options form. The default is `t`.

For more details, see [Editor Options](#).

GUI Equivalent

Command	<i>Option – Editor</i>
Form Field	<i>Add Symbol Pins as Instances (Commands tab) (Editor Options form)</i>

symbolPinIncrCount

schematic symbolPinIncrCount int *distance*

Description

Specifies the distance between a pin and the pin label. The default is 1.

GUI Equivalent

Command	<i>Create – Pin</i>
Form Field	<i>Label Offset</i> (<u>Create Pin – Schematic</u> form)

symbolPinLocate

```
schematic symbolPinLocate cyclic { "none" | "left" | "right" }
```

Description

Specifies the location of the pin label with respect to the pin. The default is `left`.

GUI Equivalent

Command	<i>Create – Pin</i>
Form Field	<i>Label Location (Create Pin – Symbol form)</i>

Virtuoso Schematic Editor User Guide

Environment Variables

symbolPinUsage

```
schematic symbolPinUsage string { "square" | "round" | "actHi" | "commActLo" |  
    "ieeeActLo" | "block" }
```

Description

Specifies whether the symbol editor sets up the Create Pin form to use `square` as the initial value for pin type when it is defined in the `schSymbolPinMasters` list. The default is `square`.

GUI Equivalent

Command	<i>Create – Pin</i>
Form Field	<i>Type (Create Pin – Symbol form)</i>

Virtuoso Schematic Editor User Guide

Environment Variables

symbolSelectAllDefault

```
schematic symbolSelectAllDefault string { "pin" | "deviceShape" | "selectionBox" |  
    "noteShape" | "pinName" | "label" | "marker" | "noteText" | "allSymObj" }
```

Description

Controls default value of the Symbol Select By Filter form. The default is allSymObj.

GUI Equivalent

Command	<i>Edit – Select – Select By Filter</i>
Form Field	N/A

Virtuoso Schematic Editor User Guide

Environment Variables

symbolSelectFilter

```
schematic symbolSelectFilter string { "pin" | "deviceShape" | "instanceBox" |  
    "noteShape" | "pinName" | "label" | "marker" | "noteText" | "allSymObj" }
```

Description

Specifies the selection filter setting for symbol objects. The default is `allSymObj`.

For more details on the *Select Filter* command, see [Editing Properties](#).

GUI Equivalent

Command	<i>Edit – Select – Filter</i>
Form Field	<i>Symbol Objects</i> (Schematic Selection Filter form)

Virtuoso Schematic Editor User Guide

Environment Variables

symbolShape

```
schematic symbolShape cyclic { "line" | "rectangle" | "polygon" | "arc" | "ellipse"  
    | "circle" }
```

Description

Specifies the symbol shape you want to add. The default is `line`.

GUI Equivalent

Command	<i>Create – Shape</i>
Form Field	<i>Shape (Add Symbol Shape form)</i>

symbolShapeFillStyle

```
schematic symbolShapeFillStyle cyclic { "outline" | "filled" }
```

Description

Specifies whether a shape is filled. *Filled* maps to `solid`. The default is `outline`.

GUI Equivalent

Command	<i>Create – Shape</i>
Form Field	<i>Fill Style</i> (<u>Add Symbol Shape</u> form)

transToSheetType

schematic transToSheetType string { "single" | "multi-sheet" }

Description

Specifies the type of sheet created when you complete the *Create – Cellview – From Cellview* command and the destination view type is *schematic*. The default is `single`.

GUI Equivalent

Command	<i>Create – Cellview – From Cellview</i>
	<i>Create – Cellview – Pin List</i>
	<i>Create – Cellview – Instance</i>
Form Field	<i>Schematic Type</i> (<u>Cellview From Cellview</u> form, <u>Cellview From Pin List</u> form, or <u>Cellview From Instance</u> form)

tsgAnnotationsForModifiedPins

```
schematic tsgAnnotationsForModifiedPins boolean boolean { t | nil }
```

Description

Sets the status for auto-annotation to be used for modified schematic and symbol views. The default is `t`.

GUI Equivalent

None

tsgExcludeInhConnPins

```
schematic tsgExcludeInhConnPins cyclic { "None" | "All" | "Named" }
```

Description

Used to specify the inherited connection pins to be excluded in the generation of a symbol view. When either *All* or *Only these* is selected in the *Exclude Inherited Connection Pins* section of the [Symbol Generation Options](#) `tsgExcludeInhConnPins` will automatically remove the names of any inherited connection pins from the entry field.

Note: The *Only these* field is automatically populated with the names of any inherited connection pins.

See also: [tsgExcludeInhConnPinsNames](#).

You should be aware of the following related issues with `tsgExcludeInhConnPins`:

- If you switch between the form field options the correct order of the inherited connection pin names may not be restored correctly.
- The form field options are only used when creating a symbol view from a schematic view. If you attempt to create a symbol view from an instance, the option to ignore inherited connections is not available.

GUI Equivalent

Command	<i>Create – Cellview – From Cellview and Edit Options</i> selected
Form Field	<i>Exclude Inherited Connection Pins: (Cellview From Cellview form)</i> ■ <i>None</i> ■ <i>All</i> ■ <i>Only these</i>

Virtuoso Schematic Editor User Guide

Environment Variables

tsgExcludeInhConnPinsNames

schematic tsgExcludeInhConnPinsNames string *pin_names*

Description

Lists the inherited connection pins to be excluded in the generation of a symbol view. This variable is only used when tsgExcludeInhConnPins is set to `Named`.

GUI Equivalent

Command	<i>Create – Cellview – From Cellview and Edit Options</i> selected
Form Field	<i>Only these (<u>Cellview From Cellview</u> form)</i>

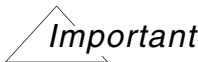
tsgTemplateType

```
schematic tsgTemplateType string { "digital" | "artist" | "PCB" | "metric" | "other" }
```

Description

Specifies which TSG template file the system loads the first time you use one of the *Create Cellview* commands to automatically generate a symbol. The default is `digital`.

The system uses the value of this variable to look up the full file path in the *tsgTemplateMasters* list, which is defined in the `schConfig.il` file.



Once you have loaded a template file, you cannot switch to another template file by just changing the value of the environment variable. To switch to another template file, you must use either the *Load* button on the Symbol Generation Options form or the `schSetAndLoadTsgTemplateType` procedural interface (PI) function.

GUI Equivalent

Command	<i>Create – Cellview – From Cellview</i> and <i>Edit Options</i> selected
	<i>Create – Cellview – From Pin List</i> and <i>Edit Options</i> selected
Form Field	<i>Load</i> (<u>Cellview From Cellview</u> form or <u>Cellview From Pin List</u> form)

Virtuoso Schematic Editor User Guide

Environment Variables

updateConn

```
schematic updateConn boolean { t | nil }
```

Description

Specifies whether the schematic editor updates connectivity during schematic checking (t) or not (nil). The default is t.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Options – Check</i>
Form Field	<i>Extract connectivity during schematic checks (Schematic Check Options form)</i>

updateInstResetCDF

```
schematic updateInstResetCDF boolean { t | nil }
```

Description

Refreshes the instance's CDF parameter values (t). The default is t.

GUI Equivalent

Command	<i>Edit – Properties – Objects</i>
Form Field	N/A

vicCheckPinOrder

```
schematic vicCheckPinOrder boolean { t | nil }
```

Description

Specifies the following when set to the default value `t`:

- The cross-view checker generates a port mismatch warning when the order in which the ports are defined in one view differs from the order defined in another view.
- The *Port Mismatch* form reports the pin order differences between views in a cell. For details, see [Resolving Pin Mismatch](#).

For details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Options – Check</i> <i>Check – Cross-View</i>
Form Field	<i>Check Pin Order</i> (Schematic Check Options form or Cross-View Checker form)

vicExactPinMatch

schematic vicExactPinMatch boolean { t | nil }

Description

When `t`, specifies that the cross-view checker generate a port mismatch warning when one view has a single pin (for example, `pin[0:7]`) and another view has multiple pins (for example, `pin[0]`, `pin[1]`, `pin[2]`, ... `pin[7]`). The default is `nil`.

For details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Options – Check</i> <i>Check – Cross-View</i>
Form Field	<i>Match Pin Names Exactly</i> (Schematic Check Options form or Cross-View Checker form)

vicIgnoreInhTermAll

schematic vicIgnoreInhTermAll boolean { t | nil }

Description

Specifies that the cross-view checker ignore all inherited terminal mismatches (t), which is the default.

GUI Equivalent

Command	<i>Options – Check</i> <i>Check – Cross-View</i>
Form Field	<i>Match Inherited Terminals – Ignore Terminals – All Terminals (Schematic Check Options form or Cross-View Checker form)</i>

vicIgnoreInhTermList

schematic vicIgnoreInhTermList string "list_of_terminals"

Description

Specifies that the cross-view checker ignore inherited terminal mismatches for the terminals listed in *list_of_terminals*.

GUI Equivalent

Command	<i>Options – Check</i> <i>Check – Cross-View</i>
Form Field	<i>Match Inherited Terminals – Ignore Terminals – Terminal Names (Schematic Check Options form or Cross-View Checker form)</i>

vicIgnoreInhTermViewList

schematic vicIgnoreInhTermViewList string "*list_of_views*"

Description

Specifies that the cross-view checker ignore inherited terminal mismatches for the views listed in *list_of_views*.

GUI Equivalent

Command	<i>Options – Check</i> <i>Check – Cross-View</i>
Form Field	<i>Match Inherited Terminals – View Names</i> (<u>Schematic Check Options</u> form or <u>Cross-View Checker</u> form)

vicIgnoreNonExistingTerms

schematic vicIgnoreNonExistingTerms boolean { t | nil }

Description

When enabled if there is an inherited terminal, a terminal with a net expression, in one view of the cell and the same name terminal does not exist in the other view of the cell, then a cross view check will not report a warning. The default is `t`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Options</i>
Form Field	<i>Ignore Terminals with Net Expressions and no Terminals in the Other View</i> (Schematic Check Options form)

vicIgnoreViewsWithNoSetPinOrder

```
schematic vicIgnoreViewsWithNoSetPinOrder boolean { t | nil }
```

Description

Specifies the criterion that the `portOrder` property must be set for a schematic or symbol view to be considered for port order consistency check. This environment variable can be used for the schematic/symbol [Cross-View Checker](#) and the *Port Mismatch* form (see [Resolving Pin Mismatch](#)). The default value is `t`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Options – Check</i>
	<i>Check – Cross-View</i>
Form Field	<i>Only check views with explicit pin order (Schematic Check Options form or Cross-View Checker form)</i>

vicInheritedPinMatch

```
schematic vicInheritedPinMatch boolean { t | nil }
```

Description

Specifies whether the cross-view checker generates a warning if an inherited terminal in one view has a different net expression than the matching terminal in another view (`t`) or not (`nil`). The default is `t`.

GUI Equivalent

Command	<i>Options – Check</i> <i>Check – Cross-View</i>
Form Field	<i>Ignore Terminals with Net Expression and no Terminal in the Other View</i> (<u>Schematic Check Options</u> form or <u>Cross-View Checker</u> form)

Virtuoso Schematic Editor User Guide

Environment Variables

vicInheritedPins

```
schematic vicInheritedPins boolean { t | nil }
```

Description

Specifies whether the cross-view checker generates a warning if a terminal is inherited in one view but is not inherited in another view (`t`) or not (`nil`). The default is `t`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Options – Check</i> <i>Check – Cross-View</i>
Form Field	<i>Match Inherited Terminals</i> (Schematic Check Options form or Cross-View Checker form)

vicLogWarnLimit

```
schematic vicLogWarnLimit int warn_limit
```

Description

Specifies the maximum number of warning and error messages generated by the cross-view checker in a single run. When set to 0, the limit is removed. The default is 1000.

GUI Equivalent

None

vicSeverity

schematic vicSeverity cyclic { "ignored" | "warning" | "error" }

Description

Specifies the severity of the cross-view check. When set to "ignored", the checker returns. The default is warning.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Options – Check</i> <i>Check – Cross-View</i>
Form Field	<i>Cross View Check Severity</i> (Schematic Check Options form or Cross-View Checker form)

vicTermSigType

```
schematic vicTermSigType boolean { t | nil }
```

Description

Checks for signal type mismatches between the corresponding pins of the schematic and symbol views of a cell. The default is `t`.

GUI Equivalent

Command	<i>Options – Check</i> <i>Check – Cross-View</i>
Form Field	<i>Match Terminal Signal Type</i> (<u>Schematic Check Options</u> form or <u>Cross-View Checker</u> form)

vicTermSupplySens

schematic vicTermSupplySens boolean { t | nil }

Description

Checks that terminal sensitivity and consistency is validated between the schematic and corresponding symbol cellviews. The default is t.

GUI Equivalent

Command	<i>Options – Check</i> <i>Check – Cross-View</i>
Form Field	<i>Match Terminal Supply Sensitivity (Schematic Check Options form or Cross-View Checker form)</i>

Virtuoso Schematic Editor User Guide

Environment Variables

vicViewList

```
schematic vicViewList string "list_of_view_names"
```

Description

Specifies the list of view names that the cross-view checker checks. The default is `symbol`
`schematic` `functional` `behavioral` `system` `opt_sch` `opt_net`.

For more details on setting schematic check options and rules, see [Checking Designs](#).

GUI Equivalent

Command	<i>Check – Options</i>
Form Field	<i>Cross View Checks – Views to Check</i> (Schematic Check Options form or Cross-View Checker form)

viewNameList

```
schematic viewNameList string "default_view_name_list"
```

Description

Specifies the list order in which view names are placed on schematic and symbol windows. The view name list controls how hierarchy traversal should expand the design except in the case where a schematic is in the context of a hierarchy configuration. The default is `schematic symbol`.

View Name List is used in conjunction with the *Edit – Hierarchy – Descend Edit* and *Edit – Hierarchy – Descend Read* commands when descendTarget is set to *use viewNameList*.

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>View Name List</i> (<u>Editor Options</u> form)

Related Topics

[Traversing a Design Hierarchy](#)

[Setting the View Precedence Order](#)

Virtuoso Schematic Editor User Guide

Environment Variables

wireBackTrace

```
schematic wireBackTrace boolean { t | nil }
```

Description

Specifies how unselected wires that are connected to selected instances, wires, and pins behave. `t` selects back to the pin or solder dot, `nil` (the default) selects only the last segment.

GUI Equivalent

Command	<i>Edit – Stretch</i>
Form Field	<i>Reroute Wire Segments Connected To (<u>Stretch – Schematic View</u> form)</i>

zoomOutButton

```
schematic zoomOutButton boolean { t | nil }
```

Description

When `t`, automatically sizes the window display to show all objects that match a specified search criterion. The default is `nil`.

GUI Equivalent

Command	<i>Edit – Select – Select By Property</i>
Form Field	<i>Zoom Out To Fit All Figures (<u>Schematic Select By Property</u> form)</i>

Photonics Environment Variables (Virtuoso Photonics Option)

Note: Virtuoso Photonics Solution - You need the *Virtuoso_Photonics_Option* license to use the functionality discussed in this section. For information about obtaining the required license, contact your local Cadence representative.

The photonics environment variables define the characteristics of the photonics solution in schematic editor. You can customize the behavior of the schematic editor by changing the value of a particular environment variable.

opticalNetColor

```
schematic opticalNetColor string "color"
```

Description

(Photonics Only) Specifies the color of optical nets. The default is `cream`. The color setting can be changed to be any color defined in the display resource file (DRF).

The line style for the optical nets can be specified using [opticalNetLineStyle](#).

GUI Equivalent¹

Command	<i>Options – Check</i>
Form Field	<i>Set Optical Wire Style</i> (Schematic Check Options form)

Related Topics

[Display Resource File Development](#) in the *Virtuoso Technology Data User Guide*.

opticalNetColoring

```
schematic opticalNetColoring boolean { t | nil }
```

Description

(Photonics Only) Specifies whether optical nets must have a specified color. The default is t.

GUI Equivalent

None

opticalNetLineStyle

schematic opticalNetLineStyle string "*lineStyle*"

Description

(Photonics Only) Specifies the line style for optical nets. The line style setting can be changed to specify any line style defined in the display resource file (DRF).

The color for the optical nets can also be specified using [opticalNetColor](#).

GUI Equivalent

Command	<i>Options – Check</i>
Form Field	<i>Set Optical Wire Style</i> (Schematic Check Options form)

Related Topics

[Display Resource File Development](#) in the *Virtuoso Technology Data User Guide*.

opticalPinColor

```
schematic opticalPinColor string "color"
```

Description

(Photonics Only) Specifies the color of optical pins. The default is `yellow`. The color can be changed to any color defined in the display resource file (DRF).

The line and fill style for the optical pins is specified using [opticalPinLineStyle](#) and [opticalPinFillStyle](#).

GUI Equivalent

Command	<i>Options – Check</i>
Form Field	<i>Set Optical Pin Color</i> (Schematic Check Options form)

Examples

```
envGetVal("schematic" "opticalPinColor")
envSetVal("schematic" "opticalPinColor" 'string "yellow")
envSetVal("schematic" "opticalPinColor" 'string "pink")
```

Related Topics

[Display Resource File Development](#) in the *Virtuoso Technology Data User Guide*.

opticalPinColoring

```
schematic opticalPinColoring boolean { t | nil }
```

Description

(Photonics Only) Specifies whether optical pins need to have a specified color. The default is t.

GUI Equivalent

None

Examples

```
envGetVal("schematic" "opticalPinColoring")
envSetVal("schematic" "opticalPinColoring" 'boolean t)
envSetVal("schematic" "opticalPinColoring" 'boolean nil)
```

opticalPinFillColor

```
schematic opticalPinFillColor string "color"
```

Description

(Photonics Only) Specifies the default color of optical pins. The default is `yellow`. The color can be changed to any color defined in the display resource file (DRF).

GUI Equivalent

None

Examples

```
envGetVal("schematic" "opticalPinFillColor") => "yellow"  
envSetVal("schematic" "opticalPinFillColor" 'string "pink") => t
```

Related Topics

[Display Resource File Development](#) in the *Virtuoso Technology Data User Guide*.

opticalPinFillStyle

```
schematic opticalPinFillStyle string "solid"
```

Description

(Photonics Only) Specifies the fill style for optical pins.

The color for the optical pins is specified using opticalPinColor.

GUI Equivalent

None

Examples

```
envGetVal("schematic" "opticalPinFillStyle")  
envSetVal("schematic" "opticalPinFillStyle" 'string "solid")
```

opticalPinLineStyle

```
schematic opticalPinLineStyle string "solid"
```

Description

(Photonics Only) Specifies the line style for optical pins.

The color for the optical pins is specified using [opticalPinColor](#).

GUI Equivalent

None

Examples

```
envGetVal("schematic" "opticalPinLineStyle")  
envSetVal("schematic" "opticalPinLineStyle" 'string "solid")
```


opticalPinStipple

```
schematic opticalPinStipple string "stipplePattern"
```

Description

(Photonics Only) Specifies the default stipple pattern used for filling shapes. The default is `solid`. The stipple pattern can be changed to any stipple defined in the display resource file (DRF).

GUI Equivalent

None

Examples

```
envGetVal("schematic" "opticalPinStipple") => "solid"  
envSetVal("schematic" "opticalPinStipple" 'string "dots") => t
```

Related Topics

[Display Resource File Development](#) in the *Virtuoso Technology Data User Guide*.

opticalSigTypePropagation

```
schematic opticalSigTypePropagation boolean { t | nil }
```

Description

(Photonics Only) Propagates the optical signal types `optical`, `singleModeOptical`, and `multiModeOptical`. The default is `t`.

GUI Equivalent

Command	<i>Options – Check</i>
Form Field	<i>Propagate the Optical Signal Type Values (<u>Schematic Check Options</u> form)</i>

srcOpticalElectricalConnection

```
schematic srcOpticalElectricalConnection cyclic { "ignored" "warning" "error" }
```

Description

(Photonics Only) Specifies the severity of the schematic rules checker (SRC) check for any optical to electrical connections. The default is `error`.

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Electrical Connections (Optical tab) (<u>Schematic Rules Checks Setup</u> form)</i>

srcOpticalMultiToSingle

```
schematic srcOpticalMultiToSingle cyclic { "ignored" "warning" "error" }
```

Description

(Photonics Only) Specifies the severity of the schematic rules checker (SRC) check for any multi mode optical to single mode optical connections. The default is `error`.

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Multi Mode to Single Mode (Optical tab) (<u>Schematic Rules Checks Setup</u> form)</i>

srcOpticalSingleToMulti

```
schematic srcOpticalSingleToMulti cyclic { "ignored" "warning" "error" }
```

Description

(Photonics Only) Specifies the severity of the schematic rules checker (SRC) check for single mode nets or outputs that are connected to multi mode inputs or nets. The default is ignored.

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Single Mode to Multi Mode (Optical tab) (<u>Schematic Rules Checks Setup</u> form)</i>

srcOpticalTooManyConnections

```
schematic srcOpticalTooManyConnections cyclic { "ignored" "warning" "error" }
```

Description

(Photonics Only) Specifies the severity of the schematic rules checker (SRC) check for optical nets to ensure each has a maximum of 2 connections. The default is `error`.

GUI Equivalent

Command	<i>Check – Rules Setup</i>
Form Field	<i>Too Many Connections (Optical tab) (<u>Schematic Rules Checks Setup</u> form)</i>

Navigator Environmental Variables

The navigator environmental variables define the behavior of how information is displayed and opened in the *Navigator*.

You can customize the behavior of the navigator by changing the value of a particular environment variable.

- autoResizeColumns
- compactBusDisplay
- defaultOpenIn
- defaultOpenMode
- fullSizeNameColumn
- showSummaryPage

autoResizeColumns

```
treeAssistant autoResizeColumns boolean { t | nil }
```

Description

Controls the automatic resizing of columns in Navigator. When set to `t`, the columns resize automatically.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("treeAssistant" "autoResizeColumns")
envSetVal("treeAssistant" "autoResizeColumns" 'boolean t)
envSetVal("treeAssistant" "autoResizeColumns" 'boolean nil)
```


compactBusDisplay

```
treeAssistant compactBusDisplay boolean { t | nil }
```

Description

Defines how bus bits are displayed in the Navigator. When set to `t`, each bus is displayed in the navigator as a container and when expanded, the associated bits are listed as unique members. A copy of each bit with associated wire shapes in the canvas is also displayed at the same level as the bus when set to `nil`. The default is `nil`.

This environment variable should be set before starting the Virtuoso session.

If a bit has no wire shape on the canvas (implicit bit), it can only be viewed by expanding the container representing the bus.

GUI Equivalent

None

Examples

```
envGetVal("treeAssistant" "compactBusDisplay")
envSetVal("treeAssistant" "compactBusDisplay" 'boolean t)
envSetVal("treeAssistant" "compactBusDisplay" 'boolean nil)
```

Virtuoso Schematic Editor User Guide

Environment Variables

defaultOpenIn

```
treeAssistant defaultOpenIn string { "new tab" | "current tab" | "new window" }
```

Description

Defines how a selected instance is opened using the Open Instance context menu. The options are to open in a new tab, the current tab, or a new window. Corresponds to the *Open In* option in the *Default Open* section of the [The Navigator Options Form](#). The default is *current tab*.

GUI Equivalent

Command	Right-click on Navigator and select <i>Options</i>
Form Field	Open In (The Navigator Options Form)

Examples

```
envGetVal("treeAssistant" "defaultOpenIn")
envSetVal("treeAssistant" "defaultOpenIn" 'string "new tab")
envSetVal("treeAssistant" "defaultOpenIn" 'string "new window")
```

Virtuoso Schematic Editor User Guide

Environment Variables

defaultOpenMode

```
treeAssistant defaultOpenMode string { "read" | "edit" }
```

Description

Defines whether a selected instance is opened using the Open Instance context menu in either *edit* or *read* mode. Corresponds to the *Mode* option in the *Default Open* tab of the The Navigator Options Form. The default is *read*.

GUI Equivalent

Command	Right-click on Navigator and select <i>Options</i>
Form Field	<i>Mode</i> (<u>The Navigator Options Form</u>)

Examples

```
envGetVal("treeAssistant" "defaultOpenMode")
envSetVal("treeAssistant" "defaultOpenMode" 'string "read")
envSetVal("treeAssistant" "defaultOpenMode" 'string "edit")
```

fullSizeNameColumn

```
treeAssistant fullSizeNameColumn boolean { t | nil }
```

Description

Specifies whether the width of the *Name* column in the Navigator assistant should match the longest object name in that column. When set to `t`, the columns are resized automatically so that the entire name is visible.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("treeAssistant" "fullSizeNameColumn")
envSetVal("treeAssistant" "fullSizeNameColumn" 'boolean t)
envSetVal("treeAssistant" "fullSizeNameColumn" 'boolean nil)
```

showSummaryPage

```
treeAssistant showSummaryPage boolean { t | nil }
```

Description

Shows the *Summary* pane or the last selected set in the Navigator when a cellview is opened for the first time in a Virtuoso session.

When set to `t`, the default value, the *Summary* pane is displayed.

When set to `nil`, the last selected set is displayed. If there are no results for the selected set, the *Summary* pane is displayed.

Note: When a cellview is subsequently opened, the Navigator displays the last set selected.

GUI Equivalent

None

Examples

```
envGetVal("treeAssistant" "showSummaryPage")
envSetVal("treeAssistant" "showSummaryPage" 'boolean t)
envSetVal("treeAssistant" "showSummaryPage" 'boolean nil)
```

Net Tracing Environmental Variables

The net tracing environmental variables define the display behavior of net hierarchical information. You can customize the behavior of net tracing by changing the value of a particular environment variable.

alternateHierSeparator

alternateHierSeparator

nt alternateHierSeparator boolean { t | nil }

Description

Controls how paths are separated on the display area of The Probes Assistant. The readability of the results can be improved by changing the path separator. The default is ‘ / ’ and the alternative is ‘ » ’.

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}

GUI Equivalent

Command	<i>Window – Assistant – Probes</i>
Form Field	N/A

Virtuoso Schematic Editor User Guide

Environment Variables

Symbol and Simulation Library Generator

The Cadence® SKILL-based symbol and simulation library generator (S/SLG) is a library management program that generates symbol and simulation views. S/SLG generates a symbol in the library and defines the complete characterization of a symbol for use in computer-aided engineering applications.

This appendix describes the S/SLG commands and program and the library management SKILL functions used with S/SLG.

- [Library Command File](#) on page 1278
- [Running the Symbol and Simulation Library Generator](#) on page 1279
- [Variables That Control Command Execution](#) on page 1283
- [Properties That Modify Values](#) on page 1283
- [Sample Library Command File](#) on page 1284

Library Command File

The library command file is one master text file that contains all text information in the library programming environment needed for S/SLG. Because this file is the only input to S/SLG, library editing becomes essentially text editing. You can generate symbol and simulation views after you create the library command file and run S/SLG.

S/SLG can do the following:

- Add or delete properties in the symbol or simulation view
- Check for library consistency
- Backannotate objects in a schematic
- Specify timing parameters

Note: All Cadence SKILL capabilities are available in S/SLG. Refer to *Virtuoso Schematic Editor SKILL Reference* for details about programming in SKILL.

Running the Symbol and Simulation Library Generator

You can run S/SLG in one of two environments:

- Graphic
- Nongraphic

Running the Symbol and Simulation Library Generator in the Cadence Graphic Environment

S/SLG is integrated into the Cadence graphic environment. If you want to do symbol generation or backannotation, you must run S/SLG in the Cadence graphic environment. Refer to [Figure B-1](#) on page 1280 for an overview.

You can run S/SLG either by typing a function or a sequence of functions in the Command Interpreter Window (CIW) or by creating a library command file.

When you create a library command file that contains S/SLG commands or SKILL functions, each command or function runs sequentially. You can load the file where you plan to save your generated symbol into S/SLG.

To load your file while you run S/SLG,

- In the CIW, type

```
lmLoadData( myFile.lm myLib )
```

where

`lmLoadData` opens your working library, loads your library command file, then closes your working library when you complete running S/SLG.

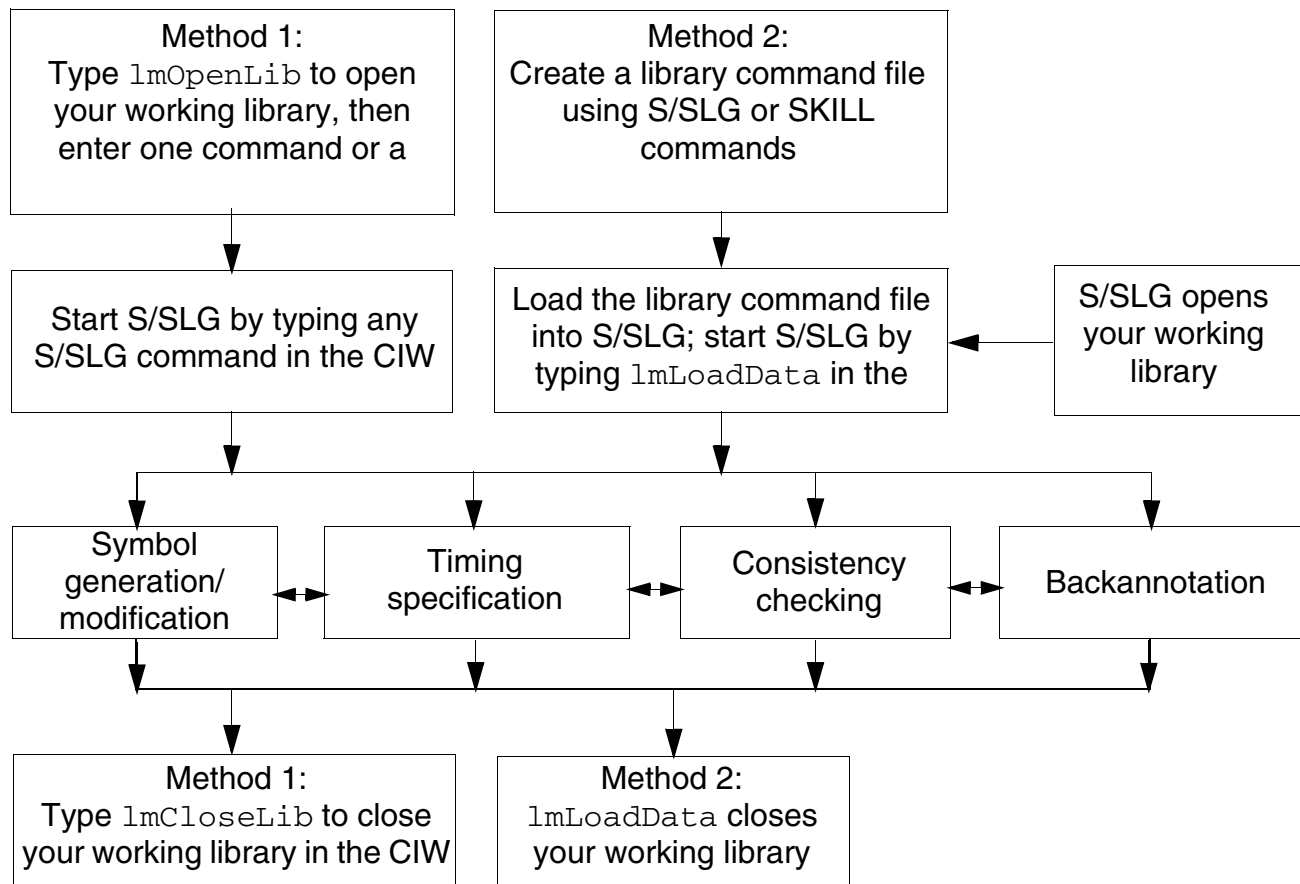
`myFile` is your library command file.

`myLib` is your working library.

Output and Error Messages

An error ends execution. Output and error messages are displayed in the CIW during execution. An exit status of `t` indicates there are no output or error messages; `nil` indicates an error occurred. Error messages can be the result of syntax errors, insufficient arguments, or undefined parameters.

Figure B-1 Diagram of Running S/SLG in the Cadence Graphic Environment



Running the Symbol and Simulation Library Generator in the Cadence Nongraphic Environment

You can run S/SLG in the Cadence nongraphic environment in a UNIX window. For running functions other than `lmDefCell`, run S/SLG in a UNIX window to display the output quickly.

Run the Cadence nongraphic program in a UNIX window to start interactive library editing; then type an S/SLG command or SKILL function.

To load a library command file and run all functions sequentially,

- In a UNIX window, type

```
lmLoadData( "myFile.lm" "myLib" )
```

where

Virtuoso Schematic Editor User Guide

Symbol and Simulation Library Generator

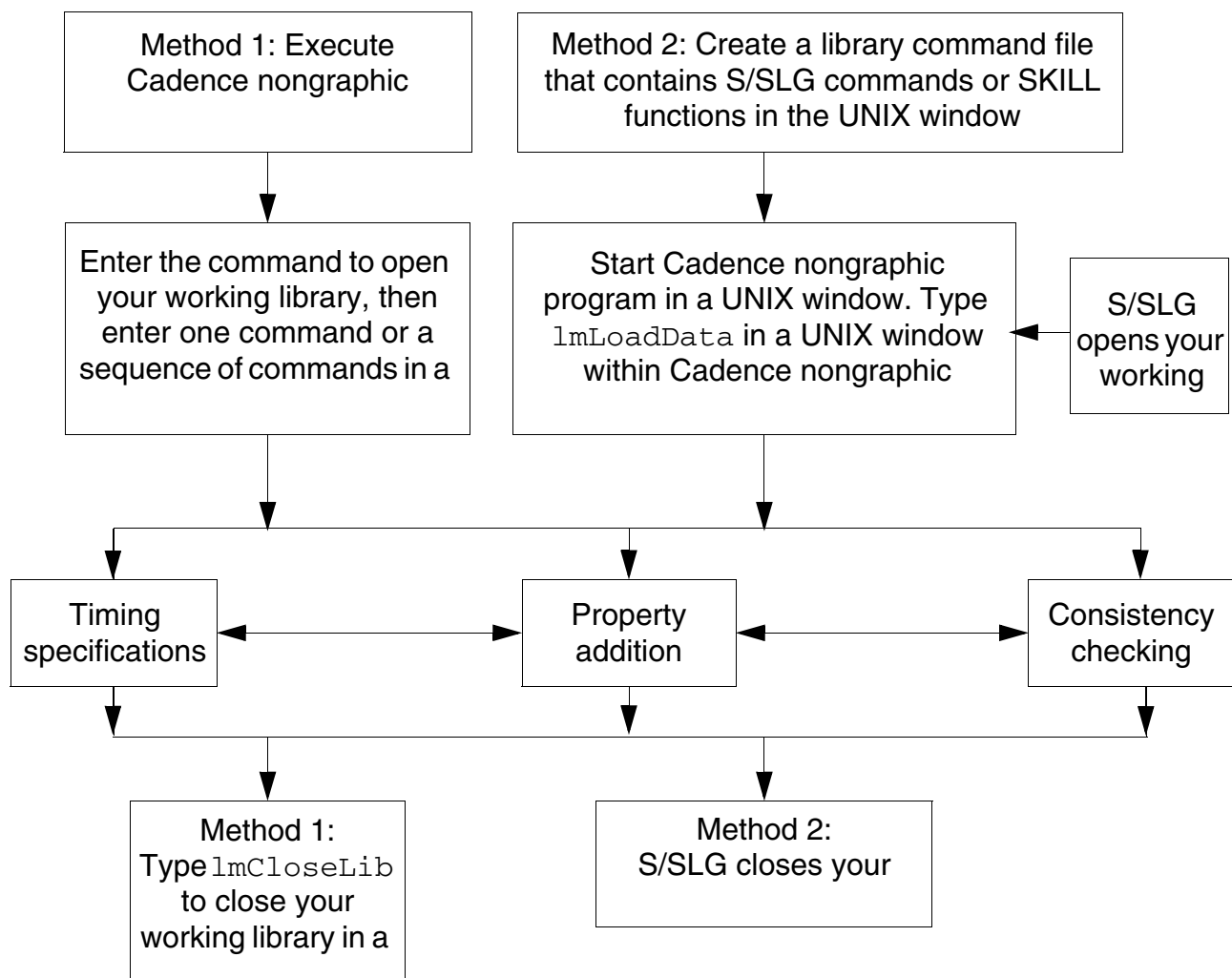
`lmLoadData` opens your working library, loads your library command file, then closes your working library when you complete running S/SLG.

`myFile` is your library command file.

`myLib` is your working library.

Refer to [Figure B-2](#) on page 1281 for an overview of the above processes.

Figure B-2 Diagram of Running S/SLG in the Cadence Nongraphic Environment



Output and Error Messages

Each function returns a running status. Output and error messages are displayed on the screen during execution. An exit status of `t` means that there are no output or error

Virtuoso Schematic Editor User Guide

Symbol and Simulation Library Generator

messages, and `nil` means an error occurred during execution. Error messages can be the result of syntax errors, insufficient arguments, or undefined parameters.

You can direct the output to a file and check it with a text editor, or you can check the output on the screen.

To direct output and error messages to a file, add the following to your library command file:

```
poport = outfile( filename )
errport = outfile( filename )
```

You can specify different files for storing the output or error messages, or you can specify the same file. If you want to store them in the same file, use the following command:

```
poport = errport = outfile( filename )
```

Variables That Control Command Execution

Predefined global variables control command execution that affect the modification of the database. The following table shows variables and how they control `lm` functions.

Variable	Default	Use
<code>symbolGen</code>	<code>t</code>	Controls <code>lmDefCell</code> execution for defining a cell
<code>taGen</code>	<code>t</code>	Controls <code>lmDefCell</code> execution for timing view generation
<code>replaceViewProp</code>	<code>nil</code>	Controls <code>lmDefViewProp</code> execution by controlling the property-adding mode
<code>replaceTermProp</code>	<code>nil</code>	Controls <code>lmDefTermProp</code> execution by controlling the property-adding mode

Properties That Modify Values

To modify the value of a view or terminal property in the Cadence environment, use the property list editor.

To delete a property, use `lmDeleteViewProp` or `lmDeleteTermProp`. Then use `lmDefViewProp` or `lmDefTermProp` in append mode to put the property with the desired value back into the database.

You can also use S/SLG commands and text editing. First, print all properties into a file. Then, edit the file by modifying the values of the target property and load the file into S/SLG to replace all original properties. Remember to set `replaceViewProp` to `t` or `replaceTermProp` to `t`. You can add parameters to the property values.

Sample Library Command File

The following example shows a text file containing several commands and variable settings. These commands and variables generate symbols, specify timing parameters, define cellview and terminal properties, and print cellview and terminal properties.

A semicolon (;) indicates a comment line.

```
lmOpenLib( "myLib" "" "" "a" )
; library must be opened in append mode in order to execute
; subsequent commands, such as lmDefCell, lmDefViewProp. This
; command is redundant if you use lmLoadData to load this library
; command file.
;
symbolGen = t
; global variable setting
;
; cell description for SN74181
;
lmDefCell( "74181"
    ; input and output terminal names
    input( A0 A1 A2 A3 B0 B1 B2 B3 S0 S1 S2 S3 CIN_ M )
    output( F0 F1 F2 F3 AEQB COUT_ Y X )
    defsymbol(
        symbolProps(
            vendorName = "TI"
            partName = "SN74181"
            refDes = "U100"
        )
    )
)
; specify time delay for generating a timing view delayTable(
    ( CIN_ COUT_ 12:13 nonInvert )
    ( (A0 A1 A2 A3) COUT_ 35ns:33ns invert )
    ( (B0 B1 B2 B3) COUT_ 35ns:33ns nonInvert )
    ( CIN_ (F0 F1 F2 F3) 13ns:12ns )
    ( (A0 A1 A2 A3) Y 17ns:17ns nonInvert )
    ( (B0 B1 B2 B3) Y 17ns:17ns invert )
    ( (A0 A1 A2 A3) X 17ns:17ns nonInvert )
    ( (B0 B1 B2 B3) X 17ns:17ns invert )
    ( (A0 B0) F0 32ns:23ns )
    ( (A1 B1) F1 32ns:23ns )
    ( (A2 B2) F2 32ns:23ns )
    ( (A3 B3) F3 32ns:23ns )
    ( (A0 A1 A2 A3 B0 B1 B2 B3) AEQB 35ns:32ns )
)
; add properties into the symbol view
lmDefViewProp(
    timeAdded = time("Apr 1 10:00:00 2000")
    author = "John Smith"
    ; define properties for logic/fault simulation if
    ; used as function model
    simulation = proplist(
        NLPElementPostamble = nlpExpr("@SILOS_FNx_Image")
        Input_Pin_List = nlpExpr("[|A0] [|A1] [|A2] \
            [|A3] [|B0] [|B1] [|B2] [|B3] [|S0] [|S1] [|S2] \
            [|S3] [|CIN_] [|M]")
        Output_Pin_List = nlpExpr("[|F0] [|F1] [|F2] [|F3] \
            [|AEQB] [|COUT_] [|Y] [|X]")
    )
)
```


Virtuoso Schematic Editor User Guide

Symbol and Simulation Library Generator

```

Pin_Net_Map = nlpExpr("\n$ 8 [|F0]=F0 [|F1]=F1 \
 [|F2]=F2 [|F3]=F3 [|AEQB]=AEQB [|COUT_]=COUT_ \
 [|Y]=Y [|X]=X \
 [|A0]=A0 [|A1]=A1 [|A2]=A2 [|A3]=A3N [|B0]=B0 \
 [|B1]=B1 [|B2]=B2 [|B3]=B3 [|S0]=S0 [|S1]=S1 \
 [|S2]=S2 [|S3]=S3 [|CIN_]=CIN_ [|M]=M") \
; The value of Input_Pin_List Output_Pin_List and
; Pin_Net_Map should be put into one line of a text
; file. Here it is split into several lines for a more
; readable format.
)
)
; define the characteristics of input/output current for
; terminals
; iol : low-level output current
; ioh : high-level output current
; iil : low-level input current
; iih : high-level input current
;
lmDefTermProp(
( output iol = 16mA )
( (F0 F1 F2 F3 COUT_ Y X) ioh = -800uA )
( AEQB ioh = 250uA )
( (A0 A1 A2 A3 B0 B1 B2 B3) iil = -4.8mA iih = 120uA )
( (S0 S1 S2 S3) iil = -6.4mA iih = 160uA )
( M iil = -1.6mA iih = 40uA )
( COUT_ iil = -8mA iih = 200uA )
)
)
; cell description for TC4531BP - 12bit parity tree
; ignore symbol generation
symbolGen = nil
lmDefCell( "4531"
; input and output terminal names
input( D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11\
"ODD|EVEN" )
output( Q )
defsymbol(
symbolProps(
vendorName = "TOSHIBA"
partName = "TC4531BP"
refDes = "U200"
)
)
)
; specify time delay
delayTable(
( (D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11) 320ns:320ns )
( "ODD|EVEN" 210ns:210ns )
)
; add properties into the symbol view
lmDefViewProp(
timeAdded = time("Apr 1 10:00:00 2000")
author = "John Smith"
; define properties for logic/fault simulation
; if used as a function model
simulation = proplist(
NLPElementPostamble =
nlpExpr("[@logic_FNx_Image]")
Input_Pin_List = nlpExpr("[|D0] [|D1] [|D2] \
 [|D3] [|D4] [|D5] [|D6] [|D7] [|D8] [|D9] \
 [|D10] [|D11] [|ODD|EVEN]") \

```

Virtuoso Schematic Editor User Guide

Symbol and Simulation Library Generator

```
Output_Pin_List = nlpExpr("[|Q|") \
Pin_Net_Map = nlpExpr("\n$ 1 [|Q|=Q [|D0|=D0 \
    [|D1|=D1 [|D2|=D2 [|D3|=D3 [|D4|=D4 [|D5|=D5 \
    [|D6|=D6 [|D7|=D7 [|D8|=D8 [|D9|=D9 \
    [|D10|=D10 [|D11|=D11 [|ODD|EVEN]=ODD|EVEN")
; The value of Input_Pin_List and Pin_Net_Map
; should be put into one line of a text file. Here it
; is split into several lines for a more readable format
)
)
;
; define the characteristics of input/output current for
; terminals
; iol : low-level output current
; ioh : high-level output current
; iil : low-level input current
; iih : high-level input current
; VDD = 5V, VSS = 0V, 25C
lmDefTermProp(
    ( output
        iol = 1.5 ; 1.5A
        ioh = -1.0 ; -1.0A
    )
    ( input
        iil = -0.01nA
        iih = 0.01nA
    )
)
)
; append properties to the symbol of SN74181
replaceViewProp = nil
lmDefViewProp( "74181"
    vdd = 5
    date = time("Apr 1 12:00:00 2000")
)
; append properties to some terminals of the symbol of TC4531
replaceTermProp = nil
lmDefTermProp( "4531"
    ("ODD|EVEN"
        check = "odd or even parity"
    )
    (Q is = "only output"
    )
)
)
; print all properties of the symbol of SN74181 in the file
; prop.74181
lmPrintViewProp( "74181" symbol "prop.74181" )
; print all terminal properties of the symbol of TC4531BP in
; the file term.4531
lmPrintTermProp( "4531" symbol "term.4531" )
; replace the properties in the symbol of SN74169
replaceViewProp = t
lmDefViewProp( "74169" symbol
    userUnits = "userUnits"
    "DBUPerUU" = 1.0
    snapSpacing = 5
    screenGridSpacing = 10
    screenGridMultiple = 4
    displayExpanded? = yes
    type = "TTL"
    manufacturerID = "TI"
```

Virtuoso Schematic Editor User Guide

Symbol and Simulation Library Generator

```
    partName = "1000"
)
; replace the terminal properties in the symbol of SN74169
replaceTermProp = t
lmDefTermProp( "74169" symbol
    (input
        iil = -2mA
        iih = 50uA
    )
    (output
        iol = 20mA
        ioH = -1mA
    )
)
; print the properties of symbol view for each cell in the current
; working library in the file sym.prop
lmPrintLibViewProp( symbol "sym.prop" )
; print the terminal properties of symbol view for each cell
; in the current working library in the term.prop file
lmPrintLibTermProp( symbol "term.prop" )
lmCloseLib
; this command is redundant if you use lmLoadData to load this
; command file.
```

Virtuoso Schematic Editor User Guide

Symbol and Simulation Library Generator

Text-to-Symbol Generator

The text-to-symbol generator (TSG) is a Cadence® application program that automatically generates symbol cellviews for the Virtuoso® Schematic Editor and subsequent simulation processes. TSG provides a quick way to generate a symbol from a list of pins in a TSG file.

This appendix describes the TSG file syntax and illustrates the use of various TSG constructs.

- [Symbol Shape Generated by TSG](#)
- [TSG Symbol Description File](#)
- [TSG Template File](#)
- [Clock and Negation Indicators](#)
- [Backannotation](#)
- [Simple File Example](#)
- [Complex File Example](#)
- [Extended File Example](#)
- [TSG Symbol Description File Structure](#)
- [Executing TSG](#)

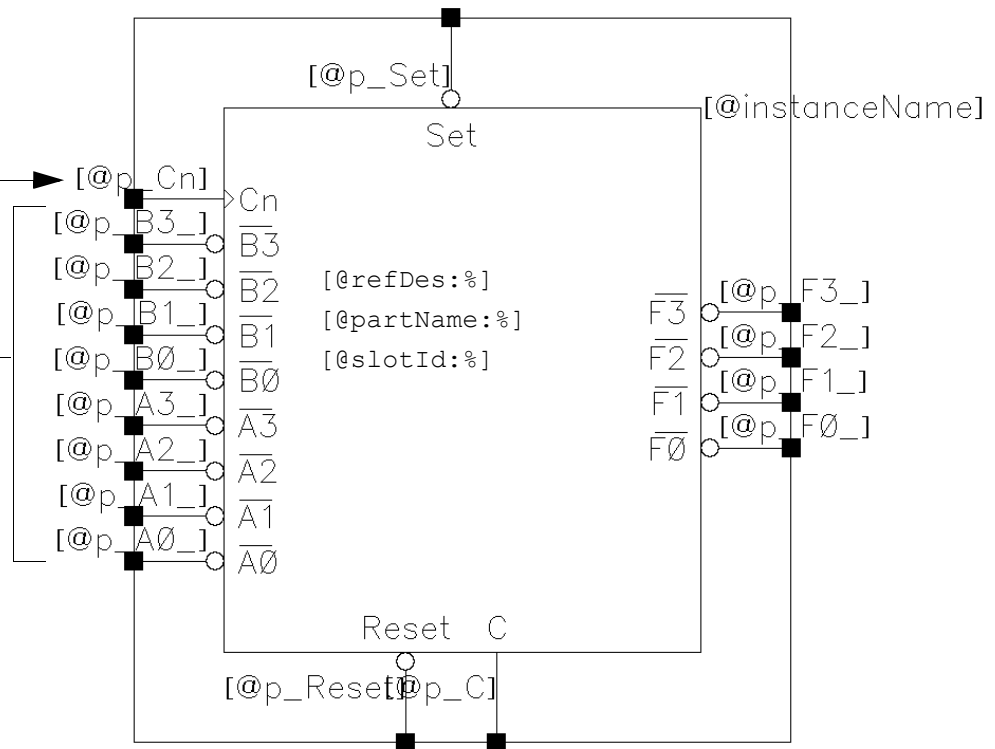
Symbol Shape Generated by TSG

All symbols generated by TSG are rectangular in shape. The default origin of a symbol is located at the upper left pin terminal. The following symbol was created by TSG.

A Symbol Master

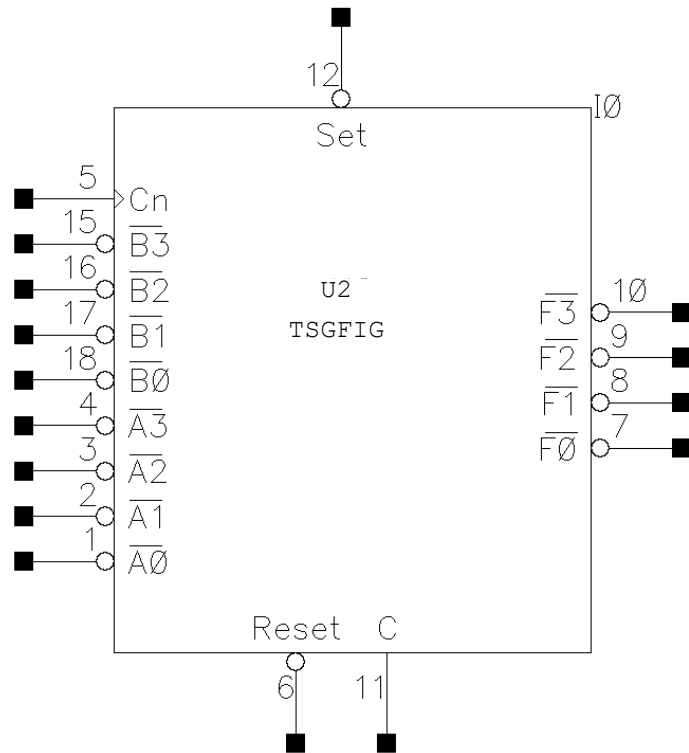
Clock pin with input
clock indicator
(arrow) →

Negative logic
types with
negation indicator
(bubble) —



An Instance

An instance of the symbol shown above when placed in a schematic



TSG Symbol Description File

The TSG symbol description file is the primary input file that controls the symbol to be generated by TSG. This ASCII file is built using the constructs detailed in this appendix. You can create a simple file containing only the cell name and the pin names, or you can add constructs to enhance the symbol description.

Defaults and error checking are built-in features of TSG. TSG has extensive error-checking and error-handling capabilities. It checks for syntax, variable-type, and logical errors in the TSG symbol description file. Depending on the severity of a detected error, TSG either exits the generation of the current symbol or displays an error message and continues processing.

TSG Template File

The TSG template file is a secondary input file that specifies default controls for symbols to be generated by TSG. The TSG template file uses the same format as the TSG symbol description file. Within this file, you can configure symbol labels, properties, and characteristics such as pin spacing, stub length, and origin placement.

A template file helps avoid unnecessarily large TSG description files. You can specify TSG constructs shared by a large group of symbols within a single template file. This allows you to create TSG symbol description files that contain only constructs specific to a particular symbol.

The first time you use TSG during a schematic editor session, a default template file is automatically loaded. This default template file is always used unless you specify an optional template file argument in the call to execute TSG, as described later.

Settings specified within the currently active template file will apply to all symbols generated by TSG unless overridden by constructs specified within the TSG symbol description file. You can find available template filenames within the *tsgTemplateMasters* list, which is defined in the `schConfig.il` file. The default template file that is chosen depends on the initial setting of the `tsgTemplateType` environment variable.

Clock and Negation Indicators

To alter the appearance of certain pins, you can specify special pin graphics: for example,

- To identify a clock pin, you can draw a clock indicator (a small triangle) at the beginning of the pin wire
- For a negative logic pin, you can draw a negation indicator (a bubble) at the beginning of the pin wire

The example TSG files and associated figures illustrate some of these capabilities. Refer to the [PinGraphicSpec](#) construct and the [wireLength](#) and [pinConnector](#) symbol parameters.

Backannotation

After laying out parts for printed circuit boards (PCB), you often need to backannotate to schematics or PCB maps parameters, such as pin numbers and reference designators of symbol instances.

You can include constructs in the TSG symbol description file that specify the properties and parameters required to support the backannotation operation.

Simple File Example

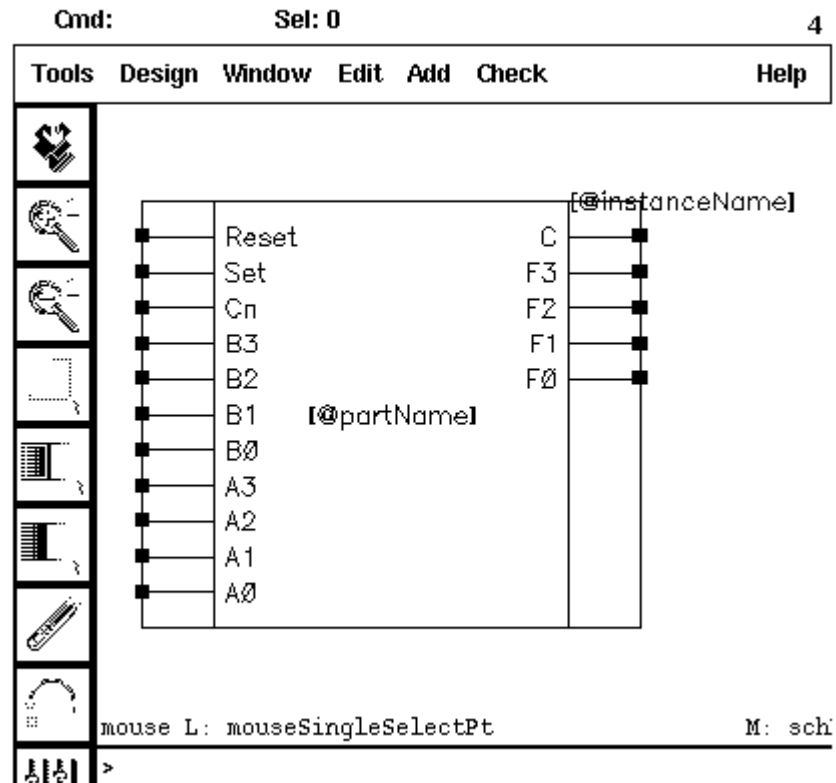
The simplest TSG symbol description for an SN74LS181 symbol has the following format:

```
defcell("tsgfig1"  
  input(A0 A1 A2 A3 B0 B1 B2 B3 Cn Set Reset)  
  output(F0 F1 F2 F3 C)  
  defsymbol(symbolProps( partName = "TSGFIG1"))  
)
```

Virtuoso Schematic Editor User Guide

Text-to-Symbol Generator

The following example shows a symbol generated from the TSG symbol description file shown above.



Complex File Example

A more elaborate version of a TSG symbol description file for an SN74LS181 symbol has the following format. This symbol was generated using the PCB package.tsg template file as an argument to the tsg command:

```
tsg( "exampleLib" "~/tsgFiles/fig2.tsg"
    prependInstallPath("samples/symbolGen/package.tsg") )
defcell("tsgfig2"
    input( A0 A1 A2 A3 B0 B1 B2 B3 Cn Set Reset)
    output( F0 F1 F2 F3 C)
    defsymbol(
        symbolProps(
            vendorName = DEMO
            partName = "TSGFIG2"
            refDes = U2 )
        pinNumSpec(
            A0:1 A1:2 A2:3 A3:4
            B0:18 B1:17 B2:16 F0:7
            F1:8 F2:9 F3:10 C:11 )
        pinLocSpec(
```

Virtuoso Schematic Editor User Guide

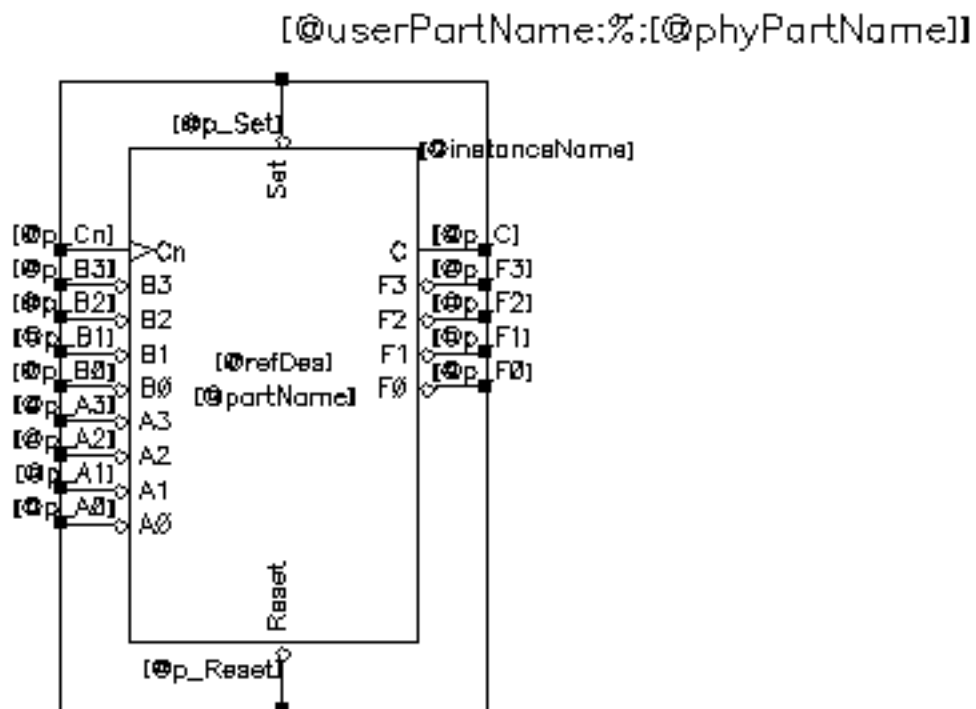
Text-to-Symbol Generator

```

topPins( Set)
bottomPins( Reset) )
pinGraphicSpec(
  A0:actLo  A1:actLo  A2:actLo  A3:actLo
  B0:actLo  B1:actLo  B2:actLo  B3:actLo
  Cn:clock  Set:actLo  Reset:actLo  F0:actLo
  F1:actLo  F2:actLo  F3:actLo    C:actHi  )
)
)

```

The following example shows a symbol generated from the TSG symbol description file shown above.



Extended File Example

The following example demonstrates many of the available TSG constructs.

```
defcell("count4"
  input("clock" "reset")
  output("a0" "a1" "a2" "a3")
  defsymbol(
    symbolProps(
      partName = "count4" ; cellview prop with valueType=string
      pinNum   = (4 (0 24))
      timeVal  = time("Jan 1 12:00:00 1993")
      color    = ("red" ("red" "green" "blue"))
      defTermProp(
        input Iih = 1.24 ) ; all input pins get property "Iih".
      defTermProp(
        all b=6 ) ; all pins get property "b".
    )
    symbolParam(
      wireSpacing = 0.125000
      wireLength  = 0.250000
      vSideLength = 0.000000
      hSideLength = 0.000000
      origin      = topLeftPin
      pinConnector = "block"
    )
    symbolLabels(
      defLabel( name("[@partName]")
        location( "(xleft + xright)/2:(ytop + ybottom)*3/4")
        labelType(NLPLLabel)
        layer(device)
        purpose(label)
      )
      defLabel( name("[@instanceName]")
        location( "xleft:(ytop + ybottom)/2")
        labelType(NLPLLabel)
        justification(upperLeft)
        layer(instance)
        purpose(label)
        apply(cellview)
        fontHeight(0.1)
      )
      defLabel( name("{pinName}")
        location( "1.15*stubLength:0")
        justification(centerLeft)
        apply(left)
      )
      defLabel( name("[@p_{pinName}]")
        location( "-stubLength/2:0.03125")
        labelType(NLPLLabel)
        layer(pin)
        purpose(annotate)
        apply(right)
      )
    )
    pinNumSpec( "a0":16 "a1":17 "a2":18 )
    pinGraphicSpec( "reset":ieeeActLo "clock":clock )
  )
)
```

Virtuoso Schematic Editor User Guide

Text-to-Symbol Generator

The following example shows a symbol generated from the TSG symbol description file shown above.



TSG Symbol Description File Structure

Basic File Structure

A TSG file is an ASCII file used as input to TSG for generating symbol cellviews. To create the file, you must use a special format that includes several constructs and keywords. All constructs have one of the following forms (CF = construct format):

CF1: =keyword([variable] ... [variable] [construct] ... [construct])

CF2: =keyword([keyword = value] ... [keyword = value] [construct] [construct])

CF3: =keyword([variable : value] ... [variable : value])

Some constructs can appear only inside certain other constructs. Some keywords can appear only once within a construct.

To define more than one symbol in a TSG file, you can use multiple `defcell` constructs. Unless otherwise specified, draw input pins on the left side, output pins on the right side, and I/O pins along the top.

To add comments to a TSG file, use the Cadence SKILL language comment characters ;
or /* */ .

An illustration of the structure of the `defcell` construct, which is the only top-level construct allowed in a TSG file, follows.

Virtuoso Schematic Editor User Guide

Text-to-Symbol Generator

Syntax

```
defcell( t_cellName
  [ t_pinType( t_pinName ... ) ]
  [ defsymbol( [ t_viewName ]
    [ symbolProps(
      t_propName = g_propValue
      ...
      [ defTermProp( t_defPinName t_propName = g_propValue ) ]
      ...
      [ defVisibleProp( t_propName = g_propValue [ labelAttr ] ]
      ...
    ) ]
    [ symbolParam(
      [ wireSpacing = f_spacing ]
      [ wireLength = f_length ]
      [ vSideLength = f_vLength ]
      [ hSideLength = f_hLength ]
      [ pinConnector = t_connector ]
      [ origin = t_origin ]
    ) ]
    [ symbolLabels(
      defLabel( name( t_labelText ) labelAttr
      ...
    ) ]
    [ controlParam(
      [ queryMode = b_boolean ]
    ) ]
    [ pinNumSpec(
      [ t_pinName : x_pinNumber ] ...
    ) ]
    [ pinLocSpec(
      [ leftPins( t_leftPinName ... ) ]
      [ rightPins( t_rightPinName ... ) ]
      [ topPins( t_topPinName ... ) ]
      [ bottomPins( t_bottomPinName ... ) ]
    ) ]
    [ pinGraphicSpec(
      [ t_pinName : t_pinGraph ] ...
    ) ]
    [ pinLogicSpec(
      [ positive( t_posPin ... ) ]
      [ negative( t_negPin ... ) ]
    ) ]
    [ clockPins( t_clockPin ...
    ) ]
  ) ]
)
```

Virtuoso Schematic Editor User Guide

Text-to-Symbol Generator

Arguments

<i>t_cellName</i>	The cell name portion of a symbol cellview into which the generated symbol is stored; must be enclosed in quotation marks.				
<i>t_pinType</i>	The type of pin: must be enclosed in quotation marks. Valid Values: <code>input</code>, <code>output</code>, <code>io</code> By default, all input pins are drawn on the left side of the symbol from top to bottom, all output pins are drawn on the right side of the symbol from top to bottom, and all I/O pins are drawn on top of the symbol from left to right. To move a pin to any other side, specify the pin in the <code>pinLocSpec</code> construct.				
<i>t_pinName</i>	The name of the pin: must be enclosed in quotation marks.				
<i>defsymbol</i>	Specifies symbol properties and parameters, control parameters, labels, and pin information. These values override applicable defaults.				
<i>t_viewName</i>	The view name of the cellview to be created. If you specify <i>t_viewName</i>, it must be the first parameter in the <code>defsymbol</code> construct. Default: <code>symbol</code>				
<i>symbolProps</i>	Creates properties on the symbol cellview or properties on sets of terminal pins.				
<i>t_propName</i>	The name of the property.				
<i>g_propValue</i>	The value of the property to set; can be any of the following: <pre><i>tnb_value</i> <i>tnb_value</i> [({ { <i>t_stringEnum</i> <i>t_string</i> } { <i>n_beginRange</i> <i>nil</i> } { <i>n_endRange</i> <i>nil</i> } })] time(<i>t_timeVal</i>) filename(<i>t_filename</i>) ilExpr(<i>t_ilExpr</i>) nlpExpr(<i>t_nlpExpr</i>)</pre> where <table><tr><td><i>tnb_value</i></td><td>An integer, a floating-point number, a string (must be enclosed in quotation marks), or Boolean (<code>t</code>, <code>nil</code>).</td></tr><tr><td><i>t_stringEnum</i></td><td>Must be enclosed in quotation marks.</td></tr></table>	<i>tnb_value</i>	An integer, a floating-point number, a string (must be enclosed in quotation marks), or Boolean (<code>t</code> , <code>nil</code>).	<i>t_stringEnum</i>	Must be enclosed in quotation marks.
<i>tnb_value</i>	An integer, a floating-point number, a string (must be enclosed in quotation marks), or Boolean (<code>t</code> , <code>nil</code>).				
<i>t_stringEnum</i>	Must be enclosed in quotation marks.				

Virtuoso Schematic Editor User Guide

Text-to-Symbol Generator

<i>t_string</i>	Must be enclosed in quotation marks.
<i>n_beginRange</i>	The beginning pin in the range. <i>nil</i> indicates the first pin.
<i>n_endRange</i>	The ending pin in the range. <i>nil</i> indicates the last pin.
<i>t_timeVal</i>	The date, time, and year, in the format "mmm dd h:mm:ss yyyy"
<i>t_filename</i>	A filename; must be enclosed in quotation marks.
<i>t_ilExpr</i>	A SKILL expression; must be enclosed in quotation marks.
<i>t_nlpExpr</i>	An NLP expression; must be enclosed in quotation marks.
	If used as part of a <i>defTermProp</i> construct, <i>g_propValue</i> does not need to be in quotation marks.
<i>defTermProp</i>	Defines properties on terminals.
<i>t_defPinName</i>	The pin name to define properties for. Valid Values: all, input, output, inputOutput, top, bottom, left, right
<i>defVisibleProp</i>	Defines a property and the label attributes information for displaying that property.
<i>labelAttr</i>	The attributes for the label; can be any of the following: <i>location("expression:expression")</i> <i>apply(t_applyString)</i> <i>labelType(t_labelType)</i> <i>format(t_format)</i> <i>layer(t_layerName)</i> <i>purpose(t_purpose)</i> <i>justification(f_justification)</i> <i>orientation(t_orientation)</i> <i>isOverbar(b_isOverbar)</i> <i>fontStyle(t_fontStyle)</i> <i>fontHeight(n_fontHeight)</i> where

Virtuoso Schematic Editor User Guide

Text-to-Symbol Generator

<i>expression</i>	A SKILL expression that evaluates to a point. <i>expression</i> is a function of the variables <i>xleft</i> , <i>xright</i> , <i>ytop</i> , <i>ybottom</i> , and <i>fontHeight</i> for labels applied to a cellview, and of the variables <i>pinSpacing</i> , <i>stubLength</i> , and <i>fontHeight</i> for labels applied to pins. Top, bottom, left, and right pin label positions must be specified separately.
<i>t_applyString</i>	Specifies whether the label is applied to either the cellview or to specific pins. Valid Values: cellview, top, bottom, left, right, input, output, inputOutput
<i>t_labelType</i>	(valid only in the defLabel construct) Specifies the label type. Valid Values: normalLabel, NLPLabel, ILLabel Default: normalLabel
<i>t_format</i>	(valid only in the defVisibleProp construct) Specifies which parts of the property name/value pair are to be displayed; must be enclosed in quotation marks. Valid Values: value, name=value, off Default: value
<i>t_layerName</i>	The layer of the label. Valid Values: Default: device
<i>t_purpose</i>	The layer purpose of the label. Valid Values: Default: annotate
<i>f_justification</i>	Valid Values: upperLeft, centerLeft, lowerLeft, upperCenter, centerCenter, lowerCenter, upperRight, centerRight, lowerRight Default: lowerCenter
<i>t_orientation</i>	The rotation of the label, in degrees. Valid Values: R0, R90, R180, R270 Default:

Virtuoso Schematic Editor User Guide

Text-to-Symbol Generator

<i>b_isOverbar</i>	Determines whether a bar (or overline) is placed over the label text that is placed between underscore characters. <i>t</i> places the bar on the text; <i>nil</i> does not. If <i>isOverbar</i> is <i>t</i>, place an underscore before and after any text you want an overbar above. For example, the label <pre>read _write_</pre> converts to <pre>read write</pre> with an overbar above <i>write</i>. Valid Values: <i>t</i>, <i>nil</i> Default: <i>nil</i>
<i>t_fontStyle</i>	The font style for a label. Valid Values: <i>euroStyle</i>, <i>gothic</i>, <i>math</i>, <i>roman</i>, <i>script</i>, <i>stick</i>, <i>fixed</i>, <i>swedish</i>, <i>milSpec</i> Default: <i>stick</i>
<i>n_fontHeight</i>	The default font height (in user units) for a label. Default: 0.0625 The smallest value that a label height can be set at is the inverse of the <i>DBUPerUU</i> (database units per user units) value, as defined in the library property. The values entered as label heights are also rounded to the nearest resolution value ($1/DBUPerUU$). For example, 0.003125 (which is half of 0.00625) will be rounded up to 0.00625, but 0.003124 will be rounded down to 0 (zero).
<i>symbolParam</i>	Specifies symbol parameters. They can be included in any order without replication. They are <i>not</i> stored in the symbol master property list.
<i>f_spacing</i>	The minimum spacing between any pair of adjacent pin wires on a side, rounded to the nearest multiple of snap spacing.
<i>f_length</i>	The length of each pin wire (a pin stub), rounded to the nearest multiple of snap spacing Default: twice the value of the <i>wireSpacing</i> field

Virtuoso Schematic Editor User Guide

Text-to-Symbol Generator

<code>f_vLength</code>	The minimum length of the vertical sides, rounded to the nearest snap spacing. If the value you supply is less than the computed default, the computed default is used.
<code>f_hLength</code>	The minimum width of the horizontal sides, rounded to the nearest snap spacing. If the value you supply is less than the computed default, the computed default is used.
<code>t_connector</code>	The default pin connector graphic (each entry is mapped into a cellview that contains the actual graphics for the connector); must be enclosed in quotation marks. Valid Values: <code>square</code> , <code>block</code> , <code>circle</code>
<code>t_origin</code>	The default placement of the symbol origin. Valid Values: <code>topLeftPin</code> , <code>bottomLeftPin</code> , <code>centerLeft</code>
<code>symbolLabels</code>	Define symbol labels. If this construct is not specified, a set of default labels is generated on the symbol.
<code>t_labelText</code>	The text of the label; must be enclosed in quotation marks.
<code>controlParam</code>	Contains control parameters to define execution behavior. This control parameter is <i>not</i> stored in the property list of the symbol master.
<code>b_boolean</code>	Use only when the specified symbol master already exists in the library. When set to <code>t</code> , TSG asks several questions interactively. When set to <code>nil</code> (the default), TSG assumes an existing symbol should be overwritten. It does <i>not</i> prompt you for permission to overwrite an existing symbol cellview. Valid Values: <code>t</code> , <code>nil</code> Default: <code>nil</code>
	This control parameter is the only one that can be included in this construct and is <i>not</i> stored in the property list of the symbol master.
<code>pinNumSpec</code>	This construct is most often used for printed circuit board design libraries.
<code>t_pinName</code>	A pin name; must also be declared in <code>input</code> , <code>output</code> , or <code>io</code> .
<code>x_pinNumber</code>	A pin number.
<code>pinLocSpec</code>	Allows you to override default pin placement locations; specifies the pin location in any order without replication.

Virtuoso Schematic Editor User Guide

Text-to-Symbol Generator

t_leftPinName The names of pins to draw on the left of the symbol; must be enclosed in quotation marks. Must also be specified in the `input`, `output`, or `io` construct.

Other pins can be drawn on the left of the symbol. Pins specified in `input` are also drawn on the left of the symbol. A pin specified in `input` but *not* specified in `pinLocSpec` is drawn on the left before those in `leftPins`. A pin name in `leftPins` cannot be specified again in the other three subconstructs.

t_rightPinName

The names of pins to draw on the right of the symbol; must be enclosed in quotation marks. Must also be specified in the `input`, `output`, or `io` construct.

Other pins can be drawn on the right of the symbol. Pins specified in `output` are also drawn on the right of the symbol. A pin specified in `output` but *not* specified in `pinLocSpec` is drawn on the right before those in `rightPins`. A pin name in `rightPins` cannot be specified again in the other three subconstructs.

t_topPinName The names of pins to draw on the top of the symbol; must be enclosed in quotation marks.

Other pins can be drawn on the top of the symbol. Pins specified in `io` are also drawn on the top of the symbol. A pin specified in `io` but not in `pinLocSpec` is drawn on the top before those in `topPins`. A pin name in `topPins` cannot be specified again in the other three subconstructs.

t_bottomPinName

The names of pins to draw on the bottom of the symbol; must be enclosed in quotation marks.

A pin name in `bottomPins` cannot be specified again in the other three subconstructs.

pinGraphicSpec Identifies special pin graphics, such as negation bubbles and clock indicators, to be drawn with the specified pins.

t_pinGraph Identifies special pin graphics, such as negation bubbles and clock indicators.

Valid Values: `actHi`, `actLo`, `ieeeActLo`, `clock`, `actLoClock`
Default: `actHi`

Each of these values must match an entry in the `schConfig.il` file.

Virtuoso Schematic Editor User Guide

Text-to-Symbol Generator

<code>pinLogicSpec</code>	Using this construct is <i>not recommended</i>, but it remains supported for backward compatibility. The <code>pinGraphicSpec</code> construct is the preferred method for specifying special pin graphics. Specifies negation indicators.
<code>t_posPin</code>	A list of pin names that indicate the positive logic type of the pins; must be enclosed in quotation marks.
<code>t_negPin</code>	A list of pin names that indicate the negative logic pin type; must be enclosed in quotation marks. Negative pins appear in the symbol as a bubble.
<code>clockPins</code>	Using this construct is <i>not recommended</i>, but it remains supported for backward compatibility. The <code>pinGraphicSpec</code> construct is the preferred method for specifying special pin graphics. Specifies the display of clock indicators.
<code>t_clockPin</code>	A list of pin names that indicate clock pins. Any pins must also be specified in <code>input</code>, <code>output</code>, or <code>io</code>. Clock pins are designated by a small triangle next to the pin.

Executing TSG

TSG runs as a separate SKILL program in the Cadence environment to generate symbols that you can place as instances in schematics.

Running TSG from SKILL requires a library name, an input filename, and optionally a template filename. TSG first reads the input parameters from the TSG file, then uses any symbol parameters contained in the template file that are not specified in the input file. This allows you to have one large template file that specifies most of the symbol parameters and smaller TSG files that contain only pin specifications. The resulting cellview is saved to the library identified in the command.

To run TSG in TSG mode,

1. Start the Cadence design environment.
2. In the CIW, type the SKILL function call

```
tsg( libName tsgFile [ tsgTemplateFile ] ) )
```

libName is the name of a preexisting library.

tsgFile is the name of the TSG symbol description text file. If the TSG file is not in the current directory, include the full path to the file.

tsgTemplateFile is the name of an optional TSG template file. If the template file is not in the current directory, include the full path to the file.

Example: `tsg( "myLib" "pathToTSGFile" )`

TSG Processing Order

When you run TSG, the following sequence of steps is performed.

1. TSG fetches the first unprocessed `defcell` from the `tsgFileName` file and parses it to extract the symbol information.

Any syntax error messages are displayed in the CIW and the program continues.

2. If the symbol does not exist in the specified library, TSG goes to step 6.
3. If you set the `queryMode` parameter to `t` (true), TSG warns you of the existence of the symbol view and asks whether to overwrite the existing symbol with one that it is going to generate.

If you respond `yes`, TSG goes to step 6. If you respond `no`, TSG goes to step 4. If you set `queryMode` to `nil` (not true), TSG goes to step 6.

4. TSG asks whether you want to save the symbol under a different cell name.

If you respond `yes`, TSG continues. If you respond `no`, the symbol is not saved and TSG goes to step 7.

5. TSG asks you to specify a new cell name for the symbol.

6. TSG generates and saves the symbol in the specified library.

7. TSG checks for an unprocessed `defcell` in the `tsgFileName` file.

8. If such constructs exist, TSG returns to step 1. If not, it stops execution and passes control to the Cadence design system.

After program execution, you can find any error messages printed by TSG in the Command Interpreter Window (CIW).

Creating Symbols Automatically from a Schematic

A convenient feature available from the schematic editor allows you to create a symbol with the same pins that are found within an existing schematic, Verilog, or other cellview.

The TSG symbol generator is automatically invoked whenever you create symbols using *Create – Cellview – From Cellview* or *Create – Cellview – From Pin List*, which are available from the schematic and symbol windows. You can use the Symbol Generation Options form to override defaults or to help create new TSG template files.

The Symbol Generation Options form can be accessed by turning on the *Edit Options* button on the Cellview From Cellview or Cellview From Pin List form.

Form Descriptions

This appendix describes the Virtuoso[®] Schematic Editor forms, in alphabetical order.

Add Custom Pin

Use this form to create and place a new custom pin. It is accessed from the symbol editor using the *Create – Custom Pin* command. F

Pin Names specifies the names for the pins you want to add to your design and the order in which you want to add them. To add more than one pin name, use a space to separate the pin names. You can also use a bus expression to name an array of pins.

Direction contains a list of pin directions you can specify for the pins you want to add. *switch* is maintained solely for backward compatibility.

Bus Expansion specifies how the system interprets bus names.

off (default) interprets a bus name as one name and generates only one pin name that includes all the bits in the bus range. For example, for the name A<0:1>, the system generates only the pin A<0:1>.

on interprets a bus name as multiple names and generates a pin for each bit of a bus range that you specify in the *Pin Names* field. For example, for the name A<0:1>, the system generates labels A<0> and A<1>.

Draw Mode specifies the line entry method.

 draws a straight line in any angle between two points.

 begins an L90 shape with a horizontal line segment and alternates between horizontal and vertical.

 begins an L90 shape with a vertical line segment and alternates between vertical and horizontal.

 draws either a horizontal or vertical line segment first, then a 45-degree angle.

 draws a 45-degree angle first, then either a horizontal or vertical line segment.

Lock Angle appears only when *Draw Mode* is set to  .

any draws a line using any angle.

45 restricts cursor movement to 45-degree angles from the starting point.

Related topics

[Creating Custom Pins as Graphic Images](#)

Add Generic – VHDL Properties

Use this form to add a VHDL property. It is accessed by clicking *Add* on the VHDL Properties form.

Name specifies the generic name.

Type specifies the generic type.

Value (Optional) specifies a value for the generic.

Related topics

[Adding VHDL Properties](#)

Add Instance

Use this form to place multiple copies of a single instance in a schematic, based on a single *lib/cell/view* name instances. It is accessed using the *Create – Instance* command.

Library specifies the library that contains the instance.

Cell specifies the instance cell name.

View specifies the instance view name.

Names specifies the names of the components you want to add to the design.

Browse opens a browser. Use the *Commands* tab in the Editor Options form (*Options – Editor* command) to specify which browser, the Library Browser or Component Browser, the system opens.

Add Wire Stubs at:

all terminals creates wire stubs for all terminals in the cellview.

Environment Variable: [allTerms](#)

registered terminals only creates wire stubs only for those terminals that are registered with a net. See also the [Register Net Name per Terminal](#) form.



Displays the Net Name Registration form. For any terminal displayed in the left column, you can enter a net name in the opposite, right, column. Once you have then selected *OK*, the net names that have been entered are registered as the default net names for the corresponding terminals. Additionally, you can unregister existing registered terminals by ensuring the adjacent column is empty.

The registered net names will then be used when wire stubs are created using either the *Create – Instance* or the *Create – Wire Stubs and Names* commands, in the current session.

Environment Variable: [addStubs](#)

Array lets you create multiple copies of the specified instance in a single operation.

Rows specifies the number of rows.

Columns specifies the number of columns.

Virtuoso Schematic Editor User Guide

Form Descriptions

Rotate turns an object in 90-degree increments counterclockwise around the current reference point.

Sideways mirrors the object across an imaginary vertical line passing through the current reference point.

Upside Down mirrors the object across an imaginary horizontal line passing through the current reference point.

(Virtuoso RF Solution Option) Click to Edit Models button opens the Stackup Editor form which lets you store substrate geometry and material properties for a stackup instance. For more details, refer to [Create Stackup](#).

Related topics

[Adding Instances](#)

Add Kanji Note

Use this form to generate a new kanji note symbol based on an existing Japanese EUC packed (multiple-width) formatted file. It is accessed using the *Create – Note – Kanji Note* command.

Kanji EUC File specifies the path for an existing EUC-packed formatted file. Leave this field blank if you are creating a new file.

File Editor Open starts the kanji text editor and creates a new EUC packed formatted file. The software automatically names the file by adding a string (`_kanji_`) and a unique number to the current schematic name and by adding the .doc suffix. The software displays the new filename in the *Kanji EUC File* field.

Library Name specifies the library where you want the software to create the new kanji symbol. When you start the kanji text editor, the software automatically places the name of the current library in this field.

Cell Name specifies the cell where you want the software to create the new kanji symbol. When you start the kanji text editor, the software automatically generates a default cellname by appending the current schematic cellname with a string (`_kanji_`) and a unique number.

The software stores the following display parameter settings as cell-level CDF parameters.

Scale Factor specifies a floating-point number for the character size. A value of 1.0 represents a character that is 50 database units high and 50 database units wide, which is the resolution used in the vector files. The default is 0.5.

Note: Because of roundoff errors, a scale factor of .25 or less can make many characters unreadable.

Horizontal Spacing specifies the prescaled *X* distance between adjacent characters. The default value is 20 database units.

Text Direction specifies how the kanji text editor writes the text in the schematic.

horizontal causes the editor to write the text from left to right.

vertical causes the editor to write the text from top to bottom.

Vertical Spacing specifies the prescaled *Y* distance between adjacent characters. The default value is 20 database units.

Virtuoso Schematic Editor User Guide

Form Descriptions

Related topics

[Configuring and Using Kanji Notes](#)

Add New Property for Search

Use this form to add a property name to the search list. It is accessed by either the Schematic Select By Property or the Symbol Select By Property forms and selecting *other* in the *Find* cyclic field.

Property Name specifies a name for the property you want to add to the list of property names on the Schematic Select By Property form.

Related topics

[Adding a Property Name to the Search List](#)

Add New Property to Find

Use this form to add a new property to search for in the Schematic Find form. It is accessed from the [Schematic Find](#) form by selecting `other` from the *Search For* field.

Property Name specifies a name of the property you want to add to the property name search list on the Schematic Find or Symbol Find form.

Related topics

[Searching for Objects That Have a Specific Property](#)

Add New Property to Find/Replace

Use this form to add a new property to find/replace in the Schematic Find form. It is accessed from the [Schematic Find](#) form by selecting `other` from the *Search For* field.

Property Name specifies a name of the property you want to add to the property name search list on the Schematic Replace or Symbol Replace form.

Related topics

[Replacing Properties](#)

Add Property – Edit Object Properties

Use this form to add a new property. It is accessed by selecting *Show* option *user* and clicking *Add* on the Edit Cellview Properties form.

Name specifies a name for the new property.

Type specifies a property type.

Value specifies a value for the property.

Note: To define most properties, you need to specify only the name, type, and value. However, when you specify *string*, *int*, or *float* as the type, one of the following fields appears on the form.

Choices appears when *Type* is set to *string*. If the string can have more than one value, type the possible values. Entries in this field must match an entry in the *Value* field.

Min and **Max** appear when *Type* is set to *int* or *float*. Specify a range for the minimum and maximum range for the integer or floating-point value.

Related topics

[Adding Cellview Properties](#)

Add Selection Box

Use this form to specify whether to draw a symbol selection box manually or automatically. It is accessed from the symbol editor using the *Create – Selection Box* command.

Use your mouse to indicate two opposite vertices of a rectangle that contains the objects you want to select, or click *Automatic*.

Automatic adds a selection box that contains all the objects that make up the symbol.

Related topics

[Adding a Selection Box](#)

Add Symbol Label

Use this form to create a label for a symbol. It is accessed from the symbol editor using the *Create – Label* command.

Label specifies the label name. The system fills in this field automatically, depending on the option you select from the *Label Choice* cyclic field.

Font Height specifies the height of the text in user units. Default font height is 0.0625.

Environment variable: symbolLabelFontHeight

Label Choice contains a list of valid label types.

instance label is an NLPLabel used to display an instance name.

device annotate is an NLPLabel used to annotate the reference designator information from a package in the PCB layout onto the schematic. Historical.

logical label is an NLPLabel used to display the part name for PCB layout. Historical.

physical label is an NLPLabel used to display the userPartName or phyPartName property values for PCB layout. Historical.

pin name is a normalLabel used to display a specified pin name.

pin annotate is an NLPLabel used to annotate the corresponding pin number from a package in the PCB layout onto the schematic. Historical.

analog instance label is an ILLabel used to display various instance name/type information. The display can be configured using the *Edit – Component Display* command, which shows the label name as cdsName() where you type a name between the parentheses. The cdsName SKILL function is internally configured and evaluated by the schematic editor and is not intended for direct evaluation by the user. It displays label information (usually placed on the layer annotate drawing7) about the cell name or instance name. The cdsName function is attached to a cell symbol view when you use the Add Symbol Label form. The Annotation Setup form is used to control the display of this label type. The CDF *Interpreted Labels Information* sections of both the library and component are also used to configure what this label displays. The pertinent *Interpreted Labels Information* parameters are instDisplayMode and instNameType.

analog pin annotate is an ILLabel used to display various information about a specified pin or terminal. The default pin name display is set up in the CDF. You can change display options using the *Edit – Component Display* command, which shows the label name as cdsTerm(""), where the terminal name is enclosed with quotation marks. The cdsTerm

Virtuoso Schematic Editor User Guide

Form Descriptions

SKILL function is internally configured and evaluated by the schematic editor and is not intended for direct evaluation by the user. It displays label information (usually placed on the layer annotate drawing) about the pin or net attached to the pin. The cdsTerm function is attached to a cell symbol view when you use the Add Symbol Label form. The Annotation Setup form is used to control the display of this label type. The CDF *Interpreted Labels Information* sections of both the library and component are also used to configure what this label displays. The pertinent *Interpreted Labels Information* parameters are termDisplayMode, termSimType, and netNameType.

analog device annotate is an ILLabel used to annotate various device or simulation-specific data onto the schematic. This lets you display component/model/operating point data which you can configure using the *Edit – Component Display* command, which shows the label name as cdsParam(1), numbered sequentially starting with 1. The specific data for each type that can be annotated is specified in the cell CDF. Multiple device annotation parameters should be entered with increasing numbers between the parentheses to indicate which parameter in the list specified in the CDF to display. The cdsParam function is internally configured and evaluated by the schematic editor and is not intended for direct evaluation by the user. It displays label information (usually placed on the layer annotate drawing) about the parameter values or backannotated parameter values. The cdsParam function is attached to a cell symbol view when you use the Add Symbol Label form. The Annotation Setup form is used to control the display of this label type. The CDF *Interpreted Labels Information* sections of both the library and component are also used to configure what this label displays. The pertinent *Interpreted Labels Information* parameters are paramDisplayMode, paramLabelSet, opPointLabelSet, modelLabelSet, paramEvaluate, and paramSimType.

Environment variable: symbolLabelChoiceText

Font Style specifies the font the system uses to display the label.

Environment variable: symbolLabelFontStyle

Label Type specifies how the system displays the label in the design.

normalLabel displays the label as a text string.

NLPLabel interprets the label according to netlisting rules and displays the result.

ILLabel interprets the label as a SKILL expression and displays the result.

Environment variable: symbolLabelType

Justification specifies the position of the cursor with respect to the text.

Environment variable: symbolLabelJustify

Virtuoso Schematic Editor User Guide

Form Descriptions

Rotate turns the note text counterclockwise in 90-degree increments around the reference point.

Related topics

[Adding Labels](#)

Add Symbol Shape

Use this form to create your own symbols. It is accessed from the symbol editor using the *Create – Shape* command.

Note: The Add Symbol Shape form expands and contracts depending on which *Shape* option you select as the options available vary depending on the shape currently selected.

Shape specifies a shape you can add to your design. You can select Line, Rectangle, Polygon, Circle, Ellipse or Arc.

Environment variable: symbolShape

Fill Style determines whether the shape is filled or not.

outline draws just the border.

filled fills in the specified shape.

Environment variable: symbolShapeFillStyle

Line Width specifies the width of the line you want to add.

narrow draws a line using the default setting of 0.0625 user units.

wide lets you edit the default width.

Environment variable: shapeLineWidth

Environment variable: shapeLineType

Draw Mode specifies the line entry method.

 draws a straight line in any angle between two points.

 begins an L90 shape with a horizontal line segment and alternates between horizontal and vertical.

 begins an L90 shape with a vertical line segment and alternates between vertical and horizontal.

 draws either a horizontal or vertical line segment first, then a 45-degree angle.

 draws a 45-degree angle first, then either a horizontal or vertical line segment.

Environment variable: drawMode

Lock Angle appears only when *Draw Mode* is set to  .

Virtuoso Schematic Editor User Guide

Form Descriptions

any draws a line using any angle.

45 restricts cursor movement to 45-degree angles from the starting point.

Related topics:

[Creating New Symbols](#)

Add Wire

Use this form to change the characteristics of wires. It is accessed using *Create – Wire (narrow)* or *Create – Wire (wide)*.

Draw Mode specifies the wire entry method.

route (default) creates a route between two points.

 draws a straight line in any angle between two points.

 begins an L90 shape with a horizontal line segment and alternates between horizontal and vertical.

 begins an L90 shape with a vertical line segment and alternates between vertical and horizontal.

 draws either a horizontal or vertical line segment first, then a 45-degree angle.

 draws a 45-degree angle first, then either a horizontal or vertical line segment.

Environment Variable: drawModeRoute

Route Method specifies how the system routes wires. This option appears when you set *Draw Mode* to *route*.

full draws complex lines consisting of right angles and up to three segments to complete the connection wire.

direct draws a straight line between two points.

flight draws a dashed, straight line between two points.

Environment Variable: defWideWireWidth

Lock Angle appears when *Draw Mode* is set to . Use this option to control the cursor movement.

any (default) lets you drag your cursor along any angle.

45 limits cursor movement to 45-degree angles.

Width specifies the wire width in user units. The default values are 0 (zero) for a narrow wire and 0.0625 for a wide wire.

Environment Variable: defWideWireWidth

Color specifies the color of the wire.

Line Style specifies the line style of the wire.

Related topics

[Adding Wires](#)

Annotation Setup

Use this form to change the annotation settings of the design components. It can be accessed using the *View – Annotations – Setup* command.

Simulation Data Directory sets the simulation data directory for annotation.

Library lists all the libraries in the design.

Cell lists all the cells within the selected library in the design.

Instance lists all the instances in the selected cell in the design.

Selected List lists all the instance names of the type selected from the Instance drop-down list.

Label displays the type of ilLabel present in the design for the selected instance.

Display Mode displays the available display modes for the ilLabels.

You can select the operating point parameter, component parameter, model parameter, instance name, cell name, pin name, net name, voltage, and current options from the drop-down list depending upon the selected ilLabel.

Expression lists the expressions for the parameter labels.

Annotate displays or hide annotations on the schematic.

Balloons displays or hide annotations on annotation balloons.

Related topics

[Using Annotation Setup Form](#)

Build Inherited View List

Use this form to select the appropriate views and constants to add to the *Inherited View List* list box. It is accessed by switching the *As Is* button off in the *Inherited View List* field on the [Set Instance Binding](#) form.

Available Constants/Views provides a selection list of available constants and views. The list is made up of constants that have been defined for this design and the union of all views of the selected instances.

Arrow keys move the selected elements into and out of the *Inherited View List* box.

Inherited View List specifies the contents of the new *Inherited View List* box that you are creating.

Add lets you type in other views to add.

Up moves the selected view in the *Inherited View List* box up one position in the list.

Down moves the selected view in the *Inherited View List* box down one position in the list.

Related topics

[Building a New Inherited View List](#)

Cellview From Cellview

Use this form to create a new cellview from an existing cellview. It is accessed using the *Create – Cellview – From Cellview* command.

Library Name specifies the library that contains the source cellview. The default library name is the current library.

Cell Name specifies the cell name. The default cell name is the current cell.

From View Name contains a list of the existing views for the source cellview.

To View Name specifies the view name of the cellview the software generates when you click *OK* or *Apply*. The *Tool / Data Type* setting initializes this field. You can edit this field.

Tool / Data Type specifies the type of cellview to be created.

Browse opens the Library Browser. You can use the browser to select the library, cell, and view names. The system automatically fills in the form with the information you specify in the browser.

Display Cellview when enabled, causes the software to open the newly created cellview in an editor window when you complete the cellview generation process.

Environment Variable: cvDisplay

Edit Options when enabled and on completion of the Cellview From Cellview form, displays the Symbol Generation Options subform. Use this form to specify other options.

Environment Variable: createCvOptions

Related topics

Automatically Creating a Cellview from Another Cellview

Cellview From Instance

Use this form to create a cellview from an instance using the command *Create – Cellview – From Instance*.

View Name specifies the type of cellview the software generates when you click *OK*. The *Tool / Data Type* setting initializes this field.

Tool / Data Type specifies the type of cellview to be created.

Display Cellview, when enabled (default), causes the system to open the newly created cellview in an editor window when you complete the cellview generation process.

Related topics

[Automatically Creating a Cellview from an Instance](#)

Cellview From Pin List

Use this form to create a cellview from existing pins, such as those from the *basic* pin library, or you can create the cellview from new pin names. It is accessed using the *Create – Cellview – From Pin List* command.

Input Pins specifies an input pin list either from any library or expressed as a new pin name.

Output Pins specifies an output pin list either from any library or expressed as a new pin name.

IO Pins specifies an input/output pin list from any library or expressed as a new pin name.

Switch Pins specifies a switch level pin list from any library or expressed as a new pin name.

Jumper Pins specifies a jumper pin list from any library or expressed as a new pin name.

Tristate Pins specifies a tristate pin list from any library or expressed as a new pin name.

Unused Pins specifies an unused pin list from any library or expressed as a new pin name.

Library Name specifies the library name for the generated cellview. The default library name is the current library.

Cell Name specifies the cell name for the generated cellview.

View Name specifies the type of cellview the software generates. The *Tool / Data Type* setting initializes this field.

Tool / Data Type specifies the type of cellview to be created; for example, *Schematic Editor – Symbol*.

Browse opens a Library Browser. Use the browser to select the library name, cell name, and view name. The system automatically fills in the form with the information you select in the browser.

Display Cellview, when enabled, causes the software to open the newly created cellview in an editor window when you complete the cellview generation process.

Environment Variable: cvDisplay

Edit Options, when enabled, displays a subform when you complete the Cellview From Pin List form. Use the subform to specify other options.

Environment Variable: createCvOptions

Virtuoso Schematic Editor User Guide

Form Descriptions

Related topics

[Automatically Creating a Cellview from a Pin List](#)

Cellviews with Evaluated Net Expression Name

Use this form to view a full list of cellviews that contain net expressions that evaluate to the selected evaluated name. It is accessed using the *Edit – Net Expression – Evaluated Names* command and selecting *List Cellviews with Selected Evaluated Name*.

Cellview shows the full list of cellviews that contain net expressions that evaluate to the selected evaluated name.

Property Name shows the property name in the list of cellviews.

Default Value shows the default value (usually a global name) in the list of cellviews.

Number of Occurrences shows the number of occurrence paths down the hierarchy to a cellview which has a net expression that evaluates to the specified name.

Related topics

[Viewing the Evaluated Net Names](#)

Change Sheet Border Size

Use this form to change the existing size and type of a schematic border, using the *Edit – Sheet Size* command.

Border Size specifies the new sheet size to use in the schematic.

A changes the sheet to an A-size sheet with a vertical orientation, as it would appear in a book.

none deletes an existing sheet border; otherwise, this option does not affect the sheet. When you select this option, the system ignores the *Border Type* field.

Border Type specifies the sheet border type.

basic (default) changes the sheet to include global and sheet-specific border titles in the title block.

continue changes the sheet to include sheet-specific border titles in the title block.

Related topics

[Editing the Sheet Border Size and Type](#)

Check Hierarchy

Use this form to define the options to be run when you choose the *Check – Hierarchy* command.

Process specifies which schematics the system checks.

every schematic checks all schematics in the hierarchy.

only those that need it (default) checks only those schematics that have been updated since the last time you saved your design.

Environment variable: checkAlways

Referenced libraries specifies whether the hierarchy check continues outside of the current library.

check activates the text field so you can edit the list. When you click *Apply* or *OK*, the system checks the libraries displayed in the field.

do not check (default) checks only the current library; the libraries that appear in the text field are not checked.

Environment variables: libList, spanLib

Open for specifies the mode in which the cellview is opened for the hierarchy check.

auto (default) checks the hierarchy in the same mode as the top cellview.

edit checks the hierarchy in editable mode.

read checks the hierarchy in read-only mode.

Environment variable: checkHierOpenMode

Save schematics specifies whether the system saves schematics when the check operation is complete.

yes (default) saves the checked schematics.

no does not save the checked schematics.

those with errors appears only when you select *yes*. This option specifies how the system handles schematics that contain errors.

Ask Me (default) prompts you to confirm which of the displayed schematics you want to save.

Virtuoso Schematic Editor User Guide

Form Descriptions

No Save does not save any schematics that contain errors.

Save saves all schematics even when they contain errors.

Environment variable: [checkHierSave](#)

View Name List displays a scrollable list of views that is used to traverse the hierarchy.

When the schematic is in the context of a design configuration or a hierarchy configuration, the following two fields appear:

Using configuration displays the configuration name using the format *lib cell view*.

Starting from displays the name of the top level cellview using the format *lib cell view*.

Related topics

[Checking a Design Hierarchy](#)

Check Label Attachment

Use this form to highlight the association between a label and its owning object (that is, a wire or an instance). It is accessed using the *Check – Label Attachment* command.

Show labels of controls which labels that you want to check as being attached to particular objects on the canvas.

wire displays wire label attachments on the canvas.

instance displays instance label attachments on the canvas.

both displays both wire and label attachments on the canvas.

none hides all label attachments on the canvas.

Close closes the Check Label Attachment form and automatically removes the label attachments. This action will also reset the selection to *none*.

Refresh updates the label display on the canvas. This can useful, for example, if objects have been moved or deleted while the Check Label Attachment form is still on display.

Related topics

[Checking a Label Attachment](#)

Component Browser

Use this form to add component instances from a library. If the *Add Instance Browser Type* option on the Editor Options form, is set to *component*, this form is accessed using the *Create – Instance* command.

Library specifies the library that contains the component you want to add.

Flatten controls whether the list box displays cell categories (default; turned off) or cells (turned on) contained in the specified library. The display changes when you toggle this option.

Filter specifies a text string that limits the list box display to categories or cells that contain the text string. The display changes when you click the *Library* option.

Remove appears only when you use the *Commands – Add Library* command to expand the form. This option lets you remove the associated library from the form.

The list box displays the categories or cells contained in the library (see *Flatten*).

Related topics

[The Component Browser](#)

Component Browser Preferences

Use this form to specify your preference on whether the Component Browser or Add Instance form is opened when the *Create – Instance* command is run. This form is accessed using the *Commands – Preferences* command.

Browser Opens with the Invocation of when turned on (default), the system opens the Component Browser when you click the *Browse* button on the Add Instance form.

Status of Browser After Closing defines what happens to the Component Browser when you close the Add Instance form.

icon (default) collapses the window into an icon.

open leaves the Component Browser window open.

closed closes the Component Browser window.

Related topics

[The Component Browser](#)

Copy

Use this form to copy an object. It is accessed using the *Edit – Copy* command.

Snap Mode specifies the angles along which you can move copied objects within the cellview.

anyAngle (default) lets you move the copied objects along any angle.

diagonal restricts the movement of copied objects to points along 45-degree, 90-degree, and 180-degree angles from the current location.

orthogonal restricts the movement of copied objects to points along the X and Y axes from the current location.

Array lets you create multiple copies of a selected object in a single operation and specify the dimensions of the array.

Rows specifies the number of rows.

Columns specifies the number of columns.

Include Design Intent lets you specify whether the design intent associated with the object is also included in the copy.

Environment variable: [copyDesignIntent](#)

Rotate turns an object in 90-degree increments counterclockwise around the current reference point.

Sideways mirrors the object across an imaginary vertical line passing through the current reference point.

Upside Down mirrors the object across an imaginary horizontal line passing through the current reference point.

Related topics

[Copying](#)

Create a Mapping Schematic Step 1 of 2

Use this form to create a mapping schematic so you can switch between design tasks without the need to modify existing cells or the design text file. It is accessed using the command *Create – Mapping Schematic*.

Lower Level Cellview Specifies the lower-level library, cell and view names.

Referenced Symbol:

Library Specifies the library that contains the cell and the view that are considered as the lower-level cellview.

Cell Specifies the cell that contains the lower-level cellview.

View Specifies the lower-level cellview. By default, you can only specify the `schematicSymbol` views in this form. However, you can set the `browserShowSchViews` to `t` for specifying the `schematicSymbol` and `schematic` views in the *Lower Level Cellview* and *Upper Level Cellview*.

Upper Level Cellview Specifies the pins of the lower-level cellview, that are to be connected to the pins in the upper-level cellview.

Referenced Symbol:

Library Specifies the library that contains the cell and the view that are to be considered as the upper-level cellview.

Note: The upper-level cellview settings default to the current library when the *Create – Mapping Schematic* option was selected.

Cell Specifies the cell that contains the upper-level cellview.

View Specifies the upper-level cellview.

Mapping Schematic Specifies the mapping schematic view name.

View Name Specifies the mapping schematic view name. The default view name is of the form `<lowerCell_cellName>`. This can be changed as required.

Environment variable: `mapCvMappingSchViewName`

Next Displays the *Create a Mapping Schematic Step 2 of 2* form. This button becomes available only after you have specified valid library, cell, and view names for both the lower- and upper-level cellview references.

Virtuoso Schematic Editor User Guide

Form Descriptions

Related topics

[Creating a Mapping Schematic](#)

Create a Mapping Schematic Step 2 of 2

Use this form to map the unmapped nets in the *Lower Level Cellview* (specified in [Step 1](#)) to the unmapped nets in the *Upper Level Cellview* (also specified in step 1). It is accessed by clicking *Next* on the [Create a Mapping Schematic Step 1 of 2](#) form.

Unmapped Nets Lists the current unmapped nets in each of the lower- and upper-level cellviews.

Mapped Nets Lists those nets that are mapped between the lower and upper-level cellviews.

Auto-Map Automatically maps the nets listed under *Unmapped Nets*, displaying the newly mapped nets under *Mapped Nets*.

Map Maps the lower- and upper-cellview nets that are currently selected under *Unmapped Nets*. This button is available only when two nets that can be mapped are selected.

Clear De-selects any maps that are currently selected under *Unmapped Nets*.

Other Setup:

Specific options in the *Other Setup* group box become available when a pin in either of the two symbol views is selected. Moreover, the options become available based on the type of pins that are currently selected.

Note: If pins in both the lower- and upper-symbol views are currently selected then all options are disabled.

Unconnected If selected, this option leaves the selected pin unconnected when *Add* is performed. This ensures that the pin is connected to a `noConn` instance.

Note: If an iterated pin is connected to a `noConn` symbol, then the `noConn` instance that is connected to that iterated pin has the same width as the iterated pin.

Connected To If selected, connects the selected pin to the specified net name.

Note: If an illegal net name has been entered, a warning message will be displayed when *Add* is subsequently selected. Additionally, if you select *Add* while this field is blank, but selected, no action will occur.

Net Expression If selected, connects the selected lower-level symbol pin to a wire with a net expression that has the specified *Property Name* and *Default Global Net*, as explained below.

Note: This option is only active when a single-bit pin is selected in the lower level symbol.

Virtuoso Schematic Editor User Guide

Form Descriptions

Property Name Specifies the net expression property name.

Default Global Net Specifies the net expression default global net name.

Net Set If selected, the lower-level instance has a `netSet` property associated with it, with the overridden net being specified in the *Override Net* field.

Note: This option is active only when a pin with a net expression is selected in the lower-level symbol.

Add Maps nets based on the specifications listed above and adds them to the *Mapped Nets* list.

Unmap Removes the mapping between the nets selected in the *Mapped Nets* list, and then displays the unmapped nets under *Unmapped Nets*.

File Specifies the location of the text file that contains the mapped net settings. You can do both, load the settings from a file or save the settings to a file.

Clicking the browse button locates the file from which you want to load the settings. The files are listed in the Choose a File form.

Load From Loads mapped net settings from the text file listed in the *File* field.

Save To Saves mapped net settings to the text file listed in the *File* field.

Related topics

[Creating a Mapping Schematic](#)

Create Block

Use this form to create multiple blocks, each with a different *lib/cell/view* name. It is accessed using the *Create – Block* command.

Library specifies the library in which you want to add a new block. The default is the library for the current design.

Cells specifies the cell name of the block you want to create. You can specify multiple cell names by using a space to separate each name. The *Cells* and *Names* fields must match in number. If there are more entries in *Cells* than in *Names*, the system assigns a name for each. If there are more entries in *Names* than in *Cells*, the system ignores the extra names.

View specifies the view you want to create. The default is *symbol*.

Names specifies the name of the cell. Only one name (instance) can be entered for each cell. There is a one-to-one correspondence between the entries in the *Cells* and the *Names* fields. You can specify multiple cell names by using a space to separate each name. The *Cells* and *Names* fields must match in number. If there are more entries in *Cells* than in *Names*, the system assigns a name for each. If there are more entries in *Names* than in *Cells*, the system ignores the extra names.

Pin Name Prefix specifies a name that the system uses to name all the pins that the system creates automatically when you connect wires to this block. The default is *pin*.

Environment variable: [pinNameSeed](#)

Block Shape lists a drawing option and a predefined set of presized blocks that you can add to the design.

freeform prompts you to drag your cursor between two points to specify the opposite corners of a rectangle. The system draws the rectangle as you drag the cursor.

The default set of presized blocks includes *small*, *medium*, *large*, *2 by 1*, *1 by 2*, *alu*, *mux4*, and *mux8*.

Related topics

[Adding Blocks](#)

Create Cellview Dialog Box – Replace or Modify

Use this dialog box to replace or modify the existing cellview. It is displayed when creating a cellview using the Cellview From Cellview form if the symbol cellview to be created already exists.

Replace does the following to a cellview:

- Discards any modifications you may have made to that cellview
- Creates only the pins, properties, selection box, labels, and other graphics; all schematic instances, schematic wires, and HDL body text are lost
- May alter the position of pins in the new symbol cellview

Modify does the following to a symbol cellview:

- Automatically edits the cellview with a minimal amount of changes to the current pin locations, graphics, and properties within the symbol cellview
- Deletes pins that are no longer needed
- Adds pins that do not yet exist in the new cellview

Pins are added near the bottom of the cellview. Use the symbol editor to reposition these new pins and to adjust any symbol graphics. The system determines attributes of the new pins by the currently active TSG template file.
- Adds or updates cellview properties and terminal properties
- Annotates the changes that were made in note format at the bottom of the cellview and in the Command Interpreter Window (CIW)
- Does not delete cellview or terminal properties

Related topics

[Automatically Creating Cellviews](#)

Create Net Expression

Use this form to add a net expression label to a wire. It is accessed using the *Create – Net Expression* command.

Property Name specifies the name of the property that can be used to override the default global signal; in other words, the name of the property whose value redefines the name of the net. The system evaluating the expression searches for the property that has this name to get the value to inherit for the net.

Default Net Name specifies the global signal name to be used for the net associated with this net expression label. The system uses the default net name—for example vdd!, gnd!, or vss!—when a netSet property has not been found on an instance of the current hierarchical path. The name must specify a global single-bit value.

Derive from property name check box enables, when selected, the *Default Net Name* field to be automatically seeded based on the *Property Name* specified. By default, *Derive from property name* is deselected.

Font specifies the appearance of the net expression label text in the design.

Height specifies the height of the net expression label text in user units. The default font height is 0.0625.

Style specifies the font used to display the net expression label.

Justification specifies the position of the cursor with respect to the text.

Entry Style specifies the mechanism for placing the net expression label on a wire or pin.

fixed offset automatically positions the net expression label at a specified distance from the wire or pin.

manual requires you to place the net expression label at a grid location on or near the wire or pin.

Horizontal Objects specifies position, offset, and rotation angle for the net expression labels in horizontal objects.

Vertical Objects specifies position, offset, and rotation angle for the net expression labels in vertical objects.

Rotation turns the label text counterclockwise in 90-degree increments around the reference point.

Note: You can edit a net expression label in the Edit Object Properties – Net Expression (*Edit – Properties – Objects*).

Virtuoso Schematic Editor User Guide

Form Descriptions

Related topics

[Adding Net Expression Labels to Create an Inherited Connection](#)

Create Note Shape

Use this form to add an annotation within a defined shape on your design. It is accessed using the *Create – Note – Shape* command.

Shape specifies a shape you can add to your design to outline the note text. The shape options available are:

- ☐ line
- ☐ rectangle
- ☐ polygon
- ☐ arc
- ☐ ellipse
- ☐ circle

Environment variable: noteShape

The options available depend on the *Shape* option you select and the form expands and contracts to accommodate this.

Dashed line check box specifies the line style used to draw the shape. When selected, a dashed line replaces the default solid line.

Environment variable: noteShapeLineStyle

Wide line check box specifies if a line setting wider than the default 0.0625 user units is to be used. This check box is displayed only when the *Shape* is set to `line`. When selected, the *Width* option is enabled for you to specify the actual width of line to be used.

Environment variable: shapeLineWidth

Width specifies the width of the line used to draw the shape and is displayed only when the *Shape* is set to `line` and the *Wide line* check box has been selected. Enter or select the width you require.

Draw Mode appears only when the *Shape* is set to `line` and `polygon`. It specifies the line entry method.

 draws a straight line in any angle between two points.

 draws two line shapes. The first is horizontal; the second is vertical.

 draws two line shapes. The first is vertical; the second is horizontal.

Virtuoso Schematic Editor User Guide

Form Descriptions

 draws either a horizontal or vertical line segment first, then a 45-degree angle.

 draws a 45-degree angle first, then either a horizontal or vertical line segment.

Lock to 45 degrees appears only when *Draw Mode* is set to . It toggles between drawing a wire using any angle (default) and restricting cursor movement to 45-degree angles from the starting point.

Environment variable: drawMode

Text specifies the text to be displayed with the shape. You can either type the text or use the *File* option to load the text from a file. You can edit the text in this window, but the *Font* options specified will only be visible when the Note Shape is created on the design.

Note: See Default Keybindings for Text Fields in the *Virtuoso Design Environment User Guide*.

File copies in a file and displays the file contents in the *Text* field. You can either enter the filename or use the *Browse (...)* button to display the *Choose a File* form, where you can select the file required. This action overwrites any text you entered in the text box. To read in the file, specify the path to the text file, click *Read File*. When the file contains characters that cannot be displayed, the system issues a warning and removes the characters before copying in the file.

Font specifies the appearance of the note text in the design. Any changes will only be visible when the Note Shape text is created on the design:

Height specifies the height of the text in user units. The default font height is 0.0625.

Style specifies the font used to display text.

Justification specifies the position of the cursor with respect to the text.

Overbar check box is a display option. You use this option to specify whether the software displays an overbar over the note text. When this option is disabled (default), the software displays underscore characters (`_`) in the note text as part of the note text string. When this option is enabled, the software interprets underscore characters (`_`) in the note text as toggle switches that control where overbars begin and end. Overbars appear above the property label, as shown in the examples.

- ☐ The note text `_abcde` puts an overbar over all 5 letters.
- ☐ The note text `ab_cde` puts an overbar over only cde.
- ☐ The note text `_abc_de` puts an overbar over only abc.
- ☐ The note text `ab_cd_ef_gh_ij` puts an overbar over only cd and gh.

Virtuoso Schematic Editor User Guide

Form Descriptions



To avoid confusion when text, labels, or names contain underscore characters, Cadence does not recommend that you enable the Overbar display option. It is provided only for backward compatibility with prior releases.

Related topics

[Adding Note Shapes](#)

Create Note Text

Use this form to add an annotation on your design. It is accessed using the *Create – Note – Text* command.

Text specifies the text to be displayed with the note. You can either type the text or use the *File* option to load the text from a file. You can edit the text in this window, but the *Font* options specified will only be visible when the note text is created on the design.

Note: See [Default Keybindings for Text Fields](#) in the *Virtuoso Design Environment User Guide*.

File copies in a file and displays the file contents in the *Text* field. You can either enter the filename or use the *Browse (...)* button to display the *Choose a File* form, where you can select the file required. This action overwrites any text you entered in the text box. To read in the file, specify the path to the text file, click *Read File*. When the file contains characters that cannot be displayed, the system issues a warning and removes the characters before copying in the file.

Font specifies the appearance of the note text in the design. Any changes will only be visible when the text is created on the design:

Height specifies the height of the text in user units. The default font height is 0.0625.

Environment variable: [noteFontHeight](#)

Style specifies the font used to display the text.

Environment variable: [noteFontStyle](#)

Justification specifies the position of the cursor with respect to the text.

Environment variable: [noteJustify](#)

Drafting maintains the normal orientation of the text as you flip a note 180-degrees around a reference point. When this option is disabled, rotated text appears upside down. The default is for this option to be *on*.

Overbar check box is a display option. You use this option to specify whether the software displays an overbar over the note text. When this option is disabled (default), the software displays underscore characters (`_`) in the note text as part of the note text string. When this option is enabled, the software interprets underscore characters (`_`) in the note text as toggle switches that control where overbars begin and end. Overbars appear above the property label, as shown in the examples.

- ☐ The note text `_abcde` puts an overbar over all 5 letters.

Virtuoso Schematic Editor User Guide

Form Descriptions

- ❑ The note text ab_cde puts an overbar over only cde.
- ❑ The note text _abc_de puts an overbar over only abc.
- ❑ The note text ab_cd_ef_gh_ij puts an overbar over only cd and gh.



To avoid confusion when text, labels, or names contain underscore characters, Cadence does not recommend that you enable the Overbar display option. It is provided only for backward compatibility with prior releases.

Rotation turns the note in 90-degree or 180-degree increments around the reference point.

Rotate Left turns the note 90-degrees anti-clockwise.

Rotate Right turns the note 90-degrees clockwise.

Flip Vertical flips the note vertically 180-degrees.

Flip Horizontal flips the note horizontally 180-degrees.

Related topics

[Adding Note Text](#)

Create Patchcord

Use this form to add a patch cord which establishes aliases between the signals of two different nets. It is accessed using the *Create – Patchcord* command.

Add net stubs automatically adds two net stubs when adding a patch cord symbol. When disabled, a simple patch cord instance will be created.

Indexing

ascending (only enabled) allows for a single prefix to be entered, and the alias net will be set up as a bus which depends upon the number of nets entered in the *Design Nets* field. The *preview* option to the right of the *Alias Nets/Prefix* option provides an example of what this bus will appear as.

auto and ascending (both enabled) instructs that the alias bus will display as `alias<0:3>`, assuming there are four design nets, otherwise it will display as `alias<3:0>`.

Alias Nets/Prefix specifies a single string (if *auto* indexing is enabled) that will provide the name to be used for the alias net or bus. If *auto* indexing is disabled, then this can be one or more, comma separated, net names. The number of net names must however match the number of nets entered in the *Design Nets* field.

Design Nets displays a list of comma separated net or bus names, for example “in, x, cntrl, data<0:3>”.

Style specifies the format of the patch cord to be placed. You can select between *straight*, which displays a horizontal layout, with both nets and patch cords in a straight line, or *curved*, where the layout is a horseshoe shape, with a vertical patch cord and horizontal nets. The *Preview* field, below the *Style* field, graphically illustrates your current selection.

Bundle Display lets you select the bundle display mode, *horizontal* or *vertical*, for the *Design Nets* if the *Add net stubs* option is enabled and the design net being placed is a bundle. The *Bundle Display* option will be disabled if the *Design Nets* option is empty or if the design net is not a bundle.

A bundle is a wire or pin name consisting of simple names separated by a comma; for example, S,R.

Related topics

[Adding Patch Cords Using the Create Patchcord Form](#)

Create Pin – Schematic

Use this form to create a schematic pin or an offsheet pin. It is accessed using the *Create – Pin* command.

Names specifies the names for the pins you want to add to the design and the order in which you want to place them. To add more than one pin name, use a space to separate each name. When you enable the Expand Busses option, you can use a bus expression to name an array of pins. You need not specify the name if you have enabled automatic pin naming by using the createPinAutoNameFromNamedNet environment variable.

Note: If setting supply and ground sensitivity, the pin name should be a single-bit terminal or single-bit terminal with vector elements. If the name is not a single-bit, a signal type can only be added if the *Expand Busses* option is also selected which will apply to the array of pins.

Direction contains a list of the directions you can specify for the pins you want to add. *switch* is maintained solely for backward compatibility: Cadence discourages its use. The default is *input*.

Usage contains a list of the pin types you can specify for the pins you want to add. The default is *schematic*. Use *offsheet* only when you are going to connect the added pins to another sheet in a multisheet schematic.

Signal Type specifies the new pin's signal type. Signal type (sigType) setting is important in the areas of design verification, consistency checking for mixed-signal designs, and to also set routing constraints. Signal types to select from are *analog*, *clock*, *ground*, *power*, *reset*, *scan*, *signal* (default), *tieHi*, *tieLo*, *tieOff*, *optical*, *singleModeOptical*, or *multiModeOptical*. Refer to Defining Signal Types.

Expand busses check box determines how the schematic editor interprets bus names.

When deselected (default) a bus name is interpreted as a single-bit name and a single pin that includes all the bits in the bus range is generated. For example, for the name A<0:1>, a pin called A<0:1> is generated.

When selected a bus name is interpreted as multiple-bit name and a pin is generated for each bit of a bus range that you specify in the *Names* field. For example, for the pin A<0:1>, pins called A<0> and A<1> are generated.

Note: This option must be selected when using multiple-bit names and signal sensitivity is required. For more details, see How to Set Supply and Ground Sensitivity.

Place multiple pins check box determines whether you can place multiple pins automatically in one operation.

Virtuoso Schematic Editor User Guide

Form Descriptions

When deselected (default) requires you to point and click to identify the location for each pin.

When selected prompts you to point and click to specify the first and second pin locations. The system uses the distance between these two points as a basis for automatically placing any remaining pins.

Net Expression section

Attach to pin check box allows you to add a net expression to the pin. The *Property Name* and *Default Net Name* fields will be automatically seeded, based on the current *Names*, if the *Derive from pin name* option is enabled.

Derive from pin name check box enables the *Property Name* and *Default Net Name* fields on the Create Pin form to be automatically seeded with values derived from the *Names* currently being added. These values will continue to update if multiple pins are added simultaneously.

Property Name specifies the name of the netSet property that can be used to override the default global signal; in other words, the name of the property whose value redefines the name of the net. The system evaluating the expression searches for the property that has this name to get the value to inherit for the net.

Default Net Name specifies the global signal name to be used for the net associated with this label. The system uses the default net name—for example vdd!, gnd!, or vss!—when a netSet property has not been found on an instance of the current hierarchical path. The name must specify a global single-bit value.

Font section

Height specifies the height of the label text in user units.

Style specifies the font used to display the label.

Justification specifies the alignment of the label. Alignments to select from are *automatic*, *lowerLeft*, *centerLeft*, *upperLeft*, *lowerCenter*, *centerCenter*, *upperCenter*, *lowerRight*, *centerRight*, or *upperRight*.

Supply Sensitivity section determines how the values entered for power and ground terminals gets attached to the specified pins' terminals.

Attach To drop-down list box lists the options to choose from while making the terminals of the next pin sensitive to the power or ground terminals. For more details, see [Setting Supply and Ground Sensitivity \(Schematic Window\)](#).

Both the terminal of the next pin will be made sensitive to any specified power and ground terminals.

Virtuoso Schematic Editor User Guide

Form Descriptions

Power terminal of the next pin will be made sensitive only to any specified power terminals.

Ground terminal of the next pin will be made sensitive only to any specified ground terminals.

None of the terminals of the next pin will be made sensitive to neither power nor ground terminals, specified or otherwise.

Power Pin type in, or select from the drop-down list (which includes all the single-bit terminals in the cellview), the names of the terminals that would make the effected pins' terminals sensitive to the entered power terminals. By default, the field shows *None* before terminals are added to the cellview.

Ground Pin type in, or select from the drop-down list (which includes all the single-bit terminals in the cellview), the names of the terminals that would make the effected pins' terminals sensitive to the entered ground terminals. By default, the field shows *None* before terminals are added to the cellview.

Rotation section

Rotate Left  turns the pin in 90-degree increments counterclockwise around the reference point.

Rotate Right  turns the pin in 90-degree increments clockwise around the reference point.

Flip Horizontal  mirrors the pin across an imaginary vertical line passing through the reference point.

Flip Vertical  mirrors the pin across an imaginary horizontal line passing through the reference point.

Related topics

[Adding Schematic and Offsheet Pins](#)

[Supply and Ground Sensitivity](#)

Create Pin – Symbol

Use this form to create a symbol pin. It is accessed from the symbol editor using the *Create – Pin* command.

Names specifies the names for the pins you want to add to the design and the order in which you want to place them. To add more than one pin name, use a space to separate each name. If you enable the *Expand Busses* option, you can use a bus expression to name an array of pins.

Note: If setting supply and ground sensitivity, the pin name should be a single-bit terminal or single-bit terminal with vector elements. If the name is not a single-bit, a signal type can only be added if the *Expand busses* option is also selected which will apply to the array of pins.

Direction contains a list of pin directions you can specify for the pins you want to add. *switch* is maintained solely for backward compatibility.

Bus Expansion specifies how the system interprets bus names.

off (default) interprets a bus name as one name and generates only one pin that includes all the bits in the bus range. For example, for the name A<0:1>, the system generates only the pin A<0:1>.

on interprets a bus name as multiple names and generates a pin for each bit of a bus range that you specify in the *Pin Names* field. For example, for the pin A<0:1>, the system generates pins A<0> and A<1>.

Type contains a list of the valid pin types.

Environment variable: symbolPinUsage

Signal Type specifies the new pin's signal type. Signal type (sigType) setting is important in the areas of design verification, consistency checking for mixed-signal designs, and to also set routing constraints. Signal types to select from are *analog*, *clock*, *ground*, *power*, *reset*, *scan*, *signal*, *tieHi*, *tieLo*, *tieOff*, *optical*, *singleModeOptical*, or *multiModeOptical*. Refer to Defining Signal Types.

Expand busses check box determines how the system interprets bus names.

When deselected (default) causes the system to interpret a bus name as a single name and generate a single pin that includes all the bits in the bus range. For example, for the name A<0:1>, the system generates only the pin A<0:1>.

When selected causes the system to interpret a bus name as multiple names and generate a pin for each bit of a bus range that you specify in the *Names* field. For example, for the pin A<0:1>, the system generates pins A<0> and A<1>.

Virtuoso Schematic Editor User Guide

Form Descriptions

Note: This option must be selected when using multiple-bit names and signal sensitivity is required. For more details, see [How to Set Supply and Ground Sensitivity](#).

Place multiple pins check box determines whether you can place multiple pins automatically in one operation.

When deselected (default) requires you to point and click to identify the location for each pin.

When selected prompts you to point and click to specify the first and second pin locations. The system uses the distance between these two points as a basis for automatically placing any remaining pins.

Label section

Location specifies where the pin name (pin label) appears in respect to the pin.

Environment variable: [symbolPinLocate](#)

Offset specifies how far, in user units, the pin name (pin label) is from the pin.

Environment variable: [symbolPinIncrCount](#)

Supply Sensitivity section determines how the values entered for power and ground terminals gets attached to the specified pins' terminals.

Attach To drop-down list box lists the options to choose from while making the terminals of the next pin sensitive to the power or ground terminals.

Both the terminal of the next pin will be made sensitive to any specified power and ground terminals.

Power terminal of the next pin will be made sensitive only to any specified power terminals.

Ground terminal of the next pin will be made sensitive only to any specified ground terminals.

None of the terminals of the next pin will be made sensitive to neither power nor ground terminals, specified or otherwise.

Power Pin type in, or select from the drop-down list (which includes all the single-bit terminals in the cellview), the names of the terminals that would make the effected pins' terminals sensitive to the entered power terminals. By default, the field shows *None* before terminals are added to the cellview.

Ground Pin type in, or select from the drop-down list (which includes all the single-bit terminals in the cellview), the names of the terminals that would make the effected pins'

Virtuoso Schematic Editor User Guide

Form Descriptions

terminals sensitive to the entered ground terminals. By default, the field shows *None* before terminals are added to the cellview.

Rotate Left  turns the pin in 90-degree increments counterclockwise around the reference point.

Rotate Right  turns the pin in 90-degree increments clockwise around the reference point.

Flip Horizontal  mirrors the pin across an imaginary vertical line passing through the reference point.

Flip Vertical  mirrors the pin across an imaginary horizontal line passing through the reference point.

Related topics

[Adding Pins as Graphic Images](#)

Create Schematic

Use this form to specify whether a single or multisheet schematic is created. It is displayed when creating a cellview from another cellview and you specify `schematic` in the *Tool / Data Type* field.

Schematic Type determines which type of schematic you create.

single creates a single sheet schematic.

multi-sheet creates a multisheet schematic and opens the first page so you can begin editing.

Environment variable: `transToSheetType`

Size specifies the size of the sheet border for the new schematic. The possible values are *A* through *F*, *A Book*, and *none*.

Type specifies the type of sheet border.

basic specifies global and sheet-specific border titles. This option creates a template only. To fill in the fields in the title box, use the *Edit – Sheet Title* command.

continue specifies a continuation border, which contains only specific border titles such as the sheet size and number.

Related topics

[Automatically Creating a Cellview from Another Cellview](#)

Create Schematic Sheet

Use this form to convert a single-sheet schematic to a multisheet schematic, using the *Create – Multisheet* command.

Number specifies the sheet number that the schematic editor creates when you click *Apply* or *OK*. The default sheet number is the next sheet number in the multisheet sequence. Valid choices are integers greater than zero (0).

Size specifies which sheet border size the editor uses for the new sheet. The valid choices are *A* through *F*, *A Book*, and *none*.

A Book creates an A-size sheet with a vertical orientation, as it would appear in a book.

none creates a new sheet without a sheet border. The system ignores the *Type* option when you select this option.

Type specifies the sheet border type.

basic (default) specifies a template that contains global and sheet-specific border titles. You use the Schematic Title Block Properties form to fill in the fields in the title box.

continue specifies the same sheet size and border titles as those used for the schematic in the current design window.

Related topics

[Converting a Single-Sheet Schematic to a Multisheet Schematic](#)

Create Wire Name

Use this form to add a wire name. It is accessed using the *Create – Wire Name* command.

Names specifies one or more names for wires. You can specify multiple names by separating each name with a space. You can also use a bus expression to name an array of wire names.

Expand bus names specifies how the system interprets bus names.

Deselected (default) interprets the bus name as a single name and generates a single name that includes all the bits in the bus range. For example, for the name A<0:1>, the system generates only the name A<0:1>.

Selected interprets the bus name as multiple names and generates a name for each bit of a bus range that you specify in the *Names* field. For example, for the name A<0:1>, the system generates names A<0> and A<1>.

Attach to multiple wires specifies how the system interprets bus names.

Deselected (default) requires you to point and click to identify each wire name location.

Selected lets you drag your mouse between two points to designate the wires that you want the system to name automatically.

Create net alias labels defines the purpose of the wire name.

Deselected (default) does not specify any net alias labels to the wire name.

Selected assigns the wire name to the corresponding wire and defines the wire name as an alternative name (usually shorter or more descriptive) for the wire.

Displays bundles vertically defines the display of comma separated bundle names.

Deselected (default) displays the bundle net names horizontally separated by a comma. For example,

a, b, c

Selected displays the bundle net names vertically so that each member net is displayed in a different row. For example,

a
b
c

Environment variable: bundleDisplayMode

Font section

Virtuoso Schematic Editor User Guide

Form Descriptions

Font Height specifies the size of your wire name text as a floating-point value, in user units. The default is 0.0625.

Environment variable: [createLabelFontHeight](#)

Font Style specifies the typeface the system uses to display the wire name.

Environment variable: [createLabelFontStyle](#)

Justification specifies the position of the cursor with respect to the wire name.

Environment variables: [createLabelAutoCenterSize](#), [createLabelAutoOffset](#), [createLabelAutoSpacer](#), [createLabelJustify](#), [createLabelAutoJustify](#)

Entry Style section

fixed offset automatically positions the wire name at a specified distance from the wire.

manual requires you to place the wire name at a grid location on or near the wire.

Horizontal Wires specifies position, offset, and rotation angle for the wire name labels in horizontal wires.

Environment variables: [createLabelHorzOffset](#), [createLabelHorzPosition](#), [createLabelHorzRotation](#)

Vertical Wires specifies position, offset, and rotation angle for the wire name labels in horizontal wires.

Environment variables: [createLabelVertOffset](#), [createLabelVertPosition](#), [createLabelVertRotation](#)

Environment variable: [createLabelOffsetMode](#)

Rotation section

Rotate Left  turns the pin in 90-degree increments counterclockwise around the reference point.

Rotate Right  turns the pin in 90-degree increments clockwise around the reference point.

Related topics

[Adding Wire Names](#)

Cross Reference Options

Use this form to display cross-reference annotations to schematic or offsheet pins. It is accessed using the *Sheet – Cross Reference* command.

Generate Cross Reference Report File specifies whether the system generates a cross-reference report file.

off (default) does not generate a cross-reference report file.

on generates a cross-reference report file.

View Report opens a view window that displays the report file you specify in the *Report File* field.

Report File specifies a name for the file where you want to store the cross-reference report.

Display Cross References in Schematic specifies whether cross-reference strings are displayed in the design window.

off does not display cross-reference strings.

on displays cross-references next to the pin in the schematic.

Cross Reference Format specifies how the system displays cross-references in the design window and report file.

Format String specifies the format of the text strings that the system displays in the design window. The default is

sheetNumber zone referenceName

Environment variable: pinRefFormat

Show Duplicate Zones determines whether the system displays duplicate pin references found within one zone.

off does not display duplicate references.

on displays all duplicate references.

Environment variable: pinRefDuplicates

Sort References By determines how the system sorts the elements within cross-reference lists.

Direction sorts the pins into groups by pin direction, then sorts the groups by sheet number and zone. The valid pin direction groups are *input pins*, *output pins*, and

Virtuoso Schematic Editor User Guide

Form Descriptions

I/O pins. switch is maintained solely for backward compatibility: Cadence discourages its use.

Sheet Number sorts the reference list by sheet number and zone.

Environment variable: pinRefSorting

Related topics

Displaying Cross-References

Cross-View Checker

Use this form to check interface consistency between views. It is accessed using the *Check – Cross View* command.

Cross View Check Severity specifies the action system performs when pin names and pin directions do not match.

ignored results in no action.

warning generates warnings.

error generates error messages.

Environment variable: vicSeverity

Views To Check displays a list of the views checked by the cross-view checker when you click *OK* or *Apply*.

Match Pin Names Exactly, when enabled, checks whether pin names match. For example, pins A<0> and A<1> in view1 and pin A<0:1> in view2 would not match.

Environment variable: vicExactPinMatch

Check Pin Order when enabled (default), checks whether the pin order is consistent in all views in the *Views To Check* list mentioned above.

Only check views with explicit pin order, when enabled (default), allows port order consistency check to run only on views with the `portOrder` property set.

Environment variable: vicIgnoreViewsWithNoSetPinOrder

Environment variable: vicCheckPinOrder

Match Inherited Terminals when enabled (default), checks whether inherited terminals match.

Ignore Terminals with Net Expression and no Terminal in the Other View when enabled (default), ignores all inherited terminal mismatches.

Environment variable: vicInheritedPinMatch

Ignore Terminals for Specified Views ignores the views specified (if any) in *View Names*.

Environment variable: vicIgnoreInhTermViewList

View Names specify a list of the views to be ignored by the cross-view checker.

Virtuoso Schematic Editor User Guide

Form Descriptions

Environment variable: [vicViewList](#)

All Terminals/Terminal Names when *All Terminals* is enabled all the terminals within the views specified in *View Names* will be ignored. When *Terminal Names* is enabled, the editable field to the right is also enabled for you to specify specific terminals to be ignored.

Environment variable: [vicIgnoreInhTermList](#)

Environment variables: [vicIgnoreInhTermAll](#), [vicInheritedPins](#)

Match Terminal Net Expressions when enabled (default), checks whether terminal net expressions match.

Environment variable: [vicInheritedPinMatch](#)

Match Terminal Supply Sensitivity when enabled (default), checks that terminal sensitivity and consistency is validated between the schematic and corresponding symbol cellviews.

Environment variable: [vicTermSupplySens](#)

Match Terminal Signal Type when enabled (default), checks for signal type mismatches between the corresponding pins of the schematic and symbol views of a cell.

Environment variable: [vicTermSigType](#)

Related topics

[Cross-View Checking](#)

Delete All Markers - OpenAccess version

Use this form to define the markers of a particular severity level, and in a particular cellview or hierarchy that are to be deleted. It is accessed using the *Check – Delete All Markers* command.

Severity defines which level of markers the system deletes in the list box and in your design.

all deletes all markers of the types selected.

none deletes none of the markers of the types selected.

fatal error deletes fatal error messages.

error deletes error messages.

acknowledged warning deletes acknowledged warning messages.

critical error deletes critical error messages.

signed off error deletes signed-off error messages.

info deletes information messages.

signed off critical error deletes signed-off critical error messages.

warning deletes only warning messages.

annotation deletes annotation messages.

Environment variable: markerDeleteSeverities

Search Scope defines the scope of the deletion.

cellview deletes markers from the current cellview only.

hierarchy starting from top cellview deletes markers throughout the entire hierarchy.

hierarchy starting from current cellview deletes markers throughout the hierarchy starting from the current cellview.

Environment variable: markerDeleteScope

Options enables you to specify whether the cellview is to be opened in edit or read mode.

Cellview Access Mode specifies the mode for opening a cellview.

Environment variable: markerDeleteAccessModeIsEdit

Virtuoso Schematic Editor User Guide

Form Descriptions

read deletes all the markers and re-opens the cellviews in edit mode with no dialog box.

edit deletes all the markers after opening the cellview in edit mode. In this mode, you can choose to select the *Confirm Access Mode Change* check box and specify the cellviews to be opened in read mode.

Confirm Access Mode Change opens the *Delete Marker* dialog box for you to confirm whether you want to open the selected cellview in edit mode. Markers are not deleted in cellviews that stay in read mode. This check box is enabled only if *Cellview Access Mode* is set to *edit*.

Environment variable: markerDeleteConfirmModeChange

Sources specifies the category of markers that are to be deleted in terms of the program that generated them.

SRC deletes schematic rules checker markers.

VIC deletes cross-view checker markers.

load deletes markers created when you opened the schematic as the source.

conn deletes connectivity markers

none appears when there are no messages in the list box.

Environment variable: markerDeleteSourcesMode, markerDeleteSources

Related topics

- Deleting All Markers
- Resolving Check Errors

Delete Generic – VHDL Properties

Use this form to delete properties that represent VHDL generics. It is accessed using the *Edit – Properties – VHDL* command and then clicking *Delete*.

Name specifies the generic name.

Note: When you select a generic in the list box before you select the command, the system fills in the form with the generic information.

Related topics

[Deleting VHDL Properties](#)

Delete Marker

Options enable you to specify whether the cellview is to be opened in edit or read mode.

Cellview Access Mode specifies the mode for opening a cellview.

Environment variable: [markerDeleteAccessModeIsEdit](#)

read deletes the selected markers and re-opens the cellviews in edit mode with no dialog box.

write deletes all the markers after opening the cellview in edit mode. In this mode, you can choose to select the *Confirm Access Mode Change* check box and specify the cellviews to be opened in read mode.

Confirm Access Mode Change opens the *Delete Marker* dialog box for you to confirm whether you want to open the selected cellview in edit mode. Markers are not deleted in cellviews that stay in read mode. This check box is enabled only if *Cellview Access Mode* to set to *edit*.

Environment variable: [markerDeleteConfirmModeChange](#)

Related Topics

[Deleting Individual Markers](#)

Delete Property – Edit Object Properties

Use this form to list all the user-defined properties available to be deleted. It is accessed by clicking *Delete* from the Edit Object Properties – Basic form.

Note: In general, you use the Delete Property form only to delete properties from the Edit Object Properties form or the Edit Cellview Properties form. Therefore, to delete properties from your cellview, you normally click *OK* or *Apply* on the form that calls the Delete Property form. However, the software deletes hierarchical properties from the database when you click *OK* or *Apply* on the Delete Property form.

The list box displays a list of all the user-defined properties.

Delete All, when enabled, deletes all the displayed properties.

Related topics

Deleting Cellview Properties

Delete Schematic Sheet

Use this form to delete a sheet when editing the index schematic of a multisheet design. It is accessed using the *Sheet – Delete* command from an index schematic.



This operation cannot be undone.

Starting Number specifies the first number of a range to delete. The default is 0.

Ending Number specifies the last number of a range to delete. If you specify same or do not specify the last sheet number in the range, only the current sheet is deleted. The default is same.

Related topics

[Deleting Sheets](#)

Descend Edit/Read

Use this form to descend into a lower-level view of an instance. It is accessed by choosing the *Edit – Hierarchy – Descend Edit / Read* command and then selecting an instance in your design to descend into.

Note: The options available vary depending on the type of instance selected. You can select from a single instance, an iterated instance, or a symbol representing a multisheet design.

Iteration Number lets you type the iteration you want to descend into. This option is available for iterated instances only.

View select the view you want to descend into.

Open for choose the mode that you want to open the selected *View* in.

edit opens the view in edit mode

read opens the view in read mode

auto opens the view dependent on current edit permission

Open in choose the method you want to display the descended cell view in. This option is available for single and multisheet instances only.

new tab opens the view in a new tab

current tab opens the view in the current tab (the default)

new window opens the view in a new window

Environment variable: descendCanvasType

Display specifies the type of sheet you want to descend into. This option is available for symbols representing multisheet designs only.

index descends into the index sheet.

sheet (default) descends into the sheet number you specify.

Related topics

Descending Using the Descend Command

Display Options

Use this form to change the window display and grid controls. It is accessed using the *Options – Display* command.

Background Color lets you choose the background color of the canvas. The color of the objects on the canvas is adjusted automatically for better visibility.

Net Expression Display lets you choose how a net expression label is displayed. The net name can vary depending on the sequence of instances through which you have descended. If you have descended down through an instance that has a netSet override property, the overriding signal name is displayed in the schematic window.

Display preferences for net expressions and evaluation for a particular descended branch of the hierarchy are as follows:

	Property gnd not found	Property gnd found and specifies mygnd!	Property gnd found and specifies a local signal “A” two levels down from the top
value only	gnd! *	mygnd! *	/I1/I2/A*
value (default)	gnd! (gnd!) *	mygnd! (gnd!) *	/I1/I2/A (gnd!) *
value [property name]	gnd! [gnd] *	mygnd! [gnd] *	/I1/I2/A [gnd] *
value [expression]	gnd! [@gnd:%:gnd!] *	mygnd! [@gnd:%:gnd!] *	/I1/I2/A [@gnd:%:gnd!] *
expression only	[@gnd:%:gnd!] *	[@gnd:%:gnd!] *	[@gnd:%:gnd!] *

Environment variable: netExprDisplay

Drag Limit lets you specify the maximum number of shapes whose movement is shown in the design window during a move or copy operation. When the total number of shapes exceeds the specified maximum, the system substitutes a bounding box for the set of shapes.

Environment variables: schMaxDragFig, symMaxDragFig

Axes specifies whether the schematic editor displays the X and Y axes when you view or edit a schematic view.

Environment variables: schDrawAxesOn, symDrawAxesOn

Virtuoso Schematic Editor User Guide

Form Descriptions

Cross Cursor controls the display of the cursor in all schematic and symbol editor windows, toggling *on* and *off* a cross-hair display to aid selection, placement etc.

Environment variables: [schCursorShape](#), [symCursorShape](#)

Object-Sensitive Pop-Up Menus lets you use the middle mouse button to open a menu of commands. The menu varies depending on the object currently under the cursor.

off disables this option. When you turn this option off, the default pop-up menu appears whenever you press the middle mouse button while an object is selected.

on (default) enables this option.

Environment variable: [sensitiveMenu](#)

Dynamic Highlighting highlights the objects in the design that are affected when certain active commands are implemented.

off disables this option.

on (default) enables this option.

Environment variables: [schDynamicHighlightOn](#), [symDynamicHighlightOn](#)

Dynamic Net Highlighting specifies whether the schematic editor dynamically highlights the items that it will select when you click them.

off disables this option.

on (default) enables this option.

Connect By Name Highlighting specifies whether the arcs need to be drawn between islands of connectivity. To show the connection between wires of the same net, you can enable this option. It also includes the scenarios where the wires are not visible on the screen at the same time.

off disables this option.

on (default) enables this option.

Environment variable: [schDynamicNetHighlightIslands](#)

Show Direct Connections shows the connection between pins.

off (default) disables this option.

on enables this option.

Virtuoso Schematic Editor User Guide

Form Descriptions

Environment variables: [schShowDirectConnects](#), [symShowDirectConnects](#)

Show Text Justification displays a plus sign (+) in the design wherever a text string begins.

off (default) disables this option.

on enables this option.

Environment variables: [schShowLabelOrigin](#), [symShowLabelOrigin](#)

Show Text Indicator displays a small rectangle when text becomes small and unreadable due to zooming out.

off (default) disables this option.

on enables this option.

Ignored Instances highlights the ignored instances in the schematic by displaying cross on the instances.

off (default) disables this option.

on enables this option.

Environment variable: [ignoreHighlightOn](#)

Display Status Banner displays an extra line of status information in the status banner of the design window. The status line displays the coordinates for the pointer or the selection count. When disabled, the *Coordinates on Status Banner* option disappears from the form.

off disables this option.

on (default) enables this option.

Environment variable: [statusBanner](#)

Coordinates on Status Banner displays the coordinates of the current mouse position in the status banner. This option appears only when *Display Status Banner* is *on*.

off (default) disables this option.

on enables this option.

Environment variable: [statusBannerXY](#)

Grid Controls

Type controls the display and the appearance of the grid.

none turns off the grid display.

dotted (default) displays a grid of dots.

line displays a grid of lines.

Environment variables: [schGridType](#), [symGridType](#)

Spacing specifies the number of user units between minor grid points on the visible grid. The default is 0.125.

Environment variables: [schGridSpacing](#), [symGridSpacing](#)

Multiple specifies the number of minor grid points. The default is 8.

Environment variables: [schGridMultiple](#), [symGridMultiple](#)

Snap Spacing specifies the minimum distance the cursor moves, in user units. The default distance (0.0625) is equal to half the default grid spacing distance.

Environment variables: [schSnapSpacing](#), [symSnapSpacing](#)

Zoom / Pan Controls

Dynamic Zooming activates dynamic zoom (*View – Dynamic Zooming*) to objects on the design canvas. The default for this option is *on*. See also [Dynamic Zoom](#).

Mode specifies the dynamic zoom setting to be used with *View – Dynamic Zooming*. Choose from:

☐ *Pan To Selected*

This is equivalent to *View – Pan*. Where, whenever an object, or a set of objects, is selected/cross-selected in an assistant, the view will pan to have the first object visible in the center of the canvas without changing the zoom factor. If a selected object is already visible in the canvas, the view will not pan to center.

Environment variable: [autoZoomPan](#)

☐ *Zoom To Selected*

If this mode is selected, dynamic zoom will perform an auto-zoom (the equivalent of *View – Zoom To Selected*) that ensures that all selected objects are visible in the design canvas window display.

☐ *Zoom To Fit*

Virtuoso Schematic Editor User Guide

Form Descriptions

If this mode is selected, a *View – Zoom To Fit* command will automatically be run after each object selection/cross-selection is made in an assistant.

Environment variable: autoZoomMode

Zoom Scale specifies an integer value indicating the percentage of visible canvas area that should display the selected set when the Dynamic Zooming mode is set to Zoom To Selected. The default value is 90(auto zoom 1 (far) to 100 (near)).

Environment variable: autoZoomScale

Dimming Controls

Display Dimming lets you set dimming *on* or *off*. When *on* is selected, the *View – Display Dimming* option is activated which controls the current dimming setting.

Environment variable: dimming

Mode lets you choose whether dimming be *Automatic* (the default); where dimming will only be enabled when at least one object is currently selected, highlighted, or probed. Or, choose to set dimming to *Always* which will enable dimming permanently (until the option setting is changed).

Environment variable: dimmingMode

Intensity lets you set the intensity (brightness) display of the dimmed objects, where 0 equals the dimmed object display being the same as the selected object, right up to 100 which will cause the dimmed objects to be “invisible” on the canvas. The default setting is 50.

Environment variable: dimmingIntensity

Related topics

Changing Display Options

Display Dimming

Edit Cellview Properties

Use this form to display the cellview properties. It is accessed using the *Edit – Properties – Cellview* command.

Show options specify which properties the system displays on the form.

system displays the system-defined properties for a selected object.

Environment variable: editPropShowSystem

user (default) displays the user-defined properties for a selected object.

Environment variable: editPropShowUser

Property displays the property names for the selected object. This section of the form expands and contracts depending on which of the *Show* options you select and whether you select an object in the design.

The default display shows only the following system-defined properties:

Library specifies the library that contains the current design.

Cell specifies the cell name of the current design.

View specifies the view of the current design.

Note: You cannot add, delete, or modify grayed-out properties.

When you select *system*, the following system-generated properties appear on the form:

CellView Type specifies the cellview type of the current design.

Mode identifies which mode you are in: edit or read only.

Environment variable: findAccessMode

Modified displays *TRUE* if the cellview has been modified during the current editing session.

schType identifies the type of schematic data.

instNamePrefix (appears only when the cellview is a symbol) defines a new prefix for instances of the symbol cellview. The prefix can be more than one character.

DBUPerUU displays the number of database units per user unit.

User Unit displays the current user unit.

Last Changed displays the date the cellview was last modified.

Virtuoso Schematic Editor User Guide

Form Descriptions

Last Checked displays the date the cellview was last checked.

Upper Right and **Lower Left** display the X and Y coordinates for the upper right corner and lower left corner of the cellview.

Note: Except for the *instNamePrefix* symbol property, you cannot add, delete, or modify system-generated properties.

Value displays the value for a property. Use *Modify* to edit values.

Display determines whether a property value is displayed in the design. When you enable one of the display options (*value*, *name*, or *both*), the information appears on the schematic near the selected object.

value displays only the value of the property.

name displays only the name of the property.

both displays both the name and the value of the property, separated by an equal sign.

off (default) displays neither the name nor the value of the property.

The *Add*, *Delete*, and *Modify* buttons appear on the form only when you enable the *Show – user* option at the top of the form. Use these options to add, delete, and modify user-defined properties.

Add displays the Add Property form, which you use to add new user properties to the Edit Cellview Properties form. To add the new properties to your cellview, click *OK* or *Apply* on the Edit Cellview Properties form.

Delete displays the Delete Property form, which lets you delete user properties. To delete properties from your cellview, click *OK* or *Apply* on the Edit Cellview Properties form.



The software deletes hierarchical properties from the database when you click *OK* or *Apply* on the Delete Property form.

Modify displays the Modify Property form, which lets you change the names and values for selected user properties. To apply the changes to your cellview, click *OK* or *Apply* on the Edit Cellview Properties form.

Related topics

[Editing Cellview Properties](#)

Edit Object Properties – Basic

Use this form to change properties of specific objects. It is accessed by selecting an object with a property you want to edit and using the *Edit – Properties – Objects* command.

You use the *Previous* and *Next* buttons along with the *Apply To* options below to locate objects in your design window. When there are multiple objects in the selected set, you use the *Next* and *Previous* buttons to locate objects in the set. You use the *Apply To* options to specify the objects the software changes.

Previous cycles backward to view the preceding object in the selected set. When you reach the first object in the selected set, this button has no effect.

Next cycles forward to view the next object in the selected set. When you reach the last object in the selected set, this button has no effect.

Apply To consists of up to three cyclic fields that you use to specify the object, or group of objects in a selected set, whose properties you want to change. The software uses one color to highlight selected objects. But when there are multiple objects in the selected set, the software uses a second color to highlight the current object. And when you configure the cyclic fields to apply changes to multiple objects, the software uses a third color to highlight all the affected objects.

Selection scope specifies whether you want to edit only the current object, only the selected objects in the specified object type group, or all the objects in the specified object type group (which includes both selected and unselected objects of the specified object type).

Object group lists the object types in the selected set. Use this option to locate the first object of a particular object type.

Object filter appears only when the selection scope is set to *all selected* or *all*. This option acts as a secondary filter for further limiting the selection scope. The options vary depending on the selected object type.

Show specifies which properties the system displays on the form.

system displays the system-defined properties for a selected object.

Environment variable: [editPropShowSystem](#)

user (default) displays the user-defined properties for a selected object.

Environment variable: [editPropShowUser](#)

Virtuoso Schematic Editor User Guide

Form Descriptions

CDF (default) displays predefined properties defined by the Component Description Format (CDF). For more details, see [Applying One or More CDF Parameter Edits to Multiple Objects](#).

Environment variable: [editPropShowCDF](#)

Environment variables: [editPropChangeAllConfirm](#), [editPropProtectCDF](#)

Related topics

[Editing Object Properties](#)

[Applying Edits to Objects and Displaying Object Types](#)

[Applying One or More CDF Parameter Edits to Multiple Objects](#)

Edit Object Properties – Instance and Block

Use this form to edit instance and block properties and their parameters and values. It is accessed by selecting an instance or block that has a property you want to edit and using the *Edit – Properties – Objects* command.

You use the *Previous* and *Next* buttons along with the *Apply To* options below to locate objects in your design window. When there are multiple objects in the selected set, you use the *Next* and *Previous* buttons to locate objects in the set. You use the *Apply To* options to specify the objects the software changes.

Previous cycles backward to view the preceding object in the selected set. When you reach the first object in the selected set, this button has no effect.

Next cycles forward to view the next object in the selected set. When you reach the last object in the selected set, this button has no effect.

Apply To consists of up to three cyclic fields (*only current*, *all selected*, *all*) that you use to specify the object, or group of objects in a selected set, whose properties you want to change. The software uses one color to highlight selected objects. But when there are multiple objects in the selected set, the software uses a second color to highlight the current object. And when you configure the cyclic fields to apply changes to multiple objects, the software uses a third color to highlight all the affected objects.

Selection scope specifies whether you want to edit only the current object, only the selected objects in the specified object type group, or all the objects in the specified object type group (which includes both selected and unselected objects of the specified object type).

Object group lists the object types in the selected set. Use this option to locate the first object of a particular object type.

Object filter appears only when the selection scope is set to *all selected* or *all*. This option acts as a secondary filter for further limiting the selection scope. The options vary depending on the selected object type. The possible choices for the instance object type are *same master* and *any master*.

Show specifies which properties the system displays on the form.

system displays the system-defined properties for a selected object.

Environment variable: editPropShowSystem

user (default) displays the user-defined properties for a selected object.

Environment variable: editPropShowUser

Virtuoso Schematic Editor User Guide

Form Descriptions

CDF (default) displays predefined properties defined by the Component Description Format (CDF).

Environment variable: [editPropShowCDF](#)

Browse opens the Library Browser, which you can use to fill in the form automatically.

Reset Instance Labels Display resets the display and placement of all instance labels to their default specifications as defined in the symbol view.

Property lists the properties that the system automatically extracts from the design database.

Value displays the values for the corresponding properties.

Display determines whether the system displays the property name and property value in the schematic.

off (default) displays neither the name nor the value.

value displays only the value.

name displays only the name.

both displays both the name and the value, separated by an equal sign.

User Property lists the user properties that exist for the current instance/block.

Master Value specifies the master symbol property value. You cannot edit this field.

Local Value specifies the local instance property value.

Display determines whether the system displays the property name and property value in the schematic.

off (default) displays neither the name nor the value.

value displays only the value.

name displays only the name.

both displays both the name and the value, separated by an equal sign.

Change All allows you to apply user property value changes to all instances in the view or to all selected view instances. This option is only available if you have already selected *Apply To: all selected* or *all*.

The options described below appear when you enable the *Show – system* option.

Virtuoso Schematic Editor User Guide

Form Descriptions

Magnification controls the scaling factor applied to the selected instance. A value of 1 (one) specifies the true size.

Origin displays the location of the instance in the cellview.

The *Add*, *Delete*, and *Modify* buttons affect only the display of properties on the Edit Objects Properties form. You must click *OK* or *Apply* to apply the changes to objects in your design.

Add displays the Add Property form, which lets you add a new property to the Edit Object Properties form.

Delete displays the Delete Property form, which lets you delete selected properties from the Edit Object Properties form.

Modify displays the Modify Property form, which lets you redefine properties currently displayed on the Edit Object Properties form.

The following fields appear for block editing in addition to the ones described for editing an instance on the previous pages.

FreezePinCreate appears only if you have selected a block. It controls whether the system adds a pin to a block when you connect a new wire to the edge of a block. When this option is disabled (default), the system automatically creates a pin when you click the edge of a block to terminate the wire. When this option is enabled, the system does not create a pin. Instead, the system automatically draws the wire to the nearest existing pin, when one exists. When no pin exists, the system does not let you connect a wire to the block. Use the *Commands* tab in the Editor Options form (*Options – Editor* command) to specify the pin directions.

The following options appear on the form only when you enable the *Show – system* option.

Pin Direction specifies a direction for the pin. *switch* is maintained solely for backward compatibility: Cadence discourages its use.

Pin Name specifies the pin name.

Net Name displays the net name.

Inherit, in general when this option is enabled, derives the pin name from the net name. However, if you delete the wire segment for a pin, the system automatically enables this option and derives the net name from the pin name.

Environment variables: editPropChangeAllConfirm, editPropProtectCDF

Virtuoso Schematic Editor User Guide

Form Descriptions

Related topics

[Editing Instance and Block Properties](#)

Edit Object Properties – Instance Pin

Use this form to edit instance pin properties and their parameters and values. It is accessed by selecting an instance pin that has a property you want to edit and using the *Edit – Properties – Objects* command.

You use the *Previous* and *Next* buttons along with the *Apply To* options below to locate objects in your design window. When there are multiple objects in the selected set, you use the *Next* and *Previous* buttons to locate objects in the set. You use the *Apply To* options to specify the objects the software changes.

Previous cycles backward to view the preceding object in the selected set. When you reach the first object in the selected set, this button has no effect.

Next cycles forward to view the next object in the selected set. When you reach the last object in the selected set, this button has no effect.

Apply To consists of up to three cyclic fields that you use to specify the object, or group of objects in a selected set, whose properties you want to change. The software uses one color to highlight selected objects. But when there are multiple objects in the selected set, the software uses a second color to highlight the current object. And when you configure the cyclic fields to apply changes to multiple objects, the software uses a third color to highlight all the affected objects.

Selection scope specifies whether you want to edit only the current object, only the selected objects in the specified object type group, or all the objects in the specified object type group (which includes both selected and unselected objects of the specified object type).

Object group lists the object types in the selected set. Use this option to locate the first object of a particular object type.

Object filter appears only when the selection scope is set to *all selected* or *all*. This option acts as a secondary filter for further limiting the selection scope. The options vary depending on the selected object type.

Show specifies which properties the system displays on the form.

user (default) displays the user-defined properties for a selected object.

Environment variable: editPropShowUser

CDF (default) displays predefined properties defined by the Component Description Format (CDF).

Environment variable: editPropShowCDF

Virtuoso Schematic Editor User Guide

Form Descriptions

Name displays the name of the instance pin.

Direction specifies the direction of the instance pin. *switch* is maintained solely for backward compatibility: Cadence discourages its use.

Net Name displays the name of the net that is connected to the instance pin.

The cyclic field to the right of the *Name* field determines whether the system displays the property value in the schematic.

The *Add*, *Delete*, and *Modify* buttons affect only the display of properties on the Edit Objects Properties form. You must click *OK* or *Apply* to apply the changes to objects in your design.

Add displays the Add Property form, which lets you add a new property to the Edit Object Properties form.

Delete displays the Delete Property – Edit Object Properties form, which lets you delete selected properties from the Edit Object Properties form.

Modify displays the Modify Property – Edit Object Properties form, which lets you redefine properties currently displayed on the Edit Object Properties form.

Environment variables: editPropChangeAllConfirm, editPropProtectCDF

Related topics

Editing Instance Pin Properties on an Instance of a Block

Edit Object Properties – Label

Use this form to edit label properties and label parameters and values. It is accessed by selecting the label whose properties you want to edit and choosing the *Edit – Properties – Objects* command.

You use the *Previous* and *Next* buttons along with the *Apply To* options below to locate objects in your design window. When there are multiple objects in the selected set, you use the *Next* and *Previous* buttons to locate objects in the set. You use the *Apply To* options to specify the objects the software changes.

Previous cycles backward to view the preceding object in the selected set. When you reach the first object in the selected set, this button has no effect.

Next cycles forward to view the next object in the selected set. When you reach the last object in the selected set, this button has no effect.

Apply To consists of up to three cyclic fields that you use to specify the object, or group of objects in a selected set, whose properties you want to change. The software uses one color to highlight selected objects. But when there are multiple objects in the selected set, the software uses a second color to highlight the current object. And when you configure the cyclic fields to apply changes to multiple objects, the software uses a third color to highlight all the affected objects.

Selection scope specifies whether you want to edit only the current object, only the selected objects in the specified object type group, or all the objects in the specified object type group (which includes both selected and unselected objects of the specified object type).

Object group lists the object types in the selected set. Use this option to locate the first object of a particular object type.

Object filter appears only when the selection scope is set to *all selected* or *all*. This option acts as a secondary filter for further limiting the selection scope. The options vary depending on the selected object type.

Attachment Label specifies which object the label belongs to.

Display determines whether the system displays the property value in the design window. You can toggle this option to *off* to suppress the display. To display labels after they have been turned off, see [Resetting Invisible Labels](#).

Height lets you specify the height of the label.

Font lets you select a font for the label.

Virtuoso Schematic Editor User Guide

Form Descriptions

Justification lets you specify where the label appears in relation its origin (placement location).

Drafting, when enabled (default), maintains the normal orientation of the text as you rotate a label 180 degrees around a reference point. When this option is disabled, rotated text appears upside down.

Overbar is only a display option. You use this option to specify whether the software displays an overbar over a label. When this option is disabled (default), the software displays underscore characters (_) in the label as part of the label text string. When this option is enabled, the software interprets underscore characters (_) in the label as toggle switches that control where overbars begin and end. Overbars appear above the label, as shown in the examples.

- The property name _abcde puts an overbar over all 5 letters.
- The property name ab_cde puts an overbar over only cde.
- The property name _abc_de puts an overbar over only abc.
- The property name ab_cd_ef_gh_ij puts an overbar over only cd and gh.



Caution

To avoid confusion when text contains underscore characters, Cadence recommends that you disable this display option. It is provided only for backward compatibility.

Related topics

[Editing Label Properties](#)

Edit Object Properties – Master Property

Use this form to change master properties. It is accessed by selecting an object with a master property you want to edit and using the *Edit – Properties – Objects* command.

You use the *Previous* and *Next* buttons along with the *Apply To* options below to locate objects in your design window. When there are multiple objects in the selected set, you use the *Next* and *Previous* buttons to locate objects in the set. You use the *Apply To* options to specify the objects the software changes.

Previous cycles backward to view the preceding object in the selected set. When you reach the first object in the selected set, this button has no effect.

Next cycles forward to view the next object in the selected set. When you reach the last object in the selected set, this button has no effect.

Apply To consists of up to three cyclic fields that you use to specify the object, or group of objects in a selected set, whose properties you want to change. The software uses one color to highlight selected objects. But when there are multiple objects in the selected set, the software uses a second color to highlight the current object. And when you configure the cyclic fields to apply changes to multiple objects, the software uses a third color to highlight all the affected objects.

Selection scope specifies whether you want to edit only the current object, only the selected objects in the specified object type group, or all the objects in the specified object type group (which includes both selected and unselected objects of the specified object type).

Object group lists the object types in the selected set. Use this option to locate the first object of a particular object type.

Attachment Label specifies which object the property belongs to.

Property lists the properties that the system automatically extracts from the design database.

<property name> appears only when the selected property is on an instance. This option does not appear when the property is on a net, a terminal, or a cellview.

Master Value specifies the master symbol property value. You cannot edit this field.

Local Value specifies the local instance property value.

Display specifies whether the system displays the property name and property value in the schematic.

Virtuoso Schematic Editor User Guide

Form Descriptions

off displays neither the name nor the value.

value (default) displays only the value.

name displays only the name of the property.

both displays both the name and the value, separated by an equal sign.

Height specifies the height of the property label.

Font specifies a font for the property label.

Justification specifies where the label appears in relation to its origin (placement location).

Drafting, when enabled (default), maintains the normal orientation of the text as you rotate a label 180 degrees around a reference point. When this option is disabled, rotated text appears upside down.

Overbar is only a display option. You use this option to specify whether the software displays an overbar over a property label. When this option is disabled (default), the software displays underscore characters (_) in the property label as part of the property label text string. When this option is enabled, the software interprets underscore characters (_) in the property label as toggle switches that control where overbars begin and end. Overbars appear above the property label, as shown in the examples.

- The property name _abcde puts an overbar over all 5 letters.
- The property name ab_cde puts an overbar over only cde.
- The property name _abc_de puts an overbar over only abc.
- The property name ab_cd_ef_gh_ij puts an overbar over only cd and gh.



To avoid confusion when text contains underscore characters, Cadence recommends that you disable this display option. It is provided only for backward compatibility.

Related topics

[Editing Master Property Values](#)

Edit Object Properties – Net Expression

Use this form to edit a net expression label. It is accessed by selecting the net expression label you want to edit and using the *Edit – Properties – Objects* command.

You use the *Previous* and *Next* buttons along with the *Apply To* options below to locate objects in your design window. When there are multiple objects in the selected set, you use the *Next* and *Previous* buttons to locate objects in the set. You use the *Apply To* options to specify the objects the software changes. For more details, see [Editing a Net Expression Label](#).

Previous cycles backward to view the preceding object in the selected set. When you reach the first object in the selected set, this button has no effect.

Next cycles forward to view the next object in the selected set. When you reach the last object in the selected set, this button has no effect.

Apply To consists of up to three cyclic fields that you use to specify the object, or group of objects in a selected set, whose properties you want to change. The software uses one color to highlight selected objects. But when there are multiple objects in the selected set, the software uses a second color to highlight the current object. And when you configure the cyclic fields to apply changes to multiple objects, the software uses a third color to highlight all the affected objects.

Selection scope specifies whether you want to edit only the current object, only the selected objects in the specified object type group, or all the objects in the specified object type group (which includes both selected and unselected objects of the specified object type).

Object group lists the object types in the selected set. Use this option to locate the first object of a particular object type.

Attachment reattaches the net expression label to a wire or pin.

Property Name specifies the name of the property that can be used to override the default global signal. In other words, the name of the property whose value redefines the name of the net. The system evaluating the expression searches for the property that has this name to get the value to inherit for the net.

Default Net Name specifies the default property name of the global signal or net name. The schematic editor uses the default net name, for example vdd!, gnd!, or vss!, when an inherited connection has not been established. The name must specify a global single-bit value.

Evaluated Name contains the result of evaluating the net expression for the current descend path of the window.

Height specifies the height of the net expression label text in user units.

Font specifies the font the system uses to display the net expression label.

Justification specifies where the label appears in relation to its origin (placement location).

Related topics

[Editing a Net Expression Label](#)

Edit Object Properties – Note Shapes

Use this form to change the line property or position of a note shape. It is accessed by selecting a note shape and choosing the *Edit – Properties – Objects* command.

You use the *Previous* and *Next* buttons along with the *Apply To* options below to locate objects in your design window. When there are multiple objects in the selected set, you use the *Next* and *Previous* buttons to locate objects in the set. You use the *Apply To* options to specify the objects the software changes.

Previous cycles backward to view the preceding object in the selected set. When you reach the first object in the selected set, this button has no effect.

Next cycles forward to view the next object in the selected set. When you reach the last object in the selected set, this button has no effect.

Apply To consists of up to three cyclic fields that you use to specify the object, or group of objects in a selected set, whose properties you want to change. The software uses one color to highlight selected objects. But when there are multiple objects in the selected set, the software uses a second color to highlight the current object. And when you configure the cyclic fields to apply changes to multiple objects, the software uses a third color to highlight all the affected objects.

Selection scope specifies whether you want to edit only the current object, only the selected objects in the specified object type group, or all the objects in the specified object type group (which includes both selected and unselected objects of the specified object type).

Object group lists the object types in the selected set. Use this option to locate the first object of a particular object type.

Property lists the properties that the system automatically extracts from the design database.

Value displays the values for the corresponding properties.

Line Style specifies whether the system uses a solid or dashed line to draw the note shape.

The form displays various options depending on the shape type you select.

The following options appear when you select a line shape.

Width specifies whether the line shape is narrow (default) or wide. You can edit the text field when you enable the *wide* option.

Points List specifies the coordinates for the points that make up the shape.

Virtuoso Schematic Editor User Guide

Form Descriptions

The following option appears when you select a polygon shape.

Point List specifies the coordinates for the points that make up the shape.

The following options appear when you select a rectangle shape.

Upper Right specifies coordinates that define the upper right-hand corner of the shape.

Lower Left specifies coordinates that define the lower left-hand corner of the shape.

The following options appear when you select either a circle shape or an ellipse shape. The *Shape* text field displays *ellipse* for both circle and ellipse shapes.

Center specifies the coordinates that define the center of the shape.

X Radius specifies the coordinate that defines the Xradius of the shape.

Y Radius specifies the coordinate that defines the Yradius of the shape.

The following options appear when you select an arc shape.

Center specifies the coordinates that define the center of the shape.

X Radius specifies the coordinate that defines the Xradius of the shape.

Y Radius specifies the coordinate that defines the Yradius of the shape.

Start Angle specifies the first point of the arc.

Stop Angle specifies the second point of the arc.

Related topics

[Editing Note Shape Properties](#)

Edit Object Properties – Note Text

Use this form to edit the text of a note after you add it to your design. It is accessed by selecting the note text and choosing the *Edit – Properties – Objects* command.

You use the *Previous* and *Next* buttons along with the *Apply To* options below to locate objects in your design window. When there are multiple objects in the selected set, you use the *Next* and *Previous* buttons to locate objects in the set. You use the *Apply To* options to specify the objects the software changes.

Previous cycles backward to view the preceding object in the selected set. When you reach the first object in the selected set, this button has no effect.

Next cycles forward to view the next object in the selected set. When you reach the last object in the selected set, this button has no effect.

Apply To consists of up to three cyclic fields that you use to specify the object, or group of objects in a selected set, whose properties you want to change. The software uses one color to highlight selected objects. But when there are multiple objects in the selected set, the software uses a second color to highlight the current object. And when you configure the cyclic fields to apply changes to multiple objects, the software uses a third color to highlight all the affected objects.

Selection scope specifies whether you want to edit only the current object, only the selected objects in the specified object type group, or all the objects in the specified object type group (which includes both selected and unselected objects of the specified object type).

Object group lists the object types in the selected set. Use this option to locate the first object of a particular object type.

Attachment performs two functions.

- When the text is already associated with an object, the software highlights the associated object in the cellview.
- Lets you associate the note text with an object in the cellview by automatically drawing a line from the text to the object when you move the cursor into the design window.

Note Text specifies the text for your note. You can either type in the text or use *Read File* to load the text from a file. You can edit any text you load into the this edit window.

Read File loads text from the specified file into the *Note Text* edit window.

Height lets you specify the height of the note text.

Virtuoso Schematic Editor User Guide

Form Descriptions

Font lets you select a font for the note text.

Justification lets you specify where the note text appears in relation to its origin (placement location).

Drafting, when enabled (default), maintains the normal orientation of the note text as you rotate a label 180 degrees around a reference point. When this option is disabled, rotated text appears upside down.

Overbar is only a display option. You use this option to specify whether the software displays an overbar over the note text. When this option is disabled (default), the software displays underscore characters (_) in the note text as part of the note text string. When this option is enabled, the software interprets underscore characters (_) in the note text as toggle switches that control where overbars begin and end. Overbars appear above the note text, as shown in the examples.

- The property name _abcde puts an overbar over all 5 letters.
- The property name ab_cde puts an overbar over only cde.
- The property name _abc_de puts an overbar over only abc.
- The property name ab_cd_ef_gh_ij puts an overbar over only cd and gh.



To avoid confusion when text contains underscore characters, Cadence recommends that you disable this display option. It is provided only for backward compatibility.

Related topics

[Editing Note Text Properties](#)

Edit Object Properties – Other Object

Use this form to edit visible user-defined object properties, values, and their display attributes. It is accessed by selecting the object property and choosing the *Edit – Properties – Objects* command.

You use the *Previous* and *Next* buttons along with the *Apply To* options below to locate objects in your design window. When there are multiple objects in the selected set, you use the *Next* and *Previous* buttons to locate objects in the set. You use the *Apply To* options to specify the objects the software changes.

Previous cycles backward to view the preceding object in the selected set. When you reach the first object in the selected set, this button has no effect.

Next cycles forward to view the next object in the selected set. When you reach the last object in the selected set, this button has no effect.

Apply To consists of up to three cyclic fields that you use to specify the object, or group of objects in a selected set, whose properties you want to change. The software uses one color to highlight selected objects. But when there are multiple objects in the selected set, the software uses a second color to highlight the current object. And when you configure the cyclic fields to apply changes to multiple objects, the software uses a third color to highlight all the affected objects.

Selection scope specifies whether you want to edit only the current object, only the selected objects in the specified object type group, or all the objects in the specified object type group (which includes both selected and unselected objects of the specified object type).

Object group lists the object types in the selected set. Use this option to locate the first object of a particular object type.

Owner specifies which object the property belongs to.

Property lists the properties that the system automatically extracts from the design database.

Name specifies the property name.

Value specifies the property value.

Display specifies whether the system displays the property name and property value in the schematic.

off displays neither the name nor the value.

Virtuoso Schematic Editor User Guide

Form Descriptions

value (default) displays only the value.

name displays only the name of the property.

both displays both the name and the value, separated by an equal sign.

Height specifies the height of the property label.

Font specifies a font for the property label.

Justification specifies where the label appears in relation to its origin (placement location).

Drafting, when enabled (default), maintains the normal orientation of the text as you rotate a label 180 degrees around a reference point. When this option is disabled, rotated text appears upside down.

Overbar is only a display option. You use this option to specify whether the software displays an overbar over a property name. When this option is disabled (default), the software displays underscore characters (_) in the property name as part of the property name text string. When this option is enabled, the software interprets underscore characters (_) in the property name as toggle switches that control where overbars begin and end. Overbars appear above the property name, as shown in the examples.

- The property name _abcde puts an overbar over all 5 letters.
- The property name ab_cde puts an overbar over only cde.
- The property name _abc_de puts an overbar over only abc.
- The property name ab_cd_ef_gh_ij puts an overbar over only cd and gh.



To avoid confusion when text contains underscore characters, Cadence recommends that you disable this display option. It is provided only for backward compatibility.

Related topics

[Editing User-Defined Object Properties](#)

Edit Object Properties – Pin

Use this form to edit pin properties and their parameters and values. It is accessed by selecting a pin that has a property you want to edit and using the *Edit – Properties – Objects* command.

You use the *Previous* and *Next* buttons along with the *Apply To* options below to locate objects in your design window. When there are multiple objects in the selected set, you use the *Next* and *Previous* buttons to locate objects in the set. You use the *Apply To* options to specify the objects the software changes.

Previous cycles backward to view the preceding object in the selected set. When you reach the first object in the selected set, this button has no effect.

Next cycles forward to view the next object in the selected set. When you reach the last object in the selected set, this button has no effect.

Apply To consists of up to three cyclic fields that you use to specify the object, or group of objects in a selected set, whose properties you want to change. The software uses one color to highlight selected objects. But when there are multiple objects in the selected set, the software uses a second color to highlight the current object. And when you configure the cyclic fields to apply changes to multiple objects, the software uses a third color to highlight all the affected objects.

Selection scope specifies whether you want to edit only the current object, only the selected objects in the specified object type group, or all the objects in the specified object type group (which includes both selected and unselected objects of the specified object type).

Object group lists the object types in the selected set. Use this option to locate the first object of a particular object type.

Object filter appears only when the selection scope is set to *all selected* or *all*. This option acts as a secondary filter for further limiting the selection scope. The options vary depending on the selected object type.

Show specifies which properties the system displays on the form.

system displays the system-defined properties for a selected object.

Environment variable: editPropShowSystem

When you enable the *Show – system* option, the form expands to display information about the pin master to show the *Magnification* and *Origin* fields.

Virtuoso Schematic Editor User Guide

Form Descriptions

Magnification controls the scaling factor applied to the selected instance. A value of 1 (one) specifies the true size.

Origin specifies the location of the instance in the design.

user (default) displays the user-defined properties for a selected object.

Environment variable: [editPropShowUser](#)

CDF (default) displays predefined properties defined by the Component Description Format (CDF).

Environment variable: [editPropShowCDF](#)

Property lists the properties that the system automatically extracts from the design database.

Value displays the values for the corresponding properties.

Display determines whether the system displays the property value in the schematic.

Direction specifies a direction for the pin. *switch* is maintained solely for backward compatibility: Cadence discourages its use.

Usage specifies whether the pin is an offsheet connector.

Signal Type specifies the signal type of the selected pin. Refer to [Defining Signal Types](#).

Pwr Sensitivity type in, or select from the drop-down menu (which lists all the single-bit terminals in the cellview), the names of the terminals that would make the current pins' terminals sensitive to the entered power terminals. For more details, see [How to Set Supply and Ground Sensitivity](#).

Gnd Sensitivity type in, or select from the drop-down menu (which lists all the single-bit terminals in the cellview), the names of the terminals that would make the current pins' terminals sensitive to the entered ground terminals. For more details, see [How to Set Supply and Ground Sensitivity](#).

Note: The specified power and ground terminals must be single-bit terminals and pre-exist in the database before the sensitivity relationship is established. If you specify terminals that do not already exist in the database, or if the terminal name is not a single-bit, this will invoke an error condition. Subsequently, no pin placement will take place, the error field will be highlighted, and an error message will be displayed in the CIW.

Add displays the Add Property form, which lets you add a new property to the Edit Object Properties form.

Virtuoso Schematic Editor User Guide

Form Descriptions

Delete displays the [Delete Property form](#), which lets you delete selected properties from the Edit Object Properties form.

Modify displays the [Modify Property form](#), which lets you redefine properties currently displayed on the Edit Object Properties form.

Environment variables: [editPropChangeAllConfirm](#), [editPropProtectCDF](#)

Related topics

[Editing Specific Object Properties](#)

[Supply and Ground Sensitivity](#)

Edit Object Properties – Pin Name

Use this form to edit pin name properties and pin name parameters and values. It is accessed by selecting a pin and choosing the *Edit – Properties – Objects* command.

You use the *Previous* and *Next* buttons along with the *Apply To* options below to locate objects in your design window. When there are multiple objects in the selected set, you use the *Next* and *Previous* buttons to locate objects in the set. You use the *Apply To* options to specify the objects the software changes.

Previous cycles backward to view the preceding object in the selected set. When you reach the first object in the selected set, this button has no effect.

Next cycles forward to view the next object in the selected set. When you reach the last object in the selected set, this button has no effect.

Apply To consists of up to three cyclic fields that you use to specify the object, or group of objects in a selected set, whose properties you want to change. The software uses one color to highlight selected objects. But when there are multiple objects in the selected set, the software uses a second color to highlight the current object. And when you configure the cyclic fields to apply changes to multiple objects, the software uses a third color to highlight all the affected objects.

Selection scope specifies whether you want to edit only the current object, only the selected objects in the specified object type group, or all the objects in the specified object type group (which includes both selected and unselected objects of the specified object type).

Object group lists the object types in the selected set. Use this option to locate the first object of a particular object type.

Object filter appears only when the selection scope is set to *all selected* or *all*. This option acts as a secondary filter for further limiting the selection scope. The options vary depending on the selected object type.

Attachment lets you reattach the pin name to the pin.

Label specifies the text for the pin name.

Value specifies whether the system displays the property value in the schematic. You can choose *off* to suppress the display.

Height specifies the height of the pin label in user units.

Font specifies a font for the pin label.

Virtuoso Schematic Editor User Guide

Form Descriptions

Justification specifies where the label appears in relation to the pin.

Drafting when enabled (default) maintains the normal orientation of the text when you rotate a label 180 degrees around a reference point. When disabled, rotated text appears upside down.

Overbar is only a display option. You use this option to specify whether the software displays an overbar over a pin name. When disabled (default), the software displays underscore characters (_) in the pin name as part of the pin name text string. When enabled, the software interprets underscore characters (_) in the pin name as toggle switches that control where overbars begin and end. Overbars appear above the pin name, as shown in the examples.

- The property name _abcde puts an overbar over all 5 letters.
- The property name ab_cde puts an overbar over only cde.
- The property name _abc_de puts an overbar over only abc.
- The property name ab_cd_ef_gh_ij puts an overbar over only cd and gh.



Caution

To avoid confusion when pin names contain underscore characters, Cadence recommends that you disable this display option. It is provided only for backward compatibility.

Related topics

[Editing Specific Object Properties](#)

Edit Object Properties – Selection Box

Use this form to change the position of the selection box. It is accessed by selecting the selection box and choosing the *Edit – Properties – Objects* command.

You use the *Previous* and *Next* buttons along with the *Apply To* options below to locate objects in your design window. When there are multiple objects in the selected set, you use the *Next* and *Previous* buttons to locate objects in the set. You use the *Apply To* options to specify the objects the software changes.

Previous cycles backward to view the preceding object in the selected set. When you reach the first object in the selected set, this button has no effect.

Next cycles forward to view the next object in the selected set. When you reach the last object in the selected set, this button has no effect.

Apply To consists of up to three cyclic fields that you use to specify the object, or group of objects in a selected set, whose properties you want to change. The software uses one color to highlight selected objects. But when there are multiple objects in the selected set, the software uses a second color to highlight the current object. And when you configure the cyclic fields to apply changes to multiple objects, the software uses a third color to highlight all the affected objects.

Selection scope specifies whether you want to edit only the current object, only the selected objects in the specified object type group, or all the objects in the specified object type group (which includes both selected and unselected objects of the specified object type).

Object group lists the object types in the selected set. Use this option to locate the first object of a particular object type.

Property lists the properties that the system automatically extracts from the design database.

Shape specifies the shape of the selection box.

Upper Right specifies the coordinates for the upper right corner of the selection box.

Lower Left specifies the coordinates for the lower left corner of the selection box.

Value specifies the values for the properties. You can edit the coordinate values when the selection scope is set to *only current*.

Virtuoso Schematic Editor User Guide

Form Descriptions

Related topics

[Editing Symbol Selection Box Properties](#)

Edit Object Properties – Symbol Label

Use this form to edit symbol label properties and symbol label parameters and values. It is accessed by selecting a symbol label and using the *Edit – Properties – Objects* command.

You use the *Previous* and *Next* buttons along with the *Apply To* options below to locate objects in your design window. When there are multiple objects in the selected set, you use the *Next* and *Previous* buttons to locate objects in the set. You use the *Apply To* options to specify the objects the software changes.

Previous cycles backward to view the preceding object in the selected set. When you reach the first object in the selected set, this button has no effect.

Next cycles forward to view the next object in the selected set. When you reach the last object in the selected set, this button has no effect.

Apply To consists of up to three cyclic fields that you use to specify the object, or group of objects in a selected set, whose properties you want to change. The software uses one color to highlight selected objects. But when there are multiple objects in the selected set, the software uses a second color to highlight the current object. And when you configure the cyclic fields to apply changes to multiple objects, the software uses a third color to highlight all the affected objects.

Selection scope specifies whether you want to edit only the current object, only the selected objects in the specified object type group, or all the objects in the specified object type group (which includes both selected and unselected objects of the specified object type).

Object group lists the object types in the selected set. Use this option to locate the first object of a particular object type.

Attachment lets you reattach the label to another component in the design.

Label specifies the label name.

Value determines whether the property value is displayed in the schematic. You can set this option to *off* to suppress the display. To display labels after they have been turned off, see [Resetting Invisible Labels](#).

Choice contains a list of valid label types.

Height specifies the height of the symbol label.

Font specifies a font for the symbol label.

Virtuoso Schematic Editor User Guide

Form Descriptions

Justification specifies where the label appears in relation to its origin (placement location).

Type specifies a label type.

normalLabel displays the label as a text string.

NLPLabel evaluates the label according to netlisting rules and displays the result.

ILLLabel evaluates the label as a SKILL expression and displays the result.

Drafting, when enabled (default), maintains the normal orientation of the text as you rotate a label 180 degrees around a reference point. When this option is disabled, rotated text appears upside down.

Overbar is only a display option. You use this option to specify whether the software displays an overbar over a label. When this option is disabled (default), the software displays underscore characters (_) in the label as part of the label text string. When this option is enabled, the software interprets underscore characters (_) in the label as toggle switches that control where overbars begin and end. Overbars appear above the label, as shown in the examples.

- The property name _abcde puts an overbar over all 5 letters.
- The property name ab_cde puts an overbar over only cde.
- The property name _abc_de puts an overbar over only abc.
- The property name ab_cd_ef_gh_ij puts an overbar over only cd and gh.



Caution

To avoid confusion when labels contain underscore characters, Cadence recommends that you disable this display option. It is provided only for backward compatibility.

Related topics

[Editing Specific Object Properties](#)

Edit Object Properties – Symbol Pin

Use this form to edit symbol pin properties and symbol pin parameters and values. It is accessed by selecting a symbol pin and choosing the *Edit – Properties – Objects* command.

You use the *Previous* and *Next* buttons along with the *Apply To* options below to locate objects in your design window. When there are multiple objects in the selected set, you use the *Next* and *Previous* buttons to locate objects in the set. You use the *Apply To* options to specify the objects the software changes.

Previous cycles backward to view the preceding object in the selected set. When you reach the first object in the selected set, this button has no effect.

Next cycles forward to view the next object in the selected set. When you reach the last object in the selected set, this button has no effect.

Apply To consists of up to three cyclic fields that you use to specify the object, or group of objects in a selected set, whose properties you want to change. The software uses one color to highlight selected objects. But when there are multiple objects in the selected set, the software uses a second color to highlight the current object. And when you configure the cyclic fields to apply changes to multiple objects, the software uses a third color to highlight all the affected objects.

Selection scope specifies whether you want to edit only the current object, only the selected objects in the specified object type group, or all the objects in the specified object type group (which includes both selected and unselected objects of the specified object type).

Object group lists the object types in the selected set. Use this option to locate the first object of a particular object type.

Object filter appears only when the selection scope is set to *all selected* or *all*. This option acts as a secondary filter for further limiting the selection scope. The options vary depending on the selected object type.

Show specifies which properties the system displays on the form.

system displays the system-defined properties for a selected object.

Environment variable: editPropShowSystem

user (default) displays the user-defined properties for a selected object.

Environment variable: editPropShowUser

Virtuoso Schematic Editor User Guide

Form Descriptions

Property lists the properties that the system automatically extracts from the design database.

Value displays the values for the corresponding properties.

Display determines whether the system displays the property value in the schematic. You can toggle this option to *off* to suppress the display.

Direction specifies a direction for the pin. *switch* is maintained solely for backward compatibility: Cadence discourages its use.

When you enable the *Show system* option, the following options appear.

accessDir options let you specify a property that defines the valid direction from which a wire segment can be attached to a pin. You can specify combinations of these options. These options are not referenced in the Virtuoso® Schematic Editor environment; they are referenced in the place-and-route environment.

left specifies that the wire segment must be routed from the left.

right specifies that the wire segment must be routed from the right.

top specifies that the wire segment must be routed from the top.

bottom specifies that the wire segment must be routed from the bottom.

none specifies that a wire segment cannot be routed to the pin.

unknown specifies that any wire segment route is unknown.

The *Add*, *Delete*, and *Modify* options affect only the display of properties on the Edit Objects Properties form. You must click *OK* or *Apply* to apply the changes to objects in your design.

Add displays the Add Property form, which lets you add a new property to the Edit Object Properties form.

Delete displays the Delete Property form, which lets you delete selected properties from the Edit Object Properties form.

Modify displays the Modify Property form, which lets you redefine properties currently displayed on the Edit Object Properties form.

Related topics

Editing Symbol Pin Properties

Supply and Ground Sensitivity

Edit Object Properties – Symbol Shape

Use this form to change the width property of a symbol shape from narrow to wide or to reposition a symbol shape using coordinates. It is accessed by selecting a shape and choosing *Edit – Properties – Objects*.

You use the *Previous* and *Next* buttons along with the *Apply To* options below to locate objects in your design window. When there are multiple objects in the selected set, you use the *Next* and *Previous* buttons to locate objects in the set. You use the *Apply To* options to specify the objects the software changes.

Previous cycles backward to view the preceding object in the selected set. When you reach the first object in the selected set, this button has no effect.

Next cycles forward to view the next object in the selected set. When you reach the last object in the selected set, this button has no effect.

Apply To consists of up to three cyclic fields that you use to specify the object, or group of objects in a selected set, whose properties you want to change. The software uses one color to highlight selected objects. But when there are multiple objects in the selected set, the software uses a second color to highlight the current object. And when you configure the cyclic fields to apply changes to multiple objects, the software uses a third color to highlight all the affected objects.

Selection scope specifies whether you want to edit only the current object, only the selected objects in the specified object type group, or all the objects in the specified object type group (which includes both selected and unselected objects of the specified object type).

Object group lists the object types in the selected set. Use this option to locate the first object of a particular object type.

Property lists the properties that the system automatically extracts from the design database.

Value specifies the type of symbol shape.

The form displays various options depending on the shape type you select.

The following options appear when you select a line shape.

Width specifies whether the line shape is narrow (default) or wide. You can edit the text field when you enable the *wide* option.

Point List specifies the coordinates for the points that make up the shape.

Virtuoso Schematic Editor User Guide

Form Descriptions

The following option appears when you select a polygon shape.

Point List specifies the coordinates for the points that make up the shape.

The following options appear when you select a rectangle shape.

Upper Right specifies coordinates that define the upper right-hand corner of the shape.

Lower Left specifies coordinates that define the lower left-hand corner of the shape.

The following options appear when you select either a circle shape or an ellipse shape. The *Shape* text field displays *ellipse* for both circle and ellipse shapes.

Center specifies the coordinates that define the center of the shape.

X Radius specifies the coordinate that defines the Xradius of the shape.

Y Radius specifies the coordinate that defines the Yradius of the shape.

The following options appear when you select an arc shape.

Center specifies the coordinates that define the center of the shape.

X Radius specifies the coordinate that defines the Xradius of the shape.

Y Radius specifies the coordinate that defines the Yradius of the shape.

Start Angle specifies the first point of the arc.

Stop Angle specifies the second point of the arc.

Related topics

[Editing Symbol Shape Properties](#)

Edit Object Properties – Wire Name

Use this form to edit wire name parameters and values. It is accessed by selecting a wire name/label on the design canvas and choosing the *Edit – Properties – Objects* command.

You use the *Previous* and *Next* buttons along with the *Apply To* options below to locate objects in your design window. When there are multiple objects in the selected set, you use the *Next* and *Previous* buttons to locate objects in the set. You use the *Apply To* options to specify the objects the software changes.

Previous cycles backward to view the preceding object in the selected set. When you reach the first object in the selected set, this button has no effect.

Next cycles forward to view the next object in the selected set. When you reach the last object in the selected set, this button has no effect.

Apply To consists of up to three cyclic fields that you use to specify the object, or group of objects in a selected set, whose properties you want to change. The software uses one color to highlight selected objects. But when there are multiple objects in the selected set, the software uses a second color to highlight the current object. And when you configure the cyclic fields to apply changes to multiple objects, the software uses a third color to highlight all the affected objects.

Selection scope specifies whether you want to edit only the current object, only the selected objects in the specified object type group, or all the objects in the specified object type group (which includes both selected and unselected objects of the specified object type).

Object group lists the object types in the selected set. Use this option to locate the first object of a particular object type.

Object filter appears only when the selection scope is set to *all selected* or *all*. This option acts as a secondary filter for further limiting the selection scope. The options vary depending on the selected object type.

Attachment performs two functions.

- When the *Label* entry is already associated with an object, the software highlights the associated object in the cellview.
- It lets you associate the *Label* entry with a wire segment in the cellview by automatically drawing a line from the wire name to the object when you move the cursor into the design window.

Label specifies the net name.

Virtuoso Schematic Editor User Guide

Form Descriptions

Value determines whether the system displays the property value in the schematic. You can toggle this option to *off* to suppress the display.

Height specifies the height of the name.

Font specifies a font for the name.

Justification specifies where the wire name appears in relation to its origin (placement location).

Bundle Display specifies the bundle display mode, *horizontal* or *vertical*. A bundle is a wire name consisting of simple names separated by a comma; for example, S,R.

Related topics

[Editing Wire Name Properties](#)

Edit Object Properties – Wire Segment

Use this form to edit the wire properties. It is accessed by selecting a wire and choosing the *Edit – Properties – Objects* command.

You use the *Previous* and *Next* buttons along with the *Apply To* options below to locate objects in your design window. When there are multiple objects in the selected set, you use the *Next* and *Previous* buttons to locate objects in the set. You use the *Apply To* options to specify the objects the software changes.

Previous cycles backward to view the preceding object in the selected set. When you reach the first object in the selected set, this button has no effect.

Next cycles forward to view the next object in the selected set. When you reach the last object in the selected set, this button has no effect.

Apply To consists of up to three cyclic fields that you use to specify the object, or group of objects in a selected set, whose properties you want to change. The software uses one color to highlight selected objects. But when there are multiple objects in the selected set, the software uses a second color to highlight the current object. And when you configure the cyclic fields to apply changes to multiple objects, the software uses a third color to highlight all the affected objects.

Selection scope specifies whether you want to edit only the current object, only the selected objects in the specified object type group, or all the objects in the specified object type group (which includes both selected and unselected objects of the specified object type).

Object group lists the object types in the selected set. Use this option to locate the first object of a particular object type.

Object filter appears only when the selection scope is set to *all selected* or *all*. This option acts as a secondary filter for further limiting the selection scope. The options vary depending on the selected object type.

Show specifies which properties the system displays on the form.

system displays the system-defined properties for a selected object.

Environment variable: [editPropShowSystem](#)

Note: The following option appears when you turn on the *system* option.

Point List displays the coordinates for the first and second point.

user (default) displays the user-defined properties for a selected object.

Virtuoso Schematic Editor User Guide

Form Descriptions

Environment variable: [editPropShowUser](#)

Property lists the properties that the system automatically extracts from the design database.

Value displays the values for the corresponding properties.

Display determines whether the system displays the property value in the schematic. You can toggle this option to *off* to suppress the display.

Net Name displays the current name of the wire. This field is editable if you have already added a wire name label to a net. Changing the value of the *Net Name* will rename all the labels associated with the selected wire's net. This will therefore not work for those nets that have not had a label applied using the *Create – Wire Name* option.

Width specifies whether the wire segment is narrow or wide. You can edit this field when you enable the *wide* option.

Signal Type specifies the signal type of the selected wire segment. Refer to [Defining Signal Types](#).

Color specifies the color of the wire.

Environment variables: [editPropShowSystem](#)

Line Style specifies the line style of the wire.

The *Add*, *Delete*, and *Modify* options affect only the display of properties on the Edit Objects Properties form.

Add displays the Add Property form, which lets you add a new property to the Edit Object Properties form. For more details, see the [Add Property – Edit Object Properties](#) form.

Delete displays the [Delete Property form](#), which lets you delete selected properties from the Edit Object Properties form.

Modify displays the [Modify Property form](#), which lets you redefine properties currently displayed on the Edit Object Properties form.

Related topics

[Editing Wire Properties](#)

Edit Pin Order

Use this form to redefine the logical pin ordering for netlisting. It is accessed using the *Edit – Properties – Pin Order* command.

Copy From copies a pin order property from a list of available views.

Move Up (arrow) moves the selected pin up one position.

Move Down (arrow) moves the selected pin down one position.

Move To Top (arrow) moves the selected pin to the top of the list.

Move To Bottom (arrow) moves the selected pin to the bottom of the list.

Save if default pin order saves the order of pins if it is the same as the default order.

Reset resets the pin order to the default order. The default port order is output input inputOutput, and the pins in each type are sorted alphanumerically.

Related topics

[Designating the Order of Pins](#)

Edit Schematic Sheet Number

Use this form to change the numbering order of sheets. It is accessed using the *Sheet – Renumber* command from an index schematic.

From Number specifies the current sheet number.

To Number specifies the new sheet number.

Related topics

[Creating a Multisheet Schematic](#)

Editor Options

Use this form to change general options for the schematic editor using the *Options – Editor* command.

Commands Tab

Add Instance Browser Type specifies which browser the system opens when you click the *Browse* button on the Add Instance form.

library (default) specifies the Library Browser.

component specifies the Component Browser.

Environment variable: browserType

Show Schematic Views in Browser shows schematic views in the Library Browser when *Add Instance Browser Type* is *library*.

Environment variable: browserShowSchViews

Block Pin Direction Rule specifies which pin direction rule the system uses to create pins automatically when you connect a wire to a block.

edge (default) specifies the pin direction based on the edge where the system creates the pin. When you add a wire to the left edge of a block, the system creates an input pin. When you add a wire to the right edge of a block, the system creates an output pin. When you add a wire to the top or bottom of a block, the system creates an I/O pin.

order of entry assigns the pin direction based on how you implement the *Create Wire* command. When you start a wire from the edge of a block, the system creates an output pin. When you terminate a pin on the edge of a block, the system creates an input pin.

Environment variable: blockDirRules

Wire Snapping lets you use the s key to quickly complete a wire drawing action. When wire snapping is enabled, the system highlights an unconnected pin in the design when you move the cursor near the pin.

Environment variable: snapEnabled

Automatic Solder Dots when enabled (default), automatically creates a solder dot when you create wire intersections. The system does not create solder dots when you draw a wire over another wire.

Environment variable: autoDot

Virtuoso Schematic Editor User Guide

Form Descriptions

Solder Dot Wire Overlap specifies the amount that a solder dot extends outward from the thickest wire segment the dot connects.

Environment variable: dotOverlap

Symbol Generator Pin Sorting specifies how the text-to-symbol generator sorts the symbol pins onto the edge of the new symbol.

alphanumeric sorts pins by pin name.

geometric sorts pins by their relative positions in the schematic and is valid only when a schematic exists for the specified cellview.

Environment variable: ssgSortPins

Add Symbol Pins as Instances specifies how the symbol editor creates pins when you add pins to a symbol using the *Create – Pin* command or creates a symbol using the *Create – Cellview – From Cellview* command.

off (default) creates a graphical symbol pin shape based on a symbol pin master. After you place symbol pin shapes, any changes you make to the original symbol pin master do not affect the symbol pin shapes based on the symbol pin master.

on creates an instance of a symbol pin master. After you place symbol pin instances, any changes you make to the symbol pin master also affect the appearance of all the existing instances created from the symbol pin master.

Environment variable: symbolPinFlatten

Repeat Commands specifies whether repeatable commands remain active after you complete an action.

off specifies that repeatable commands cannot be repeated.

on (default) specifies that repeatable commands remain active until you press the Esc key. However, some repeatable commands do not remain active when you enable this option if you have preselected an object. The *Copy*, *Move*, and *Edit* commands are examples of such commands.

Environment variable: modalCommands

Cross Selection controls whether object selection is exported to other tools.

off (default) does not export the selection to other tools.

on exports the selection to tools that are open and which can support the selection.

Environment variable: broadcast

Virtuoso Schematic Editor User Guide

Form Descriptions

Gravity affects how the *Create – Wire* command is completed. When you add a wire to a block instance, the system terminates the wire on the edge of the block. When you add a wire to another type of instance, the system terminates the wire on the instance pin.

off disables this option.

on (default) enables this option.

Environment variable: gravityOn

Minimum Marker Size specifies the minimum size of any error or warning markers that the system generates when you check your design.

Environment variable: minMarkerSize

View Name List lists the order in which view names are displayed, if they exist. You use *View Name List* in conjunction with the commands *Edit – Hierarchy – Descend Edit* and *Edit – Hierarchy – Descend Read* when descendTarget is set to *use viewNameList*.

Environment variable: viewNameList

Auto Wire Stub for Bus Terminals set the default size for creating wire stubs for bus terminals; *wide* or *narrow*. Wire stubs require a dedicated wire width, either wide or narrow. When they are wide they share the same width as other wide wires. Bus terminals take wire stubs the same as other terminals.

Environment variable: autoWireStubBusTerms

Auto Constraint Notes enables or disables the display of constraint notes.

You can also enable constraint notes using the following environment variable:

```
constraint autoConstraintNotes boolean t
```

Choosing the *enable* option in the Editor Options form automatically sets the constraint notes environment variable to `t` for new constraints, and will also create notes for existing constraints. However, if the constraint notes environment variable is set through the CIW, constraint notes will only be created for new constraints, not existing constraints.

If you still want to use the CIW to enable constraint notes, it is therefore recommended that you do this through `.cdsinit`, or before launching the tool, so that all constraint notes are created when the tool is launched.

Sync on Design Open enables or disables the synchronization of the constraint notes between schematic and constraint views. When you enable automatic generation of constraint notes, the inactive radio buttons adjacent to *Sync on Design Open* become active. By default, *disable* is selected. If you want to synchronize the constraint notes

Virtuoso Schematic Editor User Guide

Form Descriptions

between schematic and constraint views when you open the design or change the mode of the constraint view, select the *enable* radio button.

Sync on Design Check becomes active when you enable automatic generation of constraint notes. By default, *disable* is selected. Select the *enable* radio button adjacent to this control if you want the constraint notes to be synchronized whenever connectivity is updated.

Connectivity Tab

Incrementally update connectivity while editing controls which connectivity mode is used to update the connectivity. The default is on, which means that connectivity is automatically checked and updated while you edit. If switched off, batch connectivity mode is used, requiring you to explicitly request a connectivity update using one of the Check commands. For more details, see [Updating Connectivity](#).

Environment variable: [incrementalConnTimeout](#)

Bus Range Order specifies which order is used to generate buses. The default is *ascending*.

Environment variables: [netNameRangeAuto](#), [netNameRangeDescend](#)

Auto Bus Range Order auto detects the range order of the object for which a bus has to be generated and assigns the same range order to the bus.

Net Name Method specifies a net naming mechanism for schematic extraction.

prefix (default) generates a unique name for nets that you do not explicitly label or connect to a pin. The unique name consists of the prefix that you specify in the *Net Name Prefix* field followed by a number.

derived generates a net name from the driving instance name and the driving terminal name. The derived net name is of the form *instName_pinName*.

Note: Instance pins named with bundle names, for example, A, B, C, are not used to derive system-generated names.

Environment variable: [netNameMethod](#)

Net Name Separator specifies the field separator for derived names. The underscore character (`_`) is the default separator. You can edit the separator when you select the *derived* option in the *Net Name Method* field.

Environment variable: [netNameSeparator](#)

Virtuoso Schematic Editor User Guide

Form Descriptions

Net Name Prefix specifies the prefix for all new net names generated. The default is *net*. You can edit this field when you select the *prefix* option.

Environment variable: netNamePrefix

Inherited Conn Tab - System Generated Net Expressions

Pins

Derive property name and default net name from pin name enables the *Property Name* and *Default Net Name* fields on the Create Pin form to be automatically seeded with values derived from the *Pin Name* currently being added. These values will continue to update if multiple pins are added simultaneously.

Environment variable: inhConnGenPinNetExpr

Disable editing of property and default net name fields disables the net expression editing fields on the Create Pin form to prevent modification of the generated values.

Environment variable: inhConnDisablePinFields

Wires

Derive default net name from property name enables the *Default Net Name* field in the Create Net Expression, to be automatically seeded based on the *Property Name* specified. This value will continue to update if multiple pins are added simultaneously.

Environment variable: inhConnGenNetExprDefNet

Disable editing of default net name field disables editing of the *Default Net Name* field on the Create Wire Name and Create Net Expression forms to prevent modification of the generated values.

Environment variable: inhConnDisableDefNetField

Labels Tab

Pin Label Overrides

Font Style sets the font style for a label. The valid values are *euroStyle*, *gothic*, *math*, *roman*, *script*, *stick*, *fixed*, *swedish*, *milSpec*. The default is *stick*.

Environment variables: pinLabelFontStyle, pinLabelSetFontStyle

Font Height sets the default font height (in user units) for a label. The default is 0.625.

Environment variables: pinLabelFontHeight, pinLabelSetFontHeight

Virtuoso Schematic Editor User Guide

Form Descriptions

Justification determines the pin label placement. Valid Values: *upperLeft*, *centerLeft*, *lowerLeft*, *upperCenter*, *centerCenter*, *lowerCenter*, *upperRight*, *centerRight*, *lowerRight*. The default is *lowerCenter*.

Environment variables: pinLabelJustify, pinLabelSetJustify

Drafting maintains the normal orientation of the text as you rotate a label 180 degrees around a reference point. When this option is disabled, rotated text appears upside down. The default is for this option to be *on*.

Environment variables: pinLabelDrafting, pinLabelSetDrafting

Overbar determines whether a bar (or overline) is placed over the label text that is placed between underscore characters. The default is for this option to be *off*.

Environment variables: pinLabelOverbar, pinLabelSetOverbar

Ignore Properties Tab

Properties

ignore property, when set on an instance, prevents the instance from being generated in the layout. It is a Boolean type property having values as `TRUE` or `FALSE`.

lvIgnore property, when set on an instance, ignores the instance during CDL netlisting. It is a Boolean type property having values as `TRUE` or `FALSE`.

nlAction property, when set on an instance, ensures that the instance is not netlisted. It is a string type property.

`nlAction="ignore"` applies to all the simulators.

nlIgnore property, when set on an instance, prevents specific netlisters from processing the instance. The value of this property is a string containing a space-separated list of netlisters that should ignore the instance. For example, to prevent only the Spectre netlister from netlisting an instance, you can specify `nlIgnore="spectre"`.

Related topics

[Customizing Editor Options](#)

Export Schematic Hierarchy

Use this form to export your schematic design hierarchy to HTML. It is accessed in Schematics XL using the *File – Export HTML* command.

Note: Exporting design hierarchy content in HTML format from a configured or a non-configured schematic largely follows the same process. However, in a configured schematic, you do not specify the Starting Cell, View List or Stop List options (as they are already specified within the configuration itself).

Starting Cell Specify to export HTML for the sub-design starting from the *current* cellview in a traversed hierarchy, or from the *top* level design. This option is available for a non-configured schematic only.

View List / Stop List Specify the *View List* and *Stop List* (as similar to netlisting) to determine how the design hierarchy is to be traversed. This option is available for a non-configured schematic only.

Cellviews that are found on the *Stop List* will be used to terminate the traversal.

Note: The initial values specified in the *View List* field are the same as those specified in the *View Name List* option in the Editor Options form (*Options – Editor*).

Physical Config? Specify whether the *Physical Config* is to be used to perform hierarchy traversal. If you choose not to use the physical configuration in this way, standard traversal as performed by the Cadence Hierarchy Editor, will be used. This option is available for a configured schematic only.

Export to Directory Specifies the target directory.

Note: If the target directory does not exist, it will be created automatically.

Image Color Options Specify if the exported rendered images are to be in *full color*, *grayscale on white*, or *black on white*.

Note: The *full color* option is recommended for viewing the document, while *grayscale on white* and *black on white* may be more suitable for printing purposes.

Open in Browser Check this option so that Schematics XL attempts to open your default browser (the browser will be directed to the newly created index.html file in the specified target directory).

You can specify your default browser as follows:

```
setenv CDS_DEFAULT_BROWSER firefox
```

Related topics

[Exporting a Schematic Hierarchy to HTML](#)

Find Marker

If your design has errors or warnings, the *Check – Find Marker* command lets you view them by listing the corresponding error and warning messages in the Find Marker form.

To Ignore appears only when the *Show Ignored* option is *on* (default). This option ignores violations indicated by markers in future rule extractions. The schematic editor opens a column in the list box. It then displays an asterisk in the column when you apply an *ignoreCheck* property to a corresponding marker.

Reason displays marker severity levels, associated messages, and the cellview name where the errors and warnings occurred.

Zoom To Markers displays a magnified view of the specified marker in the design window if it is turned on.

Environment variable: markerZoom

Scale appears only when you turn on *Zoom To Markers*. This option lets you specify the zoom scale. The smaller the scale setting, the greater the degree of magnification.

The following four options let you control which markers the system displays in the list box.

Show Ignored displays markers for objects that have an *ignoreCheck* property. When *Show Ignored* is *on* (default), the software displays the *To Ignore* label above the list box and opens a column under the label. The software displays an asterisk in the column when you apply an *ignoreCheck* property to a corresponding marker.

Environment variable: markerShowIgnore

Ignore automatically assigns an *ignoreCheck* property to the object highlighted in the list box.

Restore deletes an *ignoreCheck* property from an object highlighted in the list box.

Restore All deletes the *ignoreCheck* property from all objects.

Severity specifies which messages are displayed in the list box.

all displays all messages of the types selected in the group box.

none displays none of the messages of the types selected in the group box.

fatal error displays fatal error messages.

error displays error messages.

acknowledged warning displays acknowledged warning messages.

critical error displays critical error messages.

signed off error displays signed-off error messages.

info displays information messages.

signed off critical error displays signed-off critical error messages.

warning displays only warning messages.

annotation displays annotation messages.

Environment variable: [markerSeverity](#)

Search Scope lets you specify the scope of a search for schematic objects.

cellview (default) searches only the current cellview.

hierarchy starting from top cellview searches the design hierarchy, starting with the *top* cellview in the window.

hierarchy starting from current cellview searches the design hierarchy, starting with the *current* cellview in the window.

Environment variable: [markerSearchScope](#)

Sources displays a button and label for the sources of the error and warning messages. When there is more than one source for the error and warning messages, you can toggle the source buttons to control which messages the system displays in the list box.

SRC specifies the schematic rules checker as the source.

VIC specifies the cross-view checker as the source

conn specifies connectivity as the source.

load specifies markers created when you opened the schematic as the source.

none appears when there are no messages in the list box.

Related topics

[Resolving Check Errors](#)

Flatten

Use this form to bring the contents of an instance up to the current level to allow editing while preserving the connectivity of the flattened instance. It is accessed using the *Edit – Hierarchy – Flatten* command.

Library corresponds to the library of the instance selected to be flattened. This field is not editable.

Cell corresponds to the cell of the instance selected to be flattened. This field is not editable.

View corresponds to the view of the instance selected to be flattened.

Connect Mode lets you specify how to connect the flattened content to the existing instance.

by-name connects the flattened content to the existing schematic using connect-by-name.

flight connects the flattened content to the existing schematic using flight lines.

wire connects the flattened content to the existing schematic using routed wires.

Environment variable: [flattenConnectMode](#)

Create symbol notes select to create a symbol note and a note line when flattening a symbol in a schematic.

Related Topics

[Flattening an Instance](#)

Go To Pin

Use this form to select the pin or offsheet connector in your design to follow throughout the multisheet. It is accessed using the *Sheet – Go To Pin* command.

Order specifies, in a multisheet schematic, the relative location of other pins that reference the same net. The choices are *next*, *previous*, *first*, and *last*.

The reference list for a pin or offsheet connector is a list of zones identifying where other pins on other sheets reference the same net. The reference lists for the pins in the multisheet schematic are generated by the *Create Pin Reference* command.

Related topics

[Zooming In to View All Cross-Referenced Pins One by One](#)

Go To Sheet

Use this form to specify which sheet within a multisheet schematic is displayed. It is accessed using the *Sheet – Go To* command.

index displays the index sheet for the multisheet schematic in the current design window.

first displays the first sheet of the multisheet schematic in the current design window.

last displays the last sheet of the multisheet schematic in the current design window.

next displays the next sheet in sequence of the multisheet schematic in the current design window.

previous displays the previous sheet in sequence of the multisheet schematic in the current design window.

number enables the text field so that you can specify which sheet of the multisheet schematic to display in the current design window.

Related topics

[Going to and from the Index Schematic](#)

Import Symbol

Use this form to import an existing symbol into the symbol window which you can then modify to create a new symbol cellview. It is accessed using the *Create – Import Symbol* command.

Library Name specifies the library that contains the symbol.

Cell Name specifies the symbol cell name.

View Name specifies the symbol view name.

Browse opens the Library Browser. You can use the browser to fill in the form with the symbol information.

Rotate turns an object in 90-degree increments counterclockwise around the current reference point.

Sideways mirrors the object across an imaginary vertical line passing through the current reference point.

Upside Down mirrors the object across an imaginary horizontal line passing through the current reference point.

Related topics

[Importing Symbols](#)

Left, Right, Top, Bottom Pin Attributes

Use this form to change the attributes of a pin. It is accessed by selecting *List* from one of the *Attributes* cyclic fields on the [Symbol Generation Options](#) form.

Name lists the pins by name.

Direction contains the direction attributes you can apply to the pins. *switch* is maintained solely for backward compatibility: Cadence discourages its use.

Pin Graphic contains the graphic attributes you can apply to the pins.

Related topics

[Editing Pin Attributes](#)

Load Label Display

Filename specifies a previously saved label display configuration file that you want to use.

Load Schematic Defaults

Use this form to load editor variables and form defaults from a specified file using the command *Options – Load Defaults*.

File Name specifies the name of the file that contains the variables and form defaults you want to load. The default is a file named .cdsenv in your home directory.

Environment variable: defaultsFile

Related topics

[Loading Form Field Default Settings](#)

Make Cell

Use this form to create a cell by cutting out an area from the current window. It is accessed using the *Edit – Hierarchy – Make Cell* command.

Cut area corresponds to the area selected on the canvas. You can edit the *Top*, *Bottom*, *Left*, and *Right* values to update the selection.

Add lets you draw a rectangle on the canvas that is added to the selection.

Subtract lets you draw a rectangle on the canvas that is removed from the selection.

Redraw lets you draw a new selection on the canvas.

Destination specifies where the new cell is to be created.

Library corresponds to the library where the new instance is to be created.

Cell corresponds to the cell where the new instance is to be created.

View corresponds to the view in which the new instance is to be created.

Overwrite existing cellview overwrites the existing cellview in the destination area.

Simplify iterated names checks whether all instances and wire labels in the selection have a common iteration factor. If they do, this option removes the common factor and creates an iterated instance of the new scalar cell.

Place symbol automatically generates a symbol for the newly created cell. If not selected, the new cell is placed directly in the current schematic.

Remove duplicated symbol pins removes any duplicate symbols that were generated during symbol placement. Only available when *Place symbol* is selected.

Related Topics

[Creating a Cellview from the Canvas \(Make Cell\)](#)

Modify Generic – VHDL Properties

Use this form to modify properties that represent VHDL generics. It is accessed by selecting the *Modify* button on the VHDL Properties form.

Name specifies the generic name.

Type specifies the generic type.

Value (Optional) specifies a value for the generic.

Related topics

[Modifying VHDL Properties](#)

Modify Property – Edit Object Properties

Use this form to modify object properties. It is accessed by selecting *Modify* on the Edit Object Properties – Basic form.

Type specifies a property type.

Name specifies the name of the property you want to modify.

Value specifies a value for the property.

Note: To define most properties, you need to specify only the name, type, and value. However, when you specify *string*, *int*, or *float* as the type, one of the following fields appears on the form.

Choices appears when *Type* is set to *string*. If the string can have more than one value, type the possible values. Entries in this field must match an entry in the *Value* field.

Min and **Max** appear when *Type* is set to *int* or *float*. Specify a minimum and maximum range for the integer or floating-point value.

Related topics

Modifying Object Properties

Move

Use this form to change the options for the *Move* command. It is accessed using the *Edit – Move* command.

Snap Mode specifies the angles along which you can move objects within the cellview.

anyAngle lets you move the objects along any angle.

diagonal restricts the movement of objects to points along 45-degree, 90-degree, and 180-degree angles from the current location.

orthogonal restricts the movement of objects to points that are along the X and Y axes from the current location.

Rotate turns an object in 90-degree increments counterclockwise around the current reference point.

Sideways mirrors the object across an imaginary vertical line passing through the current reference point.

Upside Down mirrors the object across an imaginary horizontal line passing through the current reference point.

Related topics

[Moving Objects](#)

Net Expression Available Property Names

Use this form to manipulate inherited connections. It is accessed by selecting one or more instances from the current design canvas and choosing the *Edit – Net Expression – Available Properties* command.

The upper section of the form displays the instance underneath which an inherited connection was located, the default value for that connection, and a preview of the resolution *Action* currently selected, if any.

Action Select an action to perform on the selected inherited connection (*Instance*).

■ *Leave as is*

No action is performed on the connection. It will be left to propagate upwards through the hierarchy.

Selecting this action disables all of the other option edit fields.

■ *Override with specific net*

This action resolves the selected inherited connection by overriding its default with a specific net.

A netSet property is added to the associated instance, the value of this property being the name of the specified net.

When this action is selected, the *Override net name* field is enabled.

■ *Convert to normal pin*

This action transforms the inherited connection into an explicit pin on the schematic. Connections can now be made to this pin, instead of having to use the inherited connections mechanisms.

A pin will be added to the current schematic, if required, and a netSet is created on the appropriate instance (the value being the name of the pin).

When this action is selected, the *Pin Name* and *Direction* fields are enabled.

■ *Change connection parameters*

Net expressions can be re-parameterized by assigning a netSet containing a new net expression. This action provides access to this feature by enabling the *Connection name* and *Default net name* fields to allow editing of the existing connection parameters.

Virtuoso Schematic Editor User Guide

Form Descriptions

When applied, this action constructs a new net expression and assigns it to a netSet on the appropriate instance.

Note: The *Connection name* and *Default net name* fields will initially be seeded with the current parameter values.

■ *Convert to inherited pin*

This action allows the selected net expression to propagate up through the hierarchy, and will also create an explicit pin for it at the current level of the hierarchy. This allows clients to make use of inherited connections, with the benefit of improved compatibility throughout the design flow.

This action enables the *Pin Name* and *Direction* fields, as well as the *Connection name* and *Default net name* fields to optionally allow re-parameterization of the propagated inherited connection (the latter two field values are seeded with the current parameter values, so can left as-is, if required).

On action, a new pin will be created in the current schematic and a net expression with the specified parameters is attached. If the parameters have been modified from their initial values, a newly constructed net expression is assigned to a netSet parameter on the selected instance. This step ensures that the pin and instance are using the same net expression.

Related topics:

[Manipulating Inherited Connections](#)

Net Expression Evaluated Names

Use this form to view all the different evaluated names that result from evaluations of all the net expressions in the hierarchy below a selected instance. The form is accessed using the *Edit – Net Expression – Evaluated Names* command.

Evaluated Name displays all the different evaluated names that result from evaluations of all net expressions below the selected instances for the current hierarchy.

Number of Cellviews displays the number of cellviews that contain net expressions that evaluate to a particular connection.

Number of Occurrences displays the number of occurrence paths, for example, unique lineages to net expressions that evaluate to that particular signal.

List Cellviews with Selected Evaluated Name opens the Cellviews with Evaluated Net Expression Name form showing the library, cell, and view names of the cellviews within which net expressions are evaluated to the selected evaluated result.

List Occurrences for Selected Evaluated Name opens the Occurrence Paths to Evaluated Name form showing the lineages leading to the net expressions that evaluated to the selected evaluated result.

Related topics

- [Viewing the Evaluated Net Names](#)
- [Viewing the Full List of all Hierarchical Paths](#)

New File

Use this form to open a new cellview from the schematic editor. It is accessed using the *File* – *New* command.

Library the library in which to store the new cellview.

Cell enter a new cell name or leave as-is with the current cell name entered.

View enter a name for the new cellview.

Type choose the view type, for example *schematic* or *config*, that is to be associated with the new cellview.

Open with specifies the application that the new cellview should automatically be opened with.

Always use this application for this type of file select to associate the selection made in the *Open with* field with the current *View* type selected.

Library path file specifies the library path where the new cellview is to be stored in.

Open in choose in what location the new cellview should be opened in.

new tab opens the new cellview in a new tab in the current session window.

current tab opens the new cellview in the current tab in the current session window.

new window opens the new cellview in a new session window.

Related topics

[Opening a New Cellview](#)

Occurrence Paths to Evaluated Name

Use this form to view the full list of all hierarchical paths (for example, all lineages) that can be followed down to all net expressions that evaluate to the selected evaluated name. It is accessed using the *Edit – Net Expression – Evaluated Names* command and selecting *List Occurrences for Selected Evaluated Name*.

Occurrence Path shows the full list of all hierarchical paths that can be followed down to a net expression that evaluates to the selected evaluated name.

Cellview shows the cellview name in which the net expression resides.

Property Name shows the property name of the net expression.

Default Value shows the default value of the net expression.

Display occurrence paths with gives you three choices for displaying the occurrence paths to evaluated names.

instance names sets the display of the occurrence paths to instance names.

cell names sets the display of the occurrence paths to cell names.

instance and cell names sets the display of the occurrence paths to instance and cell names.

Related topics

[Viewing the Full List of all Hierarchical Paths](#)

Open File

Use this form to open an existing cellview. It is accessed using the *File – Open* command.

Library Name lets you choose the library that contains the existing cellview.

Cell Name lets you specify a cell name by either entering the cell name in the field or by selecting it from the *Cell Names* list box.

Cells lists all the cells associated with the current *Library Name*.

View Name select the name of the view to be opened.

Type specifies the cell view type associated with the view to be opened

Browse invokes the Library Browser so you can select alternative libraries, cells, or views.

Open with specifies the application that the cellview will be opened with.

Always use this application for this type of file select to associate the selection made in the *Open with* field with the current *View* type selected.

Open for specifies to open the chosen file in *edit* or *read* mode.

Library path file specifies the library path from where the cellview is being opened from.

Open in specifies that the chosen cellview be opened in a *new tab* in the current session, the *current tab* in the current session window, or a *new window* in a new session.

Related topics

[Opening an Existing Cellview](#)

Plot Options

Use this form to set several plot options: plotter name, paper size, page orientation, size and position of plotting, area, number of copies, plot time, and plot to file. It is accessed by selecting *Plot Options* on the Submit Plot form.

Display Type specifies the plotter display type. The default is *display*, which represents a color monitor. Each plotter display type defines the colors, line styles, and stipple patterns for any layers.

Plotter Name contains a list of the available plotters. The system creates the list of available plotters and default settings from the *.cdsplotinit* file.

Paper Size contains a list of the available paper sizes. The system creates the list of available plotters and default settings from the *.cdsplotinit* file.

Orientation specifies the plot orientation of the paper.

portrait plots the drawing vertically.

landscape plots the drawing horizontally.

automatic (default) lets the software select the best orientation.

Scale displays the ratio of the drawing area of the paper to the plot data area.

Center Plot, when enabled, automatically adjusts the offset settings to print the plot in the center of the paper.

Fit to Page sets the *Plot Size* and *Offset* values to fit the plot area specified in *Paper Size*.

Plot Size specifies the length and width of the plot. The cyclic field lists the available units. The default is the width and height of the plot area of the paper specified in *Paper Size*, in user units.

Offset lets you specify the offset of the plot relative to the lower left corner of the paper using X and Y coordinates. The default is 0, which is the lower left corner.

Total Plot Size displays the sum of the *Plot Size* and *Offset* values.

Image Position shows the image position window and how the design will be plotted on one or multiple sheets.

Total Pages displays the number of pages needed.

Number Of Copies specifies the number of copies the plotter generates.

Virtuoso Schematic Editor User Guide

Form Descriptions

Local Tmp Directory specifies the location of the intermediate plot file. The default location is the value returned by `getTempDir()`.

Queue Plot Data At specifies the time and day when you want the system to generate the output plot. The cyclic fields contain lists of the available times and days.

Send Plot Only To File lets you specify the base filename where you want the system to store the output plot. The system appends `lib/cell/view` to the end of the base name that you specify. For example, when you specify a base name, such as `myPlot`, the system designates the output files as `myPlot@libName,cellName,viewName`.

Mail Log To collects log information in a file and sends the file to you through email.

Related topics

- [Setting Plot Options](#)
- [Plotting Designs](#)

Register Net Name per Terminal

Use this form to register default net names, or net expressions, on a terminal basis. It is accessed by selecting the required instance terminals and choosing the *Options – Register Net Name per Terminal* command.

Net Expression choose to specify a net expression to define the default global signal name for a connection, and the name of the property that can be used to override that global signal name.

Property Name (only displayed when *Net Expression* is checked) Specifies the name of the property that can be used to override the default global signal.

Global Net Name (only displayed when *Net Expression* is checked) Specifies the name of the global net to be used to make connections across the design hierarchy (without the need for pins and wire to be used to explicitly pass a signal through the hierarchy).

Net Name the default net name to be registered to the selected instance terminals. This option will be removed if the *Net Expression* option is checked.

Related topics

[Registering Default Net Names](#)

Renumber Instances

Use this form to renumber instances to make the numbers consecutive which you may want to do if several instances have been deleted in your cellview. It is accessed using the *Edit – Renumber Instances* command.

Sequence instances by

filling the voids fills the lowest available void with the instance with the highest index, until all the instances have been renumbered.

X+Y+ renumbers from bottom left to top right of the schematic by scanning horizontally.

X+Y- renumbers from top left to bottom right of the schematic by scanning horizontally.

X-Y+ renumbers from bottom right to top left of the schematic by scanning horizontally.

X-Y- renumbers from top right to bottom left of the schematic by scanning horizontally.

Y+X+ renumbers from bottom left to top right of the schematic by scanning vertically.

Y+X- renumbers from bottom right to top left of the schematic by scanning vertically.

Y-X+ renumbers from top left to bottom right of the schematic by scanning vertically.

Y-X- renumbers from top right to bottom left of the schematic by scanning vertically.

Start resulting sequence with index specifies the number of the first renumbered instance. The field entry limit here is $(2^{31} - 1)$.

Scope specifies which instances in the hierarchy the system renumbers.

cellview (default) renumbers the instances only in the current cellview.

hierarchy renumbers all instances within the hierarchy.

top (default) specifies that the system renumber instances from the top-level cellview.

current cellview specifies that the system renumber instances in all cellviews below and including the current cellview.

library renumbers all instances within the library. When you select *library*, a text field appears. Type the name of the library containing the instances you want to renumber.

Apply to all instances of

any master renumbers all the instances of the schematic.

Virtuoso Schematic Editor User Guide

Form Descriptions

same master rennumbers all the instances of the schematic specified by the following fields:

Library Name specifies the library containing the instances to be renumbered.

Cell Name specifies the cell containing the instances to be renumbered.

View Name specifies the view containing the instances to be renumbered.

Verbose when enabled (default), displays in the Command Interpreter Window (CIW) the old and new numbering for each renumbered instance.

Related topics

[Renumbering Instances](#)

Reset Invisible Labels

Use this form to manually or automatically reset the display of labels previously turned off on the Edit Object Properties – Label or Edit Object Properties – Symbol Label forms. It is accessed using the *View – Reset Invisible Labels* command.

visible causes the highlighted labels to remain visible in the design after you click the *Hide* button.

invisible causes the highlighted labels to revert to the invisible state after you click the *Hide* button.

Related topics

[Resetting Invisible Labels](#)

Rotate

Use this form to change how the software rotates objects. It is accessed using the *Edit – Rotate* command.

Rotate Right turns an object in 90-degree increments clockwise around the current reference point.

Rotate Left turns an object in 90-degree increments counterclockwise around the current reference point.

Flip Horizontal mirrors the object across an imaginary horizontal line passing through the current reference point.

Flip Vertical mirrors the object across an imaginary vertical line passing through the current reference point.

Related topics

[Rotating](#)

Route Flight Line

Use this form to convert wires to normal wires after adding them using the *flight* route method on the Add Wire form. It is accessed using the *Edit – Route Flight* command.

Route Method indicates how the schematic editor routes wires.

full draws complex lines to complete the connection. Can generate more than three segments to complete the connection. Usually used as a bus.

direct draws a direct line between two given points.

Note: In some cases, *full* and *direct* cannot complete the connection wire. In these cases, a flight line remains in the schematic to indicate the intended connectivity.

Related topics

[Routing Flight Lines](#)

Route Named Connection

Use this form to route physically disjoint wires or groups of wires that use wire-by-name connection. It is accessed using the *Edit – Route Named Connection* command.

Route Method indicates how the schematic editor routes wires.

full draws complex lines to complete the connection. Can generate more than three segments to complete the connection. Usually used as a bus.

direct draws a direct line between two given points.

flight draws a direct line between nets connected by name.

Line Style specifies the line style of the wire.

Color specifies the color of the wire.

Width specifies the width of the wire.

Note: In some cases, the command cannot complete the connection of the wire because of congestion. In these cases, a flight line remains in the schematic to indicate the intended connectivity.

Related topics

[Routing Named Connections](#)

[Wire-by-Name Connections](#)

Save a Copy

Use this form to save a design to a different name. It is accessed using the *File – Save a Copy* command.

Library Name of the library to save copy of view in.

Cell Name of the cell to save copy of view in.

View name of new copied view.

Related topics

[Saving Your Edits](#)

Save Label Display

Filename specifies the file in which you want the system to save the current label display configuration.

Save Schematic Defaults

Use this form to save the current environment variables and form defaults to a file using the *Options – Save Defaults* command.

File Name specifies the name of the file to store the variables and form defaults. The default is a file named `.cdsenv` in your home directory.

Environment variable: `defaultsFile`

Related topics

[Saving Form Field Default Settings](#)

Save Schematics with Errors

This form is displayed when errors are found during a design hierarchy check.

all (default) saves all the schematics listed on the form. When you turn on this option, all the selection buttons for all the schematics listed on the form are automatically enabled.

none disables all the selection buttons for all the schematics listed on the form. When you select this option, you can turn on the selection buttons for individual schematics listed on the form.

Save “<Library><cell><view>” with <number> errors? specifies a design and the number of errors in that design. This field is displayed for each schematic that contains errors.

Related topics

[Checking a Design](#)

Schematic Check Options

Use this form to set up options to be run when you choose the following commands: *File – Check and Save*, *Check – Current Cellview*, *Check – Hierarchy*, *Check – Cross View Check* (in the symbol editor). The form is accessed using the *Options – Check* command.

Connectivity Options

Extract connectivity during schematic checking specifies whether the system updates the connectivity of a cellview when you use the *Check – Current Cellview* command. When enabled (default), the system extracts connectivity from the design.

Environment variables: srcReportIncompleteChecks, updateConn

Beep When 0 Errors/Warnings when enabled (default), causes the system to beep after every save and check in even if there is no error or warning message. The benefit of enabling this option is to let you know when the check is complete. Disabling this option restores the behavior to release 4.3.x so that the system beeps only when an error or warning is found.

Environment variable: checkBeepOn

Set Signal Type from Net Name and Type Registration sets the signal type for global or registered power and ground nets, while saving a design using *Check and Save*. When this checkbox is selected, the schematic editor automatically identifies all of the global or registered power or ground nets, and sets their signal type accordingly. By default, this checkbox is not selected and the tool does not set the signal type for these nets while checking a design.

Environment variable: postProcNetSigTypes

Check and Save Action on Error specifies whether the system saves schematics that contain errors.

Save saves all schematics including those that contain errors.

No Save saves all schematics except those that contain errors.

Ask Me (default) prompts you to confirm that you want to save a schematic that contains errors. When you choose this option and click *OK*, a dialog box appears. The dialog box lists all the schematics that contain errors and then prompts you to confirm which of the displayed schematics you want to save.

Environment variable: saveAction

Virtuoso Schematic Editor User Guide

Form Descriptions

From a Multisheet Index, Check specifies which sheets in a multisheet schematic the system checks.

all checks all multisheets.

only if needed (default) checks only the multisheets that you have edited since the last check.

Environment variable: checkAlways

Wire Name Glue Distance specifies a distance, in user units, between wire names and their associated wires. During the check operation, the system looks for unglued wire names. When the checker finds an unglued wire name within the specified distance from a wire, the system glues the wire name to the wire. When the unglued wire name is beyond the specified distance, the checker marks the unglued label.

Optical Options (Virtuoso Photonics Option)

Note: Virtuoso Photonics Solution - You need the *Virtuoso_Photonics_Option* license to use the functionality discussed in this section. For information about obtaining the required license, contact your local Cadence representative.

Propagate Optical Signal Type Values when enabled (default), specifies that optical signal types are propagated from an instTerm to a net when the net has been specified as a signal and has only optical connections. A net that already has an optical signal type is not changed. For more details, see Propagating Optical Signal Types.

Environment variable: opticalSigTypePropagation

Set Optical Wire Style when enabled (default), sets the color and line style of the optical nets. For more details, see Propagating Optical Signal Types.

Environment variables: opticalNetColor, opticalNetColoring, opticalNetLineStyle

Set Optical Pin Color when enabled (default), sets the color of the optical pins to distinguish them from electrical pins. For more details, see Propagating Optical Signal Types.

Environment variables: opticalPinColor, opticalPinColoring

Schematic Rules Checks Options

Run Schematic Rules Checks when enabled (default), specifies whether the system runs the schematic rule checks when you use the *Check – Current Cellview* command.

Virtuoso Schematic Editor User Guide

Form Descriptions

Environment variable: runSRC

Rules Setup opens the Schematic Rules Checks Setup.

Cross View Checks Options

Cross View Check Severity specifies which action the system performs when pin names and directions do not match.

ignored results in no action.

warning generates warnings.

error generates error messages.

Environment variables: runVIC, vicSeverity

Views To Check displays a list of the views to be checked when you click *OK* on a Check command form.

Environment variable: vicViewList

Match Pin Names Exactly when enabled, checks whether pin names match when you click *OK* on a Check command form. For example, pins A<0> and A<1> in view1 and pin A<0:1> in view2 would not match.

Environment variable: vicExactPinMatch

Check Pin Order when enabled (default), checks whether the pin order is consistent in all views in the *Views To Check* list mentioned above.

Only check views with explicit pin order, when enabled (default), allows port order consistency check to run only on views with the `portOrder` property set.

Environment variable: vicIgnoreViewsWithNoSetPinOrder

Environment variable: vicCheckPinOrder

Match Inherited Terminals when enabled (default), checks whether inherited terminals match.

Ignore Terminals with Net Expression and no Terminal in the Other View when enabled, if there is an inherited terminal (a terminal with a net expression) in one view of the cell and the same name terminal does not exist in the other view of the cell, then a cross view check will not report a warning.

Environment variables: vicIgnoreNonExistingTerms, vicInheritedPinMatch

Ignore Terminals for Specified Views

Views Names allows you to specify the view names for which the inherited terminal mismatches should be ignored.

Environment variable: [vicIgnoreInhTermViewList](#)

All Terminals ignores all inherited terminal mismatches for the views specified in the *For Views* field.

Environment variable: [vicIgnoreInhTermAll](#)

Terminal Names allows you to specify the names of inherited terminal mismatches that should be ignored for the views specified in the *For Views* field.

Environment variable: [vicIgnoreInhTermList](#)

Environment variable: [vicInheritedPins](#)

Match Terminal Net Expressions, when enabled (default), checks whether terminal net expressions match.

Match Terminal Supply Sensitivity, when enabled (default), validates sensitivity settings between terminals in the schematic and symbol views.

Environment variable: [vicTermSigType](#)

Match Terminal Signal Type when enabled (default), checks for signal type mismatches between the schematic and symbol views of a cell.

Environment variable: [vicTermSigType](#)

Related topics

[Setting the Check Options](#)

[Checking Sensitivity Between Schematic and Symbol Cellviews](#)

Schematic Find

Use this form to find objects that have a specific property. It is accessed using the *Edit – Find* command.

Previous highlights the previous object that matched the search criteria and displays information about the object in the form. When you reach the first object that matches the search criteria, this button has no effect.

Next highlights the next object that matches the search criteria and displays information about the object in the form. When you reach the last object that matches the search criteria, this button has no effect.

Search For lets you specify the search criteria.

The cyclic field contains a list of frequently used properties.

When you select *other* from the cyclic field, the Add New Property for Find form appears.

When you select *master* from the cyclic field, you must specify a property value. You cannot use wildcards for *master* property values.

The relational operator cyclic field lets you limit the search to objects that have a value that either matches or differs from the value specified in the text entry field.

The text entry field specifies a value of the property.

Environment variable: condOp

Search Scope specifies the scope of the search.

selected set searches only for objects in the selected set.

cellview searches only for objects in the cellview currently displayed in the design window.

hierarchy expands the form to display two more option fields: *top* and *current cellview*. Specify the cellview from which you want the system to start the search. The system searches all cellviews specified in the design hierarchy.

library expands the form to display a text field. Specify the library that you want the system to search. The system searches all cellviews in the library. The library must exist in the current library search path.

Zoom To Object specifies how the schematic editor displays a matching object. When this option is selected, the editor zooms in on the search object and centers the object in the design window.

Virtuoso Schematic Editor User Guide

Form Descriptions

Environment variable: [findZoomButton](#)

Zoom Scale appears only when you turn on *Zoom To Object*. Specifies the scale used to enlarge the object.

Select adds the currently highlighted object to the selected set.

List All lists all the matching objects in the Command Interpreter Window (CIW).

Show Object Filter expands the form to show the Search Filter. You can use the filter to limit the search scope to only the objects you specify.

Schematic Objects lists the schematic object types. You can specify any combination of object types, or you can use one of the following options:

all enables the selection button for all the object types.

none disables the selection button for all the object types. You can use this option to quickly clear any existing selections before you begin to create a new selection set.

Instance Objects lists the instance object types. You can specify any combination of the object types, or you can use one of the following options.

all enables the selection buttons for all the object types.

none disables the selection button for all the object types. You can use this option to quickly clear any existing selections before you begin to create a new selection set.

Related topics

[Searching for Objects That Have a Specific Property](#)

[Search String Wildcards](#)

Schematic Replace

Use this form to find and replace properties. It is accessed using the *Edit – Replace* command.

Replace assigns the new property name to the currently highlighted matching object.

Skip highlights the next object that matches the *Search For* criteria and displays information about the object in the form.

Replace All assigns the new property name to all the matching objects.

Search For lets you specify the search criteria.

The cyclic field contains a list of frequently used properties.

When you select other from the cyclic field, the Add New Property for Replace form appears.

When you select *master* from the cyclic field, you must specify a property value. You cannot use wildcards for *master* property values.

The relational operator cyclic field lets you limit the search to objects that have a value that either matches or differs from the value specified in the text entry field.

The text entry field specifies a value of the property.

Environment variable: condOp

Replace With lets you specify values for the replace action.

The cyclic field contains a list of frequently used properties.

When you select *other*, the Add New Property for Replace form appears. Type a property name in the text entry field. When you click *Apply*, the property is added to the *Property Name* list.

The text entry field specifies a value of the property.

When you select *master* from the cyclic field, you must specify a property value. You cannot use wildcards for *master* property values. You must type the exact value.

You can use the following wildcards in the text entry field:

- * matches zero or more occurrences of any characters
- . matches any character

Virtuoso Schematic Editor User Guide

Form Descriptions

[c..] matches one of the characters in the set

Environment variable: condOp

Search Scope specifies the scope of the search.

selected set searches only for objects in the selected set.

cellview searches only for objects in the cellview currently displayed in the design window.

hierarchy expands the form to display two more option fields: *top* and *current cellview*. Specify the cellview from which you want the system to start the search. The system searches all cellviews specified in the design hierarchy.

library expands the form to display a text field. Specify the library that you want the system to search. The system searches all cellviews in the library. The library must exist in the current library search path.

Zoom To Object specifies how the schematic editor displays a matching object. When this option is selected, the editor zooms in on the search object and centers the object in the design window.

Zoom Scale appears only when you turn on *Zoom To Object*. Specifies the scale used to enlarge the object.

Preview Changes (No Edits), when enabled, writes messages to the Command Interpreter Window detailing what would happen when you click *Apply*, *Replace*, *Skip*, or *ReplaceAll*.

Save Changes, when enabled, writes changes to disk without prompting you to confirm the changes. Only cellviews in the hierarchy or library are affected; the cellview currently displayed in the design window is not affected. When disabled, prompts you to confirm the changes when you attempt to start the *File – Close* command or exit the software.

Environment variable: replSaveChanges

Show Object Filter expands the form to show the Search Filter. You can use the filter to limit the search scope to only the objects you select.

Schematic Objects lists the schematic object types. You can specify any combination of object types, or you can use one of the following options:

all enables the selection button for all the object types.

Virtuoso Schematic Editor User Guide

Form Descriptions

none disables the selection button for all the object types. You can use this option to quickly clear any existing selections before you begin to create a new selection set.

Instance Objects lists the instance object types. You can specify any combination of the object types, or you can use one of the following options.

all enables the selection buttons for all the object types.

none disables the selection button for all the object types. You can use this option to quickly clear any existing selections before you begin to create a new selection set.

Related topics

[Replacing Properties](#)

[Search String Wildcards](#)

Schematic Rules Checks Setup

Use this form to set the rules for running the SRC. It is accessed using the *Check – Rules Setup* command.

Packaged Checks automatically sets the severity levels for the following predefined checks.

- **None** sets the logical, physical, and name check severity levels to the default values.
- **Normal** sets the logical, physical, and name check severity levels to either *ignored* or *warning*.
- **Logical Only** sets all logical check severity levels to *warning*. Physical and name check severity levels are set to *ignored*.
- **Physical Only** sets all physical check severity levels to *warning*. Logical and name checks are set to *ignored*.
- **Mixed Signal** enables the *Normal* packaged checks, enables the signal type checks, and sets the *Verilog HDL Syntax (Name tab)*, *Sparse Index Bus (Name tab)*, and *Pin Direction Collision (Logical tab)* checks to *warning*.
- **VHDL** sets logical, physical, and name check severity levels to preassigned values. VHDL syntax checks are set to *error*.
- **Verilog** sets logical, physical, and name check severity levels to preassigned values. Verilog syntax checks are set to *error*.

Environment variable: [schSRCPackagedChecks](#)

Logical checks the high-level connectivity data.

Floating Nets checks for wires that are neither physically nor logically connected to a schematic or instance pin. Ignores wires that are connected by name to a schematic or instance pin.

Environment variable: [srcFloatingNets](#)

To minimize the number of messages when you set the severity levels for this check to *warning* or *error*, you should set the *Unconnected Wires* check in the *Physical Checks* section of the form to *ignored*.

Floating Input Pins checks for component input pins and schematic output pins that are not connected to component output pins, schematic input pins, or schematic I/O pins.

Environment variable: [srcFloatingInput](#)

Virtuoso Schematic Editor User Guide

Form Descriptions

Floating Output Pins checks for component output pins and schematic input pins that are not connected to component input pins, schematic output pins, or schematic I/O pins.

Environment variable: srcFloatingOutput.

Floating I/O Pins checks for component or schematic input/output pins whose signals are not connected to component output pins, schematic input pins, or any other schematic or symbol I/O pins.

Environment variable: srcFloatingIO

Floating Switch Pins checks for component or schematic bidirectional pins whose signals are not connected to component output pins, schematic input pins, or any other schematic I/O pins.

Environment variable: srcFloatingBIDIR

Shorted Output Pins checks for component output or schematic input pins that are connected to any other component output or schematic input pins.

Environment variables: srcShortedOutputs, srcShortedOutputGlobal

Offsheet Connectors checks for offsheet connectors that appear on only one sheet in a multisheet design. Also checks whether a net is used locally in more than one sheet but is not connected across the sheets with an offsheet connector.

Environment variable: srcOffsheetConnCheck

Plain Globals checks for any globals that are not the default for a net expression, that is, a check is performed for plain globals within the cell view. This check allows you to locate and protect against these unintended globals that could prevent data re-use. The default setting is *ignored*.

Environment variable: srcFixedGlobalsCheck

Pin Direction Collision checks for any inconsistencies in the direction of the bit terminals of a multi-bit terminal. For example, if set to *warning* or *error*, it detects scenarios such as where terminal A<3:0> with output direction and terminal A<0> with input-output directions, both co-exist in the schematic.

Environment variable: srcPinDirConflictCheck

Physical checks for unconnected wires, offset labels, and overlapping instances.

Unconnected Wires checks for wire segments whose end points do not physically connect to

- ☐ A schematic pin

Virtuoso Schematic Editor User Guide

Form Descriptions

- ☐ A component pin
- ☐ A label
- ☐ Another wire segment

This check is a physical check, not a logical or connectivity check.

Environment variable: srcUnconnectedWires

To minimize the number of messages when you set the severity levels for this check to *warning* or *error*, set the *Floating Nets* check in the *Logical Checks* section of the form to *ignored*.

Solder On CrossOver checks for solder connections placed on wire crossovers.

Environment variable: srcSolderOnCrossover

Overlapping Instances checks for instances that overlap other instances.

Environment variables: checkAlignOverlaps, srcInstOverlap

Percent Overlap Allowed specifies a value that defines the degree to which two instances can overlap each other. The value represents a percentage of the bounding box for the smaller instance.

Environment variables: checkAlignOverlaps, srcInstOverlapValue

Maximum Label Offset checks for unglued labels beyond a specified distance from a wire (see *Maximum Offset From Wire* option). When the checker finds unglued labels within this specified distance, the checker automatically glues the labels to the appropriate wires if the severity level is set to *warning* or *error*.

Environment variable: srcMaxLabelOffset

Maximum Offset From Wire specifies the distance that an unglued label can be located from a wire. When the checker finds unglued labels beyond the specified distance, the checker marks the unglued label (see *Maximum Label Offset* option).

Environment variable: maxLabelOffsetUU

Name checks for syntax and collisions.

Connection By Name checks whether a signal exists in more than one physically disjointed group of wires or pins. You might have inadvertently connected objects when using a name that was already used elsewhere in the schematic. For example, if a schematic contains a pin name “A” and you name a wire in the schematic “A”, the pin and the objects connected to the wire are logically connected.

Environment variable: srcConnByName

Virtuoso Schematic Editor User Guide

Form Descriptions

Unnamed Connection checks whether a net has a user-generated name. The checker flags nets without user-generated names when one of the following conditions is met:

- ☐ The net has fewer than two attached instance terminals.
- ☐ The net has a system-generated name.
- ☐ The net has wire segments.

Environment variable: srcUnnamedConnection

Instance/Net Name Collision checks whether a signal name collides with an instance name. For example, the schematic rules checker will flag an instance named “x” and a net named “x” because the names are identical. This option is especially useful for ensuring Verilog HDL and VHDL consistency.

Environment variable: srcNameCollision

Pin/Net Name Collision checks whether a pin name differs from a net name. For example, the schematic rules checker will flag a pin named “x” and a net named “y” because the names differ. It flags the condition of a pin name connected to a wire with a different net name; it does not flag connection by name.

Environment variable: srcPinNetCollision

Verilog HDL Syntax checks Verilog HDL syntax for instance and signal names in a schematic cellview. The checker flags signal names or instance names when they collide with Verilog HDL reserved words or when they are not valid Verilog HDL identifiers.

Environment variable: srcVerilogSyntax

System Verilog HDL Syntax checks System Verilog HDL syntax for instance and signal names in a schematic cellview. The checker flags signal names or instance names when they collide with System Verilog HDL reserved words or when they are not valid System Verilog HDL identifiers.

Environment variable: srcSysVerilogSyntax

VHDL Syntax performs VHDL syntax checking on instance and signal names in a schematic cellview. The schematic rules checker flags signal names or instance names when they collide with VHDL reserved words or when they are not valid VHDL identifiers.

Environment variable: srcUnnamedConnection

Instance Name Syntax controls name syntax checking for instances.

Environment variable: srcInstNameSyntax

Virtuoso Schematic Editor User Guide

Form Descriptions

Instance Name Expression specifies the regular expression the checker uses in instance name syntax checks. This field is enabled only when the *Instance Name Syntax* check is set to *warning* or *error*.

Environment variable: srcInstNameExpr

Pin Name Syntax controls name syntax checking for pins.

Environment variable: srcTermNameSyntax

Pin Name Expression specifies the regular expression the checker uses in pin name syntax checks. This field is enabled only when the *Pin Name Syntax* field is set to *warning* or *error*.

Environment variable: srcTermNameExpr

Net Name Syntax controls name syntax checking for nets.

Environment variable: srcNetNameSyntax

Net Name Expression specifies the regular expression the checker uses in net name syntax checks. This field is enabled only when the *Net Name Syntax* field is set to *warning* or *error*.

Environment variable: srcNetNameExpr

Sparse Index Bus checks that all bits of a vectored net in the range are used.

Environment variable: srcSparseIndexBus

Sparse Index Bus Only Terms filters out sparse nets which are not attached to a term from a Sparse Bus Index Check.

Environment variable: srcSparseIndexBusOnlyTerms

Inherited Connection checks existing or missing inherited connections.

Inherited Pin/Net Connections checks whether any inherited pin or net connections exist in the schematic.

Environment variable: srcInheritedConn

Missing Override Nets checks for net names that have been specified by inherited connection override properties. If the net name is not found, the appropriate severity marker will be placed on the instance where the netSet property is defined.

Environment variable: srcNoOverrideNet

Inherited Pin/Net Name Collision checks whether the name of the net driving or driven from a schematic pin matches the name of the inherited pin.

Virtuoso Schematic Editor User Guide

Form Descriptions

Environment variable: [srcInhPinNetCollision](#)

Implicit Inherited Terminals checks for all inherited signals, that the signal connects to a cellview terminal. If this is not the case, a marker is placed on the source of the inherited signal. If there are multiple sources (labels), the marker is placed on the first source. The default for this option is *ignored*.

Environment variable: [srcInhImplicitTerm](#)

Inherited Instance Pin/Pin Connections checks that inherited instance terminals are connected to nets with terminals. The default for this option is *ignored*.

Environment variable: [srcInhInstPinPinConn](#)

Constraints if enabled checks constraints members and values. For more information on this tab refer to [Constraints Checks](#).

Environment variable: [checkConstraints](#)

Note: Constraints functionality is only available in Virtuoso Schematic Editor XL. For more information on Constraints see the [Virtuoso Unified Custom Constraints User Guide](#).

Mismatched Parameters checks devices that are members of the [matchedParameters](#) constraint, confirming, or not, that the parameter has the same value on the devices.

Environment variable: [srcMatchedParameters](#)

Signal Type if enabled performs signal type (sigType) checks

The *Signal Type* tab has three sub-tabs (*Input pins*, *Output pins*, and *I/O pins*) each of which contain the same *Net Signal Type* and *Pin Signal Type* options. These tabs will however be grayed out unless the *Signal Type Checks* checkbox option is selected. For more details, see [Defining Signal Types](#).

Environment variable: [srcSigTypeIntegrity](#)

Optical (Virtuoso Photonics Option)

Note: Virtuoso Photonics Solution - You need the *Virtuoso_Photonics_Option* license to use the functionality discussed in this section. For information about obtaining the required license, contact your local Cadence representative.

Electrical Connections checks for any instances where optical nets are connected to electrical nets. The default setting is *error*.

Too Many Connections checks optical nets to ensure each has a maximum of two connections. The default setting is *error*.

Virtuoso Schematic Editor User Guide

Form Descriptions

Multi Mode to Single Mode checks multi mode nets or outputs to ensure that none are connected to single mode inputs or nets. The default setting is `error`.

Single Mode to Multi Mode checks single mode nets or outputs and flags any that are connected to multi mode inputs or nets. The default setting is `ignored`.

RF Implementation (Virtuoso RF Solution Option) if enabled, performs transmission line checks.

RF Implementation checks

Missing stackup instance checks for a stackup instance and it flags if no stackup instance is found. The default setting is `Ignored`.

Multiple stackup instances checks for more than one stackup instance and it flags if multiple are found. The default setting is `Ignored`.

Discontinuities without adjacent strips checks that the placement of each discontinuity type (tees, bends, curves, and others) is correctly constructed in relation to the transmission line and to the adjacent cells' connectivity properties (current width, transmission line format, and so on). The default setting is `Warning`.

Topology Nets with more than 2 pins checks topology nets to ensure each has maximum of 2 pins. The default setting is `Warning`.

Mixed transmission line formats checks that the placement of transmission elements within a transmission line are using the same transmission line format, stripline, microstrip, and so on. The default setting is `Warning`.

Coloring if enabled performs color checks.

Coloring Checks

Color Mismatch error checks that the instance terminal CDF color and the color constraint of its net are the same. The default setting is `Warning`.

CDF color mismatch warning checks that terminals with the same net connection have the same CDF color assignment. The default setting is `Warning`.

Related topics

[Setting Schematic Rule Checks](#)

Schematic Select By Filter

Use this form to select all the objects of one or more object types. It is accessed using the *Edit – Select – Select By Filter*.

Form Action controls whether the system selects or deselects all the specified object types when you click *OK* or *Apply*.

select selects all the specified object types.

deselect selects everything but the specified object types.

Schematic Objects lists the schematic object types.

all turns on object types.

none turns off object types.

Instance Objects lists the instance object types.

all turns on instance object types.

none turns off instance object types.

Related topics

[Selecting Objects Automatically by Type](#)

Schematic Select By Property

Use this form to limit a selection set to object types with certain property values. It is accessed using the *Edit – Select – Select By Property* command.

Find lets you specify values for the search.

The cyclic field lists frequently used properties.

When you select *other* from the cyclic field, the Add New Property for Search form appears.

The relational operators let you limit the search to objects that have a property value that either matches or differs from the value specified in the field showing the property value.

The text entry field specifies a value of the property.

When you select *master* from the list of property names, you must specify a property value of the form

`"libname cellname viewname"`

Form Action controls whether the system selects or deselects the specified object types when you click *OK* or *Apply*.

select selects all the specified object types.

deselect selects everything but the specified object types.

Zoom Out To Fit All Figures zooms the schematic editor out to display all the objects that match the selection criteria.

Environment variable: zoomOutButton

Show Object Filter expands the form to show the Selection Filter.

Schematic Objects lists the schematic object types.

all turns on schematic object types.

none turns off schematic object types.

Instance Objects lists the instance object types.

all turns on instance object types.

none turns off instance object types.

Virtuoso Schematic Editor User Guide

Form Descriptions

Related topics

Selecting Objects by Property

Schematic Selection Filter

Use this form to specify the objects you want to filter for selection. It is accessed using the *Options – Select Filter* command.

Area Partial Selection specifies whether the schematic editor includes partial objects in the selection set. Partial objects can be wire endpoints and vertices or note shape object endpoints, vertices, and edges.

When this option is turned off (default), the schematic editor does not include partial objects when you drag your cursor through the design to select several objects at once (area selection). When this option is turned on, partial objects are included when you drag the cursor through the design to select several objects at once (area selection).

Environment variable: schematicAreaPartialSelect

Schematic Objects lists the schematic object types you can select with your cursor.

all turns on all the object types.

none turns off all the object types.

Environment variable: schematicSelectFilter

Instance Objects lists the instance object types you can select. The method for selecting object types can be selected under *Scope*.

Point Selection Only enables selecting objects by pointing and clicking in the design.

Any enables selecting objects by both area selection and point selection.

The selectability of most instance subobjects is disabled by default, which allows for easier selection of the whole instance.

all turns on all the object types.

none turns off all the object types.

Note: You can also control which instance objects are available for selection using the *Instance Select Filter*, *Pin Select Filter*, *Wire Select Filter*, or *Text Select Filter* available on the *Select* toolbar. In the schematic view you can use the *Text Select Filter* option to reposition symbol labels.

Related topics

Selecting Objects by Filtering

Schematic Title Block Properties

Use this form to add or modify the title block of a multisheet schematic. It is accessed using the *Edit – Sheet Title* command.

Global Border Titles defines properties shared by all sheets.

company specifies the name of the company.

dwg_no specifies the drawing number for the current sheet.

CAGE specifies the Commercial and Government Entities supply code. CAGE replaces the Federal Supply Code for Manufacturers (FSCM) number.

Sheet Specific Border Titles specifies the properties specific to the current sheet.

rev specifies the revision level for the current sheet.

title specifies the title for the current sheet.

title1 specifies the subtitle for the current sheet.

Related topics

[Editing the Title Block](#)

Set Instance Binding

Use this form to define the view to use and choose the inherited view list. It is accessed using the *Hierarchy-Editor – Set Instance Binding* command.

OK makes changes to the selected instances in the hierarchy editor and closes the form.

Cancel closes the form.

Apply makes changes to the selected instances in the hierarchy editor and leaves the form displayed.

Defaults reverts to the default bindings and inherited view list.

Previous selects the previously selected instance in the selection list (in the schematic window).

Next selects the next instance in the selection list (in the schematic window).

Apply To

current instance applies settings to only the current instance in the selected set (or selection list).

all selected instances applies settings to all instances in the selection list.

all instances applies settings to all instances in the schematic.

Of

same master applies settings to only instances that have the same master as the current instance in the selection list.

any master applies settings to instances regardless of their masters.

Instance Name is the name of the current instance. You cannot edit this field.

View To Use identifies the view to be used when expanding down the design hierarchy. You can choose from the names of all available views for this instance.

Inherited View List specifies the view list that is inherited by all subordinate instances.

As Is retains the current setting without change.

Build displays the Build Inherited View List form that lets you create the view list by selecting views and constants or by typing in your own selections.

Virtuoso Schematic Editor User Guide

Form Descriptions

Related topics

[Setting Instance Bindings](#)

Set Label Display Simulation Data Directory

Simulation Data Directory specifies the name of the directory in which you want to store the simulation data.

Note: When a valid simulation data directory, that contains results for the schematic, is provided, the openResults API will be called to speed up the data retrieval process.

Show Views Found

This form shows views used by the instance. It is accessed by choosing the *Hierarchy-Editor – Show Views Found* command.

View Name(s) shows all the views bound to all the instances in the current schematic cellview.

Highlight Layers shows the colors used to highlight the instances with the corresponding views. The command graphically shows the current view being used for each instance.

Related topics

[Showing Views Found](#)

Stretch – Schematic View

Use this form to stretch an object while in the schematic window. It is accessed using the *Edit – Stretch* command.

Snap Mode specifies the angles along which you can stretch objects in the cellview.

anyAngle (default) lets you stretch along any angle from the current location.

diagonal restricts the movement of stretched objects to points that are along 45-degree, 90-degree, and 180-degree angles from the current location.

orthogonal restricts the stretching of objects to points along the X and Y axes from the current location.

Route Method controls how the schematic editor routes wires when you stretch an object.

full (default) draws complex lines and generates additional segments, when necessary, to complete the connection. When the schematic editor cannot complete the connection, flightlines remain in the schematic to indicate the intended connectivity.

direct draws a direct line between two given points.

flight draws a dashed line between two given points.

simple restricts the movement of objects, and the drawing of associated lines, to points along the X and Y axes from the current location. This stretch method can be used to move multiple connected/aligned objects. For example, if an instance is moved horizontally, then any instances connected to its ports, by direct vertical nets, will also be moved by the same amount. This will ensure that the relative alignment remains. The simple stretch method therefore allows for easier movement of aligned, or closely connected, devices.

Note: If you want to stretch **individual** objects, with movement restricted along the X and Y axes, you should set *Snap Mode* to *orthogonal*, and *Route Method* to *full*.

Reroute Wire Segments Connected To specifies whether the schematic editor stretches and reroutes the unselected wires that are connected to a specified object type. When you do not enable back tracing for an object type, the schematic editor stretches and reroutes only the first wire segment that is connected to the object you want to stretch. You can reroute wire segments connected to the following object types:

- Instance
- Pin
- Wire

Virtuoso Schematic Editor User Guide

Form Descriptions

When you turn on an option for an object type, the schematic editor stretches and reroutes all the connected wire segments from the selected object only back to the point where

- The wire reaches a solder dot
- The wire reaches a pin
- The wire reaches a wire of a different width
- The wire reaches another selected wire

Environment variables: [compBackTrace](#), [pinBackTrace](#), [wireBackTrace](#)

Rotate turns an object in 90-degree increments counterclockwise around the current reference point.

Sideways mirrors the object across an imaginary vertical line passing through the current reference point.

Upside Down mirrors the object across an imaginary horizontal line passing through the current reference point.

Related topics

- [Stretching in the Schematic Window](#)
- [Stretching](#)

Stretch – Symbol View

Use this form to stretch an object while in the symbol editor. It is accessed using the *Edit – Stretch* command.

Snap Mode specifies the angles along which you can stretch objects in the cellview.

anyAngle (default) lets you stretch along any angle from the current location.

diagonal restricts the movement of stretched objects to points along the X and Y axes and to points that are along 45-degree angles from the current location.

orthogonal restricts the stretching of objects to points along the X and Y axes from the current location.

Related topics

[Stretching](#)

Submit Plot

Use this form to specify your plotting requirements. It is accessed using the *File – Print* command.

Plot – **cellview** (default) section

Library Name specifies the library that contains the cellview you want to plot. For more details, see [Plotting a Library](#).

Cell Name specifies the name of the cellview you want to plot. For more details, see [Plotting a Cellview](#).

View Name specifies the view name of the cellview you want to plot. For more details, see [Plotting Part of a Schematic](#).

Area to Plot specifies the boundaries for the plot. The text field displays the coordinates for the lower right-hand and upper left-hand corners of a bounding box that represents the plot area.

Full Size (default) specifies the entire cellview and gives the coordinates in *Area to Plot*.

Select lets you type coordinates or drag your mouse across the design to specify the plot area.

Browse opens the Library Browser. Use the browser to select a cellview. The system automatically fills in the form with the information you specify in the browser.

Plot Scope section

Sheet Range (appears only when plotting from an index sheet) lets you choose the number of sheets of a multisheet schematic.

none prints only the current cellview.

all prints the entire hierarchy starting from the current cellview.

from lets you specify a start and end page to plot.

Hierarchy options control hierarchical plotting.

current level plots only the specified cellview.

hierarchy plots portions of the design hierarchy.

levels down plots a specified number of design levels.

The following fields are editable when you select *hierarchy* or *levels down*.

Virtuoso Schematic Editor User Guide

Form Descriptions

View Name List specifies a prioritized list of view names that the system looks for during hierarchical plotting.

Skip Libraries specifies a list of libraries that the system ignores during hierarchical plotting.

Skip Cells Below specifies the last level of hierarchy that the system plots during hierarchical plotting. Specify a cell name or list of cell names to identify the final level to plot. The system ignores all cells below the specified cell or list of cells.

Plot With section lets you specify the information you want the system to include on the output plot. You can specify more than one option.

header plots the header information.

notes enables the text box. Use the text box to create the note text.

grid/axes plots both the grid and axes lines.

Template File specifies a plot setup file. A plot setup file contains plot setup information for the Submit Plot form and the Plot Options form. The *Template File* options let you read in, or write to, the file you specify in this field.

Load reads in plot setup information from a plot setup file. Specify a file that contains plot setup options or a sample file. Two sample plot template files are available in the `your_install_dir/tools/dfII/samples/plot` directory. The sample files are `schPlot.il` and `schMetPlot.il`.

The system loads the default plot template file when you first use the *File – Print* command. Once the default plot template file is set, you must use the *Load* button to load another template file.

Save writes the current plot setup information to a setup file. Plot setup options are stored in a property list format.

Plot Options opens the Plot Options. You use the Plot Options form to set the plotter options.

Plot – library

Library Name specifies the library that contains the cells you want to plot.

View Name List lists the view names of the cells you want to plot.

Browse opens the Library Browser. Use the browser to select a library. The system automatically fills in the form with the information you specify in the browser.

Virtuoso Schematic Editor User Guide

Form Descriptions

Plot With options let you specify the information you want the system to include on the output plot. You can specify more than one option.

header plots the header information.

notes enables the text box. Use the text box to create the note text.

grid/axes plots both the grid and axes lines.

Template File specifies a plot setup file. A plot setup file contains plot setup information for the Submit Plot form and the Plot Options form. The *Template File* options let you read in, or write to, the file you specify in this field.

Load reads in plot setup information from a plot setup file. Specify a file that contains plot setup options or a sample file. Two sample plot template files are available in the `your_install_dir/tools/dfII/samples/plot` directory. The sample files are `schPlot.il` and `schMetPlot.il`.

The system loads the default plot template file when you first use the *File – Print* command. Once the default plot template file is set, you must use the *Load* button to load another template file.

Save writes the current plot setup information to a setup file. Plot setup options are stored in a property list format.

Plot Options opens the Plot Options. You use the Plot Options form to set the plotter options.

Plot – viewing area (Library Name, Cell Name, and View Name fields are disabled in this plot.)

Plot With options let you specify the information you want the system to include on the output plot. You can specify more than one option.

header plots the header information.

notes enables the text box. Use the text box to create the note text.

index sheets plots the index sheets for the current plot job.

grid/axes plots both the grid and axes lines.

Template File specifies a plot setup file. A plot setup file contains plot setup information for the Submit Plot form and the Plot Options form. The *Template File* options let you read in, or write to, the file you specify in this field.

Load reads in plot setup information from a plot setup file. Specify a file that contains plot setup options or a sample file. Two sample plot template files are available in the

Virtuoso Schematic Editor User Guide

Form Descriptions

`your_install_dir/tools/dfII/samples/plot` directory. The sample files are `schPlot.il` and `schMetPlot.il`.

The system loads the default plot template file when you first use the *Design – Submit – Plot* command. Once the default plot template file is set, you must use the *Load* button to load another template file.

Save writes the current plot setup information to a setup file. Plot setup options are stored in a property list format.

Plot Options opens the [Plot Options](#). You use the Plot Options form to set the plotter options.

Related topics

[Plotting a Design](#)

Symbol Find

Use this form to find objects that have a specific property in the symbol editor. It is accessed using the *Edit – Find* command.

Previous highlights the previous object that matched the search criteria and displays information about the object in the form. When you reach the first object that matches the search criteria, this button has no effect.

Next highlights the next object that matches the search criteria and displays information about the object in the form. When you reach the last object that matches the search criteria, this button has no effect.

Search For lets you specify the search criteria.

The cyclic field contains a list of frequently used properties.

When you select *other* from the cyclic field, the Add New Property for Find form appears.

When you select *master* from the cyclic field, you must specify a property value. You cannot use wildcards for *master* property values.

The relational operator cyclic field lets you limit the search to objects that have a value that either matches or differs from the value specified in the text entry field.

The text entry field specifies a value of the property.

Search Scope specifies the scope of the search.

selected set searches only for objects in the selected set.

cellview searches only for objects in the cellview currently displayed in the design window.

hierarchy expands the form to display two more option fields: *top* and *current cellview*. Specify the cellview from which you want the system to start the search. The system searches all cellviews specified in the design hierarchy.

library expands the form to display a text field. Specify the library that you want the system to search. The system searches all cellviews in the library. The library must exist in the current library search path.

Zoom To Object specifies how a matching object is displayed. When this option is selected, the schematic editor zooms in on the search object and centers the object in the design window.

Virtuoso Schematic Editor User Guide

Form Descriptions

Zoom Scale appears only when you turn on *Zoom To Object*. Specifies the scale used to enlarge the object.

Select adds the currently highlighted object to the selected set.

List All lists all the matching objects in the Command Interpreter Window (CIW).

Show Object Filter expands the form to show the Search Filter. You can use the filter to limit the search scope to only the objects you specify.

Schematic Objects lists the schematic object types. You can specify any combination of object types, or you can use one of the following options:

all enables the selection button for all the object types.

none disables the selection button for all the object types. You can use this option to quickly clear any existing selections before you begin to create a new selection set.

Instance Objects lists the instance object types. You can specify any combination of the object types, or you can use one of the following options.

all enables the selection buttons for all the object types.

none disables the selection button for all the object types. You can use this option to quickly clear any existing selections before you begin to create a new selection set.

Related topics

[Searching for Objects That Have a Specific Property](#)

Symbol Generation Options

Use this form to manage the property and attribute settings that the text-to-symbol generator (TSG) uses to generate symbols. It is accessed when Replacing an Existing Symbol Cellview and you click *Replace* on the Create Cellview dialog box.

Note: By default, the software initializes the form with information from the current cellview and the Cellview From Cellview.

Library Name specifies the library for the generated cellview.

Cell Name specifies the cell name for the generated cellview.

View Name specifies the view name for the generated cellview.

Pin Specifications lets you change pin order or move pins to different sides of the symbol.

Left Pins creates pins for the left side of the symbol.

Right Pins creates pins for the right side of the symbol.

Top Pins creates pins for the top of the symbol.

Bottom Pins creates pins for the bottom of the symbol.

Exclude Inherited Connections lets you specify whether all pins with inherited connections are to be excluded during symbol generation, or only a named set.

None no inherited connections should be included during symbol generation.

All all inherited connections should be included during symbol generation.

Only these only the named inherited connections should be included during symbol generation.

Environment variable: tsgExcludeInhConnPinsNames

Environment variable: tsgExcludeInhConnPins

Attributes

List opens a corresponding attributes form that you can use to change the pin direction and the pin graphic attributes.

Note: You can edit the tsgPinGraphicMasters variable using the *attributes* form.

Load/Save displays options you can use to specify which TSG template file the system references when creating a symbol automatically.

Virtuoso Schematic Editor User Guide

Form Descriptions

Load consists of a *Load* button, a cyclic field, and a text field. The cyclic field lists the TSG template file keywords. The text field changes when you select a keyword. When you click the *Load* button, the system loads the file displayed in the text field.

Environment variable: tsgTemplateType

Save (TSG File) specifies a path for the file you want to save.

Load Template Data From Symbol specifies a file that contains the TSG template information.

Library Name specifies the library name of the cellview that contains the TSG template information.

Cell Name specifies the cell name of the cellview that contains the TSG template information.

View Name specifies the view name of the cellview that contains the TSG template information.

Edit Attributes expands the form to display the symbol attributes options.

Pin Spacing specifies the spacing between pins.

Stub Length specifies a length for the line that represents the pin wire.

Height increases the minimum height value for the generated symbol. When set to 0 (default), TSG uses pin and label data to determine the minimum height value for the symbol.

Width increases the minimum width value for the generated symbol. When set to 0 (default), TSG uses pin and label data to determine the minimum width value for the symbol.

Pin Connector contains a list of pin connector shapes.

Environment variable: tsgConnectorMasters

Origin contains a list of possible origin locations for the generated symbol.

Reverse Pin Order creates a symbol with the pins in reverse order.

Edit Labels expands the form to display the label options.

Label contains a list of the current labels.

Name specifies a label that you want to add to the *Label* cyclic field.

Label Choice contains a list of label types.

Virtuoso Schematic Editor User Guide

Form Descriptions

Apply To contains a list of the pin types to which you can apply the label.

Label Type contains a list of the types you can assign to the label.

Orientation specifies the orientation of the label.

Font Height specifies the height of the label text.

Justification specifies the location of the label.

Font Style contains a list of the fonts you can apply to the label.

isOverbar is only a display option. You use this option to specify whether the software displays an overbar over a label in the design window. When disabled (default), the software displays underscore characters (_) in the label as part of the label text string. When enabled, the software interprets underscore characters (_) in the label as toggle switches that control where overbars begin and end.

- ☐ The property name _abcde puts an overbar over all 5 letters.
- ☐ The property name ab_cde puts an overbar over only cde.
- ☐ The property name _abc_de puts an overbar over only abc.
- ☐ The property name ab_cd_ef_gh_ij puts an overbar over only cd and gh.



Caution

To avoid confusion when text contains underscore characters, Cadence recommends that you disable this display option. It is provided only for backward compatibility.

Location specifies where on the symbol to place the label.

Add adds the pin name you specify in the *Name* field to the list under the *Label* cyclic field.

Delete removes the pin name you specify in the *Name* field from the list under the *Label* cyclic field.

Modify applies the changes you specify in the form to the specified label.

Edit Properties expands the form to display the property options.

Property contains a list of the current properties.

Name specifies a property that you want to add to the *Property List* cyclic field.

Value specifies a value for the property you specify in the *Name* field.

Virtuoso Schematic Editor User Guide

Form Descriptions

Apply To contains a list of the pin types to which you can apply the property.

Property Type contains a list of the types you can assign to the property.

Orientation specifies the orientation of the property in respect to the instance.

Font Height specifies the height of the property text.

Justification specifies the location of the property.

Font Style contains a list of the fonts you can apply to the property.

isOverbar is only a display option. Use this option to specify whether the software displays an overbar over a label in the design window. When disabled (default), the software displays underscore characters (_) in the label as part of the label text string. When enabled, the software interprets underscore characters (_) in the label as toggle switches that control where overbars begin and end.

- ☐ The property name _abcde puts an overbar over all 5 letters.
- ☐ The property name ab_cde puts an overbar over only cde.
- ☐ The property name _abc_de puts an overbar over only abc.
- ☐ The property name ab_cd_ef_gh_ij puts an overbar over only cd and gh.



Caution

To avoid confusion when text contains underscore characters, Cadence recommends that you disable this display option. It is provided only for backward compatibility.

Location specifies where on the symbol to place the property.

Add adds the property name you specify in the *Name* field to the list under the *Property* cyclic field.

Delete removes the property name you specify in the *Name* field from the list under the *Property* cyclic field.

Modify applies the property name you specify in the *Name* field to the list under the *Property* cyclic field.

Related topics

[Editing Symbol Generation Options](#)

Symbol Replace

Use this form to find and replace properties in the symbol editor. It is accessed using the *Edit – Replace* command.

Replace assigns the new property name to the currently highlighted matching object.

Skip highlights the next object that matches the *Search For* criteria and displays information about the object in the form.

Replace All assigns the new property name to all the matching objects.

Search For lets you specify the search criteria.

The cyclic field contains a list of frequently used properties.

When you select other from the cyclic field, the Add New Property for Replace form appears.

When you select *master* from the cyclic field, you must specify a property value. You cannot use wildcards for *master* property values.

The relational operator cyclic field lets you limit the search to objects that have a value that either matches or differs from the value specified in the text entry field.

The text entry field specifies a value of the property.

Replace With lets you specify values for the replace action.

The cyclic field contains a list of frequently used properties.

When you select *other*, the Add New Property for Replace form appears. Type a property name in the text entry field. When you click *Apply*, the property is added to the *Property Name* list.

The text entry field specifies a value of the property.

When you select *master* from the cyclic field, you must specify a property value. You cannot use wildcards for *master* property values. You must type the exact value.

You can use the following wildcards in the text entry field:

- * matches zero or more occurrences of any characters

- . matches any character

- [c..] matches one of the characters in the set

Virtuoso Schematic Editor User Guide

Form Descriptions

Search Scope specifies the scope of the search.

selected set searches only for objects in the selected set.

cellview searches only for objects in the cellview currently displayed in the design window.

hierarchy expands the form to display two more option fields: *top* and *current cellview*. Specify the cellview from which you want the system to start the search. The system searches all cellviews specified in the design hierarchy.

library expands the form to display a text field. Specify the library that you want the system to search. The system searches all cellviews in the library. The library must exist in the current library search path.

Zoom To Object specifies how a matching object is displayed. When this option is selected, the schematic editor zooms in on the search object and centers the object in the design window.

Zoom Scale appears only when you turn on *Zoom To Object*. Specifies the scale used to enlarge the object.

Preview Changes (No Edits), when enabled, writes messages to the Command Interpreter Window detailing what would happen when you click *Apply*, *Replace*, *Skip*, or *ReplaceAll*.

Save Changes, when enabled, writes changes to disk without prompting you to confirm the changes. Only cellviews in the hierarchy or library are affected; the cellview currently displayed in the design window is not affected. When disabled, prompts you to confirm the changes when you attempt to start the *File – Close* command or exit the software.

Show Object Filter expands the form to show the Search Filter. You can use the filter to limit the search scope to only the objects you select.

Schematic Objects lists the schematic object types. You can specify any combination of object types, or you can use one of the following options:

all enables the selection button for all the object types.

none disables the selection button for all the object types. You can use this option to quickly clear any existing selections before you begin to create a new selection set.

Instance Objects lists the instance object types. You can specify any combination of the object types, or you can use one of the following options.

all enables the selection buttons for all the object types.

Virtuoso Schematic Editor User Guide

Form Descriptions

none disables the selection button for all the object types. You can use this option to quickly clear any existing selections before you begin to create a new selection set.

Related topics

[Replacing Properties](#)

Symbol Select By Filter

Use this form to select all the objects of one or more object types. It is accessed using the *Edit – Select – Select By Filter* command.

Form Action controls whether the system selects or deselects all the specified object types when you click *OK* or *Apply*.

select selects all the specified object types.

deselect selects everything but the specified object types.

Symbol Objects lists the symbol object types.

all turns on symbol object types.

none turns off symbol object types.

Related topics

[Selecting Objects to Edit](#)

Symbol Select By Property

Use this form to limit a selection set to object types with certain property values. It is accessed from the symbol editor using the *Edit – Select – Select By Property* command.

Find lets you specify values for the search.

The cyclic field lists frequently used properties.

When you select *other* from the cyclic field, the Add New Property for Search form appears.

The relational operators let you limit the search to objects that have a property value that either matches or differs from the value specified in the field showing the property value.

The text entry field specifies a value of the property.

When you select *master* from the list of property names, you must specify a property value of the form

`"libname cellname viewname"`

Form Action controls whether the system selects or deselects the specified object types when you click *OK* or *Apply*.

select selects all the specified object types.

deselect selects everything but the specified object types.

Zoom Out To Fit All Figures zooms the schematic editor out to display all the objects that match the selection criteria.

Show Object Filter expands the form to show the Selection Filter.

Symbol Objects lists the symbol object types.

all turns on symbol object types.

none turns off symbol object types.

Related topics

[Selecting Objects by Property](#)

Symbol Selection Filter

Use this form to specify the objects you want to filter for selection. It is accessed using the *Options – Select Filter* command.

Area Partial Selection specifies whether the symbol editor includes partial objects in the selection set. Partial objects can be wire endpoints and vertices or note shape object endpoints, vertices, and edges.

When this option is turned off (default), the schematic editor does not include partial objects when you drag your cursor through the design to select several objects at once (area selection).

When this option is turned on, partial objects are included when you drag your cursor through the design to select several objects at once (area selection).

Environment variable: [symbolAreaPartialSelect](#)

Symbol Objects lists the symbol object types you can select with your cursor.

all turns on all the object types.

none turns off all object types.

Environment variable: [symbolSelectFilter](#)

Related topics

[Selecting Objects by Filtering](#)

Update Pins

Use this form to update schematic or symbol pins to match those in another view of the same cell. It is accessed using the *Edit – Update Pins From View* command.

Update from view specifies the view from which the updates are to be applied to the current cellview. A default selection is automatically selected appropriate for the current cellview.

The scrollable text area of the form displays the available updates from the selected view.

This works from schematic to symbol cellview, and symbol to schematic cellview.

The SKILL function [schHiUpdatePinsFromView](#) can also be used to invoke the form.

Related topics

[Updating Pins Across Cellviews](#)

VHDL Properties

Use this form to display VHDL properties in a cellview so you can set specific netlisting options. It is accessed using the *Edit – Properties – VHDL* command.

Generics specifies the property names and the corresponding VHDL types, organized as SKILL lists, for the netlister to use when generating the entity.

The *Add*, *Modify*, and *Delete* buttons affect only the display of properties on the VHDL Properties form. You must click *OK* or *Apply* to apply the changes to objects in your design.

Add displays the Add Generic form, which lets you add a new generic to the list box.

Modify displays the Modify Generic form, which lets you redefine the type and value for a generic displayed in the list box. When you select a generic in the list box before you select the command, the system fills in the form with generic information.

Delete displays the Delete Generic form, which lets you delete a specified generic from the list box. When you select a generic in the list box before you select the command, the system fills in the form with generic information.

Scalar Data Type specifies the default data type for the netlister to use when generating the entity.

Vector Data Type specifies the default data type for the netlister to use when generating vector ports and signals if no data type is specified on the pin or signal itself.

Use Clause specifies a list of packages to be inserted by default in the context clause of the VHDL entity generated by the netlister.

User Comment shows a string to be inserted as the first line of each VHDL entity and architecture generated by the netlister.

Related topics

[Editing VHDL Properties](#)

Configuring and Using Kanji Notes

The Virtuoso® Schematic Editor provides a user interface that lets you create or modify EUC-packed format kanji files.

Based on the information in a kanji file, the software generates a symbol cellview that contains kanji character graphics. You can then add documentation to your schematics by placing an instance of the symbol.

You can configure your system to access the *Create – Note – Kanji Note* menu option for instantiating kanji note symbols in your schematic design.

The following topics are covered in this chapter.

- [Configuring Your System](#)
- [Using Kanji Notes](#)
- [Limitations](#)

Configuring Your System

To enable the user interface to use kanji notes you need to perform the following actions:

- [Specifying the Kanji Editor](#)
- [Configuring the Kanji Menu Command](#)



The above configuration tasks must be completed before accessing *Create – Note – Kanji Note*.

Specifying the Kanji Editor

The value of the `invokeKanjiEditor` environment variable determines which kanji EUC text editor you access from the user interface.

The default value is set to:

```
schematic invokeKanjiEditor string "kterm -km euc -e vi"
```

If the default value does not work for your system, change the default setting in your `.cdsenv` or `.cdsinit` file.

Testing the Default Setting

To test whether the default value works for your system,

- At a UNIX prompt, type:

```
kterm -km euc -e vi /tmp/kanji_test
```

If a kanji editor opens, the default value works for your system.

Configuring the Kanji Menu Command

The *Create – Note – Kanji* menu option is added using the `enableKanjiNotes` environment variable.

A bindkey (`Shift<Key>k`) is also available, and both this and the `enableKanjiNotes` `envvar` value can be set in your `.cdsenv` or `.cdsinit` prior to loading the sample bindkeys.

For example:

- `.cdsenv`:
`schematic enableKanjiNotes boolean t`
- `.cdsinit`:
`envSetVal("schematic" "enableKanjiNotes" boolean t)
load(prependInstallPath("samples/local/schBindKeys.il"))`

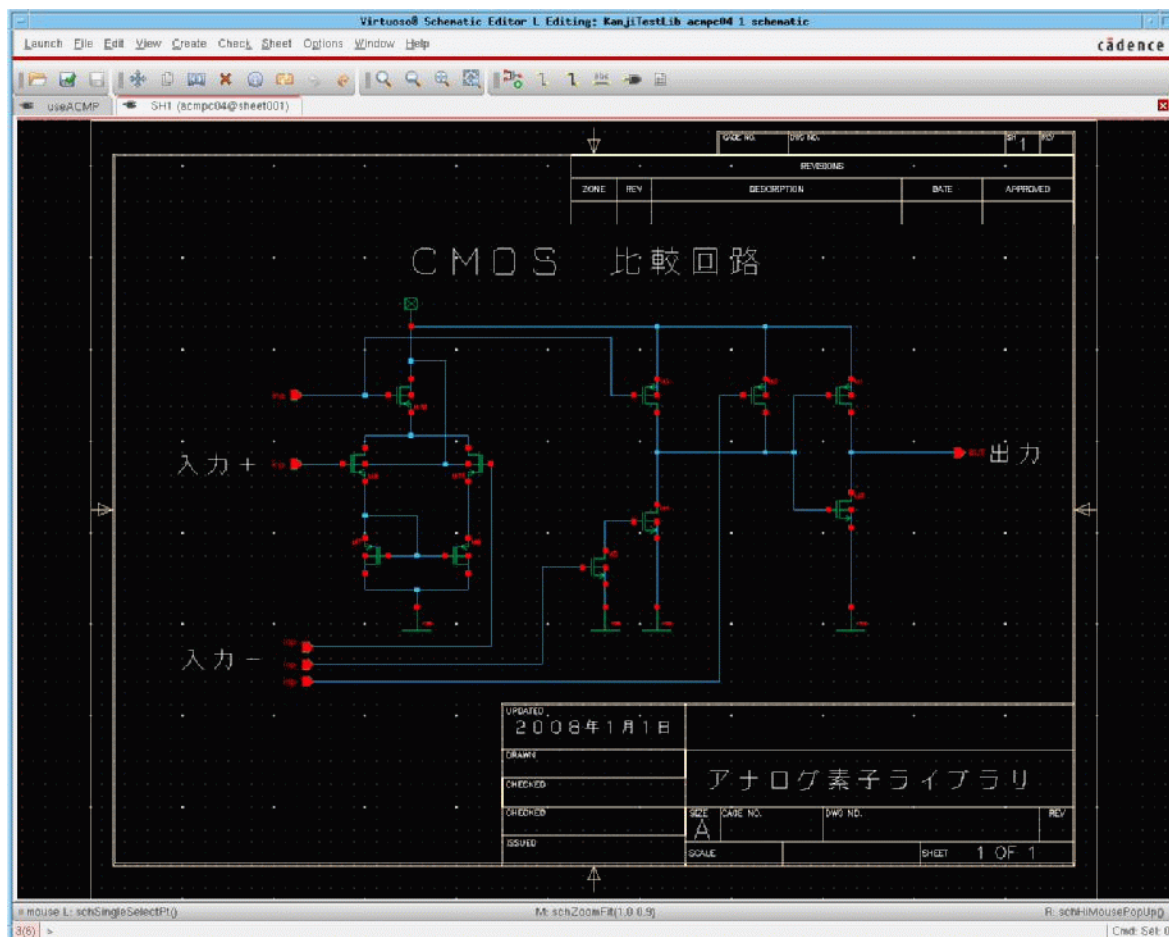


Figure E-1 Kanji Note Example

Using Kanji Notes

The *Create – Note – Kanji Note* command lets you create symbol cellviews containing kanji characters that you can instantiate in your schematic design.

The general procedure for creating and placing a kanji note is as follows:

1. Use a kanji text editor to create or edit a kanji EUC-formatted file.

Note: See also [Specifying the Kanji Editor](#).

2. Generate a symbol cellview based on the information in the Japanese EUC-formatted file.
3. Add an instance of the symbol cellview to your design.

Note: See also [schHiCreateKanjiSymbol](#) in the *Virtuoso Schematic Editor SKILL Functions Reference*.

Formats, Files, and Views

You use *Create – Note – Kanji Note* to open a kanji text editor, which you then use to create a *Japanese EUC packed (multiple-width)* formatted file.

After you quit the kanji text editor, you click on the *OK* or *Apply* button on the Add Kanji Note form. The software will then:

- Generate a *symbol* cellview
- Create a *kanji* cellview

The symbol cellview contains the graphics that represent the characters described in the EUC file. The software draws each character in the symbol by referencing the X and Y coordinates, which are stored in the two kanji vector files.

The kanji cellview, which the software stores in your library, contains an exact copy of the information in the source *Japanese EUC packed (multiple-width)* formatted file. Because this copy resides in your library, you can use the schematic editor, whenever you wish, to change the file and regenerate the associated symbol and any instances.

Note: The software stores the symbol and the kanji cellviews in the same cell.

Adding a New Note

The basic procedure for adding a new kanji note to your schematic is to generate a new symbol and place an instance in your design. There are two ways to generate a new kanji note symbol:

- [Generating a Symbol from an Existing File](#)
- [Generating a Symbol from a New File](#)

Generating a Symbol from an Existing File

To generate a new kanji note symbol based on an existing Japanese EUC packed (multiple-width) formatted file,

1. From the schematic window, choose *Create – Note – Kanji Note*.

The [Add Kanji Note](#) appears.

The screenshot shows the 'Add Kanji Note' dialog box. The 'Input' section has a 'Kanji EUC File' field and an 'Open' button. The 'Output' section has fields for 'LibraryName' (Two_Stage_Opamp), 'Cell Name' (OpAmp1_kanji_1), 'Scale Factor' (0.5), 'Horizontal Spacing' (20), 'Text Direction' (horizontal with a dropdown arrow), and 'Vertical Spacing' (20). At the bottom are 'OK', 'Cancel', 'Apply', and 'Help' buttons.

Figure E-2 The Add Kanji Note Form

2. In the *Kanji EUC File* field, type the name of the existing EUC file.
3. (Optional) Change the default kanji note display parameters.

4. Click *OK*.

The software will then:

- ☐ Generate a symbol cellview
- ☐ Create a kanji cellview
- ☐ Open the Add Instance form and make the *Add Instance* command active

5. Move your cursor into the schematic.

A box that represents the instance of the kanji note symbol is bound to your cursor.

6. Place the symbol instance by clicking in the schematic.

The kanji note appears where you click your mouse. The *Add Instance* command remains active.

7. Press the `ESC` key to end the *Add Instance* command.

The Add Instance form closes.

Generating a Symbol from a New File

To generate a new kanji note symbol based on a new Japanese EUC-packed (multiple-width) formatted file:

1. From the schematic window, choose *Create – Note – Kanji Note*.

The Add Kanji Note form appears.

2. In the *File Editor* field, click *Open*.

The software initializes the form with a filename and the kanji text editor opens the new file.

3. In the kanji text editor, enter your kanji text, save the file, and quit the kanji text editor.

4. (Optional) Change the default kanji note display parameters.

5. Click *OK*. The software will then:

- ☐ Generate a symbol cellview
- ☐ Create a kanji cellview
- ☐ Open the Add Instance form and make the *Add Instance* command active

Virtuoso Schematic Editor User Guide

Configuring and Using Kanji Notes

6. Move your cursor into the schematic. A box that represents the instance of the kanji note symbol is bound to your cursor.

7. Place the symbol instance by clicking in the schematic.

The kanji note appears where you click your mouse. The *Add Instance* command remains active.

8. Press the `ESC` key to end the *Add Instance* command.

The Add Instance form closes.

Editing an Existing Note

To change an existing kanji note symbol and all instances of the note:

1. Select the instance object in your **schematic**.

The software highlights the object.

Note: You cannot edit kanji parameters in a symbol view as CDF parameters are not available. Instead, edit the kanji parameters in the schematic, copying over to the symbol view as necessary.

2. From the schematic window, choose *Edit – Properties – Objects*.

The Edit Object Properties form appears.

Virtuoso Schematic Editor User Guide

Configuring and Using Kanji Notes

The form displays the current display parameter settings as CDF values and provides access to the *Kanji TextFile Editor*.

The 'Edit Object Properties' dialog box is shown with the following sections and controls:

- Apply To:** Two dropdown menus set to 'only current' and 'instance'.
- Show:** Checkboxes for 'system' (unchecked), 'user' (checked), and 'CDF' (checked).
- Buttons:** 'Browse' and 'Reset Instance Labels Display'.
- Property Table:**

Property	Value	Display
Library Name	kanjiTestLib	off
Cell Name	acmpc04_kanji_8	off
View Name	symbol	off
Instance Name	I36	off
- User Property Table:**

User Property	Master Value	Local Value	Display
Kanji_File_Name	kanji/kanjitxt.euc		off
nlAction	ignore		off
- CDF Parameter Table:**

CDF Parameter	Value	Display
Kanji Scale Factor	1	off
Kanji Text Direction	horizontal	off
Kanji Horizontal Spacing	20	off
Kanji Vertical Spacing	20	off
Rename Kanji Note	☒	off
- Buttons:** 'Add', 'Delete', 'Modify' (above the User Property table); 'Kanji TextFile Editor' (below the CDF Parameter table).
- Footer Buttons:** 'OK', 'Cancel', 'Apply', 'Defaults', 'Previous', 'Next', 'Help'.

Figure E-3 Edit Object Properties Form with Kanji Fields

Virtuoso Schematic Editor User Guide

Configuring and Using Kanji Notes

3. (Optional) Open the *Kanji TextFile Editor*, change the kanji note text, and close the editor.

4. The software updates the kanji cellview.

You will not see any change in the Edit Object Properties form.

You will not see any change in the schematic editor window until you click *OK* or *Apply* in the Edit Object Properties form.

5. (Optional) On the Edit Object Properties form, change the parameter values.

You will not see any change in the schematic editor window until you click *OK* or *Apply* in the Edit Object Properties form.

6. Click *OK* or *Apply*.



The software regenerates the symbol cellview, which changes all instances of the symbol cellview.

Limitations

You should consider the following limitations when working with kanji notes.

- You can use scale factor values of less than 1.0 to create smaller characters. However, due to round-off errors, scale factor values of less than 0.25 might make many characters unreadable.
- Each kanji character is converted into many lines in a symbol cellview. Therefore, very large EUC files result in symbols that require huge amounts of memory, which can slow redraw speeds significantly.
- The following character sets are not used often and are not supported in either the software or the vector files:
 - ❑ EUC half-width katakana (code set 2)
 - ❑ JIS X 0212-1990 (code set 3)
- The conversion program reads only the first 5,000 characters in an EUC file.
- If there are errors in the EUC file, the conversion program generates error messages for only the first 20 errors. The conversion process continues, however, until the program reaches the end of the EUC file or the 5,000-character limit.
- Japanese EUC packed (multiple-width) formatted files are not deleted automatically.

After you generate the symbol and kanji cellviews, the files you create with the kanji text editor are no longer needed. Although they do not take up much disk space, you should periodically remove them to maintain a clean file system.

Note: When the symbol is created, the associated kanji view will store the kanji text.

Virtuoso Schematic Editor User Guide

Configuring and Using Kanji Notes

Bindkeys in Virtuoso Schematic Editor

This appendix lists the default bindkeys defined for Virtuoso Schematic Editor.

- [Introduction](#)
- [Arranged as a Keyboard Layout](#)
- [Arranged by Functions](#)
- [Related Topics](#)

Introduction

Bindkeys tie commonly-used commands to single keys, key combinations, or mouse sequences and can be used to automate repetitive tasks and speed up your work.

Virtuoso Schematic Editor User Guide

Bindkeys in Virtuoso Schematic Editor

Arranged as a Keyboard Layout

Key	
Ctrl	
Shift	
Ctrl + Shift	

Esc	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	Delete
Cancel a Command	Launch Cadence Help		Raise most recent open form	Toggle Partial Select		Redraw the schematic					Show or Hide various Assistants		Delete
											Toggle Toolbars		
											Toggle Assistants and Toolbars		
		#											
		Toggle Anchor Magnifier											
1	2	3	4	5	6	7	8	9	0	-	=	BackSpace	\
				Route Flight Line			Check Label Attachment	Add Net to the Probe			Open Symbol or Schematic view		
Tab	Q	W	E	R	T	Y	U	I	O	P	[	]	
	Edit Object Properties	Create Wire (narrow)	Descend Read	Rotate (EF)	Direct text edit		Undo	Create Instance	Open the Display Options form	Create Pin	Zoom Out	Zoom in	
			Return		zoom to selected area								
	Edit Cellview Properties	Create Wire (wide)	Descend Edit		View Tree		Redo		Open the Editor Options form				

A	S	D	F	G	H	J	K	L	;	'		↑	
Select the Figure Pointed by the Mouse		Deselect the Figure Pointed by the Mouse	Zoom to Fit	Find Marker	Check Label Attachment			Create Wire Name		Update Magnifier Options		Pan up	
Select All	Save a Copy	Deselect all	Select Filter	Delete Marker									
Create Selection Box	Save the Design	Create a Deselection Box		Find Marker				Open Add Note Text form					
Z	X	C	V	B	N	M	,	.	/	Spacebar	←	↓	→
Zoom To Area	Check Current Cellview, Cross-View	Copy	Vector Pan	Create Block	Create Note Shape	Stretch		Toggle Magnifier		Create Wire Stubs and Names	Pan left	Pan down	Pan right
Zoom Out of Area			Go to CIW										
Go to Previous View	Check and Save		Toggle World View		Create Note text	Move							
Go to Next View													

Virtuoso Schematic Editor User Guide

Bindkeys in Virtuoso Schematic Editor

Arranged by Functions

Key	
Ctrl	
Shift	
Ctrl + Shift	

Opening	Creating	Editing	Saving	Zooming	Panning	Rotating	Toggling	Searching
O Open the Display Options form	P Create Pin	U Undo	S Save a Copy	[Zoom Out	I Pan up	R Rotate (EF)	F11 Show or Hide various Assistants	G Find Marker
Open the Editor Options form		Redo	Save the Design				Toggle Toolbars	Find Marker
	L Create Wire Name	C Copy	X Check	I Zoom in	= Pan left		Toggle Assistants and Toolbars	
			Check and Save				Toggle Anchor Magnifier	
= Open Symbol or Schematic view	N Create Note Shape	M Stretch	8 Check Label Attachment	Z Zoom To Area	I Pan down		.Toggle Magnifier	
				Zoom Out of Area				
	Create Note text	Move		Go to Next View				
F1 Launch Cadence Help	F Select Filter	Delete/Backspace Delete	H Check Label Attachment	F Zoom To Fit	→ Pan right		F3 Raise most recent open form	
	9 Add Net to the Probe	T Direct text edit			V Vector		F4 Toggle Partial Select	
L Open Add Note Text form	W Create Wire (narrow)	E Descend Read					V Go to CIW	
	Create Wire (wide)	Return Descend Edit					Toggle World View	
T View Tree	B Create Block	Q Edit Object Properties						
		Edit Cellview Properties						

Opening	Creating	Editing	Saving	Zooming	Panning	Rotating	Toggling	Searching
	F6 Redraw the schematic	F Select Filter						
	Spacebar Create Wire Stubs and Names	/ Update Magnifier Options						
	A Create Selection Box	5 Route Flight Line						
	D Create Deselection Box	A Select the Figure Pointed by the Mouse						
		Select All						
		G Delete Marker						
		I Create Instance						
		D Deselect the Figure Pointed by the Mouse						
		Deselect all						

Related Topics

- [Bindkey Storage](#)
- [Customizing Bindkeys](#)
- [Magnifier Bindkeys](#)
- [Stretching with Bindkeys](#)
- [Moving with Bindkeys](#)
- [Copying with Bindkeys](#)
- [Rotating with Bindkeys](#)
- [Direct Text Edit Bindkeys](#)
- [Using Bindkeys to Traverse the Design Hierarchy](#)

Using opticalLib to Create PDK (Virtuoso Photonics Option)

This chapter contains information about using the `opticalLib` library. The `opticalLib` library is a library in Virtuoso ADE. You access the library from the following path:

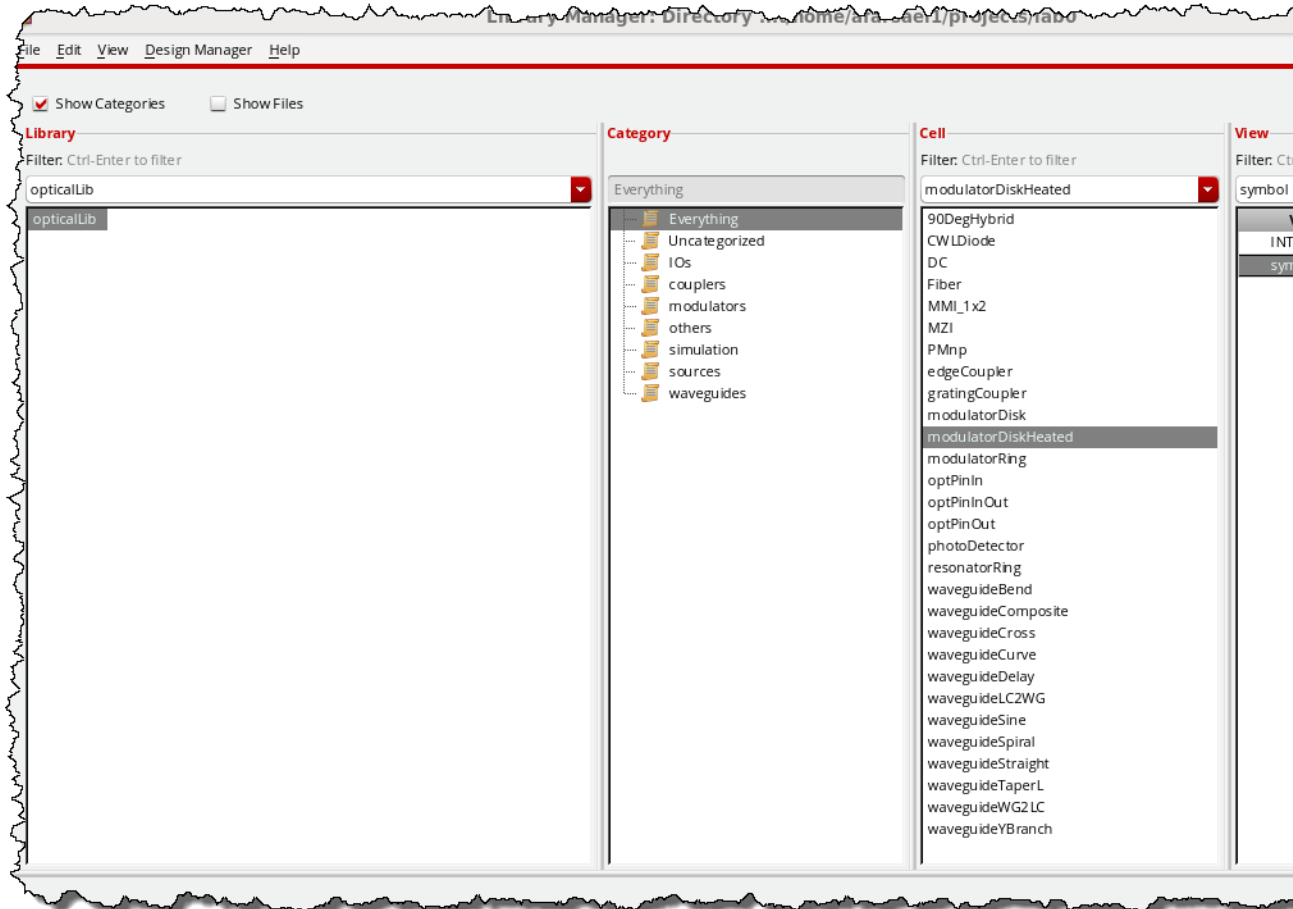
`your_install_dir/tools/dfII/etc/cdslib/opticalLib`

Ensure that you specify this path in the search path of the Set Library Search Path form.

Virtuoso Schematic Editor User Guide

Using opticalLib to Create PDK (Virtuoso Photonics Option)

The opticalLib library contains basic optical components, such as 90DegHybrid, CWLDiode, and Fiber. These basic optical symbols are used in building PDKs.



Refer to the following basic steps for creating a PDK by using the symbols available in opticalLib.

1. Add waveguide layers in the technology file.

Waveguide layers are similar to metal (conductive) layers and consist of the following components:

- `techLayers()`

These waveguide layers are usually placed after the metals and are assigned higher layer numbers to avoid conflicts with existing layers.

- `techLayerPurposePriorities()` / `techDisplays()`

Only the drawing and pin purposes are required to define these functions.

- `layerRules() -> functions()`

The waveguide layers with mask numbers higher than metal mask numbers are defined as *waveguides*.

- ❑ `constraintGroups() -> virtuosoDefaultExtractorSetup -> interconnect() -> validLayers`

These waveguide layers are added to support schematic driven layout.

2. Create the target PDK library by clicking *File -> New -> Library* in the CIW.
3. Load the technology file created in step1 into the library created in step2 by using the SKILL functions.

```
dbCreateLib( "<libName>" )
techId = techOpenTechFile( "<libName>" techGetDefaultTechName() "w")
if( isFile("techfile.tf") then
    tcLoadTechFile( techId "techfile.tf" "a")
);if
techSaveTechFile(techId)
techCloseTechFile(techId)
```

4. Copy symbols to the cellview.

The symbols are created as Pcells. Ensure that you use the SKILL function to copy the symbol graphics properly into the schematic.

```
cell = "<cellName>"
src = dbOpenCellViewByType( ddGetObj("opticalLib") cell "symbol"
"schematicSymbol" "r" )
dst = dbOpenCellViewByType( ddGetObj("<libName>") cell "symbol"
"schematicSymbol" "w" )
schCloneSymbol( dst src '(0 0) "R0" )
dbSave( dst )
```

5. Rename pins and update pin labels. If pins are being mapped to the existing layouts, the symbol pins and labels must also be updated.

```
dbFindTermByName( dst srcPin )~>name=dstPin
dbRenameNet( dbFindNetByName( dst srcPin ) dstPin )
foreach( shape dst~>shapes
if(shape~>theLabel && (shape~>theLabel == strcat("cdsTerm(\"" srcPin "\")"))
then
shape~>theLabel = strcat("cdsTerm(\"" dstPin "\")")
);if
);foreach
```

6. Create the INTERCONNECT view. The INTERCONNECT view is a copy of symbol and it is used as a stop view.

```
dbCopyCellView(
dbOpenCellViewByType( ddGetObj("<libName>") cell "symbol" "schematicSymbol"
"r" )
"<libName>" cell "INTERCONNECT" nil nil t ))
```

7. Create or update CDF for the PDK.

Virtuoso Schematic Editor User Guide

Using opticalLib to Create PDK (Virtuoso Photonics Option)

The CDF can be created or updated for a parameter by using the GUI or SKILL function.

```
dst = cdfCreateBaseCellCDF( ddGetObj("<libName>" cell) )
cdfCreateParam( dst
    ?name          "<name>"
    ?prompt        "<prompt>"
    ?defValue      "<defval>"
    ?type          "string"
    ?parseAsCEL    "yes"
);cdfCreateParam

unless(dst->simInfo
    dst->simInfo=ncons(nil)
)

unless(dst->simInfo->INTERCONNECT
    dst->simInfo->INTERCONNECT=ncons(nil)
    dst->simInfo->INTERCONNECT->componentName=cell
    dst->simInfo->INTERCONNECT->instParameters=<params>
    dst->simInfo->INTERCONNECT->termOrder=<terms>
)

cdfSaveCDF( dst )
```

Troubleshooting Schematic Connectivity

When connectivity updates result in violations, a flashing marker is displayed on each object that generates an error or warning and a corresponding message is displayed in the CIW. This appendix contains additional information about the connectivity error and warning messages issued. Each error message in the CIW is associated with a message ID. This ID can be used to locate in this appendix any relevant troubleshooting information to help you understand and resolve the issues.

The messages have been grouped as follows:

- Wire Naming
- Shorts and Misconnections
- Incompatible Connection Widths
- Tapping and Repeating
- Inherited Connections
- Patch Cords
- Indirect Shorts between Global Signals
- Indirect Shorts between Pins
- Indirect Shorts between Pins and Global Signals
- Unexpected Schematic Content
- Other Errors

You can use any of the following methods to view and manage the violation markers in a design:

- The *Check – Find Marker* command
- The Navigator assistant

Virtuoso Schematic Editor User Guide

Troubleshooting Schematic Connectivity

Inspect the problematic objects by selecting them in the Navigator assistant with Dynamic Net Highlighting enabled.

- The Annotation Browser

Right-click a violation marker in the Annotation Browser and choose *Select Nets*. You can trace the source of the nets to determine the cause of the problem.

- Virtuoso log files. See the [*Virtuoso Design Environment User Guide*](#).

Related Topics

[Updating Connectivity](#)

[Checking Designs](#)

[Resolving Check Errors](#)

[Finding Errors and Warnings in a Design](#)

[Highlighting Errors and Warnings in the CIW](#)

[Find Marker \(form\)](#)

[Annotation Browser](#)

[Dynamic Net Highlighting](#)

Wire Naming

The following error messages are included in this group:

- SCH-7060, SCH-7071, SCH-7072, SCH-7088: Incorrect Wire Label Syntax
- SCH-7064: Conflicting Labels on the Same Wire
- SCH-7065: Syntax Error in Wire Label
- SCH-7067: Incorrect Alias Label Syntax
- SCH-7068: Alias Width Does Not Match Net Width
- SCH-7069 and SCH-7084: Alias Label Not Glued to a Wire
- SCH-7070 and SCH-7077: Unconnected Wire Labels and Net Expression
- SCH-7078: Base Name Already in Use by Another Net
- SCH-7902: System-generated Net Prefix Cannot Be Used
- SCH-7903: Unexpected Format for Wire Label Glue Distance Setting

SCH-7060, SCH-7071, SCH-7072, SCH-7088: Incorrect Wire Label Syntax

The labels on the specified wires either have incorrect syntax or contain an empty label or alias.

Messages

Error: (SCH-7060) Incorrect syntax found in '<labelName>'. Used net name '<netName>'.

Warning: (SCH-7071) Ignored empty label on net '<netName>'.

Warning: (SCH-7072) Ignored empty alias on net '<netName>'.

Error: (SCH-7088) Cannot create net '<name>' because this net name is reserved for system use. Used net name '<netName>'.

Suggestions

Correct the syntax or provide the missing labels or aliases on the reported wires.

SCH-7064: Conflicting Labels on the Same Wire

There are multiple conflicting name labels attached to the same wire.

Message

Error: (SCH-7064) Ignored label '<labelName>' because multiple labels exist on wire '<labelName>'.

Suggestions

Ensure that the labels on a wire all have the same name.

SCH-7065: Syntax Error in Wire Label

The reported wire label does not follow the supported naming rules.

Message

Error: (SCH-7065) Incorrect syntax found in '<labelName>'. Used net name '<netName>'.

Suggestions

Add a correct wire name. For more information on naming conventions, see [Wire Connections and Wire Naming Conventions](#).

SCH-7067: Incorrect Alias Label Syntax

An alias label acts like a patch cord, establishing an alias between the signals of two different nets. The syntax of the specified alias label is incorrect.

Message

Error: (SCH-7067) Incorrect syntax found in alias '<labelName>'.

Suggestions

Correct the syntax of the alias label.



Although an alias label acts like a patch cord, it appears as a regular wire label and so it can be hard to identify. Cadence recommends that you use a patch cord rather than an alias label where possible.

SCH-7068: Alias Width Does Not Match Net Width

The alias label and the net it is attached to have different widths (that is, different numbers of bits).

Message

Error: (SCH-7068) Alias '<labelName>' and net '<netName>' have different widths.

Suggestions

Ensure that the label and the net it is attached to have the same width.

Related Topics

[Wire Connections and Wire Naming Conventions](#)

SCH-7069 and SCH-7084: Alias Label Not Glued to a Wire

An alias label acts like a patch cord, establishing an alias between the signals of two different nets. Labels or wire names that are unconnected to a wire are considered to be *unglued*. During Check operations, any unglued labels within a specified distance of a wire are glued to the wire. When an unglued alias label is beyond the specified distance from the wire, a marker is placed on the alias label.

Message

Warning: (SCH-7069) Label '<labelName>' is unconnected.

Error: (SCH-7084) Label '<labelName>' is unconnected.

Suggestions

Rerun the checking command after doing one of the following operations:

- Move the label closer to the wire.
- Adjust the specified distance set on the *Maximum Offset From Wire* field on the Schematic Rules Check Setup form.



Although an alias label acts like a patch cord, it appears as a regular wire label and so it is hard to identify. Where possible, Cadence recommends using a patch cord rather than an alias label.

Related Topics

[Schematic Rules Checks Setup](#)

[Patch Cord Connections and Patch Cord Naming Conventions](#)

[maxLabelOffsetUU](#) (environment variable)

SCH-7070 and SCH-7077: Unconnected Wire Labels and Net Expression

A wire name or a net expression is defined as an alias but the label is not attached to anything. Labels or wire names that are unconnected to a wire are considered to be *unglued*. During Check operations, any unglued labels within a specified distance of a wire are glued to the wire. When an unglued alias label is beyond the specified distance from the wire, a marker is placed on the alias label.

Messages

Warning: (SCH-7070) Alias '<labelName>' is unconnected.

Warning: (SCH-7077) Net expression '<netExpression>' is unconnected.

Suggestions

Open the *Edit – Properties – Object* form, select the alias label, and use the *Attachment* button to glue it to another net in the design. If the alias label is not required, delete it.

SCH-7078: Base Name Already in Use by Another Net

The same base name is used for different net types, for example, x for scalar and x<1:2> for bus.

Messages

Error: (SCH-7078) Cannot create net '<name>' because this net name is already in use by net '<netName>'. Used net name '<netName>'.

Suggestions

- Rename one of the nets. It is likely that a wire label mentions one of the problematic names.
- Split the design across multiple schematics if you want to use the same base name in different contexts.
- Check for typos in tap labels.

SCH-7902: System-generated Net Prefix Cannot Be Used

The name prefix is set to a reserved prefix, for example N__, under *Options – Editor – Connectivity – Net Name Prefix*.

Message

Error: (SCH-7902) Cannot use prefix '<string>' to create bus nets because it is reserved for system use. Specify a different prefix.

Suggestions

Prefix N__ is reserved for system use. Use a different prefix.

SCH-7903: Unexpected Format for Wire Label Glue Distance Setting

The value for glue distance property has an invalid data type.

Message

Warning: (SCH-7903) Cellview label glue distance property '<propName>' should be a float.

Suggestions

Ensure that the glue distance property is specified as a floating point number.

Related Topics

- [maxLabelOffsetUU](#)

Shorts and Misconnections

Wire labels, pins, and global symbols name the nets and signals carried by portions of a circuit. When multiple objects use the same name or a single object uses multiple names , the system cannot determine which name should be used. In these cases, the schematic editor propagates each name as far as possible and then marks the points where conflicting names meet.

The following error messages are included in this group:

- SCH-7003: Short between Nets
- **SCH-7005 to 7008: Shorted Signals across a Feed-through Pin**

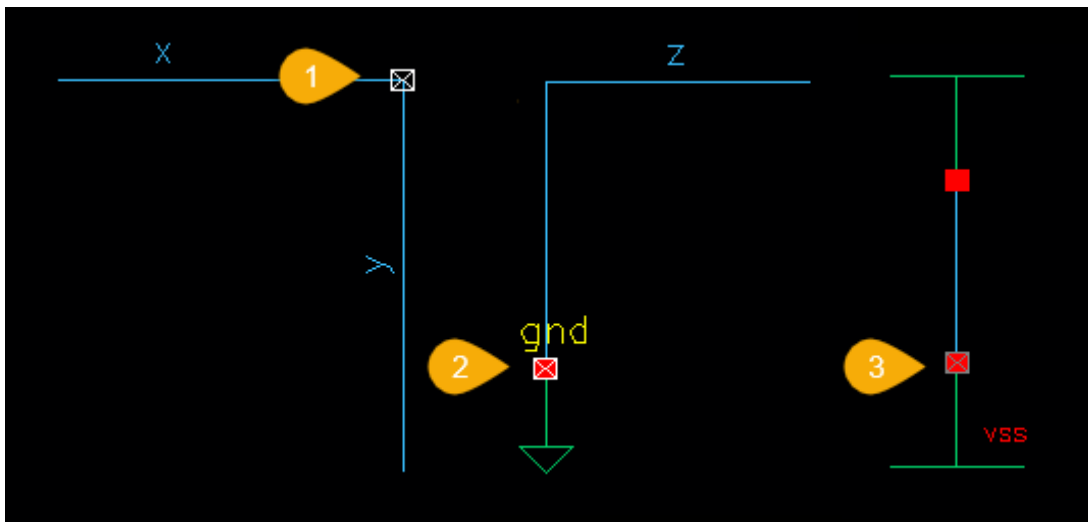
SCH-7003: Short between Nets

Two conflicting wire names cannot appear on different parts of the same wire. You can add the same wire label at multiple places on the same wire. This is useful if the wire is very long.

Note: Attaching multiple conflicting names to the same segment of wire returns error [SCH-7064: Conflicting Labels on the Same Wire](#).

Message

Error: (SCH-7003) Short between 'y' and 'x'.



1. Short between 'y' and 'x'
2. Short between 'gnd!' and 'z'
3. Short between 'vss!' and 'vdd!'

Suggestions

- To assign multiple names to a net, see [Patch Cord Connections and Patch Cord Naming Conventions](#).
- To pull out bits from a multi-bit net, see [Bus Tapping](#).

Related Topics

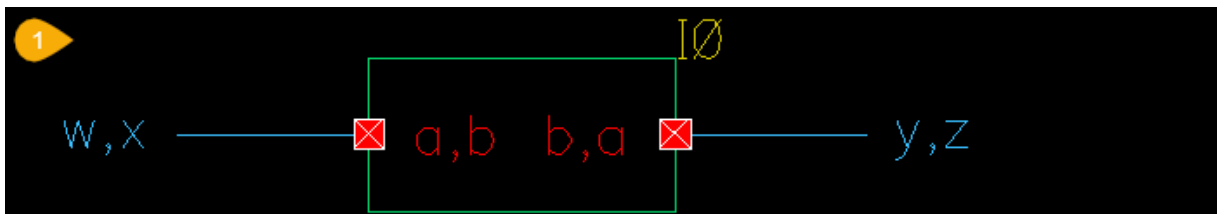
- [Resolving Check Errors](#)
- [Understanding Connectivity and Naming Conventions](#)

SCH-7005 to 7008: Shorted Signals across a Feed-through Pin

Multiple pins are using the same pin name, and the system cannot determine which should be used.

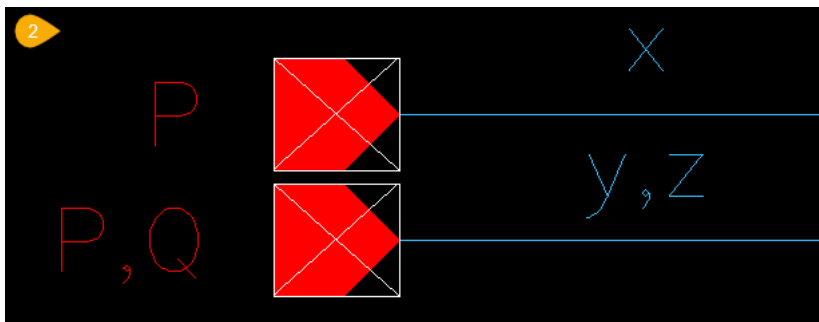
Messages

Error: (SCH-7005) Short between y and x through common pin b in a,b and b,a on instance I0.



The connection of the common pin on the specified instance terminal is causing a short circuit.

Error: (SCH-7006) Short between y and x through common pin P in P and PQ.



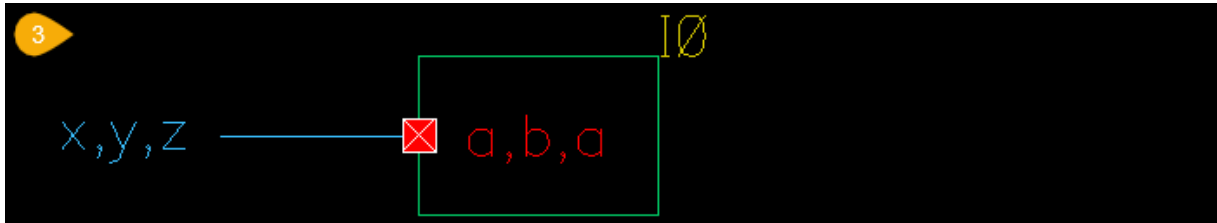
Make sure that the same signal is connected to each pin. In this case, change x to y or y, x to x, y .

However, when the pin P, Q was created, the intention might have been to create multiple pins. You can create multiple pins by separating each name with a space in the Create Pin form. Rename p in one of the pins. See [Create Pin – Schematic](#) and [Create Pin – Symbol](#) for more information.

Virtuoso Schematic Editor User Guide

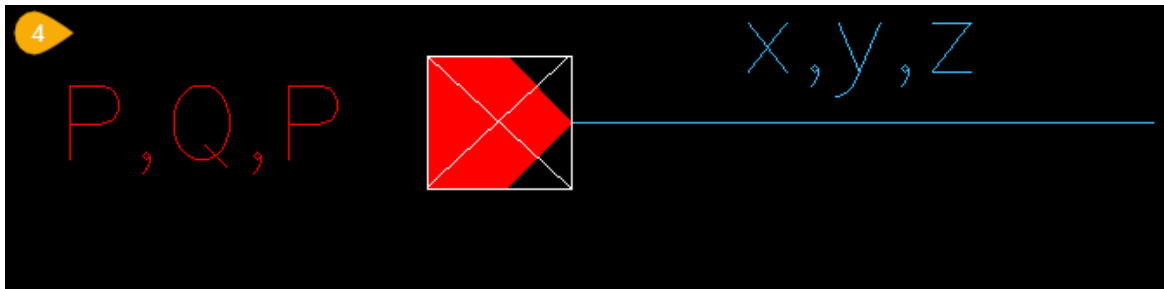
Troubleshooting Schematic Connectivity

Error: (SCH-7007) Short between z and x through pin a in a,b,a on instance I0.



The same bit name, a, appears multiple times in the instance pin a,b,a, and the net x,y,z has different nets in the corresponding positions causing a short. Ensure that the net connected to the pin has a name with the same pattern as the pin name.

Error: (SCH-7008) Short between 'z' and 'x' through pin 'P' in 'P,Q,P'.



The reported wires are causing a short circuit to the named pin.

Suggestions

- Make sure that the same signal is connected to each pin.
- Check the area on the schematic for wires overlapping the pin connection. Reconnect or replace the wire as required.
- Check the area on the schematic for instances where the wire has not fully connected with the pin. Locate the source of the electrical misconnection on the instance terminals between the nets specified in the message. Reconnect or replace the wire as required.
- Make a direct connection between two pins without an intermediate wire by placing the pins so that they touch each other, they partially or fully overlap, or one pin is contained within another. Unlike wires, pins cannot be connected by name.
- Create multiple pins by separating each name with a space in the Create Pin form.

Virtuoso Schematic Editor User Guide

Troubleshooting Schematic Connectivity

- Tapping pins: A tap cannot be drawn directly from an instance (symbol) pin. To tap a multiple-bit connection to an instance pin, you must attach a wire named with a bus name to the instance pin. You can then draw taps from the named bus.
- To connect two pins on the same net, use a `cds alias` instance.

Related Topics

- [Using Wire Draw, Width, and Route Options](#)
- [Using Wire Snap and Gravity Options](#)
- [Understanding Connectivity and Naming Conventions](#)
- [Resolving Check Errors](#)

Incompatible Connection Widths

The following error messages are included in this group:

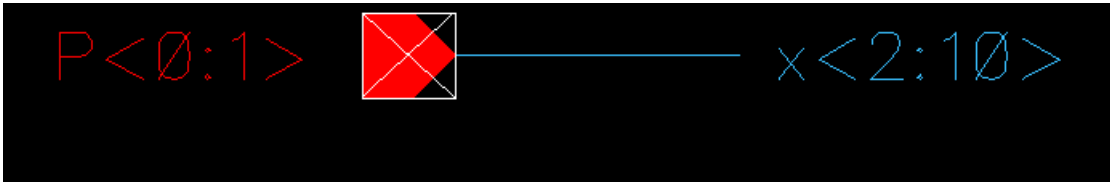
- SCH-7004 and SCH-7080: Misconnection between Unrelated Nets of Different Widths

SCH-7004 and SCH-7080: Misconnection between Unrelated Nets of Different Widths

Two nets with different widths are connected in a way that the system cannot interpret as a tap. For example, this can happen when trying to connect a multi-bit instance pin to a net.

Messages

Error: (SCH-7004) Misconnection between 'P<0:1>' (2 bits) and 'x<2:10>' (9 bits).



Error: (SCH-7080) Instance '<instanceName>' pin '<termName>' width (<numBits>) does not match '<netName>' (<numBits>).

Suggestions

- Ensure that the number of bits in the symbol pin is an exact multiple of the number of bits in the net.
- Connect the [onPageConn](#) symbol to the wire endpoint to prevent the dangling wire being highlighted as an error.
- When trying to tap certain bits of a net, ensure that all the bits being tapped are present on both sides of the connection.
- Single-bit instance pins can only connect to multi-bit nets if the multi-bit net is a repeat net (for example, "<*4>gnd!").
- When connecting a net to an instance pin with a larger number of bits, the net name must have the same number of bits as the pin name, not the instance name.

Related Topics

- [Resolving Check Errors](#)
- [Wire-to-Iterated Instance Connections](#)

Tapping and Repeating

The following error messages are included in this group:

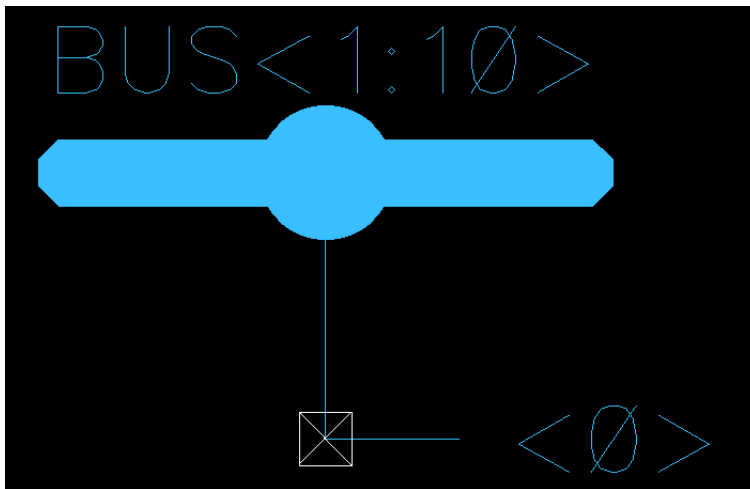
- SCH-7020 to SCH-7035, SCH-7050 and SCH-7051: Tapped Bits Out of Range
- SCH-7061: Unexpected Repeat Label
- SCH-7062: Unexpected Tap Label
- SCH-7116: Ambiguous Connection at Tap

SCH-7020 to SCH-7035, SCH-7050 and SCH-7051: Tapped Bits Out of Range

A tap must specify a member of the net from which it is tapped. If not, a corresponding message from the list below is displayed.

Messages

Error: (SCH-7020) Tap 'BUS<0>' is before the start bit of 'Bus<1:10>'.



The net BUS<1:10> from which the tap 0 is named does not contain a signal named BUS<0>.

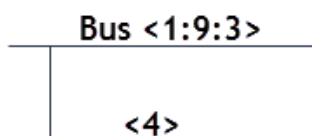
Error: (SCH-7021) Tap 'Bus<10>' is after the stop bit of 'Bus<1:9>'.

The net BUS<1:9> from which the tap 10 is named does not contain a signal named BUS<10>.

Error: (SCH-7022) Tap 'Bus<4>' is between the step bits of 'Bus<1:9>'.

Although the tapped bit value is within the range, it does not match the step bits in which the range increments.

In the vector notation <1:9:3> below, 3 is the incremental value. Every third value between 1 and 9 is a valid value or step bit in the range. Tap bit 4 is not valid because only the bits 3, 6, and 9 are valid.



Virtuoso Schematic Editor User Guide

Troubleshooting Schematic Connectivity

Error: (SCH-7023) Tap 'Bus<1:3>' start bit is before the start bit of 'Bus<2:10>'.

The tapped bit is not within the bus bit range. The first number (start bit) in the tapped bit range is lower than the start bit of the bus bit range.

The remaining messages in this section report variations of the same problem.

Error: (SCH-7024) Tap '<netName>' stop bit is before the start bit of '<netName>'.

Error: (SCH-7025) Tap '<netName>' start bit is after the stop bit of '<netName>'.

Error: (SCH-7026) Tap '<netName>' stop bit is after the stop bit of '<netName>'.

Error: (SCH-7027) Tap '<netName>' start and stop bits are before the start bit of '<netName>'.

Error: (SCH-7028) Tap '<netName>' start and stop bits are after the stop bit of '<netName>'.

Error: (SCH-7029) Tap '<netName>' straddles '<netName>'.

Error: (SCH-7030) Tap '<netName>' start bit is between the step bits of '<netName>'.

Error: (SCH-7031) Tap '<netName>' stop bit is between the step bits of '<netName>'.

Error: (SCH-7032) Tap '<netName>' start and stop bits are between the step bits of '<netName>'.

Error: (SCH-7033) Tap '<netName>' step bits are between the step bits of '<netName>'.

Error: (SCH-7034) Tap '<netName>' not in '<netName>'.

Error: (SCH-7035) Tap '<netName>' cannot tap from vector bit '<netName>'.

Error: (SCH-7050) Tap '<netName>' is not in bundle '<netName>'.

Error: (SCH-7051) Tap '<netName>' has bits other than '<netName>'.

Suggestion

In the schematic, rename the tap to a valid wire segment in the multiple-bit range that forms the bus.

Virtuoso Schematic Editor User Guide

Troubleshooting Schematic Connectivity

Related Topics

- [Tapping Multiple Bits of a Bus](#)
- [Finding the Range of Bits in a Bus](#)
- [Understanding Connectivity and Naming Conventions](#)

SCH-7061: Unexpected Repeat Label

A tap is labeled without enough context, for example, <*8>, and the system cannot determine what you intend to repeat.

Message

Error: (SCH-7061) Cannot determine what to repeat for '<labelName>'. Used net name '<netName>'.

Suggestions

Provide the full name of the net you want to use, for example, <*8> (a, b) .

Alternatively, make sure the tap is connected to a labeled net.

SCH-7062: Unexpected Tap Label

A wire is labeled with a name that looks like a tap, for example, "<1>", "3", "<a,b,c>" but does not appear in a tapping context.

Message

Error: (SCH-7062) Cannot determine what to tap for '<labelName>'. Used net name '<netName>'.

Suggestions

Provide the full name of the net you want to use and check your drawing to see if you are missing any other wire labels that give context to the tap label.

SCH-7116: Ambiguous Connection at Tap

There is a connection between two multi-bit nets whose names are permutations of each other (that is, they have the same bits but in a different order). The extractor cannot determine which name to use for the tap.

Messages

Error: (SCH-7116) Ambiguous connection at tap between nets 'A<2:0>' and 'A<0:2>' which are permutations of each other.

Suggestions

Remove the ambiguity in the net names.

Inherited Connections

The following messages are included in this group:

- SCH-7015: Conflicting Net Expressions on the Same Net
- SCH-7074 and SCH-7075: Conflict between Net Expression and Wire Label
- SCH-7079 to SCH-7087: Incomplete Net Expression Label
- SCH-7111 to SCH-7115: Circular References
- SCH-7280: Non-inherited Pin with Net Expression
- SCH-7281: Global Schematic Pin Name Inconsistent with Net Expression Default
- SCH-7282 to SCH-7284: Net Expression with Non-global Default Net

SCH-7015: Conflicting Net Expressions on the Same Net

A short circuit has been identified between the specified net expressions.

Message

Error: (SCH-7015) Short between net expression '@power:%vdd!' and '@vdd:%vdd!'.

Suggestion

In the schematic, locate the source of the electrical misconnection between the net expressions specified in the message. Reconnect or replace the wire as required.

Related Topics

[Understanding Connectivity and Naming Conventions](#)

SCH-7074 and SCH-7075: Conflict between Net Expression and Wire Label

A wire name is attached to a segment of wire that already has an attached net expression label.

Message

Error: (SCH-7074) Ignored label '<labelName>' because multiple labels exist on wire with net expression '<netExpression>'.

Error: (SCH-7075) Ignored net expression '<netExpression>' because multiple labels exist on wire with net expression '<netExpression>'.

Suggestions

Make sure each wire segment has only one label by deleting the incorrect label.

SCH-7079 to SCH-7087: Incomplete Net Expression Label

There is a malformed net expression label present. This may have been created using SKILL or by a non-Virtuoso tool.

Message

Error: (SCH-7079) Label '<labelName>' has the labelType of an evaluated label but it does not contain an expression that can be evaluated either by Virtuoso Schematic Editor or by the SKILL language to determine which is the resulting value that should be used to name a net. Used net name '<netName>'.

Error: (SCH-7082) Label '<labelName>' does not have the netExpression property so it cannot be used to name a net. Used net name '<netName>'.

Error: (SCH-7083) Label '<labelName>' has the netExpression property but it is not of NLPExpr type so it cannot be used to name a net. Used net name '<netName>'.

Error: (SCH-7085) Label '<labelName>' has the labelType of an evaluated label but it does not contain an expression that can be evaluated either by Virtuoso Schematic Editor or by the SKILL language to determine which is the resulting value that should be used to name a net. Used net name '<netName>'.

Error: (SCH-7086) Label '<labelName>' does not have the netExpression property so it cannot be used to name a net. Used net name '<netName>'.

Error: (SCH-7087) Label '<labelName>' has the netExpression property but it is not of NLPExpr type so it cannot be used to name a net. Used net name '<netName>'.

Suggestions

Delete the label and recreate it using *Create – Net Expression* or `schCreateNetExpression()`.

Related Topics

[Create Net Expression](#)

SCH-7111 to SCH-7115: Circular References

A `netSet` property value can be either a simple value, such as `vdd`, or a redirecting `netSet`, which is another net expression, such as `[@vdd:!:vdd!]`.

The redirecting expression requests that the search for `netSets` continue using the alternative property name specified in the redirecting net expression. Circular references arise when the same `netSet` property is specified containing the redirecting expression, which creates a loop.

A circular reference can also be between two `netSets` on the same instance. For example,

An instance with net expression `[@a:!:a!]` has three redirecting `netSets`:

```
a = [@b:!:c!]
b = [@c:!:d!]
c = [@b:!:e!]
```

The circular reference occurs between `b = [@c:!:d!]` and `c = [@b:!:e!]`.

A circular reference can also be achieved in multiple indirect steps.

Messages

Error: (SCH-7111) Circular reference on instance I0 between net expression `[@a:!:b!]` on property b and net expression `[@a:!:d!]` on CDF parameter a.

Error: (SCH-7112) Circular reference on instance I0 between net expression `[@a:!:b!]` on property b and net expression `[@a:!:d!]` on property a.

Error: (SCH-7113) Circular reference on instance I0 between net expression `[@a:!:b!]` on instance pin it and net expression `[@a:!:d!]` on CDF parameter a.

Error: (SCH-7114) Circular reference on instance I0 between net expression `[@a:!:b!]` on CDF parameter a and net expression `[@a:!:d!]` on CDF parameter c.

Error: (SCH-7115) Circular reference on instance I0 with net expression `[@a:!:b!]` on property a.

Suggestions

- Avoid direct or transitive circular references:

Direct: `a = [@a:!:vdd!]`

Transitive: `a = [@b:!:vdd!]` and `b = [@a:!:vdd!]`

- Rewrite the net expression to avoid circular references.

Related Topics

- [The Syntax of an Inherited Net Expression](#)
- [How Net Expressions Evaluate](#)

SCH-7280: Non-inherited Pin with Net Expression

A pin without an attached net expression connects to a net with a net expression.

Message

Error: (SCH-7280) Non-inherited pin '<termName>' connected to inherited net '<netExpression>'.

Suggestions

Move the net expression from its current position on a wire to the pin. Alternatively, add a matching net expression to the pin or remove the net expression.

Related Topics

[Inherited Connections](#)

[Inherited Connections Restrictions](#)

[Adding Schematic and Offsheet Pins](#)

SCH-7281: Global Schematic Pin Name Inconsistent with Net Expression Default

The specified pin uses a global net expression and its name is different from the default net name.

Message

Error: (SCH-7281) Pin name 'P!' uses global syntax and is inconsistent with the default net name 'b!' from its net expression '@a:%b!'.

Suggestions

Such inconsistencies are permitted, but each instance is flagged for review.

If the net expression is as you intended, ignore this message. If this is not intended, change the default net name or remove the net expression.

SCH-7282 to SCH-7284: Net Expression with Non-global Default Net

Net expressions with a non-global default net name are not valid.

Messages

Error: (SCH-7282) Net name 'local2' has a net expression '@z:local2' that does not have a global default net name.

Error: (SCH-7283) Pin name 'IN' has a net expression '@x:local' that does not have a global default net name.

Error: (SCH-7284) Instance 'I1' with pin 'Y' has a net expression '@z:local2' that does not have a global default net name.

Suggestions

Change the default net name or remove the net expression.

Patch Cords

The following error messages are included in this group:

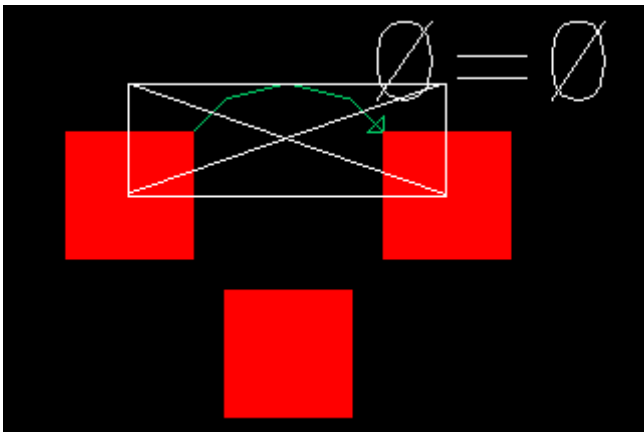
- [SCH-7090 to SCH-7092: Incorrectly Constructed Patch Cord Symbol](#)
- [SCH-7093 to SCH-7094: Incorrectly Connected Patch Cord Instance](#)
- [SCH-7095 to SCH-7102: Problems with Patch Cord Connection Expressions](#)
- [SCH-7103 to SCH-7110 and SCH-7286 Ripper Property Issues](#)

SCH-7090 to SCH-7092: Incorrectly Constructed Patch Cord Symbol

A valid patch cord must have a single source pin and a single destination pin. When a patch cord has multiple source or destination pins, it is marked as an error.

Messages

Error: (SCH-7090) Patch cord '<instanceName>' has unrecognized terminal '<termName>'.



Error: (SCH-7091) Patch cord '<instanceName>' must have a single source pin, but has <numPins>.

Error: (SCH-7092) Patch cord '<instanceName>' must have a single destination pin, but has <numPins>.

Suggestions

Ensure that each patch cord has a single source pin and a single destination pin.

Related Topics

[Patch Cords](#)

[Create Patchcord \(form\)](#)

SCH-7093 to SCH-7094: Incorrectly Connected Patch Cord Instance

You must connect both the source and destination pins of a patch cord.

Messages

Error: (SCH-7093) Patch cord '<instanceName>' requires a source net.

Error: (SCH-7094) Patch cord '<instanceName>' requires a destination net.

Suggestions

Connect both the source and destination pins of the given patch cord.

Related Topics

[Patch Cord Connections and Patch Cord Naming Conventions](#)

SCH-7095 to SCH-7102: Problems with Patch Cord Connection Expressions

These messages point out specific issues with patch cord connection expressions.

Messages

Error: (SCH-7095) Patch cord '<instanceName>' requires a '<propName>' connection expression property.

The patch cord is missing a connection expression property.

Error: (SCH-7096) Patch cord '<instanceName>' connection expression property '<propName>' must be a string."

The connection expression property for the patch cord is incorrect.

Error: (SCH-7097) Patch cord '<instanceName>' has incorrect syntax in connection expression '<expression>'.

There is a syntax error in the patch cord property.

Error: (SCH-7098) Patch cord '<instanceName>' width mismatch between <numBits>-bit source and <numBits>-bit destination sides of expression '<expression>'.

There is a width mismatch between the different bits of the patch cord.

Error: (SCH-7099) Patch cord '<instanceName>' source index '<index>' in connection expression '<expression>' exceeds width of net '<netName>'.

The source for the named patch cord expression is too wide for the net.

Error: (SCH-7100) Patch cord '<instanceName>' source index '<index>' in connection expression '<expression>' exceeds width of <numBits>-bit net.

The source for the named patch cord expression is too wide for the unique net.

Error: (SCH-7101) Patch cord '<instanceName>' destination index '<index>' in connection expression '<expression>' exceeds width of net '<netName>'.

The destination for the named patch cord expression is too wide for the net.

Error: (SCH-7102) Patch cord '<instanceName>' destination index '<index>' in connection expression '<expression>' exceeds width of <numBits>-bit net.

The destination for the named patch cord expression is too wide for the unique net.

Virtuoso Schematic Editor User Guide

Troubleshooting Schematic Connectivity

The number of bits in *srcVectorExpression* must equal the number of bits in *dstVectorExpression*.

Suggestions

Correct the specified error in the reported patch cord expression.

Related Topics

[Patch Cord Connections and Patch Cord Naming Conventions](#)

SCH-7103 to SCH-7110 and SCH-7286 Ripper Property Issues

Rippers are a special type of patch cord used in designs imported in Electronic Design Interchange Format (EDIF). They introduce explicit equivalences between the bits of two or more nets in a design.

When the system identifies an issue with a property name or property type of a ripper, it marks the issue with an appropriate message.

Messages

Error: (SCH-7103) Bus ripper '<ripperName>' requires a '<propName>' connection expression property.

The ripper is missing a connection expression property.

Error: (SCH-7104) Bus ripper '<ripperName>' connection expression property '<propName>' must be a string.

The connection expression property for the ripper must be a string.

Error: (SCH-7105) Bus ripper '<ripperName>' connection expression property '<propName>' has incorrect syntax in '<expression>'.

There is a syntax error in the ripper connection expression property.

Error: (SCH-7106) Bus ripper '<ripperName>' has width mismatch between external (SCH-<numBits>) and internal (SCH-<numBits>) sides of connection expression '<expression>'.

There is a width mismatch between the different bits of the ripper.

Error: (SCH-7107) Bus ripper '<ripperName>' requires a '<propName>' net name property.

The ripper is missing a net name.

Error: (SCH-7108) Bus ripper '<ripperName>' net name property '<propName>' must be a string.

The net name property for the ripper must be a string.

Error: (SCH-7109) Bus ripper '<ripperName>' net name property '<propName>' has incorrect syntax in '<netName>'.

There is a syntax error in the ripper net name.

Virtuoso Schematic Editor User Guide

Troubleshooting Schematic Connectivity

Error: (SCH-7110) Bus ripper '<ripperName>' destination index <index> in connection expression '<expression>' exceeds width of net '<netName>'.

The index for the named expression is too wide for the net.

Error: (SCH-7286) Bus ripper '<ripperName>' cannot patch net '<netName>' to net '<netName>'. Use a patch cord.

A ripper between the nets cannot be created. Use a patch cord instead.

Suggestion

Correct the specified error in the ripper.

Related Topics

[Virtuoso EDIF 200 Reader User Guide](#)

Indirect Shorts between Global Signals

Global nets make connections across the design hierarchy without using pins and wires to explicitly pass the signal through the hierarchy. Global nets are used primarily for power and ground connections.

The following error messages are included in this group:

- SCH-7200 to SCH-7203: Shorted Global Signals through a Patch Cord
- SCH-7220 to SCH-7223: Shorted Global Signals through an Alias Label
- SCH-7240 to SCH-7242 and SCH -7260: Shorted Globals across a Feed-through Pin

SCH-7200 to SCH-7203: Shorted Global Signals through a Patch Cord

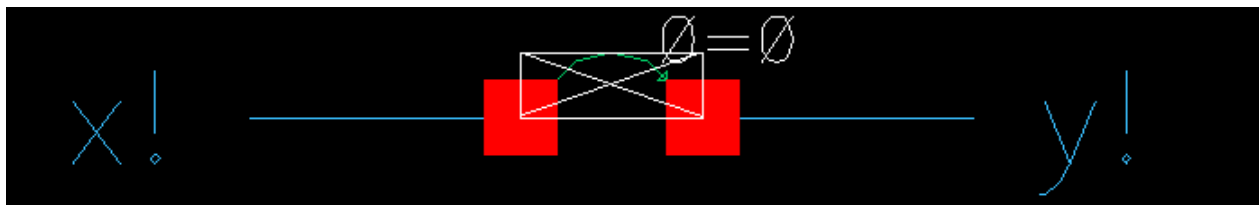
Patch cord symbols establish an alias between the signals of two different nets by combining them into a single logical net with multiple names. Unlike tap names, where you extract bits of a single net, you use patch cords to copy one net to another. You can also use a patch cord to map bits from one net to different bits of another net.

The arrow on a patch cord points to the net name that is used when a single name is required for a net with multiple names. Each logical net must have only one global name. Connecting two global nets with a patch cord is therefore marked as a problem.

When multiple object names conflict with each other, the system cannot determine which should be used. In this case, each name is propagated as far as possible and a marker is placed on the patch cord.

Messages

Error: (SCH-7200) Global signal x is aliased to global signal y by patch cord I0.



By using the vector expression $0=0$, the first bit of each net is made equivalent, and the patch cord does not have any effect on any of the other bits of either net.

Error: (SCH-7201) Global signal x! is aliased to global signal y! by patch cord P to s.

Error: (SCH-7202) Global signal x! is aliased to global signal y! by patch cord P from t.

Error: (SCH-7203) Global signal x! is aliased to global signal y! by patch cord P between s and t.

Suggestions

- Either of the nets may be connected to a schematic pin (I/O pin). An input pin cannot be connected to an output pin even if a patch cord is used in between.
- If the short between two terminals or a terminal and a global signal is intended, you can create a pseudo-short using the `cds_thru` component from the `basic` library.

Virtuoso Schematic Editor User Guide

Troubleshooting Schematic Connectivity

- If the net names contradict each other, check whether multiple names to a net have been assigned correctly.

Related Topics

- [Resolving Check Errors](#)
- [Bus Tapping](#)
- [Patch Cord Connections and Patch Cord Naming Conventions](#)
- [Understanding Connectivity and Naming Conventions](#)
- [Indirect Shorts between Pins and Global Signals](#)
- [Patch Cords](#)
- [Basic Library Symbols](#)

SCH-7220 to SCH-7223: Shorted Global Signals through an Alias Label

An alias label acts like a patch cord. It establishes an alias between the signals of two nets. When multiple object names conflict with each other, the system cannot determine which should be used. In this case, each name is propagated as far as possible and a marker is placed on the alias label.

Message

Error: (SCH-7220) Global signal x is aliased to global signal y by alias label.

Error: (SCH-7221) Global signal x is aliased to global signal y through alias '<alias>'.

Error: (SCH-7222) Global signal x is aliased to global signal y through net '<netName>' alias '<alias>'.

Error: (SCH-7223) Global signal x is aliased to global signal y through net '<netName>'.

Suggestions

- Either of the nets may be connected to a schematic pin (I/O pin). An input pin cannot be connected to an output pin even if an alias label is used.
- If the short between two terminals or a terminal and a global signal is intended, you can create a pseudo-short using the [cds_thru](#) component from the *basic* library.
- If the net names contradict each other, check if multiple names on a net have been assigned correctly.



Although an alias label acts like a patch cord, it appears as a regular wire label and so it can be hard to identify. Cadence recommends that you use a patch cord rather than an alias label where possible.

Related Topics

- [Resolving Check Errors](#)
- [Bus Tapping.](#)
- [Patch Cord Connections and Patch Cord Naming Conventions](#)

Virtuoso Schematic Editor User Guide

Troubleshooting Schematic Connectivity

- [Understanding Connectivity and Naming Conventions](#)
- [Indirect Shorts between Pins and Global Signals](#)
- [Patch Cords](#)
- [Basic Library Symbols](#)

SCH-7240 to SCH-7242 and SCH -7260: Shorted Globals across a Feed-through Pin

Two global signals are connected through instance pins that share common bits.

Messages

Error: (SCH-7240) Global signal '<netNameX>' is aliased to global signal '<netNameY>' by common pin '<termName>' in '<termName1>' and '<termName2>' on instance '<instanceName>'.

Error: (SCH-7241) Global signal '<netNameX>' is aliased to global signal '<netNameY>' through '<netName>' by common pin '<termName>' in '<termName1>' and '<termName2>' on instance '<instanceName>'.

Error: (SCH-7242) Global signal '<netNameX>' is aliased to global signal '<netNameY>' through '<netName1>' and '<netName2>' by common pin '<termName>' in '<termName1>' and '<termName2>' on instance '<instanceName>'.

Error: (SCH-7260) Global signal '<netNameX>' is aliased to global signal '<netNameY>' by common pin '<termName>' in '<termName1>' and '<termName2>'.

Suggestions

- Check that you intended to have common pins on your symbol.
- Disconnect one of the reported global signals.
- Use the [cds_thru](#) component to explicitly connect global nets.

Related Topics

- [Resolving Check Errors](#)
- [Bus Tapping](#)
- [Patch Cord Connections and Patch Cord Naming Conventions](#)
- [Understanding Connectivity and Naming Conventions](#)
- [Indirect Shorts between Pins and Global Signals](#)

Indirect Shorts between Pins

The following error messages are included in this group:

- SCH-7204 to SCH-7207: Shorted Pins through Patch Cord
- SCH-7224 to SCH-7227: Shorted Pins through Alias Label
- SCH-7243 to SCH-7245: Shorted Pins across a Feed-through Pin
- SCH-7270: Shorted Pins on Different Parts of the Same Net

SCH-7204 to SCH-7207: Shorted Pins through Patch Cord

Patch cord symbols establish an alias between the signals of two nets. Unlike tap names, where you extract bits of a single net, you use patch cords to copy one net to another. You can also use the patch cord to map bits from one net to different bits of another net.

Multiple pins are using the same patch cord, and the system cannot determine which should be used. In this case, each name is propagated as far as possible and markers are added at the points where conflicting names meet.

Messages

Error: (SCH-7204) Pin b is aliased to pin c by patch cord P.

Error: (SCH-7205) Pin b is aliased to c by patch cord P to netA.

Error: (SCH-7206) Pin b is aliased to pin c by patch cord P from netB.

Error: (SCH-7207) Pin b is aliased to pin c by patch cord P between NETA and NETB.

Suggestions

- Each patch cord has a single source (src) and destination (dst) pin. If the patch cord uses vector expressions, logically, each object on NETA is also connected to NETB, and the patch cord connects the first bit named in *srcVectorExpression* to the first bit named in *dstVectorExpression*. Check that the patch cord has been created correctly.
- To connect all the bits of the net to all the bits of the nets attached to the dst pin in the same order as they are attached to the src pin, omit the src vector expression and the equals sign (=) from the connection expression. For example, to connect two single-bit nets called NET1 and NET2 with a patch cord, a connection expression of 0 is sufficient.
- Check that the number of bits in the src vector expression equals the number of bits in the dst vector expression.
- If the net names contradict each other, check that multiple names on a net have been assigned correctly.
- If the short between two terminals is intended, you can create a pseudo-short using the *cds_thru* component from the *basic* library.
- Check if either of the nets are connected to a schematic pin (I/O pin). An input pin cannot be connected to an output pin even if a patch cord is used in between.

Virtuoso Schematic Editor User Guide

Troubleshooting Schematic Connectivity

Related Topics

- [Resolving Check Errors](#)
- [Patch Cord Connections and Patch Cord Naming Conventions](#)
- [Understanding Connectivity and Naming Conventions](#)
- [Adding Patch Cords Using the Add Instances Form](#)
- [Indirect Shorts between Pins and Global Signals](#)
- [Patch Cords](#)
- [Basic Library Symbols](#)

SCH-7224 to SCH-7227: Shorted Pins through Alias Label

Two global signals are connected because of an alias label.

Messages

Error: (SCH-7224) Pin '<termNameX>' is aliased to pin '<termNameY>' by alias label.

Error: (SCH-7225) Pin '<termNameX>' is aliased to pin '<termNameY>' through alias label '<alias>'.

Error: (SCH-7226) Pin '<termNameX>' is aliased to pin '<termNameY>' through net '<netName>' alias label '<alias>'.

Error: (SCH-7227) Pin '<termNameX>' is aliased to pin '<termNameY>' through net '<netName>'.

Suggestions

Use the [cbs_thru](#) component if you want to explicitly connect two different pins together.



Although an alias label acts like a patch cord, it appears as a regular wire label and so it is hard to identify. Where possible, Cadence recommends using a patch cord rather than an alias label.

Related Topics

- [Resolving Check Errors](#)
- [Understanding Connectivity and Naming Conventions](#)
- [Indirect Shorts between Pins and Global Signals](#)

SCH-7243 to SCH-7245: Shorted Pins across a Feed-through Pin

Two pins are connected through instance pins that share common bits.

Messages

Error: (SCH-7243) Pin '<termNameX>' is aliased to pin '<termNameY>' by common pin '<termName>' in '<termName1>' and '<termName2>' on instance '<instanceName>'.

Error: (SCH-7244) Pin '<termNameX>' is aliased to pin '<termNameY>' through '<netName>' by common pin '<termName>' in '<termName1>' and '<termName2>' on instance '<instanceName>'.

Error: (SCH-7245) Pin '<termNameX>' is aliased to pin '<termNameY>' through '<netName1>' and '<netName2>' by common pin '<termName>' in '<termName1>' and '<termName2>' on instance '<instanceName>'.

Suggestions

- Check that you have the pin arrangement you intended.
- Disconnect one of the reported pins.
- Use the [cds_thru](#) component if you want to connect multiple pins.

Related Topics

- [Resolving Check Errors](#)
- [Understanding Connectivity and Naming Conventions](#)
- [Indirect Shorts between Pins and Global Signals](#)

SCH-7270: Shorted Pins on Different Parts of the Same Net

Pins located in different sections of the same net are connected by the specified net causing a short.

Message

Error: (SCH-7270) Pins '<termNameX>' and '<termNameY>' connected through net '<netName>'.

Suggestion

Remove the net that connects the two pins on the same net.

Related Topics

- [Resolving Check Errors](#)
- [Understanding Connectivity and Naming Conventions](#)
- [Indirect Shorts between Pins and Global Signals](#)

Indirect Shorts between Pins and Global Signals

The following error messages are included in this group:

- SCH-7208 to SCH-7215: Shorted Pin and Global through Patch Cord
- SCH-7228 to SCH-7235: Shorted Pin and Global through Alias Label
- SCH-7246 to SCH-7251: Shorted Pin and Global across a Feed-through Pin
- SCH-7271: Non-global Pin Connected to Global Net
- SCH-7299: Short between Global Nets or Pins

SCH-7208 to SCH-7215: Shorted Pin and Global through Patch Cord

Global signals make connections across the design hierarchy without using pins and wires to explicitly pass the signal through the hierarchy. Global signals are used primarily for power and ground connections.

Patch cord symbols establish an alias between the signals of two different nets. Unlike tap names, where you extract bits of a single net, you use patch cords to copy one net to another. You can also use a patch cord to map bits from one net to different bits of another net.

When multiple object names conflict with each other, the system cannot determine which should be used. In this case, each name is propagated as far as possible and markers are added at the points where conflicting names meet.

Messages

Error: (SCH-7208) Pin P is aliased to global signal a! by patch cord I0.

Error: (SCH-7209) Pin P is aliased to global signal a! by patch cord '<instanceName>' to '<netName>'.

Error: (SCH-7210) Pin P is aliased to global signal a! by patch cord '<instanceName>' from '<netName>'.

Error: (SCH-7211) Pin P is aliased to global signal a! by patch cord '<instanceName>' between '<netName>' and '<netName>'.

Error: (SCH-7212) Global signal b! is aliased to S by patch cord '<instanceName>'.

Error: (SCH-7213) Global signal b! is aliased to S by patch cord '<instanceName>' to '<netName>'.

Error: (SCH-7214) Global signal b! is aliased to pin S by patch cord '<instanceName>' from '<netName>'.

Error: (SCH-7215) Global signal b! is aliased to pin S by patch cord '<instanceName>' between '<netName>' and '<netName>'.

Suggestions

- Each patch cord has a single source (src) and destination (dst) pin. If the patch cord uses vector expressions, logically, each object on NETA is also connected to NETB, and the patch cord connects the first bit named in *srcVectorExpression* to the first bit

named in *dstVectorExpression*. Check that the patch cord has been created correctly.

- To connect all the bits of the net to all the bits of the nets attached to the dst pin in the same order as they are attached to the src pin, omit the src vector expression and the equals sign (=) from the connection expression. For example, to connect two single-bit nets called NET1 and NET2 with a patch cord, a connection expression of 0 is sufficient.
- Check that the number of bits in the src vector expression equals the number of bits in the dst vector expression.
- If the net names contradict each other, check that multiple names to a net have been assigned correctly.
- If the short between two terminals is intended, you can create a pseudo-short using the [cds_thru](#) component from the `basic` library.
- Check if either of the nets are connected to a schematic pin (I/O pin). An input pin cannot be connected to an output pin even if a patch cord is used in between.

Related Topics

- [Resolving Check Errors](#)
- [Patch Cord Connections and Patch Cord Naming Conventions](#)
- [Understanding Connectivity and Naming Conventions](#)
- [Adding Patch Cords Using the Add Instances Form](#)
- [Indirect Shorts between Global Signals](#)
- [Indirect Shorts between Pins](#)
- [Patch Cords](#)
- [Basic Library Symbols](#)

SCH-7228 to SCH-7235: Shorted Pin and Global through Alias Label

A pin is connected to a global signal through one or more alias labels.

Messages

Error: (SCH-7228) Pin '<termNameX>' is aliased to global signal '<netNameY>' by alias label.

Error: (SCH-7229) Pin '<termNameX>' is aliased to global signal '<netNameY>' through alias label '<alias>'.

Error: (SCH-7230) Pin '<termNameX>' is aliased to global signal '<netNameY>' through net '<netName>' alias label '<alias>'.

Error: (SCH-7231) Pin '<termNameX>' is aliased to global signal '<netNameY>' through net '<netName>'.

Error: (SCH-7232) Global signal '<netNameX>' is aliased to pin '<termNameY>' by alias label.

Error: (SCH-7233) Global signal '<netNameX>' is aliased to pin '<termNameY>' through alias label '<alias>'.

Error: (SCH-7234) Global signal '<netNameX>' is aliased to pin '<termNameY>' through net '<netName>' alias label '<alias>'.

Error: (SCH-7235) Global signal '<netNameX>' is aliased to pin '<termNameY>' through net '<netName>'.

Suggestions

- Check the marked alias labels to ensure that your net equivalence relationships are as intended.
- Check that you have the intended global suffix ! on pins and wire labels.
- Select *View – Net Highlighting* to inspect the full set of connected objects. If your intention was to short two pins, or a pin to global, use the [cds_thru](#) component instead of alias labels.

Related Topics

- [Resolving Check Errors](#)

Virtuoso Schematic Editor User Guide

Troubleshooting Schematic Connectivity

- Understanding Connectivity and Naming Conventions
- Indirect Shorts between Global Signals
- Indirect Shorts between Pins
- Dynamic Net Highlighting

SCH-7246 to SCH-7251: Shorted Pin and Global across a Feed-through Pin

A schematic pin is connected to a global signal through one or more shared bits across multiple instance pins.

Messages

Error: (SCH-7246) Pin '<termNameX>' is aliased to global signal '<netNameY>' by common pin '<termName>' in '<termName1>' and '<termName2>' on instance '<instanceName>'.

Error: (SCH-7247) Pin '<termNameX>' is aliased to global signal '<netNameY>' through '<netName>' by common pin '<termName>' in '<termName1>' and '<termName2>' on instance '<instanceName>'.

Error: (SCH-7248) Pin '<termNameX>' is aliased to global signal '<netNameY>' through '<netName1>' and '<netName2>' by common pin '<termName>' in '<termName1>' and '<termName2>' on instance '<instanceName>'.

Error: (SCH-7249) Global signal '<netNameX>' is aliased to pin '<termNameY>' by common pin '<termName>' in '<termName1>' and '<termName2>' on instance '<instanceName>'.

Error: (SCH-7250) Global signal '<netNameX>' is aliased to pin '<termNameY>' through '<netName>' by common pin '<termName>' in '<termName1>' and '<termName2>' on instance '<instanceName>'.

Error: (SCH-7251) Global signal '<netNameX>' is aliased to pin '<termNameY>' through '<netName1>' and '<netName2>' by common pin '<termName>' in '<termName1>' and '<termName2>' on instance '<instanceName>'.

Suggestions

- Check the marked pins to ensure that your net equivalence relationships are as intended.
- Check that you have the intended global suffix ! on pins and wire labels.
- Select *View – Net Highlighting* to inspect the full set of connected objects. If your intention was to short two pins, or a pin to global, use the [cbs thru](#) component instead of pins.

Related Topics

- [Resolving Check Errors](#)
- [Understanding Connectivity and Naming Conventions](#)

Virtuoso Schematic Editor User Guide

Troubleshooting Schematic Connectivity

- Indirect Shorts between Global Signals
- Indirect Shorts between Pins

SCH-7271: Non-global Pin Connected to Global Net

A regular schematic pin (for example, DATA), connects to a wire labeled with a global wire label (for example, gnd!).

Messages

Error: (SCH-7271) Pin '<termName>' connected to global net '<netName>'.

Suggestions

- Remove the pin or wire label.
- Check for a missing ! suffix on the pin.
- Check that you intended to have a global connection.

SCH-7299: Short between Global Nets or Pins

The checker detected shorted globals or pins, but was unable to give any specific information about why this happened. It may be that a pre- or post-check trigger changed connectivity.

Messages

Error: (SCH-7299) Net '<netName>' shorts together multiple global nets and/or schematic pins.

Suggestions

- Select *View – Net Highlighting* to display connected objects.
- Check if your design environment has SKILL customizations that affect schematic check and save.

Unexpected Schematic Content

The following error messages are included in this group:

- [SCH-7066: Unexpected Missing Instance Terminal](#)
- [SCH-7285: Unexpected Pins on Sheet Border Instance](#)
- [SCH-7286: Unsupported Ripper Configuration](#)
- [SCH-7287: No-connection Annotation with Global Pin](#)
- [SCH-7288: No-connection Annotation with Net Expression](#)
- [SCH-7289: No-connection Annotation is Not Ignored by Netlister](#)

SCH-7066: Unexpected Missing Instance Terminal

There is no extended help available for this message.

Message

Error: (SCH-7066) Instance '<instanceName>' is missing an instTerm for terminal '<termName>'.

Suggestions

None.

SCH-7285: Unexpected Pins on Sheet Border Instance

The specified instance master has the property `schType=border`, suggesting that it represents a cosmetic sheet border. However, the master also has pins.

Message

Error: (SCH-7285) Invalid pin '<termName>' on sheet border instance '<instanceName>'.

Suggestions

- If the master is not supposed to be a sheet border, remove the `schType` property by the using the SKILL API `'dbDeletePropByName(master "schType")'`.
- If the master is supposed to be a border, remove the pins. For more information, see [Drawing a Sheet Border](#).

Related Topics

- [Drawing a Sheet Border](#)

SCH-7286: Unsupported Ripper Configuration

An EDIF 2.0 design is imported which contains ripper objects that function as patches, not as bus taps. This is not supported.

Message

Error: (SCH-7286) Ripper '<ripperName>' cannot patch net '<netName>' to net '<netName>'. Use a patch cord.

Suggestions

- Replace the ripper with a patch cord.
- Adjust the wire naming so that the ripper becomes a tap.

For more information, see [How EDIF 200 In Translates Connectivity](#) in the *[Virtuoso EDIF 200 Reader User Guide](#)*.

SCH-7287: No-connection Annotation with Global Pin

The specified instance master has the property `schType=noConn`, indicating that it is an annotation to mark an instance pin as intentionally not connected to anything else. However, the instance master has a global pin, which creates global connectivity.

Message

Error: (SCH-7287) No-connection annotation '<instanceName>' has global pin '<pinName>'.

Suggestions

- "If the master is not supposed to be a no-connection annotation (for example, because it is supposed to indicate a supply rail connection), remove the `schType` property by using the SKILL API `'dbDeletePropByName(master "schType")'`.
- "If the master is supposed to be a no-connection annotation, turn the symbol pin into a regular scalar pin by ensuring that the pin name does not end with "!".

SCH-7288: No-connection Annotation with Net Expression

The specified instance master has the property `schType=noConn`, indicating that it is an annotation to mark an instance pin as intentionally not connected to anything else. However, the instance master has a pin with a net expression, which creates global connectivity.

Message

Error: (SCH-7288) No-connection annotation '<instanceName>' has a pin '<pinName>' with a net expression.

Suggestions

- If the master is not supposed to be a no-connection annotation (for example, because it is supposed to indicate a supply rail connection configurable through an inherited connection), remove the `schType` property by using the SKILL API
`'dbDeletePropByName(master "schType") '`.
- "If the master is supposed to be a no-connection annotation, turn the symbol pin into a regular scalar pin by ensuring that the pin name does not end with "!".

SCH-7289: No-connection Annotation is Not Ignored by Netlister

The specified instance master has the property `schType=noConn`, indicating that is an annotation to mark an instance pin as intentionally not connected to anything else. However, the instance master does not have the property `nlAction=ignore`, meaning that it represents an object that should be included in the netlist, creating a connection.

Message

Warning: (SCH-7289) Instance '<instanceName>' is ambiguous because its master cellview has the `schType` property with value 'noConn', but does not have the `nlAction` property with value 'ignore'.

Suggestions

- If the master is not supposed to be a no-connection annotation, remove the `schType` property by using the SKILL API `'dbDeletePropByName(master "schType")'`.
- "If the master is supposed to be a no-connection annotation, add the property `nlAction=ignore`.

Other Errors

The following error messages are included in this group:

- Ungrouped Messages

Ungrouped Messages

An unexpected problem occurred while updating connectivity.

Error: (SCH-7999) Internal error while updating database: <exception>

Suggestions

Investigate and resolve the issue, then select *Run Check – Current Cellview*. If you are unable to isolate the problem, contact Cadence Customer Support for assistance.

Finding the Range of Bits in a Bus

To find the range of bits in a bus:

1. Select a bus wire on the canvas.
2. Open the Navigator assistant.

The bus is selected in the Navigator.

3. Click the + icon to see the range of bits in the bus.

In SKILL:

1. Select a bus wire on the canvas.
2. In the CIW, enter `car(getSelectedSet())->net~>signals~>name`

The bus bits are listed.

Glossary

A

array

An ordered arrangement of instances in rows and columns with uniform spacing.

assistant menu

Menu that appears at the top of an *assistant pane*. This menu differs from a *banner menu* which appears at the top of a *session window*.

assistant pane

Dockable *workspace* component that can have an *assistant toolbar* and an *assistant menu* bar. You can rearrange and resize assistant panes in your *session window*.

assistant toolbar

Toolbar that appears at the top of an *assistant pane*. This toolbar differs from a *banner toolbar* which appears near the top of a *session window*.

attribute

A mandatory characteristic of a design object or cellview, such as the wire width attribute, that you can change but cannot delete. It is a permanent and predefined part of the database. For example, a wire requires a beginning coordinate and an ending coordinate, and an instance requires an instance name.

In contrast, properties are optional; attributes are not. Properties can be added, changed, or deleted. Attributes can be added or changed, but they cannot be deleted. See also [property](#).

B

backannotation

The process of passing information generated outside the Virtuoso Schematic Editor and storing the information in the schematic editor.

banner menu

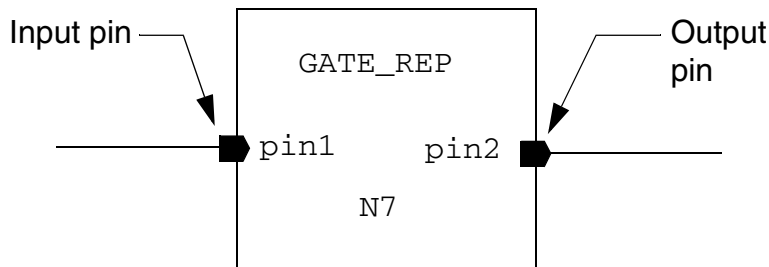
Fixed menu that appears along the top of a *session window*. A banner menu differs from an *assistant menu* which can be found at the top of an *assistant pane* (when docked).

banner toolbar

Toolbar that appears at the top of the *session window* beneath the *banner menu*. A banner toolbar differs from an *assistant toolbar* which can be found at the top of an *assistant pane*.

block

A generic instance that represents lower-level design data.



browser

A window used for displaying the names of libraries, cellviews, views, or components for a form field. Examples of browsers are the Component Browser and the Library Browser. You can use the Component Browser to display multiple libraries and their available components at the same time; to display a hierarchical list of components instead of an alphabetical list; to go directly to a specific component without scrolling.

You can use the Library Browser to view information about the libraries you have specified in your `cds.lib` file.

bundle

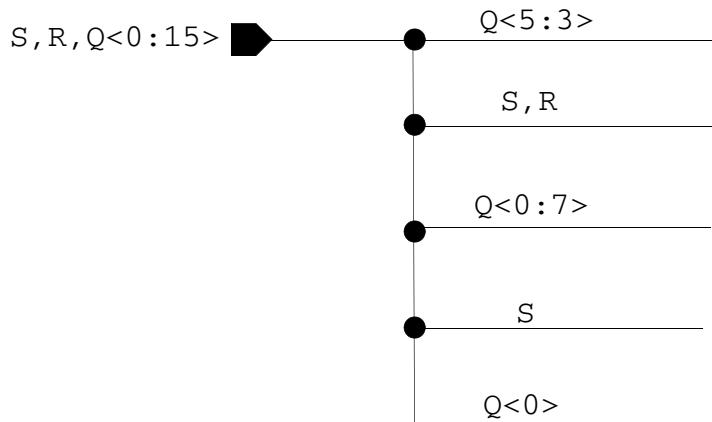
A collection of signals with different names. A bundle is represented by names separated by a comma; for example, `A,B,Data<0:15>`. See also wire.

bus

A collection of signals sharing a common base name. A bus is represented by this base name followed with a vector expression; for example, `A<0:7>`. See also wire.

bus tapping

Tapping bits of a bus or bundle gives the tapping wire segments names that are members of the base bus or bundle. For example, in the following illustration, all names on the right column are taps from the bundle $S, R, Q<0:15>$.



C

cell

A component of a design; a collection of different aspects (representations) of component implementations, such as schematic or symbol representations. A design object consisting of a set of views that can be stored and referenced independently.

An inverter and a buffer are examples of a small cell. A decoder register, arithmetic logic unit (ALU), memories, complete chips, and printed circuit boards are examples of large cells.

cellview

A specific representation (view) of a cell. A particular representation of a particular component, such as a schematic symbol of a NAND gate.

Cellviews are classified by their view type. Each cellview has a view name.

Command Interpreter Window (CIW)

The window that launches any Virtuoso Design Environment application. The CIW logs your design session and reports messages.

connectivity

The way in which components and pins are linked together in an unambiguous manner. The connectivity in a schematic is represented by wires linking the component pins to other component pins. Connectivity can also be represented by pins of the common name (connection by name). For more details, see [Updating Connectivity](#).

cross selection

The process of selecting an object in one view to automatically select the corresponding object in another Cadence application. For example, selecting an instance in a schematic cellview to cross highlight the corresponding instance in the Hierarchy Editor.

D

delay

The time interval between the manifestation of a signal at one point and the manifestation or detection of the same signal at another point.

direct manipulation

Clicking on an object and then dragging the mouse to quickly stretch, move, and copy objects.

E

editor, schematic

The interactive editor in which you can create schematic cellviews and edit objects that represent parts of the design, such as components and pins, and wires that establish connectivity between them.

editor, symbol

The interactive editor in which you can create and edit schematic symbol cellviews using labels, pins, shapes, notes, and a selection box.

F

font

The style and size of type, in the schematic editor, of the label text. The choices are *euroStyle*, *gothic*, *math*, *roman*, *script*, *stick*, *swedish*, and *milSpec*. The math font displays mathematical and Greek symbols. The *swedish* font displays Swedish language characters.

G

gate

Symbol object for combinational logic; for example, AND gates and OR gates. Gates placed in a prefabricated matrix where you provide the design for the interconnect. Gate

array designs are cheaper to manufacture because only the interconnects have to be custom made.

global signal or global net

A signal that is connected by name across all levels of a design hierarchy without using pins. In schematics, a global signal is determined by name; if the signal name ends with an exclamation point (!), it is considered global. A signal that is explicitly passed everywhere in a design is not considered global. A global signal connects to other signals with the same name elsewhere in the hierarchy without requiring an explicit connection through the hierarchy. See also Inherited Connections.

H

hierarchical design

Division of portions of large designs into manageable views that can be created, represented symbolically, edited, and called by or connected to other such views in a design, optimizing gate production through tool speed and distribution of work load across a design organization.

I

ilInst

The `ilInst` SKILL variable provides self-reference to an instance during graphic display of the cellview containing the instance. The typical usage is for IL labels, such as labels that evaluate a SKILL expression and display the results. For example, to display the value of an instance property, you create an IL label with the value `ilInst~>propertyName`. The expression `ilInst~>propertyName` is evaluated for all instances of the cell and the resulting values are displayed in the graphic editor.

The `ilInst` SKILL variable is only evaluated for the instances in the top display level. That is, only the instances in the schematic can have the `ilInst` SKILL variable value defined. If, for example, you are instantiating schematics in schematics, `ilInst` can only be determined for the instances in the top schematics, and all subinstances are evaluated with reference to the subschematic in which they are instantiated.

ilInstPath

The `ilInstPath` SKILL works in a similar manner to the `ilInst` SKILL variable, but is evaluated for the instances at all display levels. You should use `ilInstPath`, rather than `ilInst`, when instantiating schematics in schematics.

ILLabel

IL labels are expressions that are evaluated by the SKILL language to determine the value. When you place an instance in your schematic, the system evaluates the SKILL expression and displays the resulting value. For example, the system displays the IL label `ilInst~>name` in the symbol window, but displays a value—for example, `I3`—in the schematic window. Other types of labels are *normal* and *NLPLabel*.

inherited connections

A facility for specifying global signals that may be overridden for a particular branch of the hierarchy. Net expression labels are used to specify the default signal and the property name which the global signal looks for. The `netSet` properties can be placed on instances to override the default signal for a branch of the hierarchy.

inherited net expression

A use of the NLP expression to support declaring an inherited connection.

instance

An instantiation of a lib cell symbol. It is a database object that represents a master cellview. You can have several instances of the same cellview in a design.

instance iteration

A compact way of representing repeated instances of a symbol in your schematic. Particularly useful in bus-type or data-flow architectures that have identical structures to handle each bit on the bus.

interpreted label

Represents the group of `ILLabel` and `NLPLabel` because their values are always derived.

J

justification

Position of the label's origin with respect to the label. The cursor is always attached to the origin point. The choices are *upperLeft*, *upperCenter*, *upperRight*, *centerLeft*, *centerCenter*, *centerRight*, *lowerLeft*, *lowerCenter*, and *lowerRight*. For example, if you specify *lowerCenter*, the cursor is centered below the label.

M

multisheet

The use of several pages to represent a schematic, connected by pins. You use two levels of hierarchy, the index and sheet levels. This partition improves readability and produces better plot output.

multisheet, index

A schematic representation that is given the instance of all the sheets. This schematic is not intended to be edited manually.

multisheet, pin

Connects nets between sheets in a multisheet schematic. A pin can connect a single signal or multiple signals (buses) of a terminal.

multisheet, sheet

A schematic containing the logical description of a sheet that can be connected to other sheets with offsheet pins.

N

net

A database object that describes one or more signals. Logical implementation of a single signal or a group of signals that implements the connections to instance pins and pins of the cellview containing the net.

net expression

A use of the NLP expression to support declaring an inherited connection.

net expression label

The object for displaying the value of net expressions to the user.

net, global

Used to establish connectivity between nets in different levels of the hierarchy. Indicated by an exclamation mark added to the end of the name; for example, `vdd!`

NLP

Netlist property. NLP expressions, besides being used for netlisting, are also used to display evaluated property values. The syntax of NLP expressions can be used to support inherited net expressions.

NLPLabel

A label type. Other types of labels are `Normal` and `ILLabel`. Labels that are of `NLPLabel` type undergo special NLP evaluation. In general, NLP expressions define a property name to look for, where to look for the property, a format string used to substitute a found property value, and a default value to use if the property is not found. The at sign (`@`) in the net expression label specifies the beginning of a property substitution expression, as `[@instanceName]`. It specifies how NLP label expressions are interpreted.

P

patch cord

A special connection symbol used to establish aliases between the signals of two different nets.

Unlike tap names, where you extract bits of a single net, you can use patch cords to copy one net to another. You can also use the patch cord to map bits from one net to different bits of another net.

pin

Represents the connection point for a single signal or multiple signals of the terminal.

pin, offsheet

Connects nets between sheets in a multisheet schematic. A pin can connect a single signal or multiple signals (buses) of a terminal.

property

A characteristic of a design object or cellview that affects the object and can be edited or deleted. Properties can be mandatory or optional. See also [attribute](#).

S

session window

Current running instance of a particular tool. You can have more than one session window open at a time, each with a *workspace* of its own.

signal

A database object that represents a scalar, logical entity that connects one or more member instance terminals. A scalar net is equivalent to a signal.

Ties together all the nets that implement the signal. Can belong to a number of different nets. See also [wire](#).

solder dot

Small filled-in circle used to identify connections between intersecting wires that are part of the same net.

T**task**

Work objective. For example, a hierarchical task comprising a complex objective such as optimizing device sizing or a simple task such as adding a new pin to a schematic.

technology file

A file that defines all the physical information required for a design. It defines information such as layers, physical design rules for compaction, and devices.

terminal

Logical counterpart of the pin that lets you connect from the next higher level of hierarchy to the net associated with the pin.

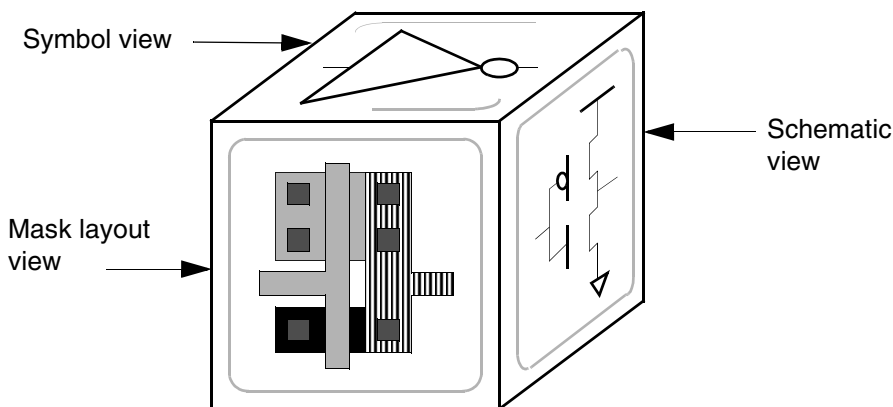
A database object that makes a net available for connection on instances of a given cellview.

V**view, schematic**

Contains connectivity data and graphics that describe the logical views of a design.

view, symbol

Graphic representation of the interface between the current level of hierarchy to the level below the symbol that is instantiated.



W

window element

Any configurable part of a window. For example, a *banner menu* or an *assistant pane*.

wire

A line that connects a pin and an instance to represent a net in a schematic. When you draw a wire between a pin and an instance, you create a net.

wire name

Designates the name of a net and signals represented by the wire.

workspace

Selection of *window elements* comprising a layout of menus, toolbars, and *assistant panes*. You can select and use Cadence workspaces or save and use customized workspaces.