

Virtuoso Electrically Aware Design Flow 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 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

<u>Introduction to Virtuoso Electrically Aware Design Flow</u>	29
<u>Licensing Requirements</u>	30
<u>What is Electromigration?</u>	31
<u>A Snapshot of The Electrically Aware Design Flow</u>	32

2

<u>Running Simulations and Saving Electrical Data</u>	35
<u>Setting up Testbenches and Corners</u>	36
<u>Preparing the EAD Setup for Simulation</u>	37
<u>Enabling the EAD Mode</u>	37
<u>Specifying the Save Options</u>	39
<u>Specifying the Signals to Save Electrical Data</u>	39
<u>Choosing the Type of Current to be Saved</u>	40
<u>Choosing a Method to Save the Current Data</u>	41
<u>Selecting Signals to Save Currents</u>	43
<u>Running Simulations and Viewing Output Current Data</u>	47
<u>Viewing the Current Data</u>	47
<u>Viewing the Current Data for Reliability Analysis Results</u>	52
<u>Creating Datasets</u>	53
<u>Viewing Datasets</u>	58
<u>Filtering Signals in the EAD Results View</u>	62
<u>Working with VDR Datasets</u>	63
<u>Opening the VDR Dataset Form</u>	63
<u>Viewing Simulation Datasets</u>	65
<u>Deleting Simulation Datasets</u>	67
<u>Exporting Simulation Datasets</u>	67
<u>Viewing CSV Datasets</u>	67
<u>Creating a CSV Dataset</u>	69
<u>Deleting CSV Datasets</u>	70
<u>Exporting CSV Datasets</u>	70

3

<u>Running the Pre-Layout Electromigration Check</u>	73
<u>Setting Up Environment Variables for the Pre-EM check</u>	73
<u>Running the Pre-EM Check</u>	75
<u>Pre-EM Report</u>	78
<u>Performing What-If Analysis Based on the Pre-EM Check Results</u>	82
<u>Preparing the Setup for What-If Analysis</u>	82
<u>Editing Terminal Currents in Datasets for What-If Analysis</u>	84
<u>Editing EM Temperature for What-If Analysis</u>	85

4

<u>Extracting Parasitics in the Layout View</u>	87
<u>Starting EAD Browser for a Layout</u>	88
<u>Starting EAD from a Layout Window</u>	88
<u>Starting EAD Browser</u>	88
<u>Starting Layout EAD from a Schematic View</u>	88
<u>Starting EAD from a Layout Window</u>	90
<u>The EAD Workspace</u>	94
<u>EAD Toolbar</u>	95
<u>EAD Browser</u>	98
<u>EAD Browser Toolbar</u>	100
<u>Extraction Corner and EM Dataset Controls</u>	105
<u>Summary Pane</u>	107
<u>Detail Pane</u>	113
<u>Viewing Resistance Paths between Terms</u>	137
<u>Changing Data Display in the EAD Browser</u>	139
<u>Toggling Dual Pane and Single Pane View</u>	139
<u>Sorting Columns</u>	140
<u>Showing and Hiding Columns</u>	141
<u>Filtering Rows</u>	143
<u>Creating Net Groups</u>	144
<u>Managing Net Groups</u>	146
<u>Searching Nets</u>	147
<u>Managing EAD Setups</u>	150

Virtuoso Electrically Aware Design Flow Guide

<u>Preparing an EAD Setup</u>	151
<u>Saving an EAD Setup</u>	152
<u>Saving a Setup</u>	152
<u>Creating a New Setup</u>	153
<u>Referencing a New Setup</u>	154
<u>Loading an EAD Setup</u>	155
<u>Related Topics:</u>	156
<u>Editing Process Settings</u>	157
<u>Using Custom Variables in ICT-EM Files</u>	159
<u>Checking Layer Mapping</u>	159
<u>Deleting an EAD Setup</u>	160
<u>Working with EAD States</u>	161
<u>Saving an EAD State</u>	161
<u>Restoring an EAD State Automatically</u>	162
<u>Loading an EAD State</u>	162
<u>Extracting and Analyzing Parasitics</u>	163
<u>Overview of Parasitic Extraction in Layout EAD</u>	163
<u>Extracting Parasitics on Interconnect Wires Between Devices</u>	164
<u>Extracting Parasitics inside Devices</u>	165
<u>Extracting Parasitics for Devices and Interconnect</u>	166
<u>Extracting Parasitics for Devices and Overlapping Interconnect</u>	166
<u>Extracting Parasitics for Local Interconnect Layers</u>	167
<u>Extracting Parasitics from Strong, Weak, or Must Connections</u>	168
<u>Customizing Extraction of Parasitics</u>	169
<u>Assigning Shapes for Extraction</u>	169
<u>Preventing Extraction Inside Devices</u>	171
<u>Stopping Extraction at the Device Boundary</u>	172
<u>Specifying the Hierarchy Levels and Hierarchy Shape Type</u>	172
<u>Identifying Instances to be Ignored during Extraction</u>	174
<u>Modifying EM Analysis for Wide Pins</u>	176
<u>Extracting Parasitics in Layout EAD</u>	183
<u>High Precision C and R Extraction</u>	186
<u>Customizing High precision R and C Extraction</u>	187
<u>finFET Support in High-Precision C Extraction</u>	192
<u>iQuantus FS C Extraction (Advanced Nodes Layout EAD Only)</u>	192
<u>Viewing and Analyzing Parasitics</u>	194

Virtuoso Electrically Aware Design Flow Guide

<u>Viewing Mesh for Selected Nets</u>	198
<u>Comparing Parasitic Values for Multiple Nets</u>	201
<u>Using Automatic Update Mode</u>	203
<u>Distributed Computation of Extraction Process</u>	205
<u>Automatic submission of jobs for all nets to an LSF farm</u>	205
<u>Automatic submission of jobs for all nets to dedicated servers</u>	206
<u>Customized distributed computation of net groups using different job policies</u>	207
<u>Working with Job Policies</u>	211
<u>Setting Up a Job Policy</u>	211
<u>Using Distributed Processing Jobs</u>	213

5

Performing Electromigration Checks and Fixing Violations 219

<u>Running EM Analysis to Identify Violations</u>	221
<u>Verifying the Setup for EM Analysis</u>	221
<u>Running an EM Check</u>	224
<u>Running EM Checks with High Precision R and C Extraction</u>	228
<u>Viewing Current Density and Vectors</u>	229
<u>Viewing Voltage Drop</u>	233
<u>Identifying EM Hot Spots</u>	235
<u>Highlighting EM Violations</u>	237
<u>Running EM Checks on a CUSTOM Dataset</u>	240
<u>Using Custom Datasets for EM Checks</u>	240
<u>Creating User-Defined Custom Datasets</u>	242
<u>Saving CSV Templates for User-Defined Custom Datasets</u>	251
<u>Reloading and Updating User-Defined Custom Datasets</u>	253
<u>Viewing EM Information in the EAD Browser</u>	255
<u>Viewing Delta T Information</u>	260
<u>Viewing Electrical Information in the Layout Window</u>	267
<u>Using Info Balloons</u>	267
<u>Using Point-To-Point Info Balloons</u>	271
<u>Snapping Start and End Points to Existing Nodes</u>	273
<u>Displaying Info Balloons with Start and End Points from Different Layers</u>	276
<u>Displaying Resistance Paths Between the Start and End Points of an Info Balloon</u>	277

Virtuoso Electrically Aware Design Flow Guide

<u>Updating Info Balloons to Reflect Layout Edits</u>	279
<u>Saving Info Balloons to XML Files</u>	279
<u>Using Measurement Display</u>	280
<u>Highlighting and Fixing Violations in the Layout Window</u>	282
<u>Fixing Violations Using EM Driven Trunk Optimization</u>	286
<u>Important Point to Note</u>	288
<u>Optimizing EM Driven Trunks</u>	288
<u>Expanding Trunks to Fix EM Violations</u>	288
<u>Shrinking Oversized Trunks</u>	306
<u>Shrinking and Expanding Trunks</u>	308
<u>Running EM Analysis on Power Grids</u>	311
<u>Prerequisites</u>	313
<u>Specifying the Source Current</u>	313
<u>Using EAD Menu</u>	313
<u>Using the defaultCurrent Environment Variable</u>	314
<u>Using the CUSTOM Dataset</u>	314
<u>Running EM Analysis on the Scratch Cellview</u>	315
<u>Modifying the Original Layout</u>	315
<u>Running EM Analysis on the Original Grid</u>	316
<u>Specifying Mapping between Nets and Pin Layers for EM Check</u>	317
<u>Example</u>	318
<u>Generating EAD Dataset</u>	321

6

<u>Introduction to EM Calculator</u>	325
<u>Opening the EM Calculator</u>	326
<u>EM Calculator Graphical User Interface (GUI)</u>	327
<u>EM Calculator Toolbar</u>	328
<u>User-Specified Fields</u>	329
<u>Results Table</u>	331
<u>Working with the EM Calculator</u>	334
<u>Calculating EM Limits</u>	335
<u>Running What-If Analysis</u>	337
<u>Exporting EM Results</u>	338
<u>Loading EM Results</u>	339

7

Resimulating Designs with Extracted Parasitics	341
<u>Running Simulations with Layout Parasitics</u>	342
<u>Preparing the Parasitic Setup</u>	342
<u>Creating a Netlist Cellview with Layout Parasitics in Virtuoso Layout EAD</u>	343
<u>Creating a Netlist Cellview with Layout Parasitics in Virtuoso ADE Assembler</u>	344
<u>Creating a DSPF File with Layout Parasitics in Virtuoso Layout EAD</u>	347
<u>Running Simulations and Verifying Specifications</u>	349
<u>Comparing the Simulation Results</u>	349
<u>Using Extracted Parasitics to Run Spectre EM/IR Analysis</u>	351
<u>Using Parasitic Values from Multiple Sources</u>	355
<u>Setting the Source of a Parasitic Estimate to a Layout View</u>	355
<u>Creating Layout Stitching Estimates for a Cell</u>	357
<u>Running Advanced Simulation Runs to Improve Designs</u>	358
<u>Viewing and Updating Layout Parasitics in VSE XL</u>	359
<u>Important Points to Note</u>	362
<u>Viewing Parasitics</u>	365
<u>Probing Resistances for Nets, Terminals, or Instance Terminals Selected on Schematic</u>	367
<u>Probing Capacitance for Nets Selected on Schematic</u>	369
<u>Filtering the Coupling Caps Table by Layer</u>	372
<u>Finding a Specific Resistance Path</u>	373
<u>Extracting Parasitics</u>	373
<u>Viewing and Editing Constraints</u>	373
<u>Viewing and Editing the Max Resistance Constraint in the EAD Browser</u>	374
<u>Viewing and Editing the Max Capacitance Constraint in the EAD Browser</u>	375
<u>Viewing and Editing the Max Coupling Capacitance Constraint in the EAD Browser</u>	377
<u>Annotating Parasitics on the Schematic</u>	379
<u>Modifying Parasitics on the Schematic</u>	380
<u>Building Netlist to Run Simulations</u>	382
<u>Annotating Parasitics from Schematic EAD Browser</u>	384

8

<u>Quantus-Pegasus Integration for Partial Layouts</u>	1
<u>Creating Pegasus/Quantus Setup for Parasitics Extraction</u>	2
<u>Creating Process and Setup Files Manually</u>	3
<u>Extracting Parasitics from a Partial Layout</u>	4
<u>Extracting LDE Parameters</u>	5
<u>Viewing Log Files</u>	6

A

<u>Form Descriptions</u>	9
<u>EAD Setup</u>	10
<u>EAD Options</u>	12
<u>General</u>	13
<u>Extraction</u>	16
<u>EM</u>	22
<u>Environment</u>	25
<u>Comments</u>	27
<u>EAD Setup Options</u>	27
<u>EAD Pegasus/Quantus Create Setup File Dialog Form</u>	28
<u>EAD Process Settings</u>	29
<u>EAD Process Settings – Corners</u>	30
<u>EAD Process Settings – Layer Mapping</u>	35
<u>EAD Process Settings – Vias</u>	36
<u>EAD Process Settings – Cell Shape Types</u>	37
<u>EAD Job Policy Editor</u>	39
<u>EAD Net Options</u>	41
<u>Parasitics & LDE Setup</u>	42
<u>General Tab</u>	42
<u>Layout Tab</u>	43
<u>DSPF Tab</u>	48
<u>P2T EM Optimization</u>	50
<u>The Pre-EM Setup Form</u>	51

B

<u>Environment Variables</u>	53
<u>elec.gui</u>	54
<u>createVoltageConstraints</u>	56
<u>Description</u>	56
<u>GUI Equivalent</u>	56
<u>Examples</u>	56
<u>customVoltageConstraintFunc</u>	57
<u>Description</u>	57
<u>GUI Equivalent</u>	57
<u>Examples</u>	57
<u>customVmaxCalc</u>	58
<u>customVminCalc</u>	59
<u>enableCustomVminVmaxCalc</u>	60
<u>enableEADMode</u>	60
<u>saveAll</u>	62
<u>useHierarchyLevel</u>	63
<u>hierarchyLevel</u>	64
<u>enableIdc</u>	65
<u>idcScaleFactor</u>	66
<u>enableIavg</u>	67
<u>iavgScaleFactor</u>	68
<u>enableIsignal</u>	69
<u>saveWaveforms</u>	70
<u>clipWaveforms</u>	71
<u>clipFrom</u>	71
<u>clipTo</u>	72
<u>skipLibListForDataset</u>	73
<u>Description</u>	73
<u>GUI Equivalent</u>	73
<u>Examples</u>	73
<u>vdrcPerf</u>	74
<u>Description</u>	74
<u>GUI Equivalent</u>	74
<u>Examples</u>	74

Virtuoso Electrically Aware Design Flow Guide

<u>vdrPerfSigCount</u>	75
<u>Description</u>	75
<u>GUI Equivalent</u>	75
<u>Examples</u>	75
<u>vdrCsvDatasetDir</u>	76
<u>Description</u>	76
<u>GUI Equivalent</u>	76
<u>Examples</u>	76
<u>vdrCsvDatasetFileSuffix</u>	77
<u>Description</u>	77
<u>GUI Equivalent</u>	77
<u>Examples</u>	77
<u>vdrCsvDatasetNameSuffix</u>	78
<u>Description</u>	78
<u>GUI Equivalent</u>	78
<u>Examples</u>	78
<u>showOnlyTopLevelNets</u>	79
<u>Description</u>	79
<u>GUI Equivalent</u>	79
<u>Examples</u>	79
<u>warnOnCacheEdit</u>	80
<u>warnOnCacheSave</u>	81
<u>warnOnDatasetOverwrite</u>	82
<u>elec.results</u>	83
<u>dcResultName</u>	83
<u>tranResultName</u>	83
<u>ei.dataSet</u>	84
<u>excludeCurrentsInCI</u>	84
<u>storeRelativePaths</u>	85
<u>transferPerf</u>	85
<u>ei.setup</u>	86
<u>enableWorstCaseDatasetForVoltages</u>	87
<u>Description</u>	87
<u>GUI Equivalent</u>	87
<u>Examples</u>	87
<u>expandIteratedInstancesForCurrents</u>	88

Virtuoso Electrically Aware Design Flow Guide

<u>expandIteratedInstancesForVoltages</u>	89
<u>msps.layout</u>	90
<u>individualInstCdfCallbacks</u>	91
<u>layoutEAD</u>	92
<u>auxSearchPaths</u>	92
<u>checkEM</u>	93
<u>defaultJobPolicy</u>	94
<u>doAutoLayerMapping</u>	94
<u>excludedNets</u>	95
<u>excludedNetsExceptions</u>	95
<u>extractConnectivity</u>	97
<u>extractNetsWithConstraints</u>	98
<u>flattenOtherInstances</u>	99
<u>highlightNetShapes</u>	99
<u>includeEmptyNets</u>	100
<u>loadSubconductorLayers</u>	100
<u>processSettings</u>	101
<u>processStrappedMosfets</u>	102
<u>resistancePathLimit</u>	102
<u>stampNullNetTiles</u>	103
<u>updateMode</u>	104
<u>updateStyle</u>	104
<u>useCorner</u>	105
<u>useSetup</u>	105
<u>xlNetsOnly</u>	106
<u>layoutEAD.calc</u>	107
<u>showEstimatedResistance</u>	107
<u>Description</u>	107
<u>GUI Equivalent</u>	107
<u>Examples</u>	107
<u>layoutEAD.em</u>	108
<u>autoDatasetCustomProc</u>	111
<u>Description</u>	111
<u>GUI Equivalent</u>	111
<u>Examples</u>	111
<u>autoDatasetDeviceLengthPropList</u>	113

Virtuoso Electrically Aware Design Flow Guide

<u>Description</u>	113
<u>GUI Equivalent</u>	113
<u>Examples</u>	113
<u>autoDatasetMacroProp</u>	114
<u>Description</u>	114
<u>GUI Equivalent</u>	114
<u>Examples</u>	114
<u>autoDatasetDeviceWidthPropList</u>	115
<u>Description</u>	115
<u>GUI Equivalent</u>	115
<u>Examples</u>	115
<u>autoDatasetDistributionMethod</u>	116
<u>Description</u>	116
<u>GUI Equivalent</u>	116
<u>Examples</u>	116
<u>adjustedCurrentMsgRelTol</u>	117
<u>customSHESetupFile</u>	118
<u>Description</u>	118
<u>GUI Equivalent</u>	118
<u>Examples</u>	118
<u>currentDensityMPVDefault</u>	119
<u>cdfPercentage</u>	119
<u>computeIRDropWhenLoadingSetup</u>	120
<u>checkResWidthAndEMLength</u>	122
<u>Description</u>	122
<u>GUI Equivalent</u>	122
<u>Examples</u>	122
<u>defaultCurrent</u>	122
<u>deltaT</u>	123
<u>distributeTermCurrent</u>	124
<u>dynamicAnalysis</u>	124
<u>eadVerbosityLevel</u>	125
<u>Description</u>	125
<u>GUI Equivalent</u>	125
<u>Examples</u>	125
<u>emViolThresholdPercent</u>	126

Virtuoso Electrically Aware Design Flow Guide

<u>enableDynamicRead</u>	126
<u>enableSelfHeatingEffects</u>	127
<u>enableViaElseRule</u>	127
<u>forceSegmentTotal</u>	128
<u>icemLayerMapFile</u>	129
<u>includeDeltaT_Margin</u>	129
<u>lifetime</u>	130
<u>ignoreResistorLimit</u>	131
<u>lifetimeUnits</u>	131
<u>metalRegionSearchExtent</u>	132
<u>minCurrent</u>	133
<u>odRegionSearchExtent</u>	133
<u>preloadWaveforms</u>	134
<u>preserveResistorCurrents</u>	135
<u>printAdjustedCurrents</u>	136
<u>resistorLimit</u>	137
<u>resPathsResistorLimit</u>	137
<u>rmsMetalLineNumAC</u>	138
<u>rmsMetalLineNumDC</u>	138
<u>scale</u>	139
<u>scalingFactor</u>	139
<u>scratchLib</u>	140
<u>segmentLength</u>	141
<u>sheSetupFile</u>	142
<u>Description</u>	142
<u>GUI Equivalent</u>	142
<u>Examples</u>	142
<u>setTermNodeAsReference</u>	143
<u>skipViaStackMetalResistor</u>	143
<u>sumThreshold</u>	144
<u>termPinLayersForEMCheck</u>	145
<u>Description</u>	145
<u>GUI Equivalent</u>	145
<u>Examples</u>	145
<u>useDatasetClipInfo</u>	146
<u>useDatasetTemperature</u>	146

Virtuoso Electrically Aware Design Flow Guide

<u>useMaxRule</u>	147
<u>viaOverlapPercentage</u>	147
<u>warnOnEmLayer</u>	148
<u>warnOnEmLimit</u>	149
<u>writeEMResultsToOA</u>	149
<u>preEMCheckLayers</u>	150
<u>preEMDescendIntoPCell</u>	151
<u>preEMDetailReport</u>	151
<u>preEMLogFile</u>	152
<u>preEMViolThresholdPercent</u>	152
<u>minResistorCurrent</u>	153
<u>layoutEAD.pex</u>	154
<u>autoInjectionPointFile</u>	157
<u>Description</u>	157
<u>GUI Equivalent</u>	157
<u>Examples</u>	157
<u>autoInjectionPointDebug</u>	158
<u>Description</u>	158
<u>GUI Equivalent</u>	158
<u>Examples</u>	158
<u>capacitanceMethod</u>	158
<u>checkInterrupt</u>	159
<u>couplingCThreshold</u>	160
<u>coupleToBulkNets</u>	161
<u>doSlotting</u>	162
<u>dspfCapUnits</u>	162
<u>dspfCouplingCapPrefix</u>	163
<u>dspfFileName</u>	163
<u>dspfGroundCapPrefix</u>	164
<u>dspfResistorPrefix</u>	164
<u>dspfResUnits</u>	165
<u>dspfSubNodeDelimiter</u>	165
<u>dspfWriteInstances</u>	166
<u>dspfWriteSubNodes</u>	166
<u>enableErosionEffects</u>	167
<u>enableLoadingEffects</u>	168

Virtuoso Electrically Aware Design Flow Guide

<u>enableWbeEffects</u>	168
<u>enableWeeEffects</u>	169
<u>excludePolyDiffusionCap</u>	170
<u>extractDegenerateNets</u>	171
<u>extractLocalInterconnectVias</u>	171
<u>extractNetsWithOpens</u>	172
<u>extractNonOrthoNets</u>	172
<u>extractResOnDiffusion</u>	173
<u>extractToPins</u>	173
<u>extractionPrimitives</u>	174
<u>enableCapOnlyModels</u>	174
<u>fillPurposeNames</u>	176
<u>fillType</u>	176
<u>forceExtraction</u>	177
<u>groundNetName</u>	177
<u>hideDummyResistors</u>	178
<u>Description</u>	178
<u>GUI Equivalent</u>	178
<u>Examples</u>	178
<u>hierarchyLevels</u>	179
<u>hierarchyShapeType</u>	180
<u>insertResNodeLabelPurpose</u>	181
<u>insertResNodes</u>	181
<u>instTermHierarchyLevels</u>	183
<u>instTermSuperconductive</u>	184
<u>maxResistorLength</u>	184
<u>netHierarchyLevels</u>	185
<u>nonTermDiffusionShapeType</u>	186
<u>promotePorts</u>	186
<u>recordCoupledCap</u>	187
<u>relativeCThreshold</u>	188
<u>resScaleFactor</u>	188
<u>Description</u>	188
<u>GUI Equivalent</u>	188
<u>Examples</u>	188
<u>gndCapScaleFactor</u>	189

Virtuoso Electrically Aware Design Flow Guide

<u>Description</u>	189
<u>GUI Equivalent</u>	189
<u>Examples</u>	189
<u>couplingCapScaleFactor</u>	189
<u>Description</u>	189
<u>GUI Equivalent</u>	189
<u>Examples</u>	189
<u>slottingPurposeNames</u>	190
<u>stackViaArrayHandling</u>	191
<u>skipPowerGround</u>	191
<u>supportSDRDynamicDataset</u>	192
<u>Description</u>	192
<u>GUI Equivalent</u>	192
<u>Examples</u>	192
<u>swapLithoBiasDirection</u>	192
<u>temperature</u>	193
<u>threads</u>	194
<u>type</u>	195
<u>unusedInstTermShapeType</u>	196
<u>viaInstanceMode</u>	197
<u>viaShapesScanMode</u>	198
<u>warnOnLayerMapping</u>	199
<u>layoutEAD.solver</u>	200
<u>autoThreadsPerNetOverride</u>	201
<u>convergenceCThreshold</u>	201
<u>finWidth</u>	202
<u>finSpacing</u>	202
<u>hpcMetalFillDefFileName</u>	203
<u>hprDisableViaClustering</u>	204
<u>hprStampDataFileName</u>	205
<u>includePolyFringingCap</u>	206
<u>modelNonGateFormingPolyCap</u>	206
<u>modelViaCapacitanceEffect</u>	207
<u>resistanceMeshDensity</u>	208
<u>resistanceMeshDensityFileName</u>	208
<u>resistanceMeshFileName</u>	210

Virtuoso Electrically Aware Design Flow Guide

<u>resistanceMesherThreadCount</u>	211
<u>resistanceMesherTimeOut</u>	212
<u>resistancePathThreshold</u>	213
<u>resistanceFieldFileName</u>	214
<u>showVoltageDropViolation</u>	215
<u>tileExtractor</u>	216
<u>tileExtractorThreadCount</u>	216
<u>useNumberOfGPU</u>	217
<u>qrcTechFileName</u>	219
<u>qrcLayerSetupFileName</u>	220
<u>qrcCapgenFileName</u>	221
<u>hpcEngine</u>	222
<u>layoutEAD.gui</u>	223
<u>autoHideParasiticHighlights</u>	225
<u>applyLayerFilterOnCanvasHighlight</u>	226
<u>capacitanceFixedSuffix</u>	226
<u>clickClickUpdateDelay</u>	227
<u>clickClickUseNetCache</u>	228
<u>detailedParasiticTables</u>	228
<u>dualPaneMode</u>	229
<u>emResultsShowAllResistors</u>	230
<u>emTableResultType</u>	230
<u>enableFindNet</u>	231
<u>fitToSelected</u>	232
<u>hideCheckLayerMapping</u>	233
<u>Description</u>	233
<u>GUI Equivalent</u>	233
<u>Examples</u>	233
<u>hideHPRCExtraction</u>	234
<u>Description</u>	234
<u>GUI Equivalent</u>	234
<u>Examples</u>	234
<u>fitToSelectedZoomScale</u>	234
<u>highlightSelectedNets</u>	235
<u>layerFilterOnEMCanvas</u>	235
<u>loadTablesOnDemand</u>	237

Virtuoso Electrically Aware Design Flow Guide

<u>Description</u>	237
<u>GUI Equivalent</u>	237
<u>Examples</u>	237
<u>includeSkippedNets</u>	237
<u>maxTooltipLength</u>	238
<u>netGroupsPath</u>	239
<u>parasiticHighlightColor</u>	239
<u>P2PSnapping</u>	240
<u>resistanceDisplay</u>	240
<u>displayP2PResistancePath</u>	241
<u>resistancePathsTableTimeout</u>	242
<u>saveStateOnClose</u>	243
<u>saveUserSpecificSetups</u>	243
<u>selectNetsInLayout</u>	244
<u>selectedParasiticHighlightColor</u>	244
<u>significantDigits</u>	245
<u>statePath</u>	245
<u>strikeOutDeletedDatasets</u>	246
<u>updateP2POnDrag</u>	247
<u>violationLevels</u>	248
<u>violationPacketStipple</u>	248
<u>violationTransparency</u>	249
<u>warnOnTemperatureMismatch</u>	249
<u>warnOnTemperatureOutOfRange</u>	250
<u>warnWhenEMDataSourcesInvalid</u>	251
<u>warnWhenNoParasiticsForDSPF</u>	251
<u>loadStateOnOpen</u>	252
<u>enableEMCalculator</u>	253
<u>Description</u>	253
<u>GUI Equivalent</u>	253
<u>Examples</u>	253
<u>showEMPlannerWindow</u>	254
<u>Description</u>	254
<u>GUI Equivalent</u>	254
<u>Examples</u>	254
<u>showRecommendedTracks</u>	255

Virtuoso Electrically Aware Design Flow Guide

<u>Description</u>	255
<u>GUI Equivalent</u>	255
<u>Examples</u>	255
<u>showTracksWithoutWSP</u>	256
<u>Description</u>	256
<u>GUI Equivalent</u>	256
<u>Examples</u>	256
<u>useWSPGroupNames</u>	257
<u>Description</u>	257
<u>GUI Equivalent</u>	257
<u>Examples</u>	257
p2t	258
<u>EMStackTwigs</u>	258
<u>EMFillStackedTrunksWithVias</u>	259
<u>EMAllowDRCViolations</u>	259
<u>EMOptimizationMode</u>	260
<u>EMTrunkStrandMode</u>	260
<u>EMTrunkTapering</u>	261

C

<u>Electrically Aware Design Functions</u>	263
<u>List of EAD-related OCEAN Commands</u>	265
<u>elecCreateDataSetCSVTemplate</u>	266
<u>Description</u>	266
<u>Arguments</u>	266
<u>Value Returned</u>	267
<u>Examples</u>	267
<u>elecDeleteDataSets</u>	268
<u>Description</u>	268
<u>Arguments</u>	268
<u>Value Returned</u>	268
<u>Example</u>	268
<u>eadDeleteScratchLib</u>	270
<u>Description</u>	270
<u>Arguments</u>	270

Virtuoso Electrically Aware Design Flow Guide

<u>Value Returned</u>	270
<u>Example</u>	270
<u>elecGetEnableEADMode</u>	271
<u>Description</u>	271
<u>Arguments</u>	271
<u>Value Returned</u>	271
<u>Example</u>	271
<u>elecSetEnableEADMode</u>	272
<u>Description</u>	272
<u>Arguments</u>	272
<u>Value Returned</u>	272
<u>Example</u>	272
<u>elecGetCurrentData</u>	274
<u>Description</u>	274
<u>Arguments</u>	274
<u>Value Returned</u>	274
<u>Example</u>	274
<u>elecGetDataSetNames</u>	276
<u>Description</u>	276
<u>Arguments</u>	276
<u>Value Returned</u>	276
<u>Example</u>	276
<u>elecGetDataSetParamsPropValue</u>	277
<u>Description</u>	277
<u>Arguments</u>	278
<u>Value Returned</u>	278
<u>Example</u>	278
<u>elecMakeDefaultDatasetName</u>	279
<u>Description</u>	279
<u>Arguments</u>	279
<u>Value Returned</u>	279
<u>Examples</u>	280
<u>elecUpdateDataSetParamsPropValue</u>	282
<u>Description</u>	282
<u>Arguments</u>	282
<u>Value Returned</u>	282

Virtuoso Electrically Aware Design Flow Guide

<u>Example</u>	283
<u>elecGetVoltageData</u>	284
<u>Description</u>	284
<u>Arguments</u>	284
<u>Value Returned</u>	285
<u>Examples</u>	285
<u>elecGetVsyncEnabled</u>	286
<u>Description</u>	286
<u>Arguments</u>	286
<u>Value Returned</u>	286
<u>Example</u>	286
<u>elecSetVsyncEnabled</u>	287
<u>Description</u>	287
<u>Arguments</u>	287
<u>Value Returned</u>	287
<u>Example</u>	287
<u>elecVdrCreateConstraints</u>	288
<u>Description</u>	288
<u>Arguments</u>	288
<u>Value Returned</u>	289
<u>Example</u>	289
<u>elecVdrCreateDeltaV</u>	291
<u>Description</u>	291
<u>Arguments</u>	291
<u>Value Returned</u>	292
<u>Example</u>	292
<u>eadAddNetsToNetGroup</u>	294
<u>Description</u>	294
<u>Arguments</u>	294
<u>Value Returned</u>	295
<u>Example</u>	295
<u>eadCreateDataSetFromLayCSV</u>	296
<u>Description</u>	296
<u>Arguments</u>	296
<u>Value Returned</u>	296
<u>Examples</u>	296

Virtuoso Electrically Aware Design Flow Guide

<u>eadCreateDataSetFromSchCSV</u>	298
<u>Description</u>	298
<u>Arguments</u>	298
<u>Value Returned</u>	298
<u>Examples</u>	298
<u>eadCreateDataSetLayCSVTemplate</u>	299
<u>Description</u>	299
<u>Arguments</u>	299
<u>Value Returned</u>	299
<u>Examples</u>	299
<u>eadCreateNetGroup</u>	300
<u>Description</u>	300
<u>Arguments</u>	300
<u>Value Returned</u>	301
<u>Example</u>	301
<u>eadDeleteNetGroup</u>	302
<u>Description</u>	302
<u>Arguments</u>	302
<u>Value Returned</u>	302
<u>Example</u>	302
<u>eadExtractParasiticsForDesign</u>	303
<u>Description</u>	303
<u>Arguments</u>	303
<u>Value Returned</u>	303
<u>Example</u>	303
<u>eadGetAllEMDatasets</u>	304
<u>Description</u>	304
<u>Arguments</u>	304
<u>Value Returned</u>	304
<u>Examples</u>	304
<u>eadGetNetsInNetGroup</u>	305
<u>Description</u>	305
<u>Arguments</u>	305
<u>Value Returned</u>	305
<u>Example</u>	305
<u>eadGetTriggerDesc</u>	306

Virtuoso Electrically Aware Design Flow Guide

<u>Description</u>	306
<u>Arguments</u>	306
<u>Value Returned</u>	306
<u>Example</u>	306
<u>eadGetLocCurrentLimit</u>	307
<u>Description</u>	307
<u>Arguments</u>	308
<u>Value Returned</u>	308
<u>Example</u>	308
<u>eadJobClose</u>	309
<u>Description</u>	309
<u>Arguments</u>	309
<u>Value Returned</u>	309
<u>Example</u>	309
<u>eadJobStatus</u>	311
<u>Description</u>	311
<u>Arguments</u>	311
<u>Value Returned</u>	311
<u>Example</u>	311
<u>eadJobSubmit</u>	313
<u>Description</u>	313
<u>Arguments</u>	313
<u>Value Returned</u>	313
<u>Example</u>	313
<u>eadLoadSetup</u>	314
<u>Description</u>	314
<u>Arguments</u>	314
<u>Value Returned</u>	314
<u>Example</u>	314
<u>eadPreEMSetUserStopCVList</u>	315
<u>Description</u>	315
<u>Arguments</u>	315
<u>Value Returned</u>	315
<u>Examples</u>	315
<u>eadRegTrigger</u>	316
<u>Description</u>	316

Virtuoso Electrically Aware Design Flow Guide

<u>Arguments</u>	316
<u>Value Returned</u>	317
<u>Examples</u>	317
<u>eadRegisterCustomEMVariable</u>	318
<u>Description</u>	318
<u>Arguments</u>	318
<u>Value Returned</u>	318
<u>Example</u>	318
<u>eadRemoveNetsFromNetGroup</u>	319
<u>Description</u>	319
<u>Arguments</u>	319
<u>Value Returned</u>	320
<u>Example</u>	320
<u>eadRemoveParasitics</u>	321
<u>Description</u>	321
<u>Arguments</u>	321
<u>Value Returned</u>	321
<u>Example</u>	321
<u>eadSetEMVariable</u>	322
<u>Description</u>	322
<u>Arguments</u>	322
<u>Value Returned</u>	322
<u>Example</u>	322
<u>elecTransferDataSets</u>	323
<u>Description</u>	323
<u>Arguments</u>	323
<u>Value Returned</u>	324
<u>Example</u>	324
<u>eadUnregTrigger</u>	326
<u>Description</u>	326
<u>Arguments</u>	326
<u>Value Returned</u>	326
<u>Example</u>	326
<u>eadInitializeDesign</u>	327
<u>Description</u>	327
<u>Arguments</u>	327

Virtuoso Electrically Aware Design Flow Guide

<u>Value Returned</u>	327
<u>Example</u>	327
<u>eadExtractParasitics</u>	328
<u>Description</u>	328
<u>Arguments</u>	328
<u>Value Returned</u>	328
<u>Example</u>	328
<u>eadRemoveDesignParasitics</u>	330
<u>Description</u>	330
<u>Arguments</u>	330
<u>Value Returned</u>	330
<u>Example</u>	330
<u>eadRunEM</u>	331
<u>Description</u>	331
<u>Arguments</u>	331
<u>Value Returned</u>	331
<u>Examples</u>	331
<u>eadSaveSummaryTableToCSV</u>	333
<u>Description</u>	333
<u>Arguments</u>	333
<u>Value Returned</u>	333
<u>Examples</u>	333
<u>eadSetCurrentDataset</u>	335
<u>Description</u>	335

D

<u>Generating an EAD Technology File</u>	337
<u>Prerequisites</u>	338
<u>Generating an EAD Technology File</u>	338
<u>Example Scenarios of EAD Technology File Generation</u>	344
<u>Updating an EAD Technology File</u>	347
<u>EM Rule Support in EAD</u>	348
<u>Current Density (JMAX) Keywords</u>	351
<u>Rules for Defining EM Parameters</u>	352
<u>Rules for Specifying Via Area</u>	359

Virtuoso Electrically Aware Design Flow Guide

<u>EM Rule Selection Priority</u>	360
<u>Solvers for Generation of Capacitance Models</u>	364
<u>Guidelines for Using a Solver</u>	365

Virtuoso Electrically Aware Design Flow Guide

Introduction to Virtuoso Electrically Aware Design Flow

The reducing sizes of advanced integrated circuits means that wire widths are becoming smaller and the lengths of connecting wires longer. In addition, there is an exponential increase in the density of the current flowing through the wires. As a result, integrated circuits become more vulnerable to electromigration (EM), which is the gradual dislocation of metal atoms in a conductor.

To produce a reliable and sustainable integrated circuit, it is therefore essential that you perform comprehensive electromigration checks and ensure that the physical design of every design component is electrically correct by construction and optimized to meet the design intent.

The Virtuoso Electrically Aware Design (EAD) flow lets you capture the current data from design simulations, extract and visualize RC parasitics as you edit the layout, perform EM checks and fix violations. You can further extract parasitics from a partial or a complete layout and rerun simulations to check if the output specifications are met. This flow guide describes the Virtuoso EAD flow in detail and explains how to use the various tools to perform EM analysis for your design.

This chapter describes the following:

- [What is Electromigration?](#)
- [A Snapshot of The Electrically Aware Design Flow](#)

Licensing Requirements

The EAD flow uses the following licenses:

License Number	Product Name
95600	Virtuoso Layout Suite EAD If the license 95800 is already checked out, 95600 is not used.
95800	Virtuoso Layout Suite EXL
95810	Virtuoso Layout Suite MXL
95511	Virtuoso Advanced Node Option for Layout
95512	Virtuoso Advanced Node Option for Layout Standard To use Width Spacing Patterns (WSPs), you require either 95511 or 95512.
95513	Virtuoso Advanced Node Option for GAA
95260	Virtuoso ADE Assembler To run simulation and save electrical data using Virtuoso ADE Assembler
95510	Virtuoso Implementation Aware Design Option Used for resimulation of design using the extracted parasitics. (Optional, if 95600 is not available) If 95600 is already checked out, 95510 cannot be used.

What is Electromigration?

Electromigration (EM) is the term used to describe the movement of atoms in a solid conductor resulting from collisions between the flow of electrons and metal atoms in connecting wires.

This movement of atoms is caused by high current stress in metal wires generated by the movement of electrons. As the electrons move through a metal wire, they collide with the atoms in the wire, causing the wires to become heated. If enough electrons collide with a metal atom over a period of time, the metal atom can move in the direction of the electron flow, potentially causing

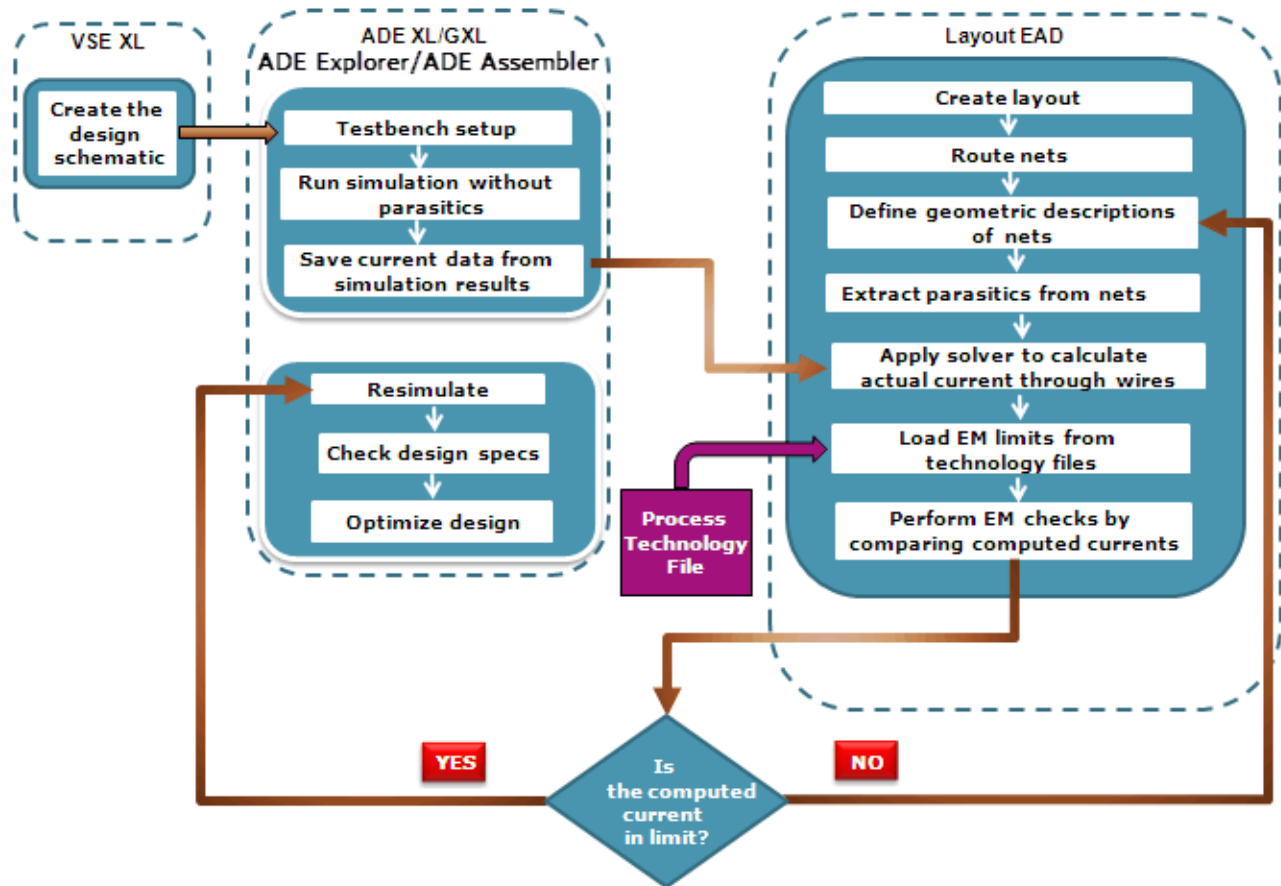
- An open circuit if enough atoms move and cause the wire to break
- A short to an adjacent metal wire if enough atoms move to the same location

EM can damage the interconnect wires and vias, thereby causing degradation in the performance of an integrated circuit. Therefore, it is important to observe the electrical impact of the physical design of every component.

The EAD flow in Virtuoso involves various steps to perform comprehensive EM checks for your design and make it electrically correct. The following section provides a brief description of this flow.

A Snapshot of The Electrically Aware Design Flow

The following diagram provides a snapshot of the EAD flow.



You begin by creating a schematic, running simulations, and modifying your design so that it meets the desired specifications. When the specifications are met, you then configure the testbench to save the current data (average, RMS, and peak) that will be used for EM checking in Layout EAD.

EM depends on various factors, such as the local current density (current per unit area), the homogeneity of the current flow through a region, and the EM length. All these factors are captured for different conductor layers and vias as rules in technology files for each process technology specification.

Before performing the EM checking, launch the EAD Browser in either Layout EXL or Layout MXL. Next, configure the EAD setup to load the required process settings from technology files. These settings are used during parasitic extraction and EM checking.

Virtuoso Electrically Aware Design Flow Guide

Introduction to Virtuoso Electrically Aware Design Flow

As you create or modify the layout of your design, an extraction engine in Layout EAD lets you extract and visualize resistance and capacitance parasitics on nets. An integrated high precision solver also lets you perform critical net analysis by extracting capacitance and resistance on critical nets with a high degree of accuracy.

You can now run EM checks to compare the current data with the standard EM limits specified in the technology files. The EAD Browser in Layout EAD shows detailed reports to indicate the pass or fail status of the EM checks for each net in the layout.

Note: You can also configure layout to dynamically extract parasitics and run EM checks as you are creating or modifying the layout.

You can then fix the reported EM violations by modifying the placement and routing of components in the layout or by changing the geometric properties of the nets, and then rerun the EM checks to verify the results.

Note: One of the main benefits of EAD is that it can be used to analyze parasitic information for both partially and fully completed layouts. Not all devices and nets need to be routed in order to benefit from EAD and you can place and route devices while the application is running.

At any point in the flow, you can take the parasitics from your partially complete or completed layout and resimulate your design in ADE Assembler to check if the output specifications are still met. You can compare the results of an ideal simulation with those of a parasitic resimulation and, based on those results, further optimize your design and verify the performance and reliability of every physical design component to ensure that the layout is electrically correct by construction.

The following chapters in this flow guide describe the Virtuoso EAD flow in detail and explain how to use the various Virtuoso tools to perform EM analysis and parasitic resimulation for your design.

Related Videos

- [Electrically Aware Design Flow in Virtuoso](#)
- [Advanced Node Electromigration](#)

Related Blog

[Virtuosity: In-design Electromigration Analysis - An efficient way to make layouts electrically correct](#)

Virtuoso Electrically Aware Design Flow Guide

Introduction to Virtuoso Electrically Aware Design Flow

Running Simulations and Saving Electrical Data

The first step towards electromigration checking is to run simulations for your design and to save the electrical data. This chapter describes how to use the EAD menu in ADE Assembler to save the electrical data in the simulation results. It further describes how to explore the electrical data in the EAD view on the Results tab.

This chapter includes the following topics:

- [Setting up Testbenches and Corners](#)
- [Preparing the EAD Setup for Simulation](#)
- [Running Simulations and Viewing Output Current Data](#)

For details on the other features in ADE Assembler, see the [*Virtuoso ADE Assembler User Guide*](#).

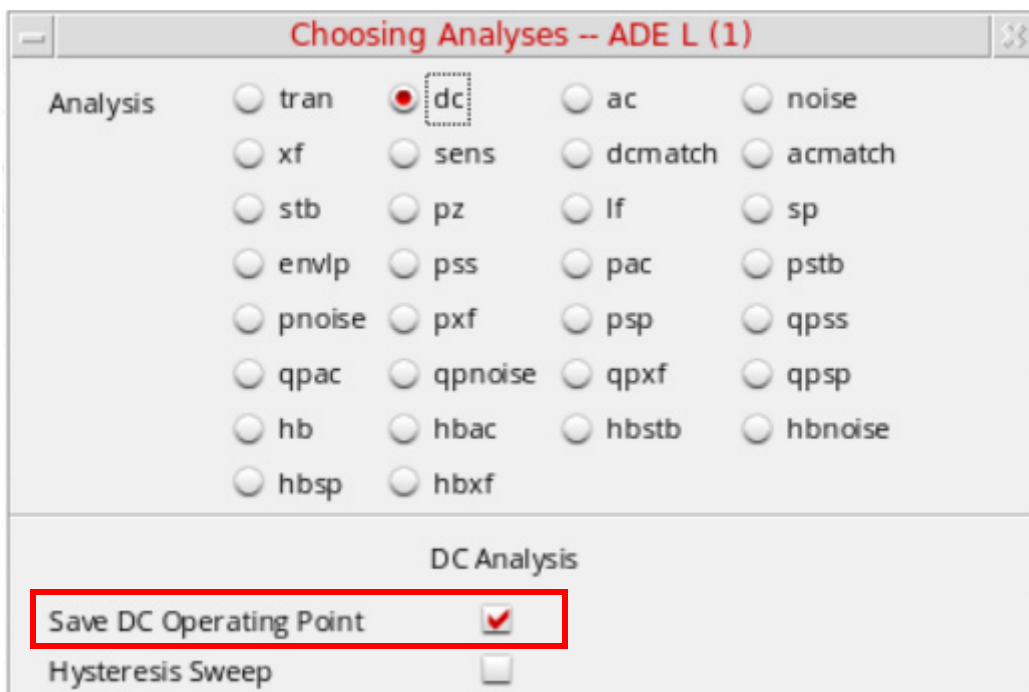
Setting up Testbenches and Corners

Open your design in ADE Assembler, and set up the tests and corners for simulation. For details on how to set up the simulation environment, refer to the following topics:

- [Specifying Tests and Analyses](#)
- [Working with Design Variables and Instance Parameters](#)
- [Adding Corners](#)

There is no specific setting for the electromigration analysis in the test and corner setup. However, it would help to note the following points:

- Ensure that you set up DC operating point analysis to save the direct current (Idc) values and transient analysis to save the average current (Iavg) values from the simulation results.
- For DC analysis, enable the *Save DC Operating Point* check box to save the simulation data. Otherwise, the [dataset](#) would remain blank.



- It would also be helpful to run simulations at the temperature at which you need to perform the EM analysis.

Note: If there are multiple tests, ensure that they all include a common design.

Preparing the EAD Setup for Simulation

Electromigration analysis is done using the currents loaded to the terminals of a net. Therefore, before running a simulation, you need to specify the terminals for which you need to save the current data. You can also specify if you want to save the complete data or process the data and save only selected values.

The following sections explain how to use the EAD Setup form to specify the EAD-related settings before running simulations:

- Enabling the EAD Mode
- Specifying the Save Options
- Selecting Signals to Save Currents

Enabling the EAD Mode

To enable the EAD mode, do the following:

1. Choose *EAD — Setup*.

Note: The shortcut for the EAD menu is `Alt+E`.

The **EAD Setup** form is displayed.

2. Select the *Enable Electrical Data Capture for EAD Flow* check box.

This ensures that ADE Assembler saves the electrical data in every simulation run as per the settings given in the other group boxes of this form.

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

Note that when you select this check box, all other sections on this form are activated, as shown below.

The screenshot shows the 'EAD Setup' dialog box with the following sections and options:

- Enable Electrical Data Capture for EAD Flow** (checked)
- Pre EM Check** (checked)
- Design Selection**
 - Design Under Test: IR_drop (dropdown), CurrentMirror (dropdown)
- Save Options**
 - Signal Selection**
 - All Signals** (selected radio button)
 - Hierarchy Stop Level** 99 (dropdown, checked)
 - Selected Signals (Using the EAD -> Signal Selection menu option) (radio button)
 - Electromigration Checking**
 - Type: DC Current (Idc) (checked), Average Current (Iavg) (checked), Waveforms for RMS and Peak Checks (Isignal) (unchecked)
 - Scale: 1.00 (dropdown)
 - Voltage Dependent Rules**
 - Type: Min and Max Voltage (Vmin/Vmax) (unchecked)
 - Scale: 1.00 (dropdown)
- Waveform Processing Options (For Iavg, Vmin and Vmax)**
 - Process Waveforms Inside Simulator (MMSIM only) (radio button)
 - Process Waveform Post Simulation** (selected radio button)
 - Clip Waveforms From: (text box) To: (text box) (unchecked)

Annotations on the right side of the dialog box:

- Signals for which data is to be saved (points to the Signal Selection section)
- Options to save current data (points to the Electromigration Checking section)
- Options to save voltage data (points to the Voltage Dependent Rules section)
- Waveforms processing options (points to the Waveform Processing Options section)

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

3. Select the *Pre EM Check* check box if you want to enable the Pre-EM check mode.

4. Ensure that the design for which you need to perform electromigration analysis is displayed in the *Design Selection* section.

If required, you can select a different library or cell name for which you need to enable the EAD mode.

Important Points to Note

- ❑ For the selected design cell, the tool uses the first available schematic in the view name list specified by the `viewNameList` environment variable set for Virtuoso Schematic Editor.
- ❑ You can set default values for the fields in the EAD Setup form using the corresponding environment variables. For the list of variable names, see [Appendix B](#).

Specifying the Save Options

You can specify the following save options:

- Signals for which the electrical data is to be saved
- Type of current to be saved

Specifying the Signals to Save Electrical Data

The *Save Options* section of the EAD Setup form specifies the signals for which you need to save the electrical data from the simulation results.

You can select any one of the following two options:

- **All Signals:** This is the default option which specifies that the current data is to be saved for all the signals and instances in the selected design. Along with this, the *Hierarchy Stop Level* option is also selected and is by default set to 99. This option specifies the design hierarchy level upto which the current data for all the instances and pins is saved.

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

By default, the data for a maximum of 99 hierarchy levels is saved, but you can change this value as required. A value of 0 saves the terminal currents for instances and pins at the design level. Similarly, a value of 1 saves the terminal currents for instances and pins at the design level and one level down in the hierarchy.

Note: In case of iterated instances, the current data is saved for only one instance. This data is internally replicated across instances while transferring from schematic to layout, that is, when you update the electrical data in Virtuoso Layout XL.

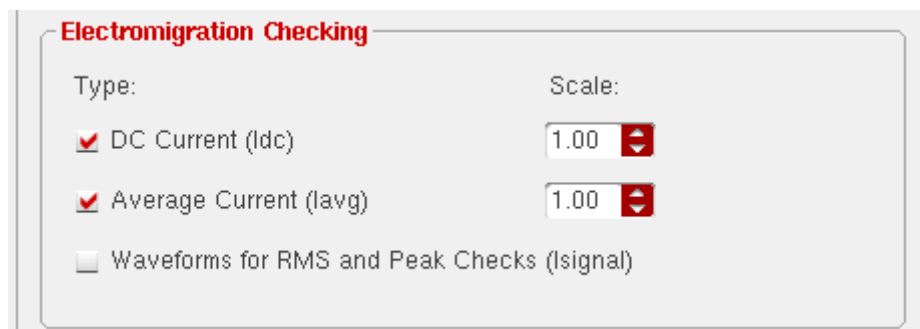
Related environment variables:

- ☐ [expandIteratedInstancesForCurrents](#)
- ☐ [expandIteratedInstancesForVoltages](#)
- **Selected Signals:** This option specifies that the current data is to be saved for a selected set of signals. This option helps in saving disk space and simulation time.

Note: When you choose the *Selected Signals* option, you also need to select the signals using the Parasitics and Electrical Setup assistant. For more details, refer to [Selecting Signals to Save Currents](#).

Choosing the Type of Current to be Saved

For the signals specified in the previous section, you can choose to save the DC current, average current, or waveforms.



Electromigration Checking

Type:	Scale:
☒ DC Current (<i>Idc</i>)	1.00
☒ Average Current (<i>Iavg</i>)	1.00
☐ Waveforms for RMS and Peak Checks (<i>Isignal</i>)	

You can select the following three options to specify the type of electrical data to be saved:

- **DC current (*Idc*):** Select this check box to save the DC current. Also specify a scale value, which is the multiplier by which the electrical data is to be scaled before transferring it to layout.
- **Average current (*Iavg*):** Select this check box to save the average current computed for the transient waveform data and specify a scale value for it.

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

According to the specified current type, ADE Assembler calculates the DC or average current for terminals and saves one value per terminal for each simulation.

- **Waveforms for RMS and Peak Checks (Isignal):** Select this check box to save the transient waveform data for the selected signals so that it can be used for dynamic EM analysis. You can select this check box in the following two cases:
 - ❑ When you need to save only the waveform data for dynamic analysis and there is no requirement to save the scalar results (DC or average current values). In this case, you can choose to select only the waveforms and not the scalar data.
 - ❑ When you need to process the signal values during a simulation to compute scalar results and you also need to save waveforms for dynamic analysis in the layout view. That is, when you select the *Process Waveforms Inside Simulator* check box to process the scalar data, you can also select *Waveforms for RMS and Peak Checks* to save waveforms.

Important Points to Note

- For DC current to be saved, DC analysis should be enabled in the testbench setup.
- For average current to be saved, transient analysis should be enabled in the testbench setup.
- The scale factor is used to scale the current data only while transferring the scalar currents from the schematic to the layout view. It is also possible to specify a scaling factor on the EM tab of the EAD Options form. The default scale factor for both DC current and average current is set to 1.00 and the data is not scaled.

Choosing a Method to Save the Current Data

Data requirement for static and dynamic EM analysis are different. Therefore, depending on the volume of current data and the type of electromigration checks to be run, you need to choose a method to save the data.

The *Waveform Processing Options* group box on the EAD Setup form provides options to specify the method to be used.

Waveform Processing Options (For Iavg, Vmin and Vmax)

☒ Process Waveforms Inside Simulator (MMSIM only)

☐ Process Waveform Post Simulation

☒ Clip Waveforms From: 10n To: 30n

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

You can choose from the following options:

- *Process Waveforms Inside Simulator*: Select this option if you choose to process the current waveforms during simulation and save only the calculated electrical data to the disk. This data is used when performing static electromigration checks.

Note: This option is available only for Spectre and APS simulators.

- *Process Waveforms Post Simulation*: Select this option if you choose to process the current waveforms by reading the simulation data from disk. The saved data can be processed later while performing dynamic electromigration checks, for example, Peak EM or RMS. It is important to consider that if you have selected a large number of devices and terminals for EM checking, a large amount of disk space is required for this option.
- *Clip Waveforms*: If you know the time interval for which the electrical data is relevant, select the *Clip Waveforms* check box and specify the time interval in the *From* and *To* fields. You can also use VAR expressions in the *From* and *To* fields. For more information, refer [Virtuoso ADE Assembler User Guide](#).

Note: If you have selected *Process Waveforms Post Simulation*, clipping waveforms helps in saving disk space as it saves the waveforms only for the selected time period.

Important

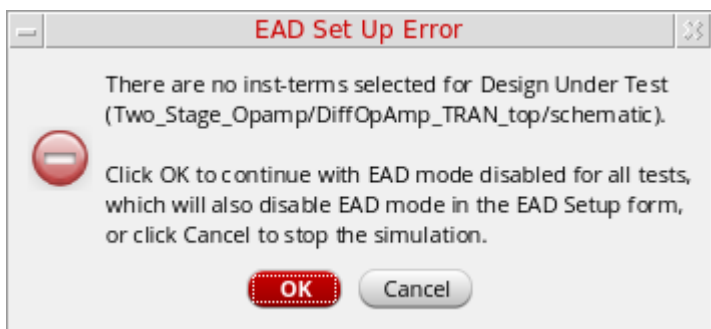
For dynamic peak and RMS electromigration checking, full waveforms should be saved to disk.

Selecting Signals to Save Currents

If you have chosen to save the current data for selected devices and terminals in the design (that is, you have chosen the *Selected Signals* options on the [EAD Setup](#) form), select the required nets and terminals, and create electrical selection setups using the **Parasitics & Electrical Setup** assistant in Schematic Editor XL.

Important

If you do not select any signal and run a simulation to save the electrical data, ADE Assembler displays the following message and prompts you to either continue the simulation with EAD mode disabled or run the simulation again after selecting instance terminals on the schematic of the DUT.

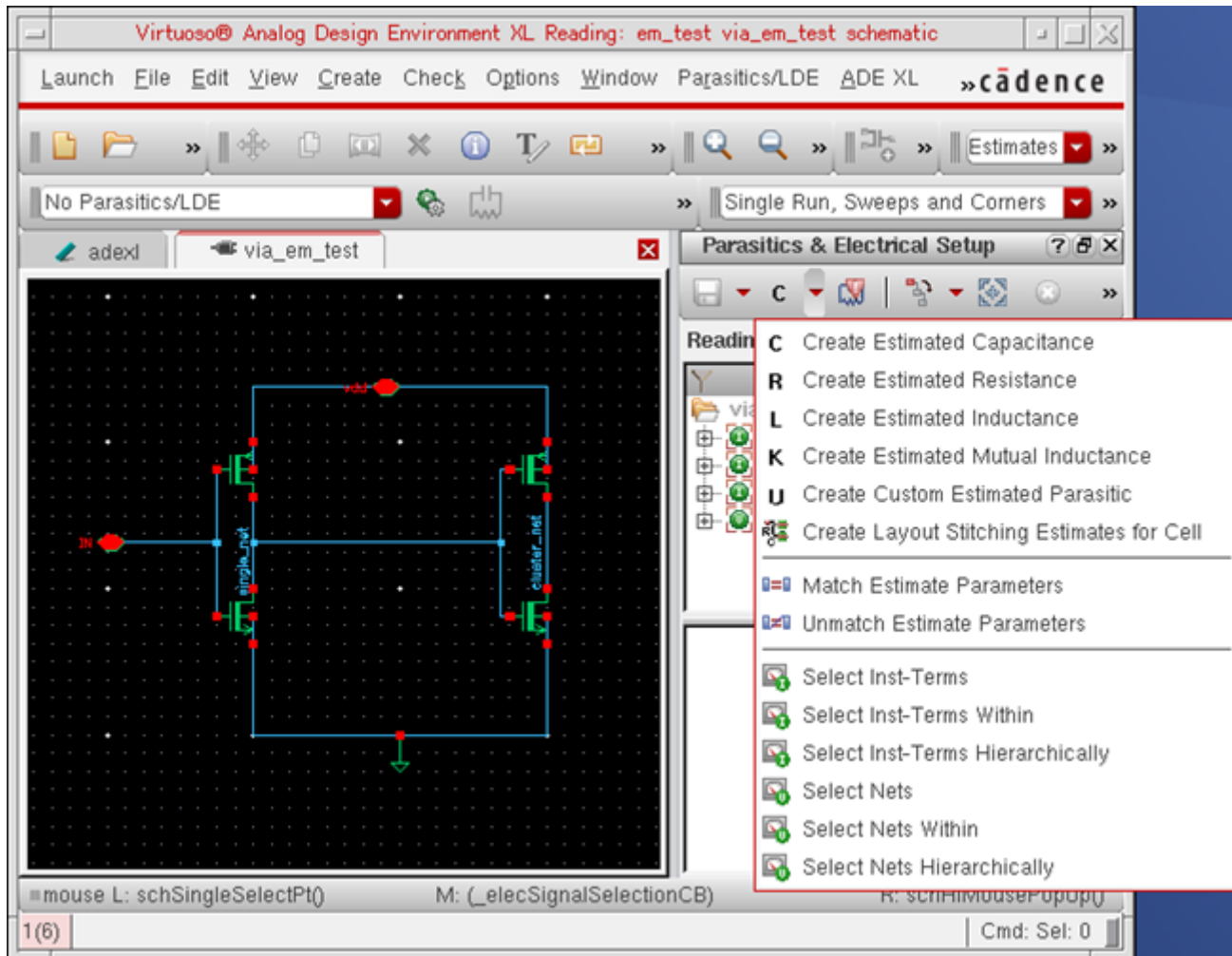


Similarly, in the [Voltage Dependent Rules flow](#), if you do not select any net, a similar message is displayed to either disable the EAD mode or to select nets on the schematic.

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

The *Create* menu on the Parasitics & Electrical Setup toolbar provides commands to add selected terminals to an electrical selection setup.



You can use the following commands to create selection setups or to add the selected terminals to an existing setup:

- *Select Inst-Terms*: Includes all the terminals for the selected instances
- *Select Inst-Terms Within*: Includes the terminals of the instances within the currently selected instances. Note that the tool traverses only one level down the hierarchy for the selected instances and selects all the terminals from that level.
- *Select Inst-Terms Hierarchically*: Includes all the terminals for all the instances in the complete design hierarchy

For example, to create an electrical selection setup for the terminals associated with one or more instances, you need to perform the following steps:

Virtuoso Electrically Aware Design Flow Guide

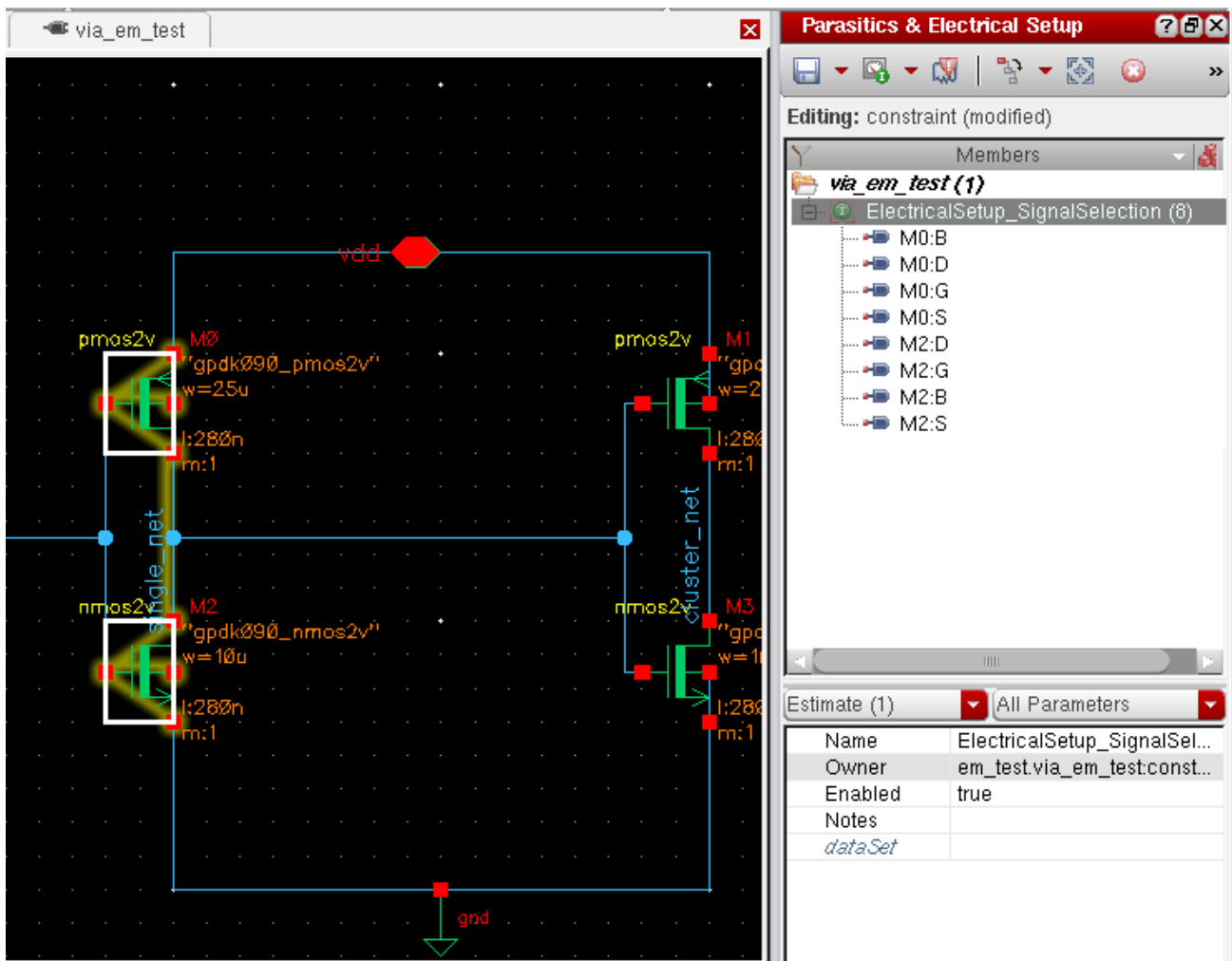
Running Simulations and Saving Electrical Data

1. In the Design Environment XL window, select *EAD — Select Signals*.

The schematic view of the design opens with the `Parasitics-Estimates` workspace, which displays the **Parasitics & Electrical Setup** assistant.

2. Ensure that the schematic is opened in the edit mode.
3. Press the *Ctrl* key and click on the desired instances in the design schematic.
4. In the **Parasitics & Electrical Setup** assistant, select *Create – Select Inst-Terms*.

A new selection setup with a default name, `ElectricalSetup_SignalSelection`, is created and added to the existing list for the design, as shown in the figure given below.



Note: In the example shown above, instances `M0` and `M2` were selected to create an electrical selection setup for all the terminals of these two instances. The tool extracted the details of terminals and listed their names under the

Virtuoso Electrically Aware Design Flow Guide

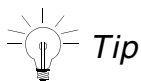
Running Simulations and Saving Electrical Data

ElectricalSetup_SignalSelection selection setup. The count of the selected terminals is also displayed to the right of the selection setup name.

5. (Optional) Select a particular terminal name in the terminal list of the selection setup and edit the properties, if required.
6. Select *Save Estimates* on the Parasitics & Electrical Setup toolbar to save the new selection setup.



The tool internally saves the electrical selection setups in the constraint view of the design.



Tip

You can modify an existing electrical selection setup by adding more instances or nets to the existing list.

Running Simulations and Viewing Output Current Data

After specifying the test and corner details and creating the EAD setup, click *Run Simulation*  on the Run toolbar. The EAD flow supports all simulators with the following limitations:

- Only AMS Designer simulator version IUS 13.2 and up is supported
- Live processing mode is supported only with Spectre. Post-processing mode (which uses OASIS interfaces to perform the necessary settings), works with all simulators

ADE Assembler runs the simulations and displays the results on the Results tab. When the EAD mode is enabled, by default, the results are displayed in the **EAD** results view.

Note: If the required license, 95510: Virtuoso_Variation_Analysis_Op, is not available before running the simulation, ADE Assembler displays a message box to confirm if the simulation is to be continued without enabling the EAD flow. If you click *OK*, ADE Assembler clears the *Enable Electrical Data Capture for EAD Flow* check box on the EAD Setup form and proceeds with the ADE Assembler simulation with EAD disabled. To enable EAD in the subsequent runs, you need to select the *Enable Electrical Data Capture for EAD Flow* check box again.

You can now view the electrical data in the results and create datasets to be used for electromigration analysis in Layout EAD.

The following sections describes how to view the electrical data from the simulation results and how to create datasets that are referenced in the layout view.

- [Viewing the Current Data](#)
- [Creating Datasets](#)
- [Viewing Datasets](#)

Viewing the Current Data

The current data, *Idc* and *Iavg*, for the selected signals is saved as part of the testbench results. To view this data, ensure that the results of the correct history are loaded and select the **EAD** view on the *Results* tab of the Design Environment XL window.

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

The EAD view is displayed, as shown in the figure below.

Outputs Setup Results

EAD Append

Test TRAN Point 1 Corner Nominal

DUT Masters ALL DUT Instance ALL Group By Flat

DUT Dataset \$dutCellName_\$dutInstance_Interactive.1_Nominal_1_27 Create View PreEM Violations

Search: Search

Name	Idc	Iavg
Test: TRAN, Design Point: 1, Corner: Nominal		
/I3/I0/AVDD	664.2n	52.32u
/I2/I0/NM1/S	-458.6n	-28.35u
/I2/I1/NM0/B	-4.85p	152.1n
/I3/I0/PM0/D	-436.4n	-10.88u
/I3/I0/NM1/G	230.4n	-12.95u
/I3/I0/PM1/D	-108.5n	-41.66u
/I3/I1/NM0/S	-205.6n	-58.5u
/I2/I1/NM1/G	230.4n	-13.49u
/I3/I1/PM0/D	-436.5n	-12.59u
/I2/I1/PM0/G	-119.5n	21.2u
/I3/I1/NM0/G	-172.1p	13.89u

Interactive.0 Interactive.1

Results table

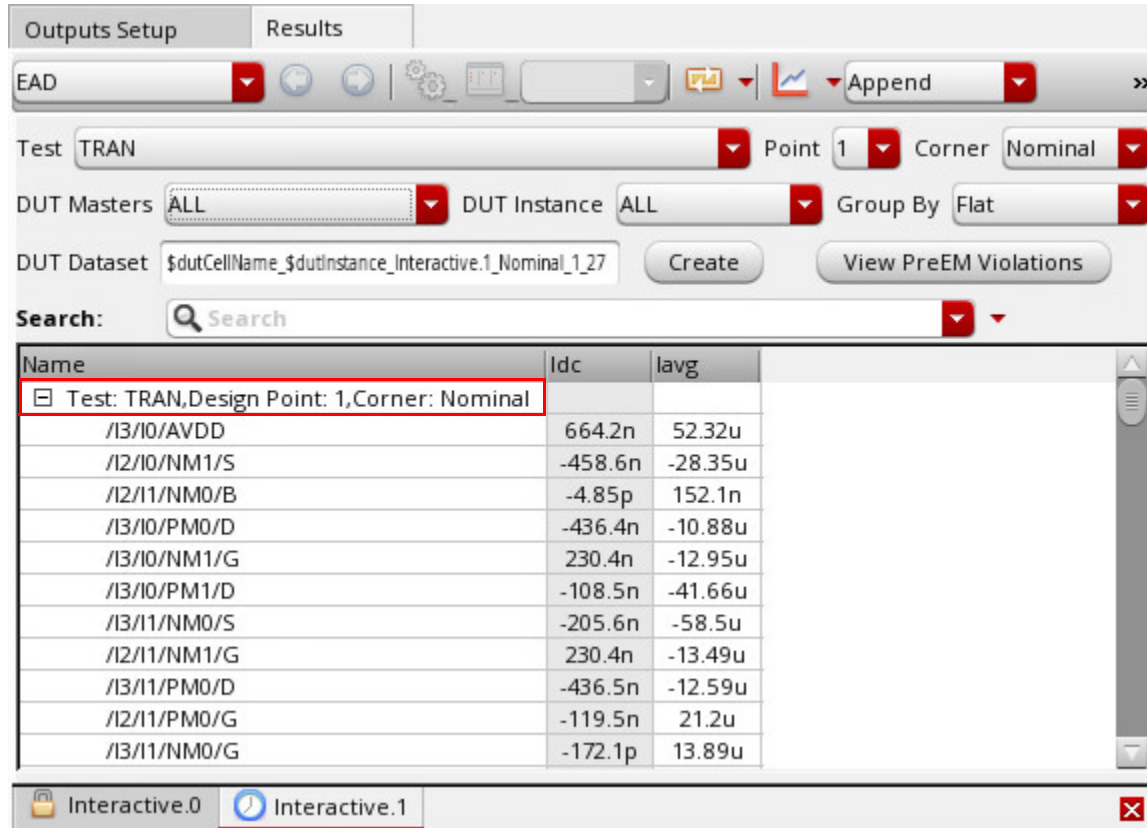
Note: If the simulation is run for reliability analysis, a few extra data filter drop-down lists are displayed in the EAD results view. For more details, refer to [Viewing the Current Data for Reliability Analysis Results](#).

By default, the data for the first test, first design point and the first corner in the ADE Assembler results database is displayed in the results table. The test name, design point

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

number, and the corner name is displayed in the title of the result tree node, as highlighted below.



The result tree for every corner shows the DC current and average transient current values in the *Idc* and *Iavg* columns, respectively. In addition, if you chose to save the waveforms for RMS and peak checks, the *Isignal* column shows waveform icons. This indicates that waveforms have also been saved in the results directory. You can click the waveform symbol in the *Isignal* column to plot the waveforms.

You can change the values in the various drop-down lists given at the top of the results table to apply a desired filter criteria.

The drop-down lists that control the display of the current data in the results table are shown below.

Drop-Down List Displays...

<i>Test</i>	the current or voltage data for the selected test. By default, name of the first test in the results database is selected.

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

Drop-Down List	Displays...
<i>Point</i>	the current or voltage data for the selected design point. By default, the first design point is selected.
<i>Corner</i>	the current or voltage data for the selected corner. By default, name of the first corner saved in the results database is selected.
<i>DUT Masters</i>	the current or voltage data for the selected design master. By default, name of the DUT that you selected in the EAD Setup form is selected. For hierarchical designs, if you had chosen to traverse through multiple levels, the names of the masters for instances within the traversed design hierarchy are listed in the <i>DUT Masters</i> list. You can select any DUT master name to view the current data for it. Select <code>ALL</code> to display the current data for all the DUT masters. Note that this option is used only to display the current data. You cannot create a dataset for all the DUT masters.
<i>DUT Instance</i>	the current or voltage data for the selected design instance. You can either select the name of another instance to display the current data for it or select <code>ALL</code> to display the current data for all the instances in the DUT.

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

Drop-Down List Displays...

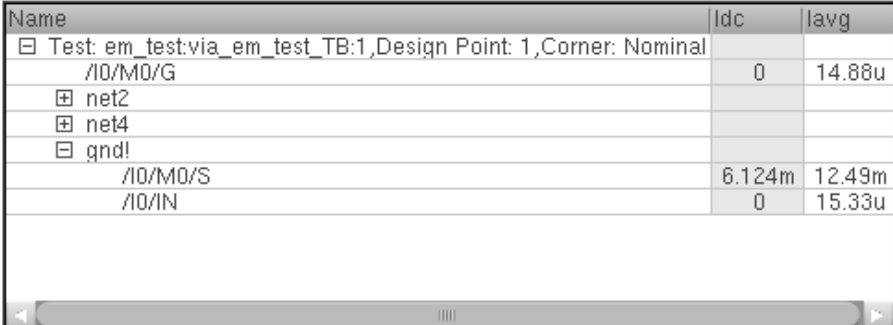
Group By

A flat view of the current or voltage data for all the signals, or groups the rows according to the selected grouping parameter.

The available grouping parameters are listed below.

- **Flat:** Displays a flat view with the current data for all the design instances, and signals (or terminals). This is the default selection.
- **Instance:** Re-groups the current data instance-wise.
- **Instance Master:** Re-groups the current data instance master-wise.
- **Signal:** Re-groups the current data signal-wise.

For example, signal-wise regrouping displays the current data, as shown below.



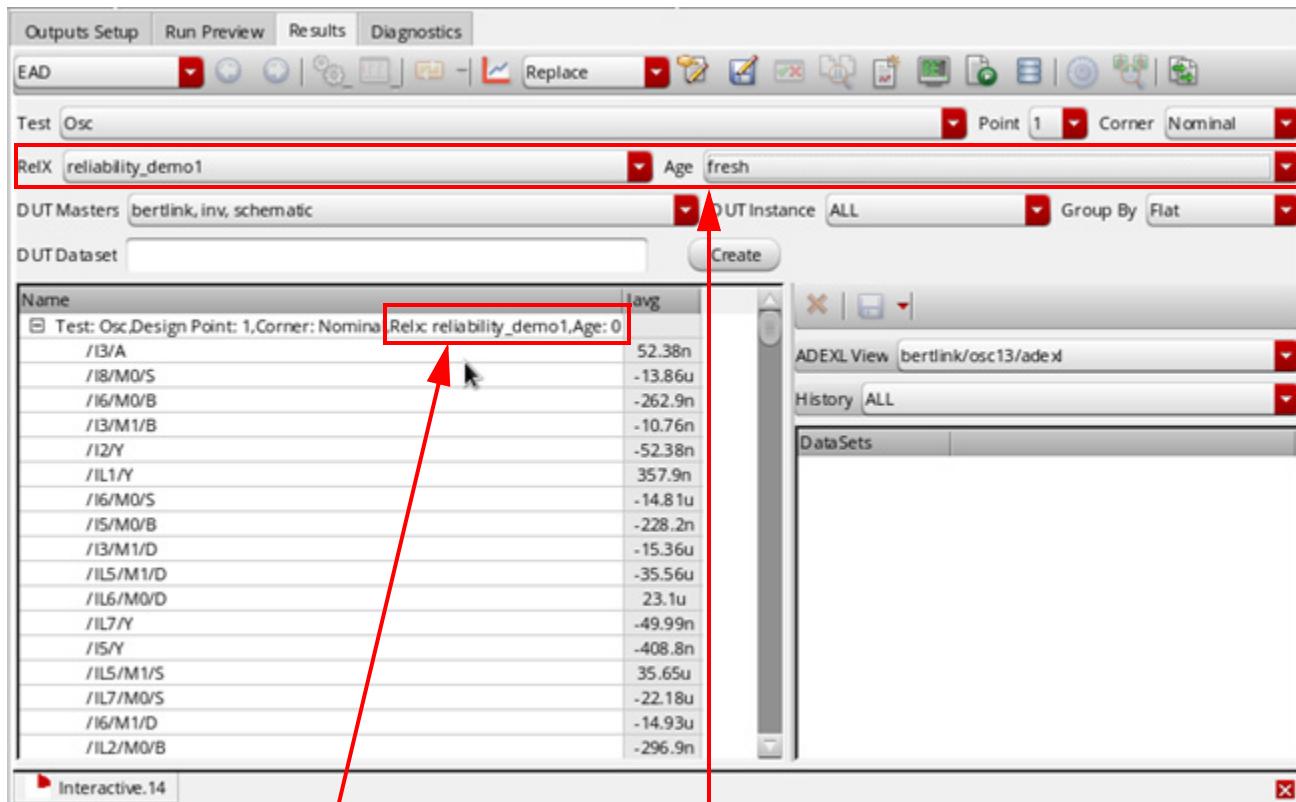
Name	Idc	Iavg
[-] Test: em_test:via_em_test_TB:1,Design Point: 1,Corner: Nominal		
/I0/M0/G	0	14.88u
[+] net2		
[+] net4		
[-] gnd!		
/I0/M0/S	6.124m	12.49m
/I0/IN	0	15.33u

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

Viewing the Current Data for Reliability Analysis Results

When the ADE Assembler simulation is run for reliability analysis, and the results data is saved for fresh, stress, and aged tests, additional drop-down lists are displayed on the Results tab, as shown below.



Additional information in the tree node

Additional filters for the results of reliability analysis

Using the *RelX* drop-down list, you can choose the name of reliability analysis setup for which you need to view the results. Using the *Age* field, you can choose the age for which you need to view the results of fresh and aged tests.

Note that reliability analysis name and age value are also displayed in the title of the result tree node.

Creating Datasets

For electromigration analysis, you need to use the current data in the layout view. For that, you need to save the data in datasets, which can be further referenced from Virtuoso Layout EAD.

For more details on how to use a dataset in the layout view, refer to [Loading an EAD Setup](#) in Chapter 3.

To save the current data displayed in the results table, open the results in the EAD results view and perform the following steps:

1. Select appropriate values in the following drop-down lists:

Drop-down List	Select...
<i>Test</i>	Name of the test for which you want to save the current or voltage data
<i>Point</i>	Point ID If simulation results have only voltage data, this field is disabled when the environment variable <code>ei.dataSet.excludeCurrentsInCI</code> is set to <code>nil</code> and the environment variable <code>ei.setup.enableWorstCaseDatasetForVoltages</code> is set to <code>t</code> .
<i>Corner</i>	Name of the corner for which you want to save results. Select <code>ALL</code> to save the results for all the corners. When you select <code>ALL</code> , multiple datasets are created; one for each corner. Name of a dataset is the string specified in the <i>DUT Dataset</i> field followed by <code>_<incremental_count></code> . If simulation results have only voltage data, this field is disabled when the environment variable <code>ei.dataSet.excludeCurrentsInCI</code> is set to <code>nil</code> and the environment variable <code>ei.setup.enableWorstCaseDatasetForVoltages</code> is set to <code>t</code> .
<i>DUT Masters</i>	Name of the DUT master. Select <code>ALL</code> to save the results for all the instance masters in the design hierarchy. When you select <code>ALL</code> , <i>DUT Instance</i> is also set to <code>ALL</code> . Note: The dataset table is not visible when you select <code>ALL</code> in this drop-down list. To view the datasets, select any value other than <code>ALL</code> .

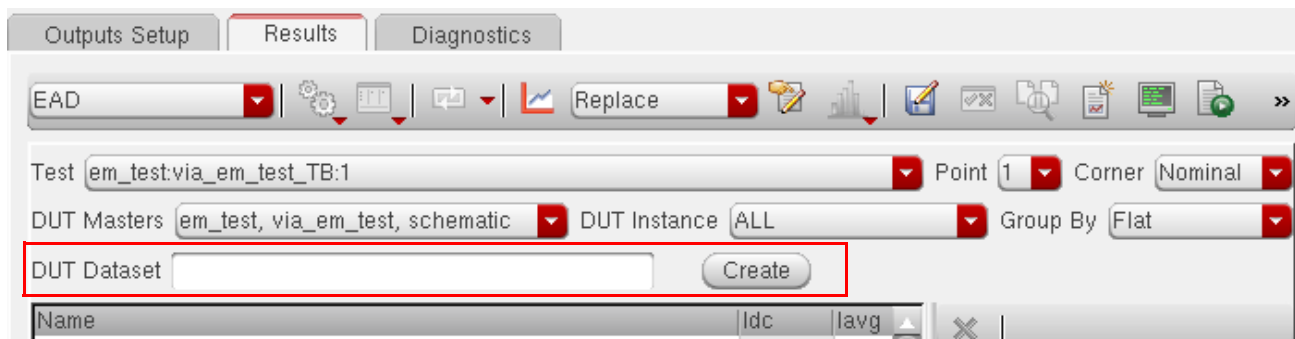
Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

Drop-down List	Select...
<i>DUT Instance</i>	Name of the DUT instance for which you want to save results in the dataset. Select ALL to create dataset for all the instances in the master selected in <i>DUT Masters</i>. If simulation results have only voltage data, this field is disabled when the environment variable <code>ei.dataSet.excludeCurrentsInCI</code> is set to <code>nil</code> and the environment variable <code>ei.setup.enableWorstCaseDatasetForVoltages</code> is set to <code>t</code>.

Note: If the design specified in the *DUT Masters* drop-down list is not editable, the *DUT Dataset* field and the *Create* command button are disabled.

- Specify a name for the dataset in the *DUT Dataset* field. Depending on the number of datasets being created, if multiple datasets are created, the name specified in this field is used as a prefix.



Note: If no name is specified for the dataset, the default name `dataset_<incremental_count>` is used to name the dataset. For more details on the naming convention of the datasets, refer to [Important Points to Note](#).

- Click *Create*.

Depending on the filters on the **EAD** results view and your choice to save the current or voltage data on the **EAD Setup** form, ADE Assembler creates one or more datasets as described below.

How many datasets are created?

The datasets are created for unique combinations of values in the *Point*, *Corner*, *DUT Masters*, and *DUT Instance* drop-down lists. For example, if the *Corner* drop-down list is set to **All** and an instance name is selected in the *DUT Instance* filter, a separate dataset

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

is created for each corner. The number of datasets created is the total number of corners. Each dataset contains results only for the selected DUT instance.

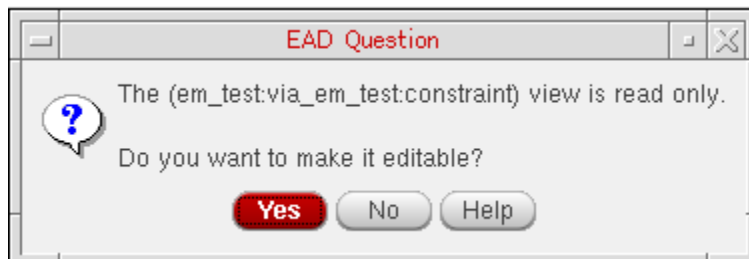
Similarly, if both the *Corner* and *DUT Instance* drop-down lists are set to `ALL`, a dataset is created for each unique combination of point, corner, and instance. The total number of datasets created is the product of the number of corners and the number of instances. Each DUT master can have a different set of DUT instances. Therefore,

If the *DUT Masters* drop-down list is set to `ALL`, multiple datasets are created, one for each unique combination of corner, DUT master, and DUT instance. The number of datasets created for each DUT master is the product of the number of instances it contains and the number of corners.

Note: If the environment variable `ei.dataSet.excludeCurrentsInCI` is set to `nil` and the environment variable `ei.setup.enableWorstCaseDatasetForVoltages` is set to `t`, only a single worst-case simulation voltage dataset having worst *Vmin* and *Vmax* values for all nets is created.

Where are the datasets saved?

These datasets are saved in the constraint view of the design. If the constraint view is not editable, the tool displays the following message and prompts you to confirm if the constraint view can be modified.



Click **Yes** to make the constraint view editable.

Note: If you are sure that ADE Assembler can always make the constraint view editable and it is not required to confirm every time before editing, set the `warnOnCacheEdit` environment variable to `nil`. ADE Assembler will suppress the display of the message box shown above. Instead, it makes the constraint view editable and creates the dataset. After completion, it displays the following information message is displayed in the CIW:

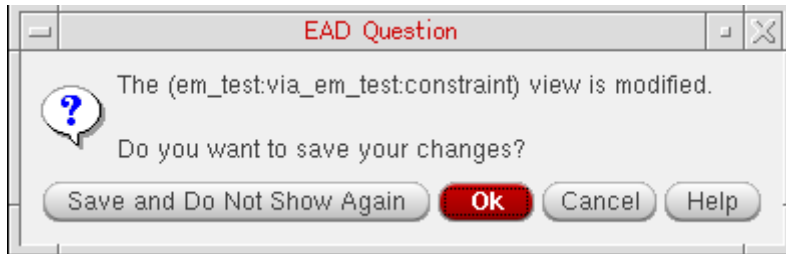
```
--> Creating Dataset ddl on DUT Master: "<lib-name>", "<cell-name>", "<schematic-view-name>"
```

```
INFO (CMGR-5020): Created constraint of type 'currentData' in cache '<constraint-view-name>' with name '<dataset-name>'.
```

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

Next, the tool displays the following message and confirms that the constraint view has been modified and prompts you to save the changes.



Click **OK** to save the changes.

Note: To suppress the display of the message box shown above, set the `warnOnCacheSave` environment variable to `nil`. ADE Assembler will save the constraint view and display the following information message in the CIW.

```
INFO (CMGR-5030): "<constraint-view-name>" saved.
```

ADE Assembler saves the dataset. If a dataset already exists with the given name, ADE Assembler displays a message box to confirm that you want to delete the existing dataset and overwrite it with the new one. To suppress the display of the message box, set the `warnOnDatasetOverwrite` environment variable to `nil`.

Important Points to Note

- The naming convention of the datasets depends on two factors, the value in the *Corners* or *DUT Instance* drop-down lists on the EAD results view and type of data in the simulation results being saved. If you choose to save the data for multiple DUT instances or corners, a separate dataset is created for each corner and DUT instance combination and an incremental number is used for unique identification.

If the simulation results contain both the current and voltage data (because you had selected the check boxes in both the *Electromigration Checking* and *Voltage Dependent Rules* sections on the *EAD Setup* form), separate datasets are created for the current and voltage data. The dataset names are suffixed by `_I` or `_V` to identify the dataset type.

For example, if the value in the *DUT Dataset* field is `myDataset` and the results contain the current as well as voltage data, the datasets are named as `myDataset_0_I`, `myDataset_0_V`, `myDataset_1_I`, `myDataset_1_V`, and so on.

- If you use a design management system, the constraint view needs to be checked out before you can create the dataset. Therefore, the *Create* button will be grayed out until the constraint view is checked out.

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

- When a dataset is created, it contains only the data that is currently displayed in the results table. Therefore, ensure that you have applied the required filters before creating a dataset.
- If ADE Assembler opens the constraints cache while creating or deleting a dataset, it closes the cache after the action is complete.
- If the constraints cache is already opened in the read-only mode and ADE Assembler changes the state to editable before creating or deleting a dataset, it resets the mode to read-only after the action is complete.
- A dataset is saved as a part of the constraint view of the design. However, by default, the actual current data is not saved in it. The dataset in the constraint view contains only a reference to the simulation results database specified by the `adexl.results saveDir` environment variable.

You can choose to save the current data results as part of the constraint view. For this, you need to set the `excludeCurrentsInCl` environment variable to `nil`. By default, this variable is set to `t`.

Important

It is recommended not to save the current data results in the constraint view if you are saving the data for all the terminals in a design. This is because the size of the constraint view will increase and it will take more time to save or open.

- By default, a dataset includes the absolute paths to the simulation results psf directory and the results database (rdb file) where the current data is saved. These paths are saved and displayed as dataset properties in the constraint cellview, as shown in the figure below.

DataSets	
TranDc_1	
lavq	1.00
ldc	1.00
cornerName	Nominal
currentsExcluded	true
designPoint	1
instName	/I0
pointID	1
simResDB	/net/ciclinux81/export/home/sb...
simResDir	/net/ciclinux81/export/home/sb...
simResName	tran-tran
temperature	27

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

You can choose to save relative paths by setting the `storeRelativePaths` environment variable to `t` before running a simulation. In this case, the results directory is saved in the same directory as that of the cellview, as shown in the figure below.

DataSets	
try2	
lavg	1.00
ldc	1.00
cornerName	Nominal
currentsEx...	true
designPoint	1
instName	/I0
pointID	1
resultTypes	lavg\$ldc\$signal
simResDB	em_test/via_em_test/adexl/results/data/Int...
simResDir	em_test/via_em_test/adexl/results/data/Int...
simResName	tran-tran
temperature	27

- When you create a dataset, ADE Assembler automatically locks the history. This is because the current data is saved as a part of the ADE Assembler history and locking the history saves the data from unintentional deletion or removal when the maximum history count is reached. The ADE Assembler history is used later for dynamic electromigration checks and dynamic read.
- The `simResDB` and `simResDir` paths saved in the constraints view are used to load the datasets in the *EM Dataset* drop-down list on the EAD Browser in Layout EAD. Also see [EM Dataset](#).

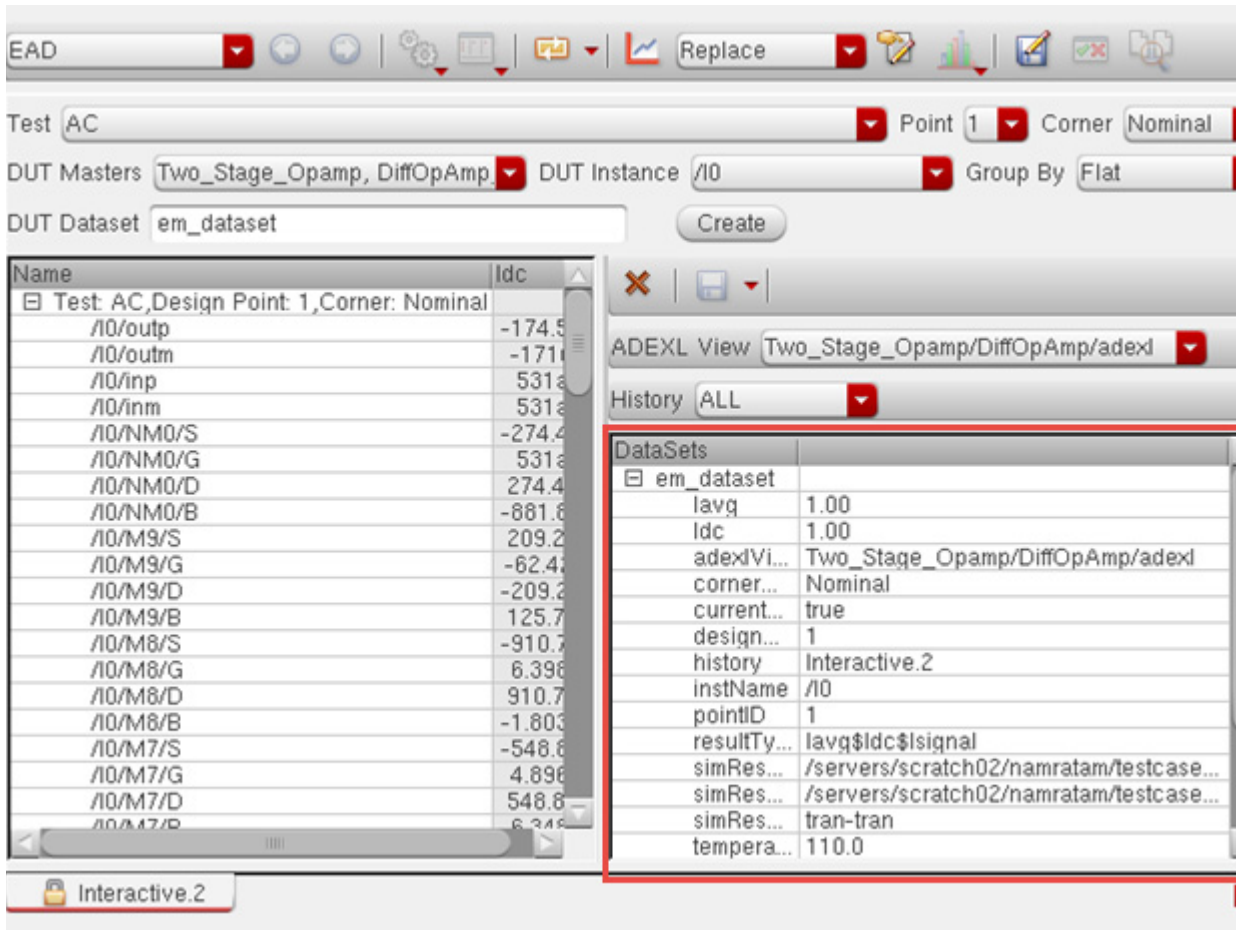
Viewing Datasets

The datasets that contain the current data are saved in the constraint view for the design under test. You can view datasets for a design in any of the following two ways:

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

- ➔ In the dataset table in the EAD results view, as shown in the figure given below.



Select the name of the view from the *ADEXL View* or *Maestro View* drop-down list. All the histories for that view are listed in the *History* drop-down list.

Note: Note that the name of the drop-down list for view selection changes depending on the current tool being used to run simulations and create datasets. For ADE Assembler, the field name is *Maestro View*.

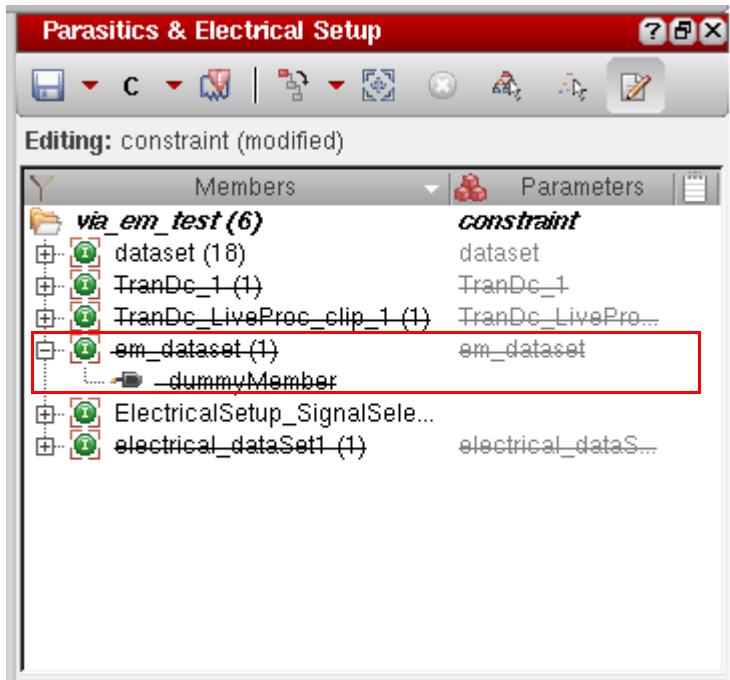
By default, the *History* drop-down list is set to ALL, but you can select a specific history for which you want to view the datasets. All the datasets for the selected history of the given adexl or maestro view are listed in the dataset table.

Expand a dataset to view details, such as the scaling factors (lavg and Idc), test name, corner name, and temperature values.

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

- ➔ In the **Parasitics & Electrical Setup** assistant, as shown in the figure below.



When you expand the dataset name, it does not show any details. Only a dummy member, `dummyMember`, is shown. This is because the dataset does not include any schematic element. Instead, it contains only the references to the current data in the results database.

Note the following points regarding datasets:

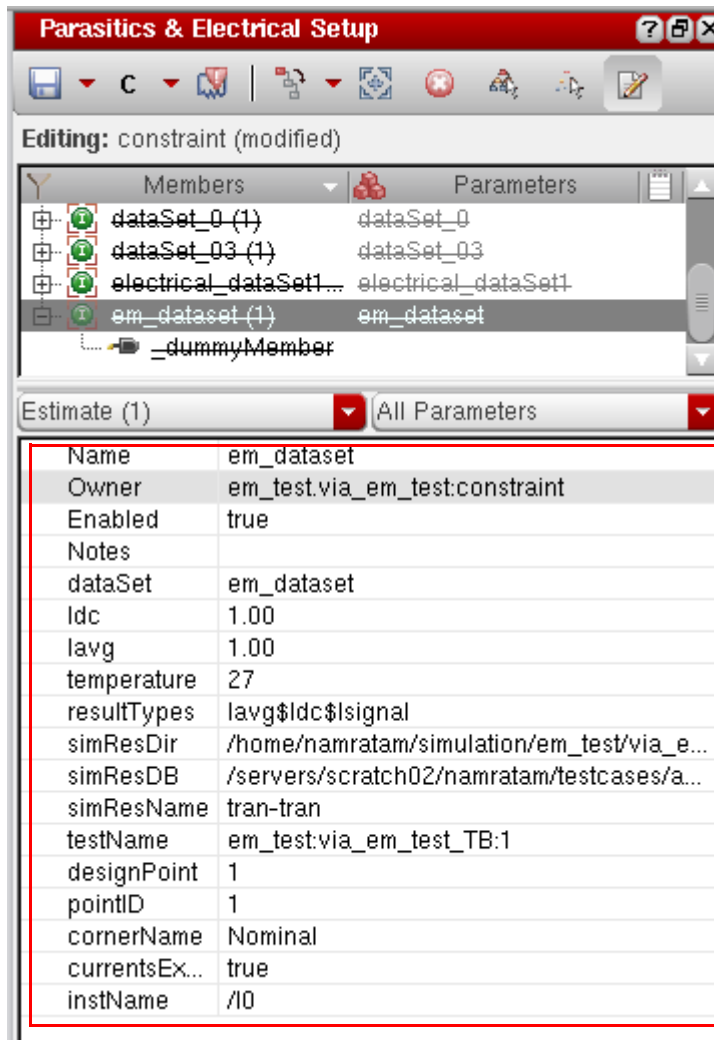
- ❑ You can save multiple datasets from the results of a simulation run, where each dataset can use a different set of filters, such as a different corner name or instance name.
- ❑ To delete a dataset from the Results view, select its name in the dataset table, right-click and choose *Delete DataSet*. Alternatively, select a dataset and click  on top of the table.

You can also delete datasets from the **Parasitics & Electrical Setup** assistant.

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

- ❑ In the **Parasitics & Electrical Setup** assistant, click a dataset name to open the properties section of this assistant. The properties displayed for a dataset are same as those displayed in the dataset table on the Results tab.



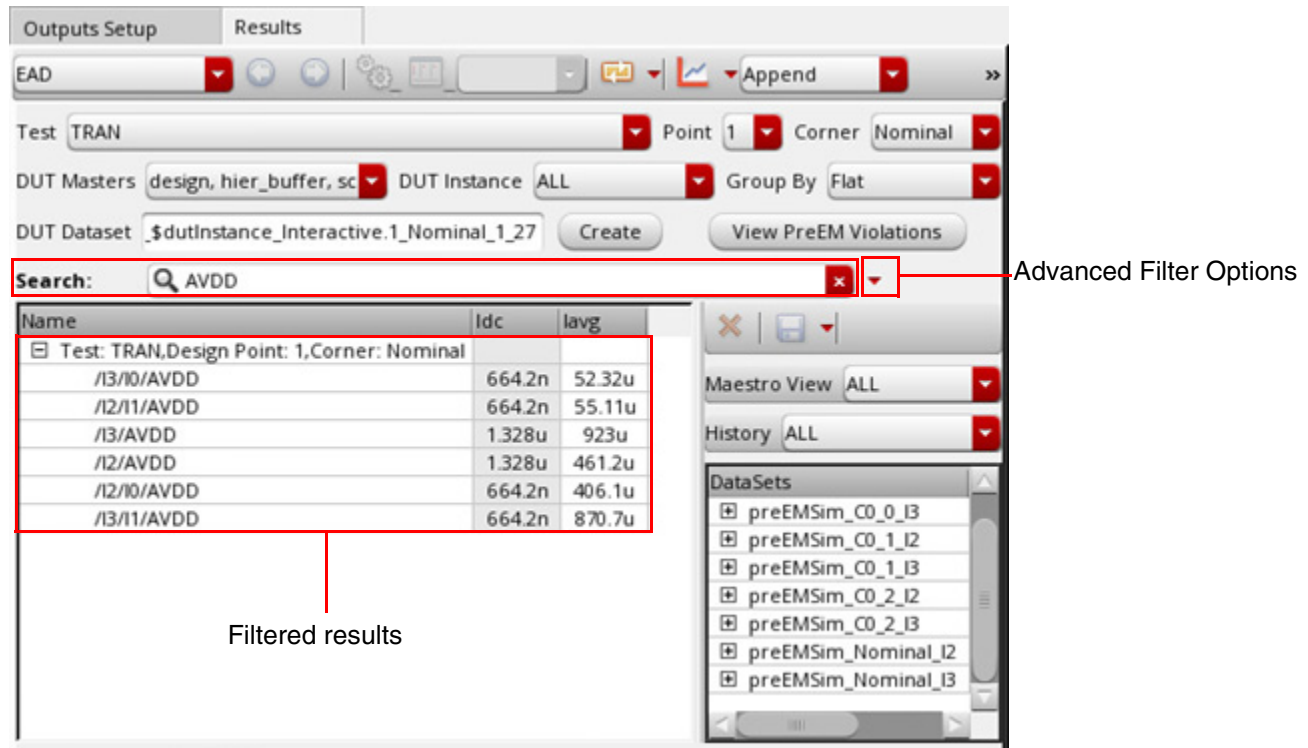
- ❑ You can edit the properties, such as, scale values for ldc and lavg, from both the Results tab or from the **Parasitics & Electrical Setup** assistant.

Filtering Signals in the EAD Results View

To filter signals in the EAD results view:

- ➔ In the *Search* field, type the name of the signal that you want to search.

Search results are displayed interactively as you type.



You can also use the following advanced filter options to define the search criteria.

Advanced Filter Options	Description
<i>Find With</i>	Using this option, you can set the filter criteria to search: ■ All of the words specified in the <i>Search</i> field ■ Any of the words specified in the <i>Search</i> field ■ The exact phrase specified in the <i>Search</i> field ■ Results that do not contain the words specified in the <i>Search</i> field

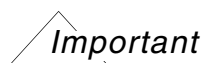
Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

Advanced Filter Options	Description
<i>Match Word</i>	Using this option, you can set the filter criteria to match: ■ <i>Substring</i>: displays the column with the values containing the filter criteria ■ <i>Prefix</i>: displays the columns with values starting with the filter criteria ■ <i>Exactly</i>: displays the columns with values matching exactly with the filter criteria ■ <i>Suffix</i>: displays the columns with values ending with the filter criteria
<i>Using Case</i>	Using this option, you can set the filter criteria to: ■ <i>Insensitive Match</i>: filters the results regardless of the case specified in the filter field ■ <i>Sensitive Match</i>: filters the results matching the case specified in the filter field

Working with VDR Datasets

The VDR Dataset form enables you to view, refresh, and delete voltage information derived from simulation datasets. The form lists the nets in each dataset and the minimum and maximum voltages allowed for each net. You can also create a new CSV dataset from an existing CSV file.



This feature requires an EXL license or an MXL license.

Opening the VDR Dataset Form



Ensure that the *Enable Electrical Data Capture for EAD Flow* option is selected on the EAD Setup form before you open the VDR Dataset form.

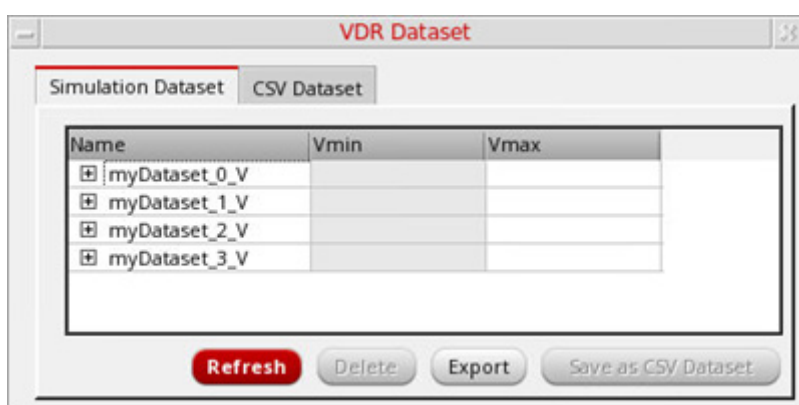
To open the VDR Dataset form, do one of the following:

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

- From the menu bar of Virtuoso ADE Assembler, choose *EAD — VDR Dataset*.
 - From the menu bar of Virtuoso Schematic Editor, choose *VDR — VDR Dataset*.
- Note:** The VDR menu is disabled if the `vdrConstraintGroupName` environment variable is set to an empty string.
- From Virtuoso Layout Suite EXL, choose *Tools — Voltage Dependent Rules — VDR Dataset*.

The *VDR Dataset* form opens.



The VDR Dataset form has following tabs:

- *Simulation Dataset*

The *Simulation Dataset* tab displays a list of all voltage datasets that are generated for the selected DUT after running simulations.

Using this tab, you can perform the following tasks:

- ☐ Viewing Simulation Datasets
- ☐ Deleting Simulation Datasets
- ☐ Exporting Simulation Datasets

- *CSV Dataset*

The *CSV Dataset* tab displays a list of CSV datasets.

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

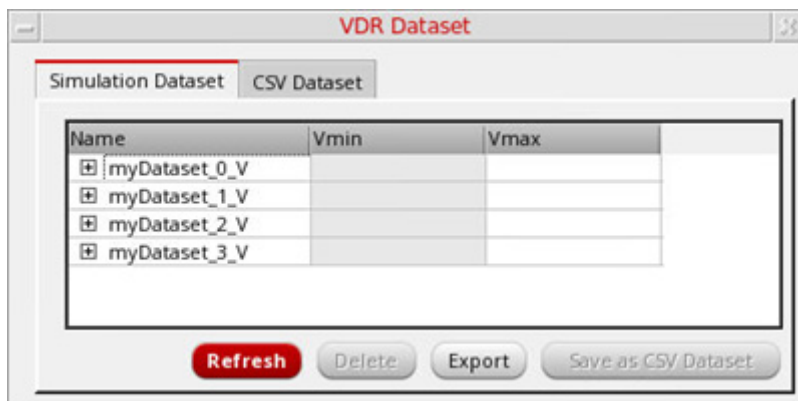
- ❑ [Viewing CSV Datasets](#)
- ❑ [Creating a CSV Dataset](#)
- ❑ [Deleting CSV Datasets](#)
- ❑ [Exporting CSV Datasets](#)

Viewing Simulation Datasets

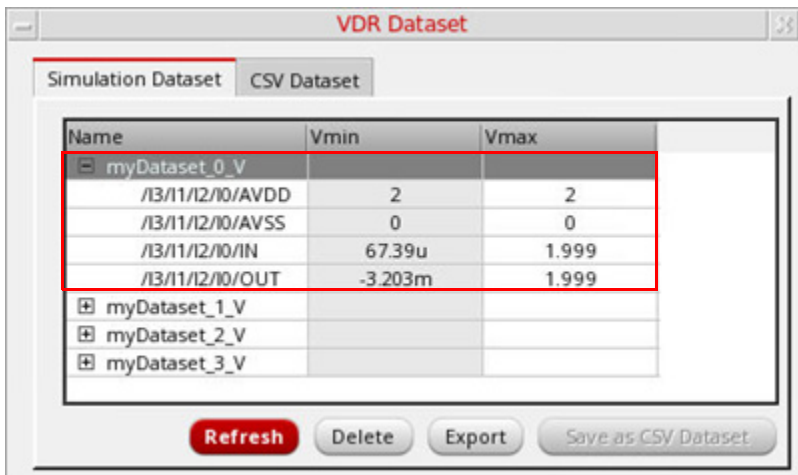
To view voltage datasets generated after a simulation is run for a DUT:

1. Open the VDR Dataset form. For more information, see [Opening the VDR Dataset Form](#).

The VDR Dataset form opens, displaying a list of simulation datasets available for the selected DUT.



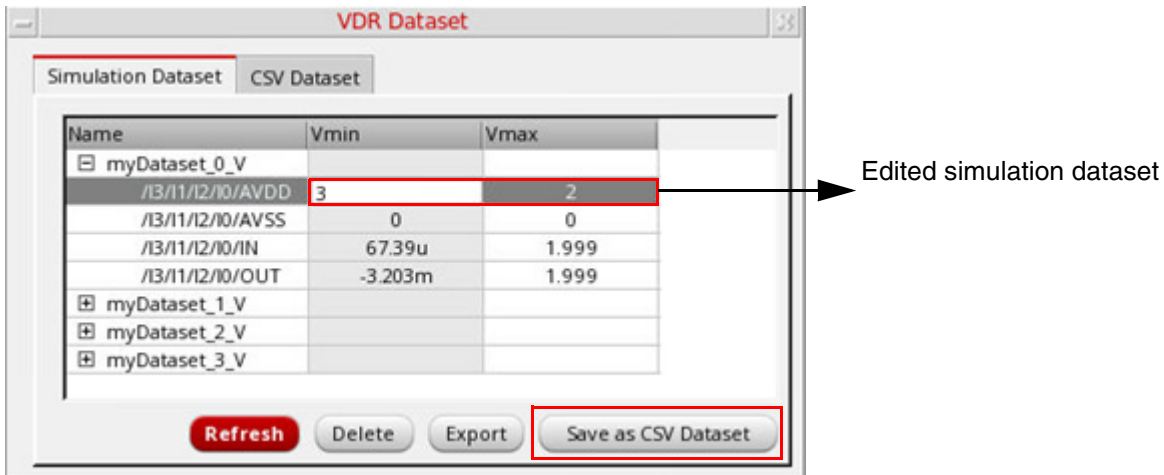
2. In the *Name* column, click the '+' icon next to a dataset name to view the minimum and maximum voltage for the nets in the dataset.



Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

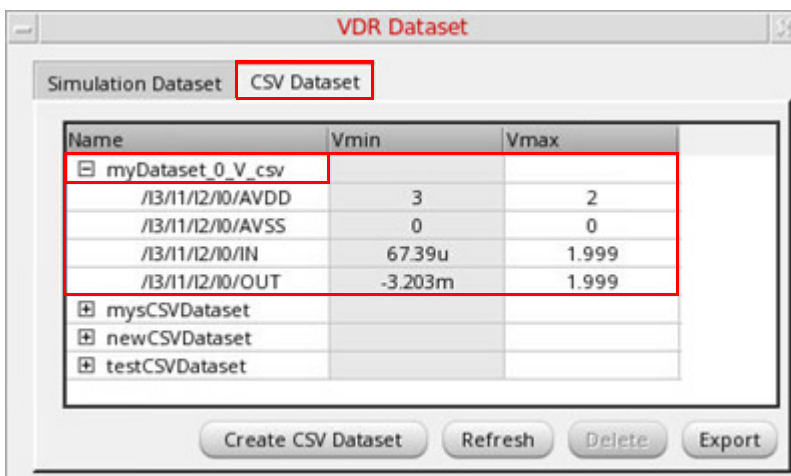
- Double-click the cells in the *Vmin* and *Vmax* columns to edit the minimum and maximum voltages for the nets in the datasets.



Note: The *Save as CSV Dataset* command appears dimmed and becomes available only when you edit a simulation dataset.

- Click *Save as CSV Dataset* to save the edited simulation dataset as a CSV dataset.

The edited simulation dataset is saved as CSV dataset with the name `nameOfSimulationDataset_suffix` and displayed in the *CSV Dataset* tab of the VDR Dataset form. The default suffix for the CSV dataset name is `csv`. A CSV file with the same name is also saved in the current working directory.



- If new simulations have been run, thus creating new simulations datasets, you can view them by clicking the *Refresh* command.

Deleting Simulation Datasets

To delete simulation datasets:

- ➔ On the *Simulation Dataset* tab of the VDR Dataset form, select the name of datasets you want to delete, and then click *Delete*.

The selected simulation datasets are deleted.

Exporting Simulation Datasets

To export simulation datasets in a CSV file:

1. On the *Simulation Dataset* tab, select the name of datasets that you want to export, and then click *Export*.

The *EAD Save CSV For Simulation Dataset* form opens.

2. In the *File name* field, enter the name of the CSV file in which you want to export the simulation dataset.
3. Click *Save*.

The simulation datasets are exported to a CSV file. You can use this file to create CSV datasets.

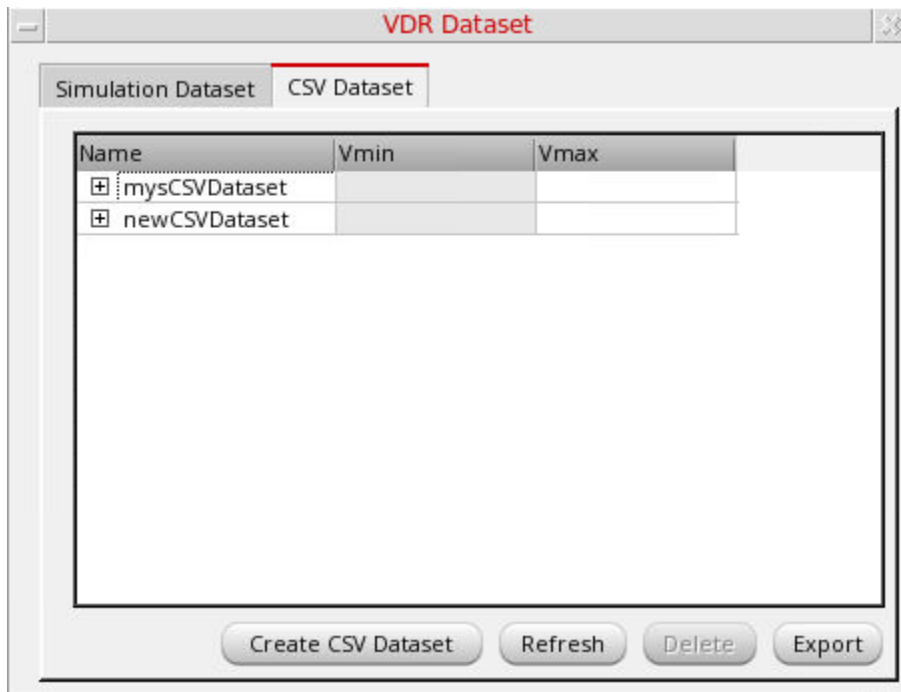
Viewing CSV Datasets

To view CSV datasets:

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

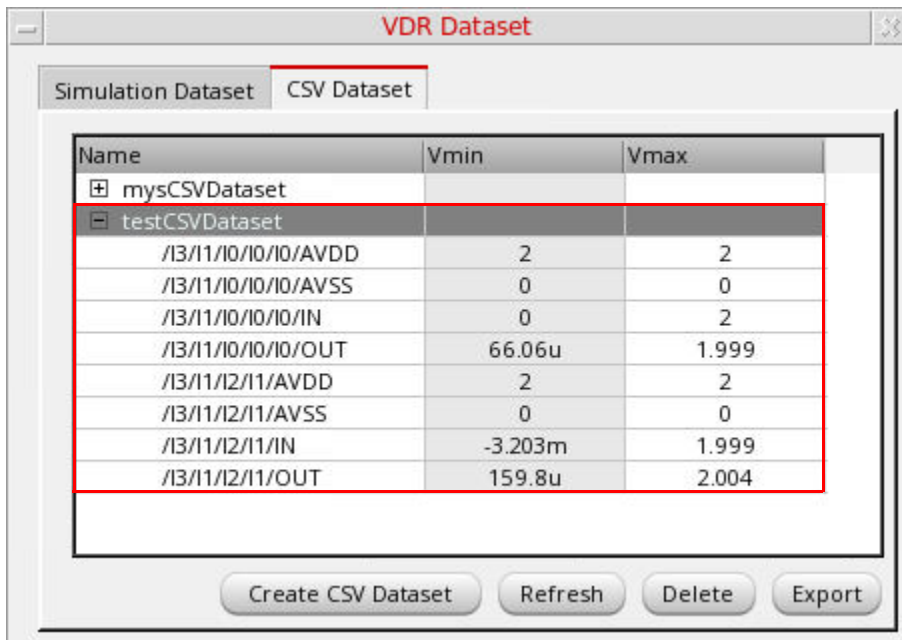
1. Click the *CSV Dataset* tab on the VDR Dataset form to display a list of available CSV datasets.



Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

2. In the *Name* column, click the '+' icon next to a dataset name to view the minimum and maximum voltage for the nets in the dataset.



3. You can also edit the values of minimum and maximum voltages for a net by double-clicking the cells in the *Vmin* and *Vmax* columns against that net.
4. Click *Refresh* to view any new CSV datasets that have been created after you have opened the VDR Dataset form.

Creating a CSV Dataset

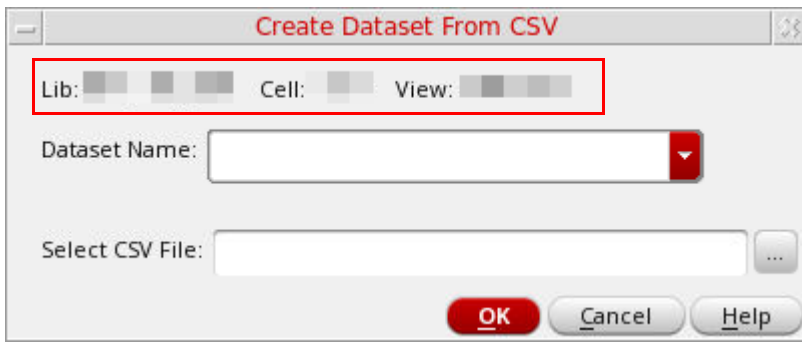
To create a CSV Dataset from a CSV file:

1. On the *CSV Dataset* tab, click *Create CSV Dataset*.

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

The *Create Dataset From CSV* form opens, displaying the library cellview information of the currently selected DUT master.



2. In the *Dataset Name* field, either enter a name with which you want to create the CSV dataset or select an existing name from the drop-down list.
3. In the *Select CSV File* field, click the *ellipsis (...)* command to open the *Choose a File* form.
4. Select a CSV file that you want to use to create the CSV dataset, and then click *Open*.
The CSV file must have entries for maximum and minimum voltages for nets.
5. Click *OK* on the *Create Dataset From CSV* form.

The CSV dataset is created and added to the list on the *CSV Dataset* tab.

Deleting CSV Datasets

To delete CSV datasets:

- ➔ On the *Simulation Dataset* tab of the VDR Dataset form, select the name of datasets you want to delete, and then click *Delete*.

The selected CSV datasets are deleted.

Exporting CSV Datasets

To export CSV datasets to a CSV file:

1. On the *CSV Dataset* tab, select the name of datasets that you want to export, and then click *Export*.

The *EAD Save CSV For Simulation Dataset* form opens.

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

2. In the *File name* field, enter the name of the CSV file in which you want to export the dataset.
3. Click *Save*.

The datasets are exported to a CSV file.

Virtuoso Electrically Aware Design Flow Guide

Running Simulations and Saving Electrical Data

Running the Pre-Layout Electromigration Check

Pre-layout EM check on a schematic help you find whether the MOS device terminals carry more current than the maximum allowed current based on the pin layer shape and size. The limits for each conductor are specified in the EAD technology file. In case of EM violations, EAD may recommend number of fingers or fins for a given device to be EM-compliant.

PreEM check reads the values set for the environment variable `layoutXL`
`lxFingeringNames` to recognize the names of the device finger CDF parameters.

You can run the pre-layout EM check (pre-EM check) on the specified layers of a Pcell. This helps in identifying and resolving EM violations even before creating the actual circuit layout.

Running the pre-EM check automatically creates a scratch layout view based on the device parameters that you specify. It then runs the EM check on the scratch layout and generates a report that lists EM violations. The report also provides recommendations on how to resolve the reported violations, if any.

The pre-EM check reports violations at the terminals where J/J_{max} ratio is greater than or equal to 1. The default threshold percentage value is 100. You can modify the threshold value using the `preEMViolThresholdPercent` environment variable.

Setting Up Environment Variables for the Pre-EM check

Ensure that you set the following environment variables before running the pre-EM check:

- `useCorner`
- `useSetup`
- `preEMCheckLayers`
- `preEMDetailReport`
- `preEMLogFile`

Virtuoso Electrically Aware Design Flow Guide

Running the Pre-Layout Electromigration Check

■ preEMViolThresholdPercent

You can also use the Pre-EM Setup form to specify the extraction setup, extraction corner, and pre-em check layers.

To set these options:

1. From the menu bar of the ADE Assembler, choose *EAD – Run Pre-EM Check*.

The Pre-EM Setup form opens.

2. In the *Extraction Setup* field, specify the path to the file containing the setup options.

For example: `"./cadence/dfII/EAD/1/setup/gpdk45.ini"`

3. In the *Extraction Corner* field, specify the extraction corner that is mapped to an ICT file or eadTechFile to be used for extraction and the source of the EM models to be used for EM Checking.

For example, `nominal`.

4. In the *Pre-EM Check Layers* field, specify the comma-separated list of layers on which pre-EM check is performed.

For example, `Via1, Via2, Via3`

5. Click *OK*.

Related Topics

The Pre-EM Setup Form

Running the Pre-EM Check

You can run the pre-EM check as part of simulation runs in ADE Explorer, and ADE Assembler. For this, you need to first enable the pre-EM check mode in EAD setup.

During the simulation run, a scratch layout is internally created and the pre-EM check is run on the scratch layout. After the simulation is complete, electrical data is automatically saved as CSV datasets and a pre-EM report is generated.

Follow these steps to enable and run the pre-EM check:

1. From the menu bar, choose *EAD – Setup*.

Virtuoso Electrically Aware Design Flow Guide

Running the Pre-Layout Electromigration Check

The *EAD Setup* form is displayed.

EAD Setup

☒ Enable Electrical Data Capture for EAD Flow

☒ Pre EM Check

Design Selection

Design Under Test:

Save Options

Signal Selection

☒ All Signals

☒ Hierarchy Stop Level

☐ Selected Signals (Using the EAD -> Signal Selection menu option)

Electromigration Checking

Type: ☒ DC Current (Idc) Scale:

☒ Average Current (Iavg)

☐ Waveforms for RMS and Peak Checks (Isignal)

Voltage Dependent Rules

Type: ☐ Min and Max Voltage (Vmin/Vmax) Scale:

Waveform Processing Options (For Iavg, Vmin and Vmax)

☐ Process Waveforms Inside Simulator (MMSIM only)

☒ Process Waveform Post Simulation

☐ Clip Waveforms From: To:

OK **Cancel** **Apply** **Help**

2. Select the *Enable Electrical Data Capture for EAD Flow* check box.
3. Select the *Pre EM Check* check box.

Note: The *Pre EM Check* check box is enabled only if the *Enable Electrical Data Capture for EAD Flow* check box is selected.

4. Ensure that the design for which you need to perform electromigration analysis is displayed in the *Design Selection* section.

Virtuoso Electrically Aware Design Flow Guide

Running the Pre-Layout Electromigration Check

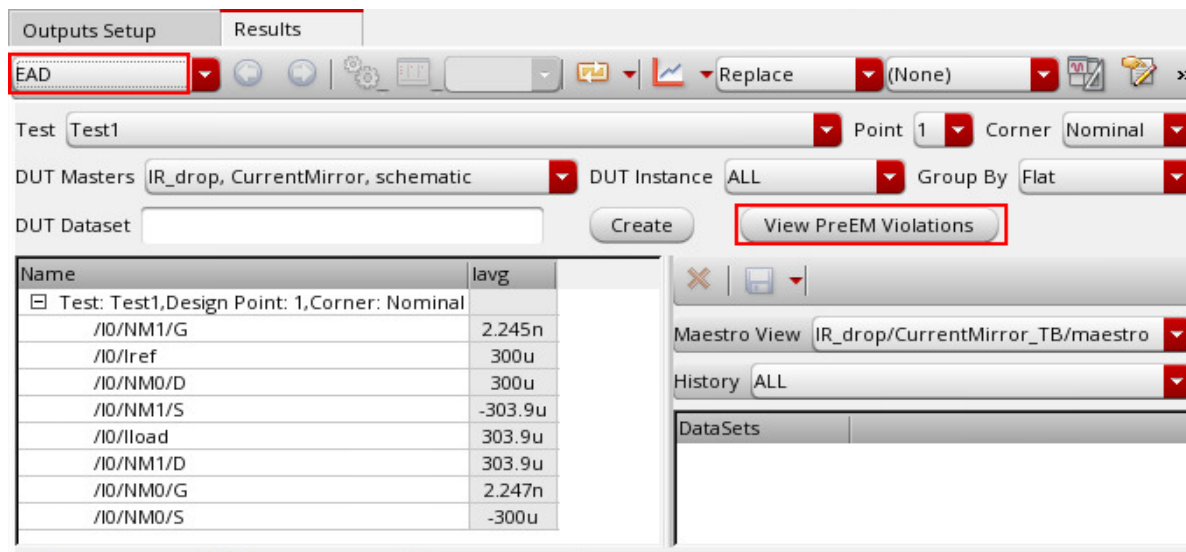
5. Specify the signal and methods to save electric data. For more information about how to set these options, refer to [Preparing the EAD Setup for Simulation](#).
6. Click *OK*.
7. On the *Run* toolbar, click *Run Simulation*.

Note: If simulation results are already available, select *EAD – Run Pre-EM Check* to run the pre-EM check. This helps you save time by eliminating the need of running the simulation again.

The steps to run the pre-EM check are same in ADE Explorer and ADE Assembler.

After the simulation finishes, the EM report opens, displaying the details of the instances and their terminals where EM violations are reported. The report also provides recommendations on how to resolve the reported violations. For more information about the pre-EM report, refer to [Pre-EM Report](#).

If you want to view the EM report again, click *View PreEM Violations* in the EAD results view as shown in the following figure.



Note: The *View PreEM Violations* command is displayed only if you have enabled the pre-EM check in the EAD setup.

Pre-EM Report

The pre-EM report provides the following details:

- Information about the layout and schematic cellviews, and datasets used for EM check
- A summary of pass, fail, and warning statuses of the EM check and recommended number of fingers to resolve the EM violations, if any
- Details of the instances and their terminals where EM violations are reported—J/Jmax ratio, m-Factor, no of terms, no of current fingers, and recommended number of fingers to resolve the reported EM violations
- Runtime information in the following format:

Pre-EM check started at: *start date and time* and ended at: *end date and time*
with elapsed time (wall clock): *elapsed time* s.

Note: To view the detailed pre-EM report, ensure that the `preEMDetailReport` environment variable is set to `t`.

Virtuoso Electrically Aware Design Flow Guide

Running the Pre-Layout Electromigration Check

The following figure shows an example of a pre-EM report:

Information about the layout and
schematic cellviews and datasets

Summary of the pass or fail status of
the EM check and recommendations

```
*****
Utility : Pre-EM checker
Program : sub-version IC6.1.8-64b.500.1.EA32
Started at : Aug  9 15:48:34 2021
User name : urajput
Host name : 
Directory : 
CADENCE Design Systems, Inc.
*****
INFO : Top layout cellview : IR_drop/CurrentMirror/PRE_EM_CHECK_TEMP_LAYOUT_
INFO : Top schematic cellview : IR_drop/CurrentMirror/schematic
INFO : Checked by dataSet(s) ("preEMSim_CO_0" "preEMSim_CO_1" "preEMSim_Nominal")

**SUMMARY**
INSTANCE      CHECK STATUS      RECOMMENDATION (IF ANY)
/NM0           FAIL              6
/NM1           FAIL              6

For details of each instance see below:
-----
Instance name: /NM0, Master cell view: gpdK045/nmoslv_3/symbol
-----
InstTerm: /NM0:S
  Current: "Iavg" (Dataset: "preEMSim_CO_1"), Temperature: 110.0
  Layer: Metall
  WARN ==> Isim: 0.299984 mA, Ilimit: 0.246400 mA, (Isim/Ilimit)*100: 121.75%, mFactor: 2,
  Per m-factored device -> No. of terms: 2, Width per term(Reference-term 'NM0.l:S'):
  0.06, Fingers: 2, Recommended Fingers: 4
InstTerm: /NM0:D
  Current: "Iavg" (Dataset: "preEMSim_CO_0"), Temperature: 110.0
  Layer: Metall
  WARN ==> Isim: 0.299996 mA, Ilimit: 0.123200 mA, (Isim/Ilimit)*100: 243.50%, mFactor: 2,
  Per m-factored device -> No. of terms: 1, Width per term(Reference-term 'NM0.l:D'):
  0.06, Fingers: 2, Recommended Fingers: 6
-----
Instance name: /NM1, Master cell view: gpdK045/nmoslv_3/symbol
-----
InstTerm: /NM1:S
  Current: "Iavg" (Dataset: "preEMSim_CO_0"), Temperature: 110.0
  Layer: Metall
  WARN ==> Isim: 0.304029 mA, Ilimit: 0.246400 mA, (Isim/Ilimit)*100: 123.39%, mFactor: 2,
  Per m-factored device -> No. of terms: 2, Width per term(Reference-term 'NM1.l:S'):
  0.06, Fingers: 2, Recommended Fingers: 4
InstTerm: /NM1:D
  Current: "Iavg" (Dataset: "preEMSim_CO_0"), Temperature: 110.0
  Layer: Metall
  WARN ==> Isim: 0.304073 mA, Ilimit: 0.123200 mA, (Isim/Ilimit)*100: 246.81%, mFactor: 2,
  Per m-factored device -> No. of terms: 1, Width per term(Reference-term 'NM1.l:D'):
  0.06, Fingers: 2, Recommended Fingers: 6
-----
INFO : Pre-EM check completed.
Pre-EM check started at: Aug  9 15:48:34 2021 and ended at: Aug  9 15:48:46 2021 with elapsed time (wall
clock): 11.27s.
```

Runtime information of the Pre-EM check

Details of the instances and their terminals where
EM violations are reported

Virtuoso Electrically Aware Design Flow Guide

Running the Pre-Layout Electromigration Check

The Pre-EM checker reports a warning for instance terminals, if any of the following condition is met:

- There are no shapes connected to the instance terminal.
- There is no current passing through the instance terminal.
- There is no mapping in the layout for the instance terminal in the schematic.

****SUMMARY****

INSTANCE	CHECK STATUS	RECOMMENDATION(IF ANY)
/GAIN/I_CC_P/MP_DUMMY	WARN	0

Instance name: /GAIN/I_CC_P/MP_DUMMY, Master cell view: /symbol

Inst Term: /GAIN/I_CC_P/MP_DUMMY:S
WARN ==> No Shapes
Inst Term: /GAIN/I_CC_P/MP_DUMMY:G
WARN ==> Isim: NO CURRENTS
Inst Term: /GAIN/I_CC_P/MN_DUMMY:G
WARN ==> Isim: NO CURRENTS

If any of the instance terminals have reported a warning, WARN is printed in the CHECK STATUS for that instance in the Pre-EM report.

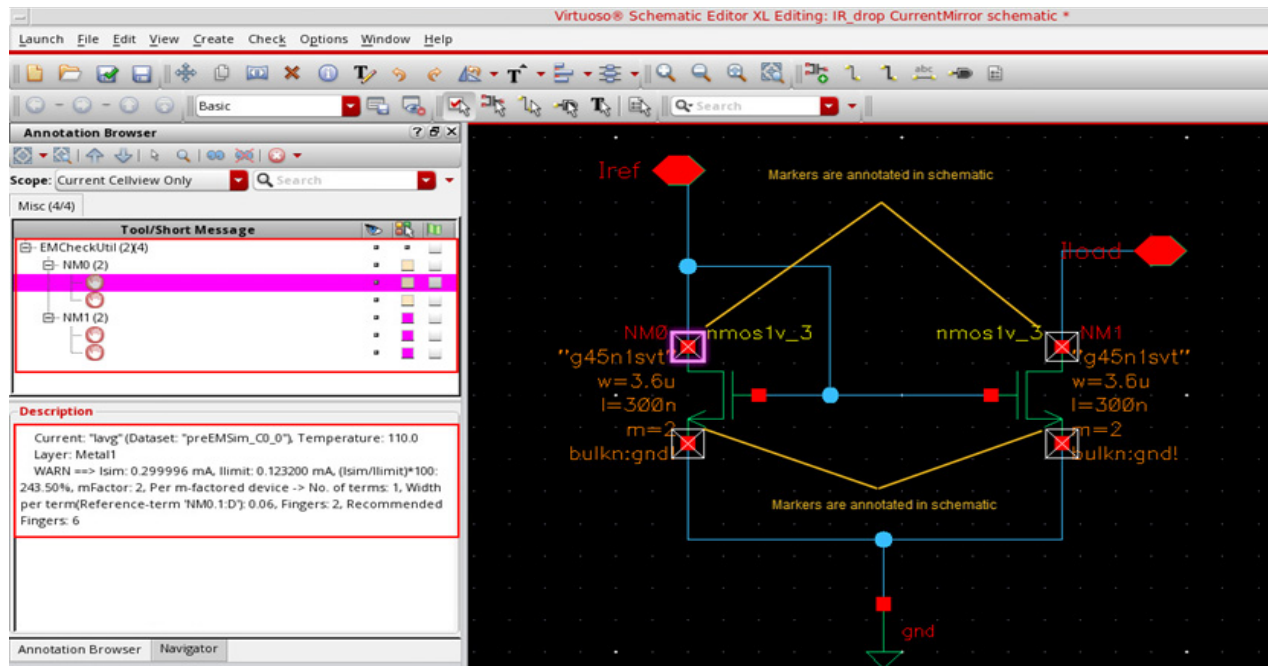
If any of the instance terminals have reported a fail status, FAIL is printed in the CHECK STATUS for that instance in the Pre-EM report.

The priority order of the statuses is as follows: FAIL > WARN > PASS.

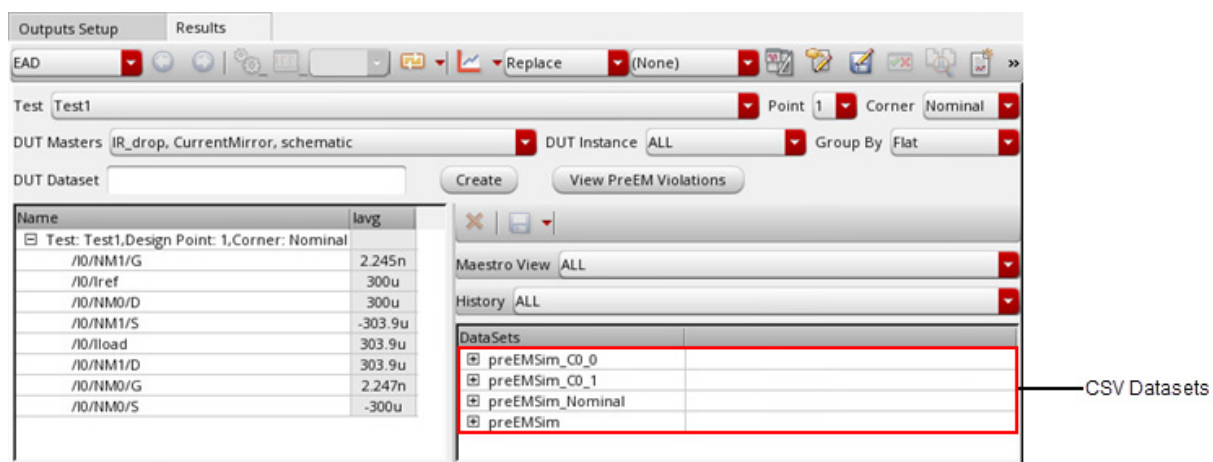
Virtuoso Electrically Aware Design Flow Guide

Running the Pre-Layout Electromigration Check

You can also view the details of the reported violations in the Annotation Browser of Virtuoso Schematic Editor, as shown in the following figure.



If required, you can view the saved CSV datasets by loading the history results in ADE Assembler. To view the datasets, in the EAD results view, from the *Maestro View* and *History* drop-down lists, select *ALL*. The CSV datasets are displayed in the datasets table as shown in the following figure.



Performing What-If Analysis Based on the Pre-EM Check Results

After analyzing the violations reported in the pre-EM check, you can make the suggested changes in the design, and verify the results by running the pre-EM checks again in ADE Explorer, or ADE Assembler.

Alternatively, you may want to reuse the results saved in the CSV datasets, and run a what-if analysis by making some changes in the setup, for example, changing the EM temperature or electrical data in the dataset. You can then run the EM checks using the EAD Browser in Virtuoso Schematic Editor, as explained below.

Preparing the Setup for What-If Analysis

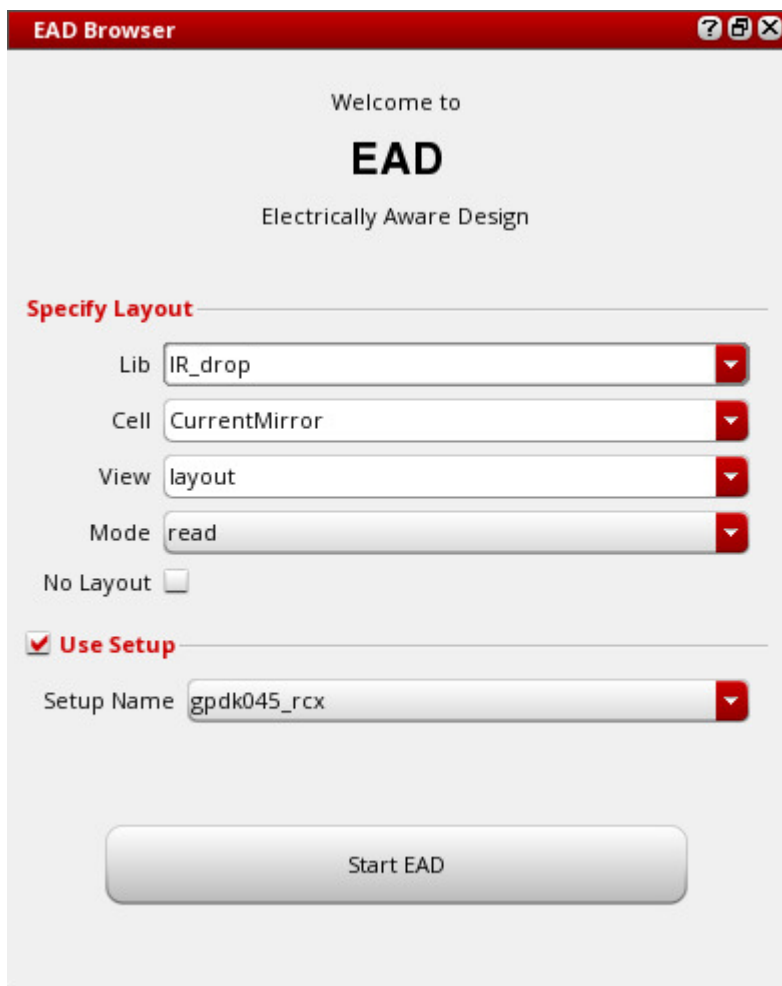
Follow these steps to prepare the set up for what-if analysis:

1. Open the design under test (DUT) in Virtuoso Schematic Editor.
2. From the schematic editor menu, choose *Launch – Plugins – EAD*.

Virtuoso Electrically Aware Design Flow Guide

Running the Pre-Layout Electromigration Check

The EAD Browser assistant is displayed.



3. Select the *No Layout* check box.
4. In the *Use Setup* section, ensure that the correct setup is selected in the *Setup Name* drop-down list.
5. Click *Start EAD*.
6. Import the dataset that has the latest electrical data used for running the pre-EM check by performing the following actions:
 - a. Click the *Update with latest electrical data from simulation* button to reload the electrical data for the datasets that were saved during the last simulation run.
 - b. From the *Electrical Dataset* drop-down list, select the CSV dataset that was created after the simulation run.

Virtuoso Electrically Aware Design Flow Guide

Running the Pre-Layout Electromigration Check

Important

The CSV datasets are available only if you have run the pre-EM check in ADE Assembler or ADE Explorer.

7. By default, the EM checker is disabled in the Schematic EAD. To enable the EM checker, perform the following actions:
 - a. On the *EAD Browser* toolbar, click the drop-down arrow next to *Run EM check* to display additional commands.
 - b. Click *Disable EM Checker* to remove the check mark.

Note: Click *Close* if a message box appears, displaying the temperature mismatch warning.

After the setup is ready, you can run what-if analysis. The following sections describes a few examples of performing the analysis:

- [Editing Terminal Currents in Datasets for What-If Analysis](#)
- [Editing EM Temperature for What-If Analysis](#)

Editing Terminal Currents in Datasets for What-If Analysis

Follow these steps to edit the values of terminal current and to analyze its effect on EM violations:

1. In the *EAD Browser* of Virtuoso Schematic Editor, click the *EM* tab.

Note: If the EM tab is disabled, first enable it by removing the check mark from the *Disable EM Checker* command in the EAD Browser toolbar.

2. Select a net in the summary pane and click *Edit currents* on the EM tab toolbar to display the *Terminal Electrical Values* table. The Terminal Electrical Values table in the detail pane lists all the terminals on the selected net with the electrical values from the dataset.

Note: Ensure that you have selected the correct dataset from the *Electrical Dataset* drop-down list.

3. Double-click the cell next to the terminal to edit its current value.
4. On the *EAD Browser* toolbar, click *Run Pre-EM Checker* (.
5. Select the location where you want to save the log file that contains the detailed report of EM analysis, and then click *Save*.

Virtuoso Electrically Aware Design Flow Guide

Running the Pre-Layout Electromigration Check

The EM report opens. Analyze the effect of changing the terminal current on the EM violations.

Editing EM Temperature for What-If Analysis

Follow these steps to edit the EM temperature and to analyze its effects on the EM violations:

1. In the *EAD Browser* toolbar of Virtuoso Schematic Editor, click the drop-down arrow next to the Options button to display additional commands, and then choose *Edit Options*.

The *EAD Options* form opens.

2. Click the *EM* tab.
3. In the *Temperature* section, edit the value of EM temperature in the *EM Temperature* field.
4. Click *OK*.
5. On the *EAD Browser* toolbar, click *Run Pre-EM Checker* ().
6. Select the location where you want to save the log file that contains the detailed report of EM analysis, and then click *Save*.

The EM report opens. Analyze the effect of EM temperature on the EM violations.

Virtuoso Electrically Aware Design Flow Guide

Running the Pre-Layout Electromigration Check

Extracting Parasitics in the Layout View

This chapter explains how you extract and analyze parasitic information from a partial or complete layout view using the EAD Browser for a layout view. EAD Browser is an assistant you use to drive your electrically-aware design session.

One of the main benefits of the EAD Browser is that it can be used to analyze parasitic information for both partially and fully completed layouts. Not all devices and nets need to be routed before you launch EAD Browser, and you can place and route devices while the application is running.

This section explains how to start EAD Browser from a schematic or layout view, and provides details on the EAD workspace and EAD Browser. It goes on to explain how to define and manage EAD setup and process information and then use that setup to extract parasitics for your design and analyze the results in the EAD Browser and layout canvas.

- [Starting EAD Browser for a Layout](#)
- [The EAD Workspace](#)
- [Managing EAD Setups](#)
- [Extracting and Analyzing Parasitics](#)

 Important

It is essential that you generate a correctly bound, XL-compliant layout in Layout XL. EAD does not support flat layouts imported from GDSII.

 Video

You can view a video demonstration for this feature at [Verifying the Parasitic Information for a Partial Layout](#).

Starting EAD Browser for a Layout

Starting EAD from a Layout Window

You can also start EAD from a layout window. To do this,

- ➔ From the menu bar, *Launch – Layout EAD*.

Virtuoso Layout Suite EAD is launched and The EAD Workspace is loaded. This workspace opens the EAD Browser assistant on the left of the layout canvas.

Starting EAD Browser

You start Layout EXL or Layout MXL in the same way as Layout XL; either from a schematic view that is open in the Virtuoso Schematic Editor or from a layout view that is open in Virtuoso Layout Suite.

EAD and its various features are enabled only while you are in the Layout EXL or in Layout MXL application. When you close Layout EXL or switch to another VLS tier, all EAD-specific data and features are removed from the layout window.

Starting Layout EAD from a Schematic View

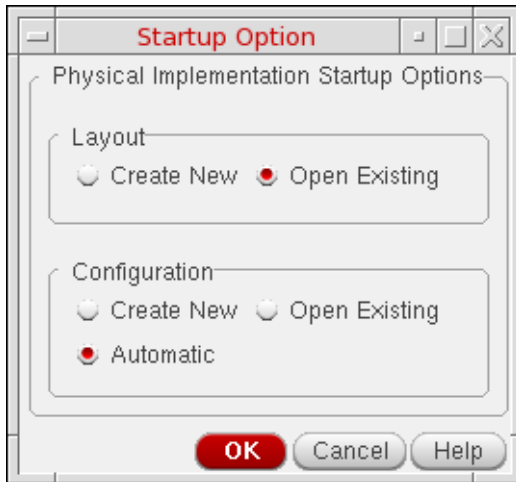
To start Layout EAD from a schematic view,

1. From the schematic editor menu bar, choose *Launch – Layout EXL*, or *Launch – Layout MXL*.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

2. The Startup Option form is displayed.



3. In the *Layout* group box, choose whether you want to create a new layout cellview or open an existing one.
4. In the *Configuration* group box, choose *Automatic* and then click *OK*.

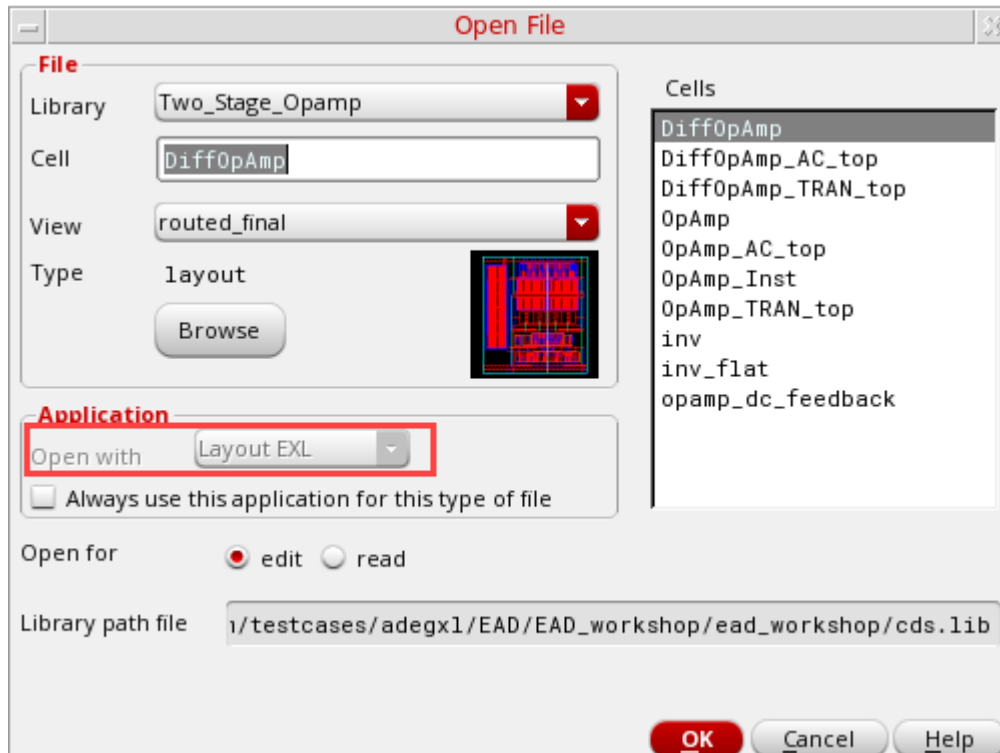
One of the following happens.

- ☐ If you chose to create a new layout view, the New File form is displayed.
Click *OK* to create the new layout cellview.
- ☐ If you chose to open an existing layout view, the Open File form is displayed.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Specify the *Library* and *Cell* names and choose the *View* you want to open from the cyclic field.



Click **OK** to open the cellview.

5. In Layout EXL, choose EAD from the *Workspace Configuration* drop-down list.

The EAD workspace displays the EAD Toolbar and three stacked assistants: the EAD Browser, the Navigator, and the Palette.

Starting EAD from a Layout Window

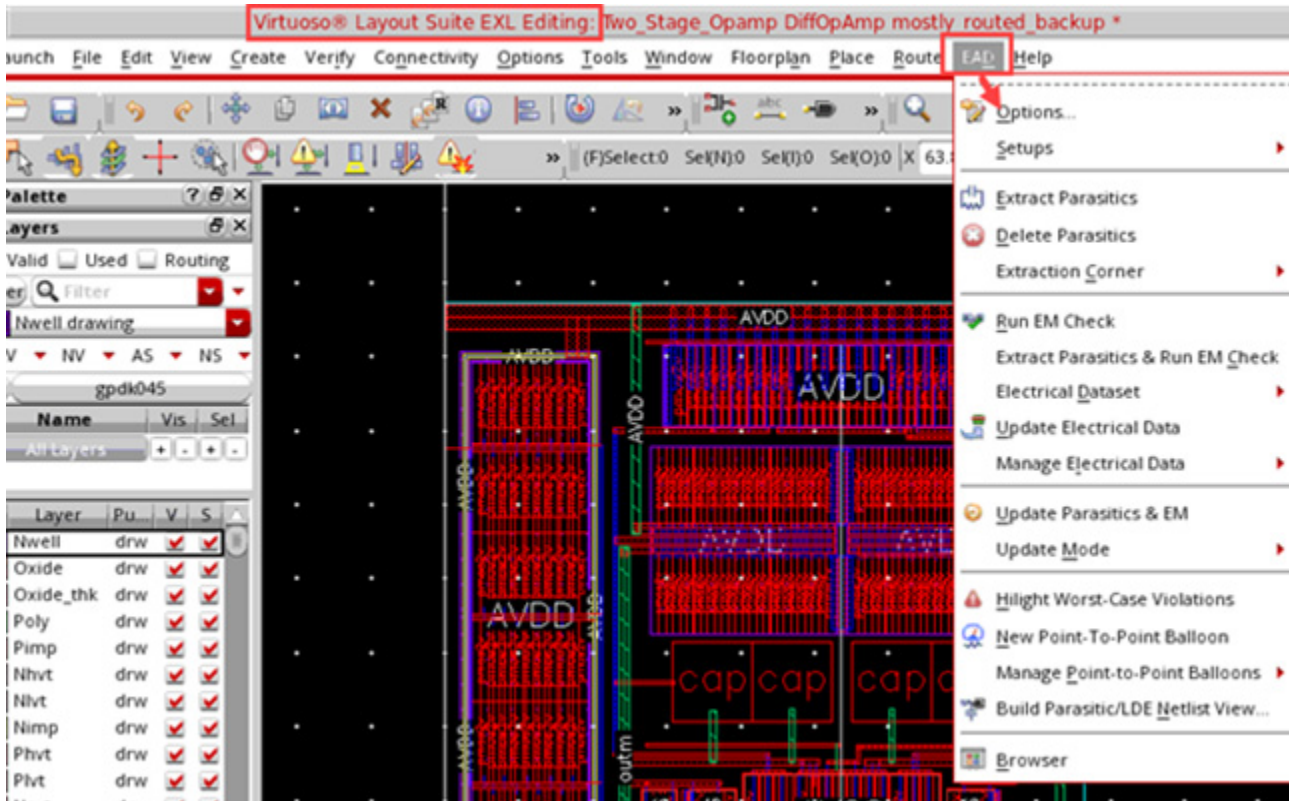
You can also start EAD from a layout window. To do this,

1. From the menu bar, *Launch – Layout EXL*.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Virtuoso Layout Suite EXL is launched. The *EAD* menu is also added to the menu bar, as shown below.

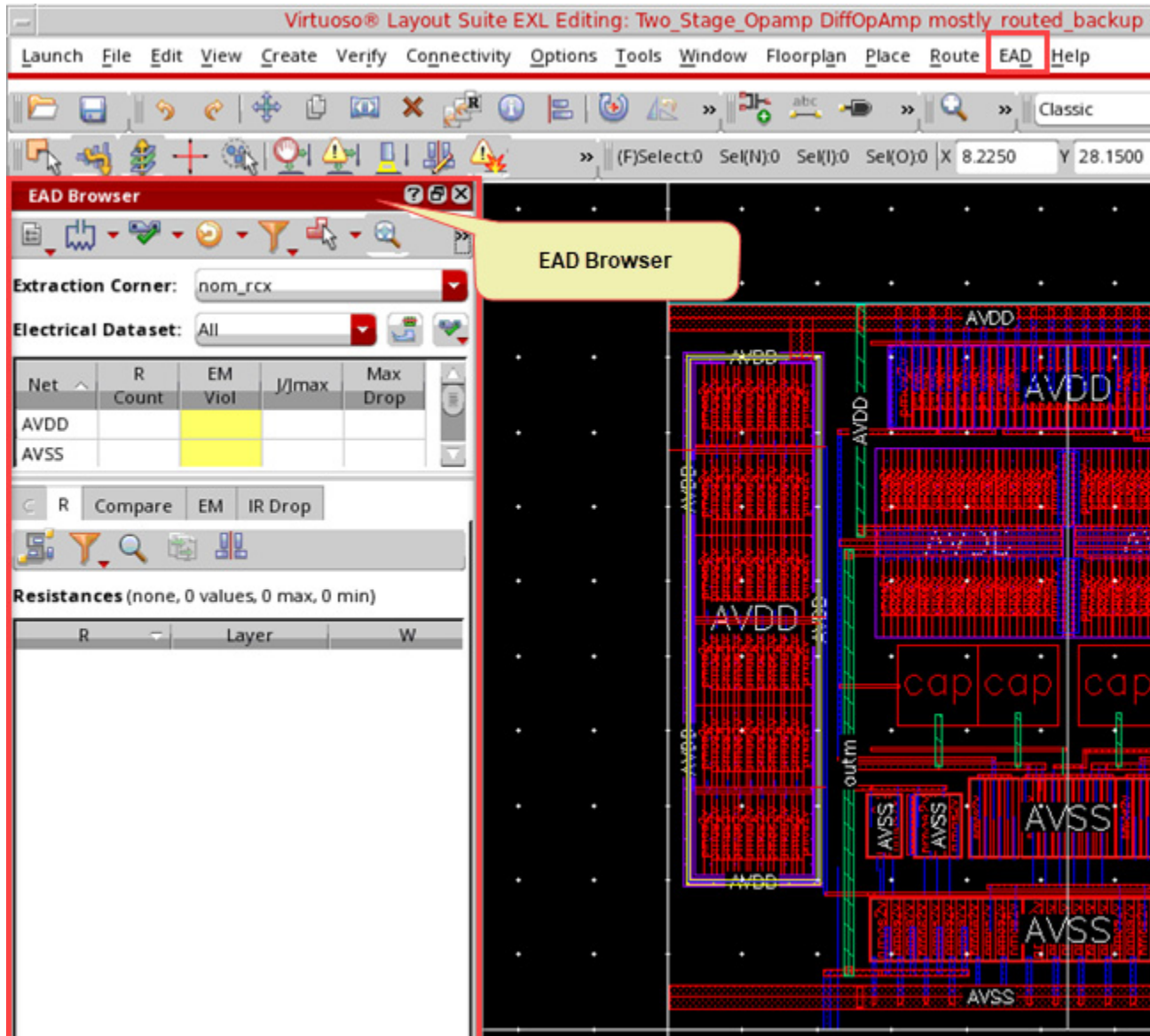


2. Choose *EAD* – *Browser*.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

The EAD Browser is displayed on the left of the layout canvas, as shown below.

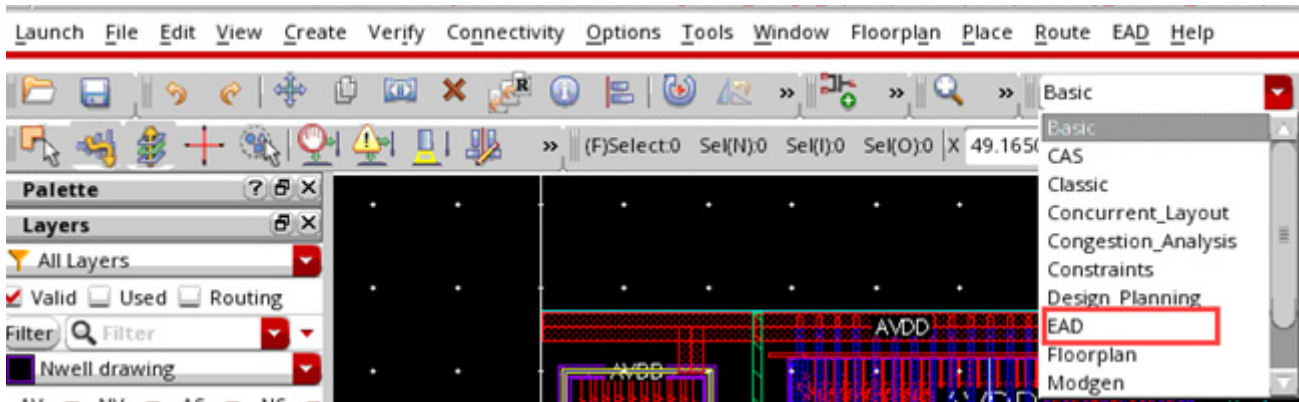


Note: Alternatively, choose EAD from the *Workspace Configuration* drop-down list, as

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

shown below.

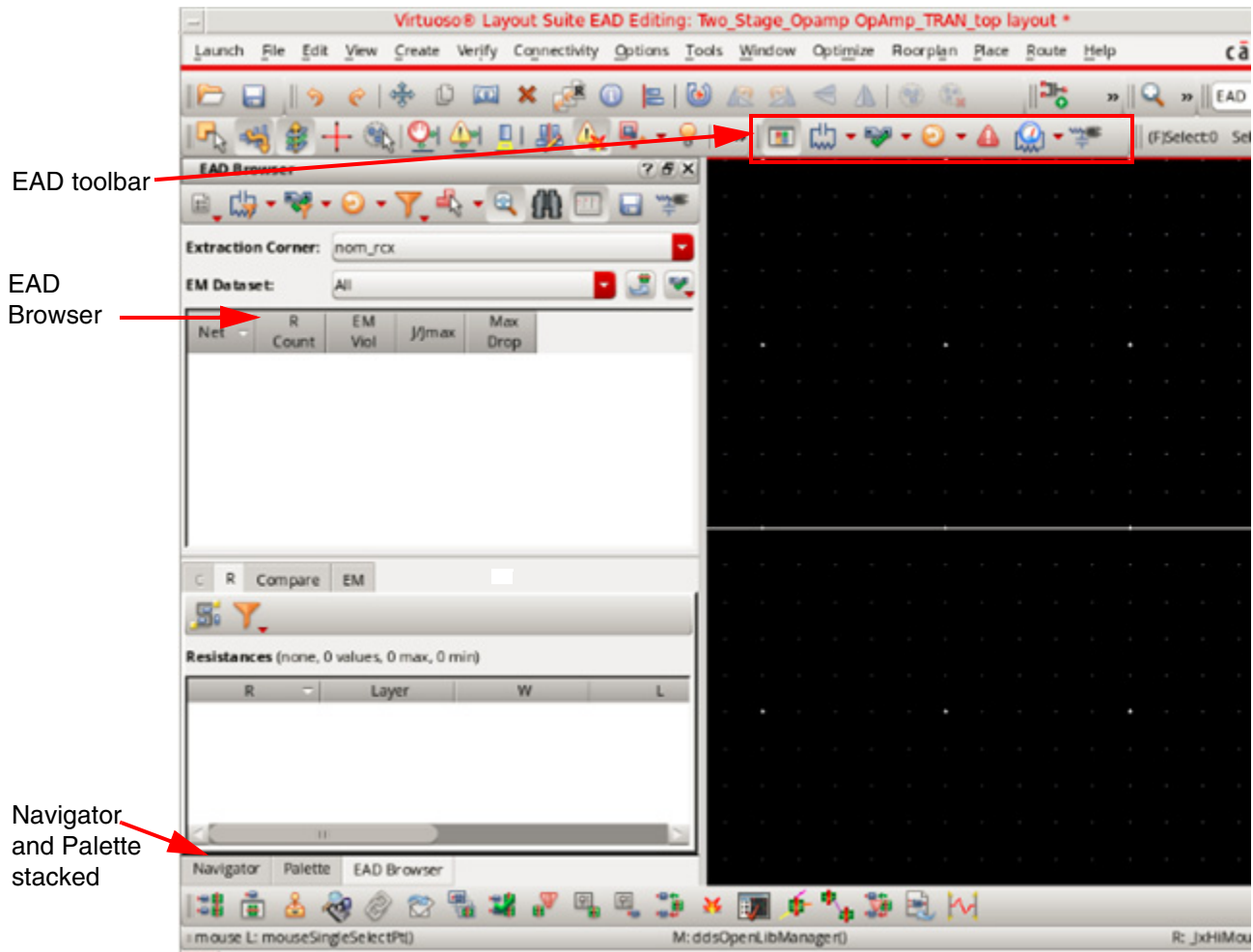


The EAD workspace displays the EAD Toolbar and three stacked assistants: the EAD Browser, the Navigator, and the Palette.

Note: To exit EAD from Virtuoso Layout Suite EXL, choose *Launch – Plugins*, and then click *EAD* to clear the check mark.

The EAD Workspace

The *EAD* workspace is designed to help you drive parasitic extraction and EM analysis, and is installed automatically when you launch Layout EAD from the layout or schematic view.



The workspace comprises the following elements:

- **EAD Toolbar**, which lets you access high level Layout EAD functions with a single mouse click. For more information, see [EAD Toolbar](#).
- **EAD Browser** assistant, which you use to drive parasitic extraction and EM analysis for your design. For more information, see [EAD Browser](#).
- **Navigator** assistant, which shows a hierarchical representation of all the devices, nets, and pins in the design. When the layout of the top level is in view, if you select a hierarchical net in the Navigator assistant, the corresponding net is highlighted in the EAD Browser. When multiple nets are selected, EAD uses a different color for each net.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

For more information, see [Navigator](#).

- **Palette** assistant, which lets you define how layers, objects, and guides are displayed in the layout window. For more information, see [Palette Panels](#) in the *Virtuoso Layout Suite L User Guide*.

EAD Toolbar

The EAD toolbar lets you access the main EAD functions with a single mouse click.



These commands are available in Advanced Nodes Layout EAD Only.

For more information on the individual buttons on the toolbar, see the table below.

Icon	Commands	Lets you...
	<i>Toggle EAD Browser</i>	Toggle the display of the EAD Browser assistant in the layout window.
	<i>Extract Parasitics</i> <i>Delete Parasitics</i>	Extract and delete parasitics for the current design. See Extracting and Analyzing Parasitics for more information. Note: You can also delete parasitic information programmatically using the elecUpdateDataSetParamsPropValue SKILL function. To delete parasitics for selected nets only, select one or more nets in the Summary Pane, right-click and choose <i>Delete Parasitics</i>.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Icon	Commands	Lets you...
	<i>Run EM Check</i>	Run electromigration checks for all the nets in the current design.
	<i>Extract Parasitics & Run EM Check</i>	You can optionally extract parasitics and update electrical data before the EM check is run; and update the electrical data for the design.
	<i>Update Electrical Data, Extract Parasitics, & Run EM Check</i>	
		Also see: Running EM Analysis to Identify Violations
	<i>Update Electrical Data</i>	Update the electrical data for the design
	<i>Delete Electrical Data</i>	Delete the electrical data already extracted and displayed in the EAD Browser.
	<i>Refresh Existing Datasets</i>	Write electrical data information for the existing datasets to the terminals or instance terminals for which the current data is missing or out of sync because they are newly created, modified, or recreated due to layout edits.
	<i>Create User-Defined Dataset</i>	Save custom datasets with a user-defined name.
		Also see: Creating User-Defined Custom Datasets
	<i>Disable EM</i>	Disable EM checks

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Icon	Commands	Lets you...
	<i>Update Parasitics & EM</i> <i>Manual Update Mode</i> <i>Automatic Update Mode</i>	Update the parasitic and electromigration data for the current design. ■ In <i>Manual Update Mode</i>, you must click the button to update the information shown in the EAD Browser ■ In <i>Automatic Update Mode</i>, the information is updated dynamically as you are editing your design When you run these commands, any net that has been extracted previously is updated even if the parasitics have been deleted due to some other action. Bindkey: <code>Ctrl+u</code> See Extracting Parasitics in Layout EAD for more information.
	<i>Highlight Worst-Case Violations for All Nets</i>	Highlight the worst-case EM violations in the layout window. While using this feature, dim the design in the canvas using the <i>Dim</i> command on the Options toolbar of VLS XL. This enhances the view of the main electromigration issues in your design.
	<i>Create New Point-to-Point Info Balloon</i> <i>Close All Point-to-Point info Balloons</i>	Select two points on a net to see electrical information between those two points. See Using Point-To-Point Info Balloons for more information. Note: The info balloons remain visible on the canvas until you close them using the command provided.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Icon	Commands	Lets you...
	<i>Build Parasitic/LDE Netlist View for Simulation</i>	Open the <u>Build Parasitic/LDE Netlist View</u> form that you can use to generate a <code>netlist_layout</code> view for running parasitics resimulations.
	<i>Optimize Trunk Width for Complete Network</i>	Run EM trunk optimization commands. The three commands are <i>Expand to Fix EM Violations</i> , <i>Shrink Oversized Trunks</i> , and <i>Shrink and Expand</i> .
	<i>Select Trunk Optimization Style</i>	Select the style for optimizing a trunk to resolve EM violations. The three styles are <i>Widen - No Stranding</i> , <i>Parallel Stranding</i> , and <i>Stacked Stranding</i> .
	<i>Toggle Trunk Tapering for Complete Network</i>	Enable trunk tapering during trunk widening. Trunk tapering only works for the Widen style and not for parallel or stacked stranding.

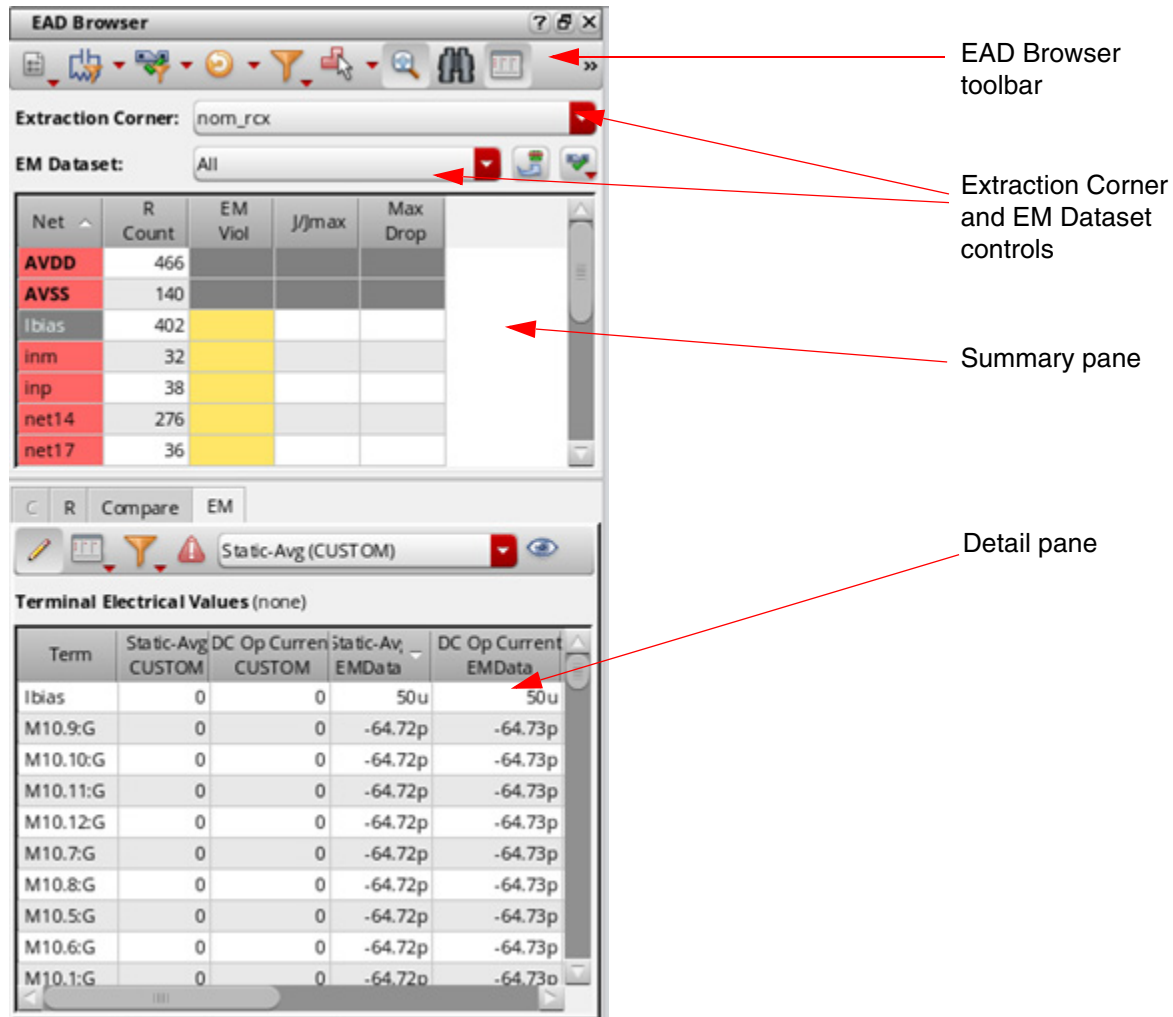
EAD Browser

You use the **EAD Browser** to obtain visual feedback on the parasitics and electromigration issues in your design and to view the effects of any changes you make in order to solve these issues. It provides information about your design beyond that available in either the

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Annotation Browser and Navigator assistants and supports all the relevant Layout XL commands.



The browser has four main components:

- EAD Browser Toolbar
- Extraction Corner and EM Dataset Controls
- Summary Pane
- Detail Pane

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

EAD Browser Toolbar

The EAD Browser toolbar lets you access the main Layout EAD functions with a single mouse click (three of the buttons are also present on the main [EAD Toolbar](#).)



The buttons are organized from left to right to reflect the basic Layout EAD flow. For more information on the functions available, see the table below.

Icon	Commands	Lets you...
	<i>Options</i>	Choose a predefined setup to use for the current session or open the EAD Options form to change any settings or create a new setup if required. See Managing EAD Setups for more information.
	<i>Extract Parasitics for Filtered Nets</i> <i>Delete Parasitics for Filtered Nets</i>	Extract and delete parasitics only for the filtered nets displayed in the EAD Browser.
	<i>Extract Parasitics For All Nets</i> <i>Delete Parasitics For All Nets</i>	Extract and delete parasitics for all the nets in the current design. The commands to extract or delete parasitics for filtered nets are grayed when no filters are applied. See Extracting and Analyzing Parasitics for more information. Note: You can also delete parasitic information using the elecUpdateDataSetParamsPropValue SKILL function.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Icon	Commands	Lets you...
	<i>Run EM Check On Filtered Nets</i>	Run electromigration checks only for the filtered nets displayed in the EAD Browser.
	<i>Run EM Check On All Nets</i>	Run electromigration checks for all the nets in the current design. The <i>Run EM Check On Filtered Nets</i> command is grayed when no filters are applied.
	<i>Extract Parasitics & Run EM Check Update Electrical Data, Extract Parasitics, & Run EM Check</i>	You can optionally extract parasitics and update electrical data before the EM check is run; and update the electrical data for the design.
	<i>Update Electrical Data</i>	Transfers the electrical datasets from ADE Assembler, or schematic to the layout. While doing this, if EAD finds that the name of the dataset being transferred matches with the name of an existing dataset at the layout side, it compares the timestamps of the two. If their timestamp is same, no action is taken. If their timestamp is different, the tool shows a dialog box and prompts you to confirm whether the dataset at the layout side should be overwritten. If you confirm, it deletes the existing dataset and copies the newer version to the layout. Before deleting the dataset at the layout side, EAD ensures that the results database is accessible. If, due to some reason, the results database is not accessible, the transfer of that dataset is skipped, and the version already saved with the layout is retained.
	<i>Delete Electrical Data</i>	Deletes the electrical data already extracted and displayed in the EAD Browser.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Icon	Commands	Lets you...
	<i>Refresh Existing Datasets</i>	Writes electrical data information for the existing datasets to the terminals or instance terminals for which the current data is missing or out of sync because they are newly created, modified, or recreated due to layout edits.
	<i>Create User-Defined Dataset</i>	Save custom datasets with a user-defined name. For more details, refer to Creating User-Defined Custom Datasets .
	<i>Disable EM Checker</i>	Disable EM checks
	<i>Update Parasitics & EM</i>	Update the parasitic and electromigration data for the current design. ■ In <i>Manual Update Mode</i>, you must click the button to update the information shown in the EAD Browser ■ In <i>Automatic Update Mode</i>, the information is updated dynamically as you are editing your design Any net that has previously been extracted is automatically updated in this mode, even if the parasitics have been deleted due to some other user action in the meantime. Bindkey: <code>Ctrl+u</code> See Extracting Parasitics in Layout EAD for more information.
	<i>Manual Update Mode</i>	
	<i>Automatic Update Mode</i>	
	<i>Filtering options</i>	Filter the data in the browser to show only those nets that meet the selected filter criteria. See Filtering Rows for more information.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Icon	Commands	Lets you...
	<i>Toggle ignore selection</i>	Specify how selections in the canvas affect the EAD Browser. ■ <i>Ignore Selection:</i> Canvas selection has no impact on the EAD Browser ■ <i>Select Full Network:</i> Selecting a single shape on a net in the canvas shows data for the entire net in the EAD Browser ■ <i>Select Partial Network:</i> The EAD Browser shows data only for the shapes selected in the canvas
	<i>Automatically zoom display to selected network or parasitic</i>	Automatically pan and zoom to the correct area of the layout view when you select a net or parasitic in the EAD Browser tables. Environment variable: fitToSelected
	<i>Search</i>	Locate results for a particular net by typing the name of the net into a text box. See Searching Nets for more information.
	<i>Toggle dual pane view</i>	Switch between <i>Single pane view</i> and <i>Dual pane view</i>. ■ <i>Dual pane view</i> is the default and has the summary table at the top and the detail table underneath it. ■ <i>Single pane view</i> has all the tabs listed on a single panel. See Toggling Dual Pane and Single Pane View for more information.
	<i>Export table data</i>	Export all the data shown in the tables in either HTML or CSV format.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

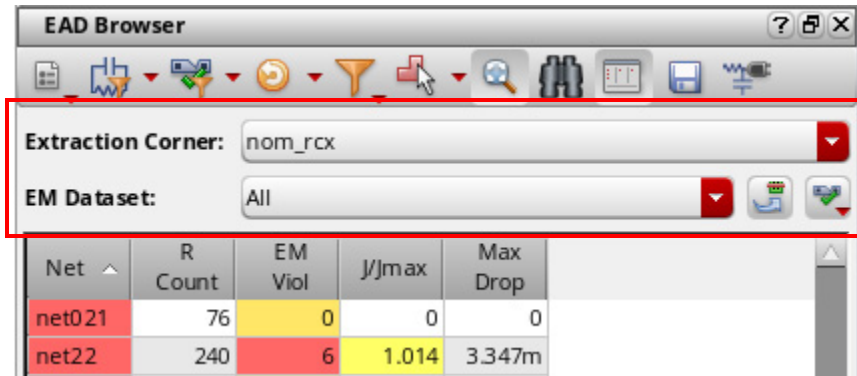
Icon	Commands	Lets you...
	<i>Build Parasitic/LDE Netlist View for Simulation</i>	Opens the Build Parasitic/LDE Netlist View form that you can use to generate a <code>netlist_layout</code> view to be used for running parasitics resimulations.
	<i>Point-To-Point Info Balloons</i>	Interactively shows EAD parameters for the segment between any two points on a net. See the Using Point-To-Point Info Balloons section for more information.
	<i>Save current EAD state</i>	Opens the <i>EAD - Save State</i> form, which you can use to save the current EAD state. See the Saving an EAD State section for more information.
	<i>Load EAD state</i>	Loads an EAD state that has been saved earlier. See the Loading an EAD State section for more information.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Extraction Corner and EM Dataset Controls

Use these controls to specify which extraction corner results are displayed and which dataset is used to run EM checks and display results.



Form Field

Specifies...

Extraction Corner

Which extraction corner results are displayed in the EAD Browser.

The default extraction corner is defined in the process settings file (.ini file) that is saved to the specified location, for example:

```
/.cadence/dfII/EAD/1/process/gpdk045.ini
```

Note: These corners are defined during EAD Setup in ADE Assembler and mapped to an ICT file, which contains the process information for the corner in question. See [Setting up Testbenches and Corners](#) for more information.

Virtuoso Electrically Aware Design Flow Guide

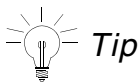
Extracting Parasitics in the Layout View

Form Field	Specifies...
<i>EM Dataset</i>	Which EM dataset is used to run EM checks and display results. The default is <i>All</i>, which means that EM results are reported for all available datasets. By default, the datasets are read from the paths saved in the <code>simResDB</code> and <code>simResDir</code> properties of the constraint cellview, and their names are displayed in the <i>EM Dataset</i> drop-down list. Also see: Creating Datasets. However, if you need to use results from some other dataset, you can use ADE Assembler to overwrite the existing dataset with new simulation data, or load the history results that were used to save the other dataset. Alternatively, you can use <code>elecUpdateDataSetParamsPropValue</code> to set the paths to another existing dataset. To set these paths, you need the <code>*.rdb</code> files and the <code>psf</code> directories, which can be saved outside the ADE Assembler results hierarchy. For the <code>psf</code> directories, ensure that they point to the lowest level of the path. That is, they should include the raw data from simulation. For example, if the results database and the <code>psf</code> directory to be used are saved in the <code>/home/projectA/simulation/</code> directory and the name of the dataset for which you need to change the paths is <code>emData</code>, you can set the paths as shown below: <pre>elecUpdateDataSetParamsPropValue("Two_Stage_Opamp" "DiffOpAmp" "schematic" "emData" "simResDB" "/home/projectA/simulation/vdd13/ vdd13.rdb") elecUpdateDataSetParamsPropValue("Two_Stage_Opamp" "DiffOpAmp" "schematic" "emData" "simResDir" "/home/projectA/simulation/ vdd13/psf")</pre> After setting the paths, click <i>Update Electrical Data</i> on the EAD Browser toolbar to update the electrical data from the specified results database. At any time, you can check the paths to the results database and the <code>psf</code> directory by using <code>elecGetDataSetParamsPropValue</code>. If you specify relative paths in the constraint view or by using <code>elecUpdateDataSetParamsPropValue</code>, Layout EAD resolves them relative to the following directories in the given order of preference: ■ Path specified by <code>adex1.results_saveResDir</code> ■ Path specified by <code>asimenv.startup_projectDir</code> ■ Current working directory

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Form Field	Specifies...
■ <i>Update with latest electrical data</i> 	That all datasets that have been changed in ADE Assembler are updated in Layout EAD. Use this if you have updated your datasets or created new datasets in ADE Assembler while the layout view is open. Note: Datasets that are unchanged in ADE Assembler are skipped. The names of datasets that no longer exist on the schematic side are struck through in the drop-down list (for example, <code>dataset_0</code>).
■ <i>Select EM checks</i> 	Select which EM checks are enabled and disabled. ■ To enable all EM checks, select Enable All Checks. ■ To disable all EM checks, select Disable All Checks. The EM temperature that you set in the EM tab of the EAD Options form is displayed next to the name of the EM dataset. Enable All Checks Disable All Checks ✓ Static-Avg 60 ✓ DC Op Current 60 Dynamic-Peak 60 Dynamic-Rms 60



Tip

You can define a custom callback procedure and register it using the `eadRegTrigger` SKILL function to perform any customized tasks after the dataset transfers to the layout view are complete.

Summary Pane

The summary pane shows a net-specific summary of the parasitic and EM information currently available for each net in the design. The columns displayed depend on the extraction mode that is set.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

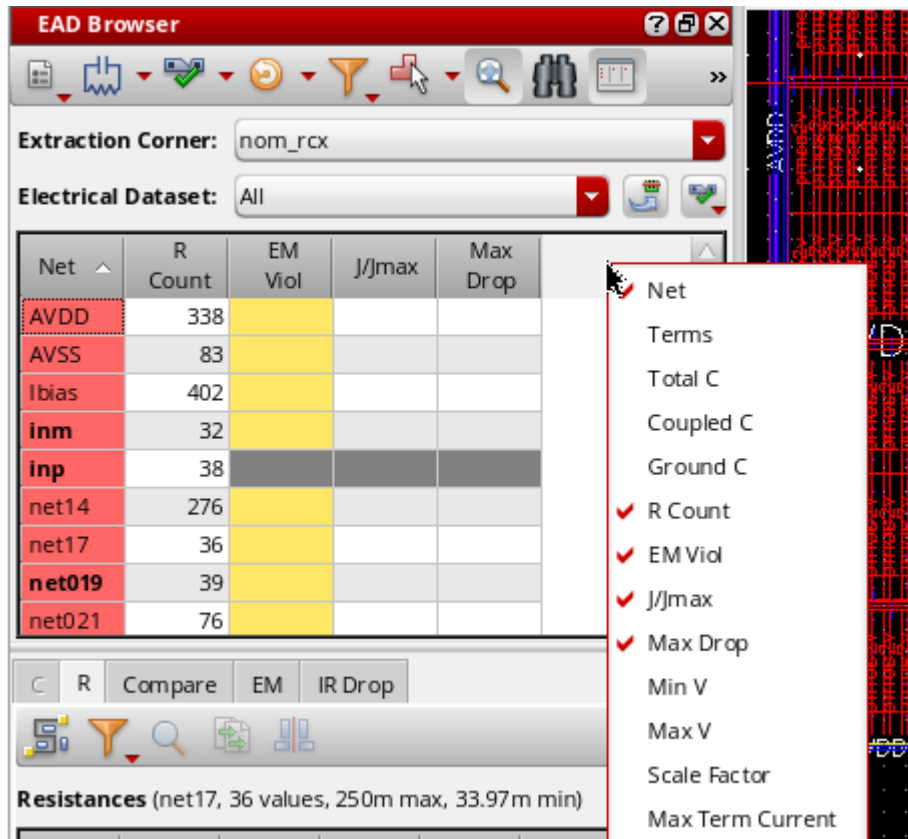
The following table lists all the columns available in the summary pane:

Column Name	Shows...
Net	The names of the nets in the current design.
Terms	The number of terminals on each net. This column is hidden by default.
Total C	Total capacitance on each net. Note: This column is visible only when the C or RC extraction mode is selected on the <i>Extraction</i> tab of EAD Options form.
Coupled C	Total coupled capacitance on each net. Note: This column is visible only when the C or RC extraction mode is selected on the <i>Extraction</i> tab of EAD Options form.
Ground C	Total grounded capacitance on each net. Note: This column is visible only when the C or RC extraction mode is selected on the <i>Extraction</i> tab of EAD Options form.
R Count	Total resistance on each net. This column is visible only when the R or RC extraction mode is selected on the <i>Extraction</i> tab of EAD Options form.
EM Viol	The number of violations reported for each net.
J/J_{max}	The ratio of actual current to maximum current.
Max Drop	The maximum voltage drop value for each net.
Scale Factor	The scaling factor used to scale the current data.
Max Term Current	Maximum terminal current out of all terminals of the net. This column is hidden by default.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

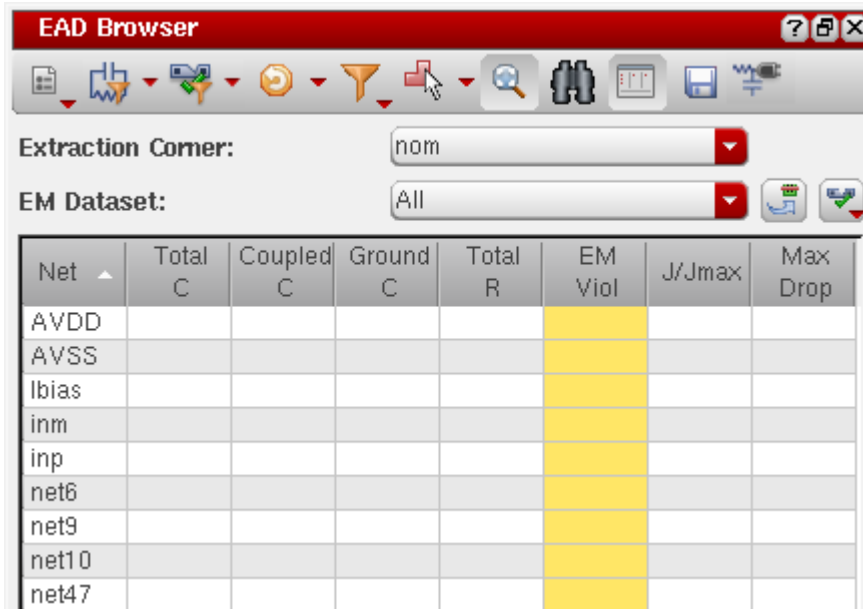
Some of the columns remain hidden by default. To view any hidden column, right-click in the table header and select the column name from the context menu.



Virtuoso Electrically Aware Design Flow Guide

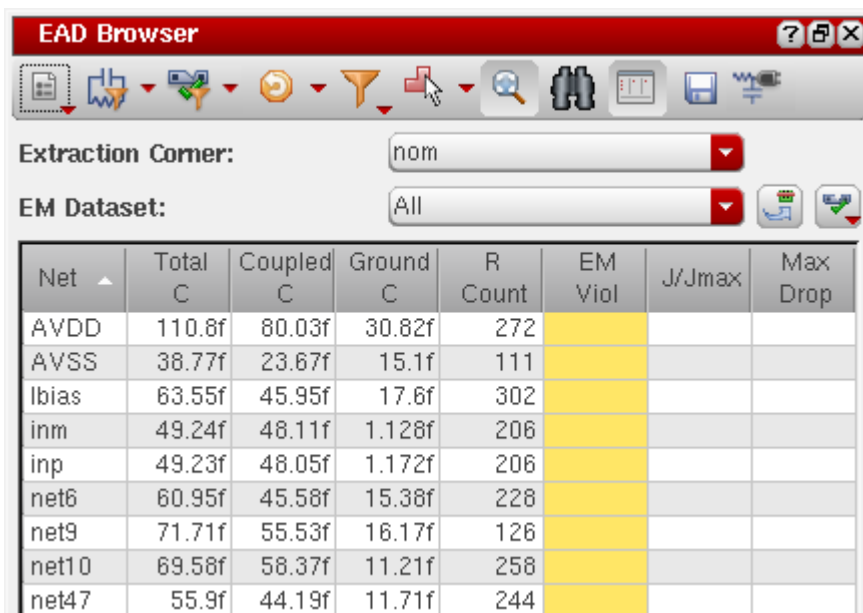
Extracting Parasitics in the Layout View

When you first launch Layout EAD, the table lists all the nets in the current design but does not extract parasitics or run any EM checks by default.



Net	Total C	Coupled C	Ground C	Total R	EM Viol	J/Jmax	Max Drop
AVDD							
AVSS							
Ibias							
inm							
inp							
net6							
net9							
net10							
net47							

After you perform parasitic extraction, the table is updated to show parasitics (total, coupled, and grounded capacitance values and total number of resistors) for the listed nets.



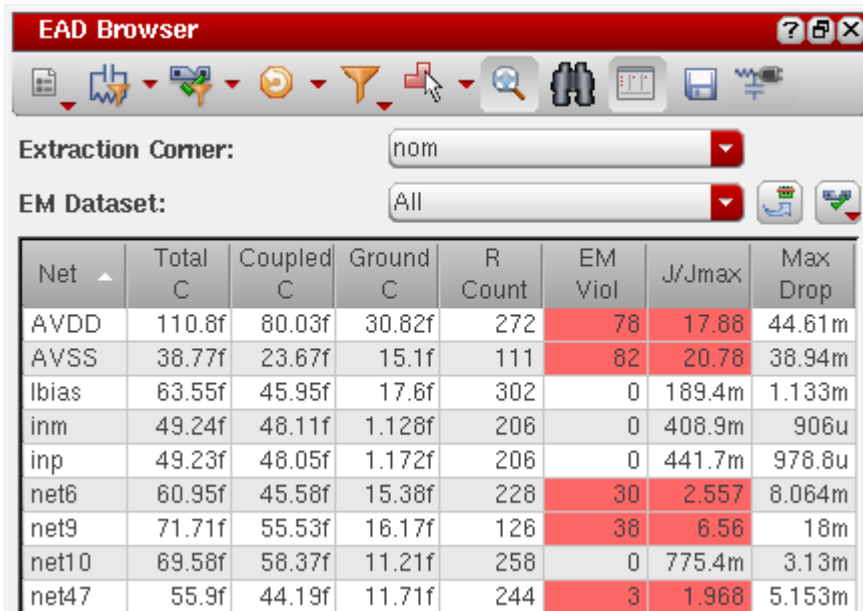
Net	Total C	Coupled C	Ground C	R Count	EM Viol	J/Jmax	Max Drop
AVDD	110.8f	80.03f	30.82f	272			
AVSS	38.77f	23.67f	15.1f	111			
Ibias	63.55f	45.95f	17.6f	302			
inm	49.24f	48.11f	1.128f	206			
inp	49.23f	48.05f	1.172f	206			
net6	60.95f	45.58f	15.38f	228			
net9	71.71f	55.53f	16.17f	126			
net10	69.58f	58.37f	11.21f	258			
net47	55.9f	44.19f	11.71f	244			

If electrical constraints, such as the Max Capacitance constraint or the Max Resistance constraint, are applied on nets, and the extracted parasitic data violated the constraints, the corresponding cells are highlighted in red. For more details, refer to the [Detail Pane](#).

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

When you have run an EM check, the summary table updates to show color-coded electromigration information for the listed nets (only some of this information is shown below).



The screenshot shows the EAD Browser window with a toolbar and two dropdown menus. The 'Extraction Corner' is set to 'nom' and the 'EM Dataset' is set to 'All'. Below these is a table with 8 columns: Net, Total C, Coupled C, Ground C, R Count, EM Viol, J/Jmax, and Max Drop. The table lists data for nets AVDD, AVSS, lbias, inm, inp, net6, net9, net10, and net47. The 'EM Viol' and 'J/Jmax' columns are color-coded: red for violations and yellow for skipped checks.

Net	Total C	Coupled C	Ground C	R Count	EM Viol	J/Jmax	Max Drop
AVDD	110.8f	80.03f	30.82f	272	78	17.88	44.61m
AVSS	38.77f	23.67f	15.1f	111	82	20.78	38.94m
lbias	63.55f	45.95f	17.6f	302	0	189.4m	1.133m
inm	49.24f	48.11f	1.128f	206	0	408.9m	906u
inp	49.23f	48.05f	1.172f	206	0	441.7m	978.8u
net6	60.95f	45.58f	15.38f	228	30	2.557	8.064m
net9	71.71f	55.53f	16.17f	126	38	6.56	18m
net10	69.58f	58.37f	11.21f	258	0	775.4m	3.13m
net47	55.9f	44.19f	11.71f	244	3	1.968	5.153m

The indication of the different color codes is as given below.

- Red indicates that EM violations were detected
- Yellow indicates that the EM check was skipped for the net

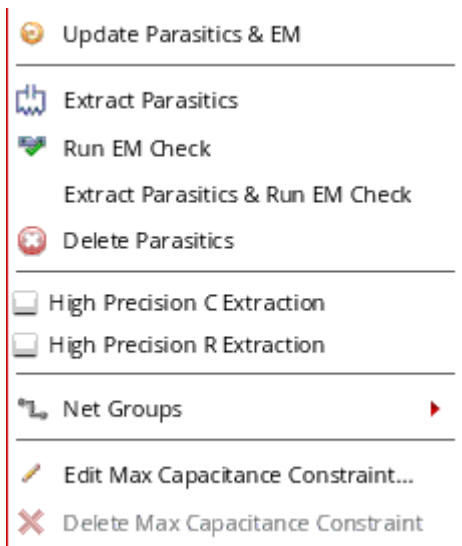
You can find out more about any individual value by clicking one of the table cells to open the *EM* tab in the Detail Pane.

You can right-click in one of the cells to open the context menu, which provides a quick access to commands to update the parasitic and EM data, extract parasitics, and run the EM check for the selected net. For example, to delete parasitics for selected nets only, select one or

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

more nets, right-click and choose *Delete Parasitics*. The context menu for a net in the summary pane is given below.



Important Points to Note

- By default, EAD shows the nets in your design from the perspective of the top-level cell of the window. When you edit a cell in place, the list of nets shown in the summary table does not change, unless the cell you are editing in place has been edited and saved in another session since it was loaded into the current session. If nets were added or deleted in that other session, then those changes are reflected when the edit-in-place occurs in the current session.

The display of nets in the summary pane also depends on the setting of the extractNetsWithConstraints environment variable, which controls the display of nets based on the EAD constraints applied to them. If this variable is set to `t`, only those nets that have EAD constraints applied on them are displayed in the summary table.

- By default, the tool performs a fast C and fast R extraction which is based on the models in the technology file, but the extracted data might not meet high accuracy levels. Therefore, to extract capacitance and resistance on critical nets with a higher degree of accuracy, you can selectively enable high precision C and R solvers. This creates a corresponding High Precision C Extraction and High Precision R Extraction constraint based on the options specified on *Extraction* tab of EAD Options form. It is also recommended to use high precision C extraction when the layout contains non-orthogonal nets. When you turn off high precision extraction, these constraints are deleted.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- The `excludePolyDiffusionCap` and `includePolyFringingCap` environment variables can be used to control the inclusion of both gate-to-diffusion overlap and fringing capacitance.
- You can also use the features in the EAD Toolbar and table headers to change the way data is displayed in the summary pane. See [Changing Data Display in the EAD Browser](#) for more information.

Detail Pane

Clicking a value in the summary table opens the appropriate tab in the detail pane, which shows a breakdown of the resistance and capacitance values for each of the selected nets. Information is organized in five tabs; one each for [capacitance](#), [resistance](#), [comparisons](#), and [EM checks](#), and [IR Drop](#).

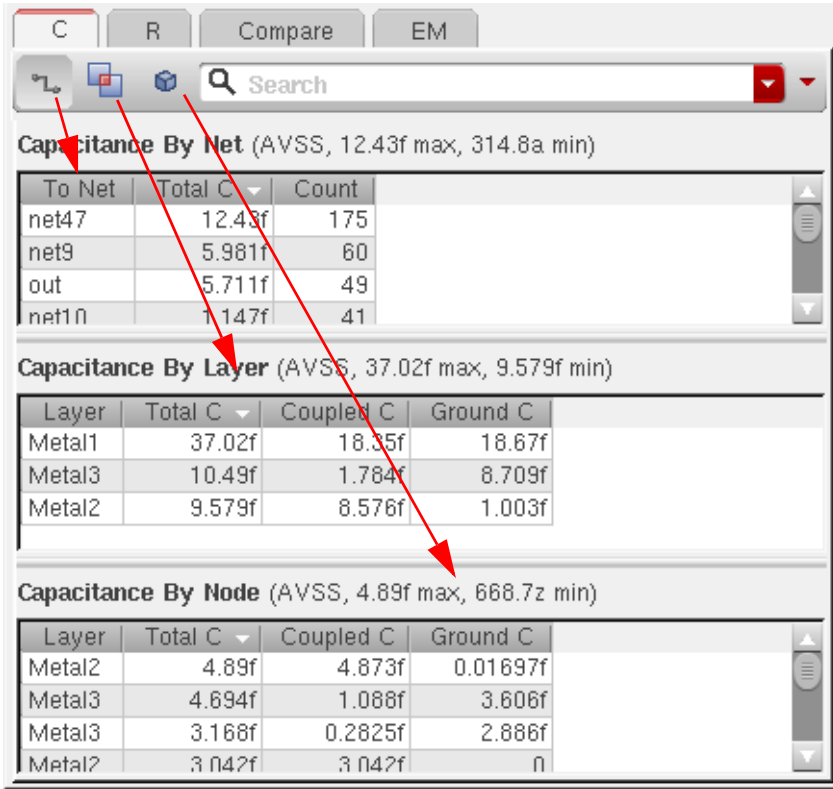
The *C* and *R* tabs are enabled based on the extraction mode set in the [Extraction](#) form. For example, if you are performing only C extraction, the *R* tab is disabled. Similarly, if you disable EM checking using the *Disable EM Checker* command, the *EM* tab is also disabled.

Note: The display of the detailed parasitic tables in the details pane depends on the value set for the `detailedParasiticTables` environment variable or the *Detailed Parasitic Tables* check box on the Environment tab of the EAD Options form.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- The **C** tab shows capacitance information by net, by layer, and by node.



The screenshot shows the 'C' tab in the Virtuoso interface. It contains three tables: 'Capacitance By Net', 'Capacitance By Layer', and 'Capacitance By Node'. Red arrows point to the 'C' tab, the 'Capacitance By Net' table, and a value in the 'Capacitance By Node' table.

Capacitance By Net (AVSS, 12.43f max, 314.8a min)

To Net	Total C	Count
net47	12.43f	175
net9	5.981f	60
out	5.711f	49
net10	1.147f	41

Capacitance By Layer (AVSS, 37.02f max, 9.579f min)

Layer	Total C	Coupled C	Ground C
Metal1	37.02f	18.35f	18.67f
Metal3	10.49f	1.784f	8.709f
Metal2	9.579f	8.576f	1.003f

Capacitance By Node (AVSS, 4.89f max, 668.7z min)

Layer	Total C	Coupled C	Ground C
Metal2	4.89f	4.873f	0.01697f
Metal3	4.694f	1.088f	3.606f
Metal3	3.168f	0.2825f	2.886f
Metal2	3.042f	3.042f	0

When you select one or more values in the *Capacitance By Node* table, the corresponding nodes are highlighted in the layout canvas.

Note: By default, the rows are displayed in the decreasing order of values in the *Total C* column of a table. However, if constraints are applied to check capacitance values on nets, the nets with violating constraints are displayed on top to bring them in focus, as

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

shown below.

EAD Browser

Extraction Corner: **nom_rcx**

Electrical Dataset: **All**

Net	Total C	Coupled C	Ground C	R Count
AVDD	21.45f	11.32f	10.13f	122
AVSS	10.43f	2.42f	8.011f	83
Ibias	1.812f	0.3164f	1.496f	6
inm	1.636f	0.8748f	0.761f	38
inp	2.226f	1.111f	1.115f	30
net14	9.65f	2.943f	6.707f	207
net17	2.745f	0.8083f	1.937f	36
net019	4.313f	0.8344f	3.479f	41
net021	5.921f	2.875f	3.046f	79

Capacitance By Net (inm, 0.3997f max, 9.024e-06f min)

To Net	Total C	Count
outm	0.2408f	2
net17	0.3997f	18
AVSS	0.1152f	6

Coupling Caps (inm, 0.8748f coupled, 0.761f ground)

C	Layer	To Net	To Layer
0.2283f	Metal1	outm	Metal1
0.115f	Metal1	AVSS	Metal1
0.0793f	Metal1	net14	Metal1

Constraint Manager

Show: **All**

Type (167)

- Symmetry (2)
- Symmetry (1)
- Symmetry (2)
- Net Priority (2)
- High Precision C Extraction (2) 1
- High Precision C Extraction (1) 5
- High Precision R Extraction (1) 1
- Max Coupling Capacitance (2)
- Max Coupling Capacitance (2)
- Max Coupling Capacitance (2)**
 - inm
 - outm
- Modgen (28)
- Symmetry (2)
- Symmetry (2)
- Symmetry (2)
- Symmetry (2)
- Symmetry (2)

Constraint (1) **All Parameters**

Name	Constr_95
Owner	Two_Stage_Opamp.DiffOp
Enabled	true
Status	not checked
Notes	
Coupled Capacitance	100a

Search

Capacitance By Net (inm, 0.3997f max, 9.024e-06f min)

Coupling Caps (inm, 0.8748f coupled, 0.761f ground)

Navigator **Palette** **EAD Browser**

Callout: Nets that violate constraints are displayed on top

You can use the Search field to filter capacitance details for selected nets. Alternatively, you can use the filter command on the EAD Browser to view details for only selected nets, or the nets with constraints, or violated constraints. For more details on filters, refer to [Filtering Rows](#).

If *Automatically zoom display to selected network or parasitic* is switched on, the display automatically pans and zooms to the selected nodes.

- The **R** tab shows the terminal-to-terminal resistance paths for the net selected in the summary pane. For each resistance path, the total resistance value, R count, and the names of from and to terminals are displayed. Clicking on a row in the *Resistance*

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Paths table shows the list of resistances on this path in the *Resistances* table, as shown below.

The screenshot shows the EAD Browser interface with the **R** tab selected. The **Resistance Paths** table for net044 is displayed, showing a list of resistance paths with their total resistance, count, and source/destination nodes. The **Resistances** table for net044 is also shown, listing individual resistance values, layers, widths, lengths, and cut areas. A red circle highlights the value 4.747 in the Resistances table, and a red arrow points to the corresponding net044 in the layout canvas on the right.

Resistance Paths (net044, 2137 paths)

Total R	Count	From	To
92.29	44	M0.25:G	M0.20:G
92.36	49	M0.25:G	M0.19:G
92.43	52	M0.25:G	M0.18:G
92.5	55	M0.25:G	M0.17:G

Resistances (net044, 258 values, 899.7 max, 100.5m min)

R	Layer	W	L	Cuts	CutArea
7.865	Metal1	0.0135	1.45		
4.747	Metal2	0.045	3.525		
2.865	Metal1	0.2465	9.605		
2.507	Metal2	0.2495	10.35		
2.203	Metal1	0.248	7.425		
1.695	Metal2	0.2415	6.77		

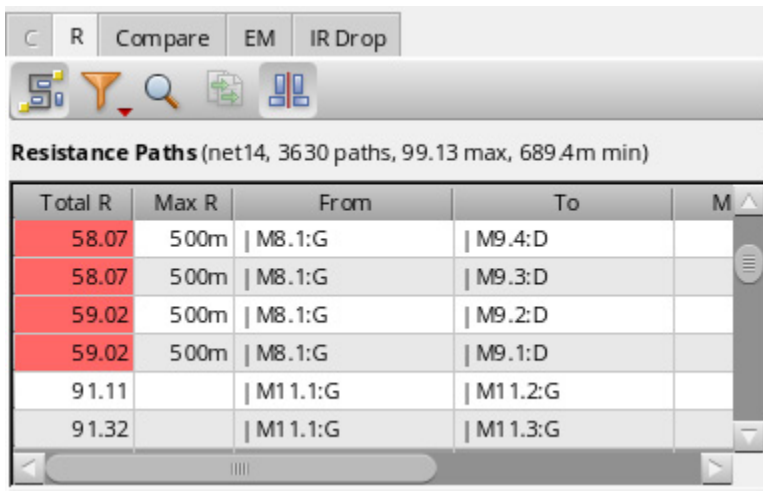
Important Points to Note

- If the *Detailed Parasitic Tables* check box on the *Environment* tab of the EAD Setup form is selected, by default, the *R* tab shows only the *Resistances* table. Click  on the toolbar of the *R* tab to show the *Resistance Paths* table that lists all the resistance paths for a net.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- ❑ If the number of resistors on a particular net exceeds the value specified by the `resPathsResistorLimit` environment variable, EAD resolves only a limited number of resistance paths. For the remaining paths, the *Total R* and *Count* columns show `Click to Resolve`. You need to resolve the remaining resistance paths manually.
- ❑ When you select a net in the summary pane and the *Resistance Paths* table is displayed, EAD looks for the resistance paths for that net. If resistance paths are found for the net, it displays all the available paths in the table. After that, it calculates resistances and displays the values in the *Total R* and *Count* columns.
- ❑ If the Max Resistance constraint is applied between two terminals or instance terminals on the selected net, EAD verifies the constraint. It highlights the resistance paths where the total R between those terminals or instance terminals exceeds the maximum resistance value specified by the constraint. All the paths violating the Max Resistance constraint are displayed on top of the table, as shown below.



Resistance Paths (net14, 3630 paths, 99.13 max, 689.4m min)

Total R	Max R	From	To	M
58.07	500m	M8.1:G	M9.4:D	
58.07	500m	M8.1:G	M9.3:D	
59.02	500m	M8.1:G	M9.2:D	
59.02	500m	M8.1:G	M9.1:D	
91.11		M11.1:G	M11.2:G	
91.32		M11.1:G	M11.3:G	

If the Max Resistance constraint is applied on a single member (terminal or instance terminal), EAD finds paths to all other terminals or instance terminals on that net, and highlights those paths where total R exceeds the maximum resistance value.

Note: If you change the members of the Max Resistance constraint after extracting the parasitics, it is recommended to re-extract the parasitics to view the correct violations.

- ❑ If the Max Voltage Drop constraint is applied on a single member (terminal or instance terminal), EAD finds paths to all other terminals or instance terminals on that net, and highlights those paths where total voltage drop exceeds the maximum voltage drop value.
- ❑ If you want to view the maximum possible current on each resistance path in the *Resistance Paths* table, you can show the hidden *Max Current* column by

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

selecting the name of this column in the context-sensitive menu of the column headings. This column shows the maximum current limit possible for a resistance path over all active datasets. The tooltip of the column shows the name of the dataset for which this maximum current limit is calculated.

The command to display the *Max Current* column is not available for the nets with high-precision resistance enabled.

Note: If you add, delete, enable, or disable the EM checks to be run, the maximum current values are deleted from the *Max Current* column. Re-extract the parasitics to view the maximum current values corresponding to the selected set of EM checks.

- ❑ For a large net with many terminals, the number of resistance paths could go into thousands or more, and the resolution all the resistance paths can be time consuming. In such cases, the tool might become unresponsive until this is done. To avoid this, EAD allocates only a limited time to this activity of calculating the resistance paths. This time limit is specified by the resistancePathsTableTimeout environment variable. By default, this variable is set to 2 (seconds), but you can reset it to the duration you want the tool to spend on reading the resistance paths in your designs. If you set this variable to 0, the time limit is removed.

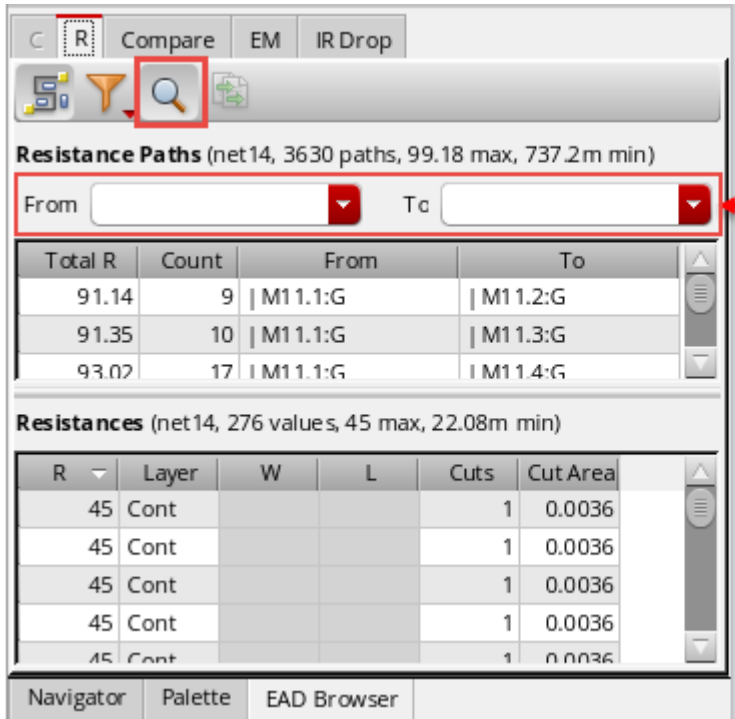
If the resistance paths search process is taking a long time, EAD displays the progress on the status bar of the Virtuoso Layout EAD window. If required, you can interrupt the process at any time by pressing `Ctrl+C`.

- ❑ It is possible that for large designs, EAD does not identify all the resistance paths for the selected net in the allocated time. If you do not see the path of your interest listed in the Resistance Paths table, you can search for it using the search feature. For this,

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

click  on the toolbar of the *R* tab. The *From* and *To* fields are displayed at the top of the table, as shown below.



Resistance Paths (net14, 3630 paths, 99.18 max, 737.2m min)

From	To

Resistances (net14, 276 values, 45 max, 22.08m min)

R	Layer	W	L	Cuts	Cut Area
45	Cont			1	0.0036
45	Cont			1	0.0036
45	Cont			1	0.0036
45	Cont			1	0.0036
45	Cont			1	0.0036

Navigator | Palette | EAD Browser

Fields to enable search of R paths

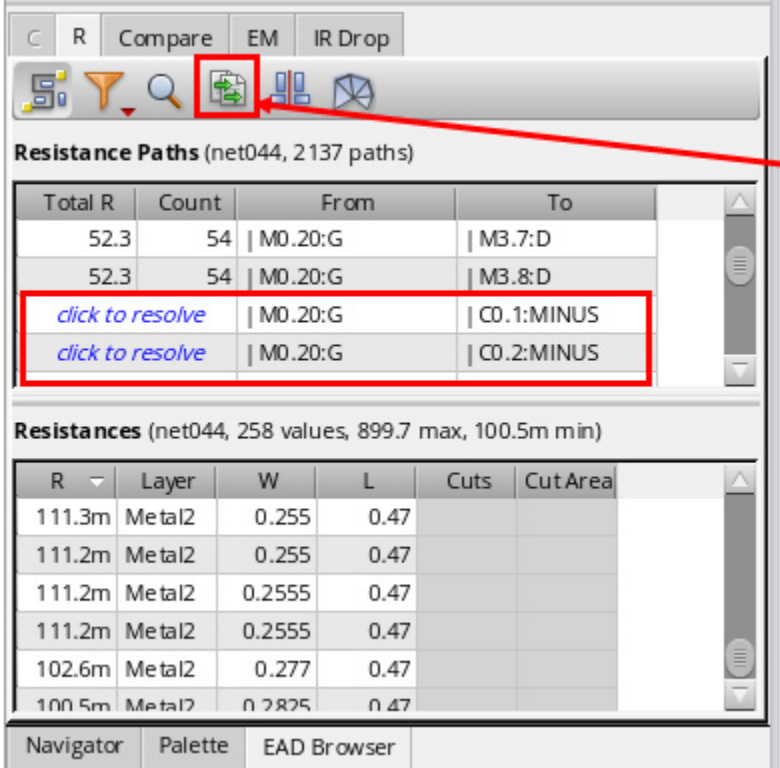
Depending on your requirement, enter the name of instance terminal in the *From* and *To* fields. You can use wildcard characters to specify the instance terminal names in the these fields. When both from and to terminal instance names are given, the tool searches for the resistance paths between them. However, if either of these values is given, the paths from or to the specified instance terminals are displayed.

- ❑ If all the resistance paths are not resolved within the time limit set by resistancePathsTableTimeout, the *Resistance Paths* table shows only the from

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

and to instance terminal names. The *Total R* and *Count* columns show [click to resolve](#), as shown below.



Resistance Paths (net044, 2137 paths)

Total R	Count	From	To
52.3	54	M0.20:G	M3.7:D
52.3	54	M0.20:G	M3.8:D
click to resolve		M0.20:G	C0.1:MINUS
click to resolve		M0.20:G	C0.2:MINUS

Resistances (net044, 258 values, 899.7 max, 100.5m min)

R	Layer	W	L	Cuts	CutArea
111.3m	Metal2	0.255	0.47		
111.2m	Metal2	0.255	0.47		
111.2m	Metal2	0.2555	0.47		
111.2m	Metal2	0.2555	0.47		
102.6m	Metal2	0.277	0.47		
100.5m	Metal2	0.2825	0.47		

Navigator | Palette | EAD Browser

Resolve resistance paths

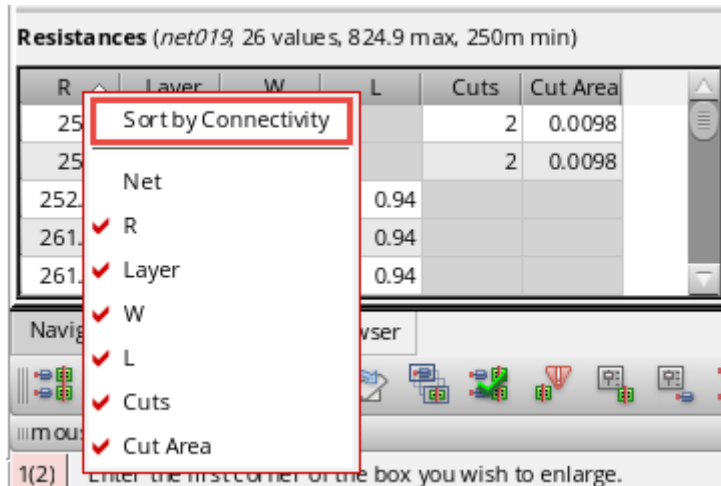
In this case, there are two ways to resolve resistances:

- ☐ Click *Resolve Resistance Paths* () on the toolbar to resolve all the incomplete paths.
- ☐ Click *click to resolve* to calculate the resistance for the selected resistance paths of your interest.
- ☐ Paths with constraints are given higher priority. They are always solved and displayed at the top.
- ☐ By default, the resistance values in the *R* column of the *Resistances* table are displayed in the order in which they are connected in the resistance path. You can display the rows as per the ascending or descending order of the values displayed

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

in any column by clicking the corresponding column header. To restore the default order, right-click a column header and choose the *Sort By Connectivity* command.



- Use the options of the *Select which layers are displayed in the table* command on the toolbar to show or hide rows for selected layers. This command is visible when either the detailedParasiticTables environment variable is set to `t` or the *Detailed Parasitic Tables* check box is selected on the Environment tab of the EAD Options form.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- The **Compare** tab shows the maximum difference and the % error between the values on two or more selected nets. Clicking on a net in the *Coupling Capacitance by Net* table highlights the net in the layout canvas.

The screenshot shows the 'Compare' tab in the Virtuoso EDA tool. It contains three tables and a layout view.

Parasitics Summary

	inm	lbias	AVSS	Max Diff	% Error
Total C	45.58f	59.11f	57.09f	13.53f	29.68%
Coupled C	29.76f	44.86f	28.71f	16.16f	56.28%
Ground C	15.82f	14.25f	28.39f	14.14f	99.24%
R Chunt	38	431	191	419	2153.77%

Coupling Capacitance by Net

Net	inm	lbias	AVSS	Max Diff	% Error
AVDD		25.88f	0.3148f	25.57f	8120.90%
AVSS	0.979f				
inm			0.979f		
inp	0.002882f		1.073f	1.07f	37117.81%

Total Capacitance by Layer

Layer	inm	lbias
Metal1	1.179f	7.235f
Metal2	0.4653f	3.655f
Metal3		
Poly	43.94f	48.22f

The layout view on the right shows a circuit board design with various components and traces. A red box highlights a specific area, and red arrows point from the tables to this area, indicating the net being compared.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

For more information on the *C*, *R*, and *Compare* tabs, see [Viewing and Analyzing Parasitics](#) and [Comparing Parasitic Values for Multiple Nets](#).

- The **EM** tab shows any electromigration violations present for the net selected in the summary pane. The color coding is controlled by *EM Violation Highlighting* settings in the [Extraction](#) tab. If there are no violations for the selected net, the table is empty.

Note: By default, EAD displays only those resistors with violations in the *EM* tab (i.e., those that have a ratio of J/J_{max} greater than 1 or the minimum violation level whichever is lower). To see **all** resistors, set the `emResultsShowAllResistors` environment variable to `t`.

The screenshot shows the EM Results table with the following annotations:

- Select the result type to be shown for each EM check:** Points to the 'Worst-case' dropdown menu.
- Select which layers are shown in the table:** Points to the 'Layer' column header.
- Set the highlight transparency level:** Points to the transparency slider icon.
- Shows All resistors:** Points to the 'Show All resistors' button.
- Show legends table:** Points to the 'Show legends table' button.
- Edit currents:** Points to the 'Edit currents' button.
- Export EM results to a CSV File:** Points to the 'Export EM results to a CSV File' button.
- Highlight EM violations (Overlay current levels onto the layout):** Points to the 'Highlight EM violations' button.

R	J/J _{max} Worstcase	CUSTOM	CUSTOM	max DC Op Current EMData	J/J _{max} DC Op Current EMData	Layer	W	L	Cuts	Cut Area
329.6m	1.911	0	0	1.911	1.452	Metal2	0.4	2.2395		
141.9m	1.835	0	0	1.835	1.394	Metal2	0.4	0.94		
141.9m	1.682	0	0	1.682	1.278	Metal2	0.4	0.94		
141.9m	1.529	0	0	1.529	1.162	Metal2	0.4	0.94		
141.9m	1.376	0	0	1.376	1.045	Metal2	0.4	0.94		
1.378	1.27	0	0	1.27	964.7m	Metal3	0.6	13.69		
1.433	1.27	0	0	1.27	964.7m	Metal3	0.6	14.24		
141.9m	1.223	0	0	1.223	929.3m	Metal2	0.4	0.94		
141.9m	1.07	0	0	1.07	813.1m	Metal2	0.4	0.94		
382.9m	957.8m	0	0	936.2m	957.8m	Metal2	0.4	2.585		
41.67m	946m	0	0	946m	718.7m	Via2			12	0.058
41.67m	946m	0	0	946m	718.7m	Via2			12	0.058

Note: Click the *Edit currents* button to manually add currents to the *CUSTOM* dataset in cases where there is no current data available from ADE Assembler. You can also save the dataset with a user-defined name. See [Running EM Checks on a CUSTOM Dataset](#) for more information.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- ❑ Click the *Result type* command to control what values are displayed in the *EM* table.

<i>J/Jmax</i>	Shows the ratio of actual current to maximum current. A value greater than or equal to 1.0 is a violation.
<i>(J/Jmax)*W</i>	Shows the ratio of actual current to maximum current multiplied by the resistor width.
<i>(J/Jmax)*N</i>	Shows the ratio of actual current to maximum current multiplied by the number of existing via cuts. This provides a hint on the number of via cuts that could possibly fix the EM violation reported for the via resistor.
<i>Current</i>	Shows the current from simulation results
<i>Max Current</i>	Shows the maximum current that can flow through this resistor
<i>Percent of Max Current</i>	Shows the value as a percentage of the maximum current

- ❑ You can set the `emTableResultType` environment variable to set the value to be displayed by default.
- ❑ Click the *Select which layers are displayed in the table* command and select one of the following options to control which layers are shown in the *EM* table.
 - *Show All Layers*: To display all the layers.
 - *Hide All Layers*: To hide all the layers.
 - *Show Only Via Layers*: To display only the Via layers.
 - *Show Only Conductor Layers*: To display only the conductor layers.
- ❑ Choose a *Highlight* overlay to apply to current values for the selected nets.

An overlay color-codes the currents calculated for the selected nets based on the EM information available for that net and the violation highlighting colors defined in the legend table for the *EM* tab.

Use the *Highlight Transparency* slider to change the transparency of the violation highlighting in the canvas. For example, you might increase the transparency to make it easier to see the underlying geometry.

See [Highlighting and Fixing Violations in the Layout Window](#) for more information.

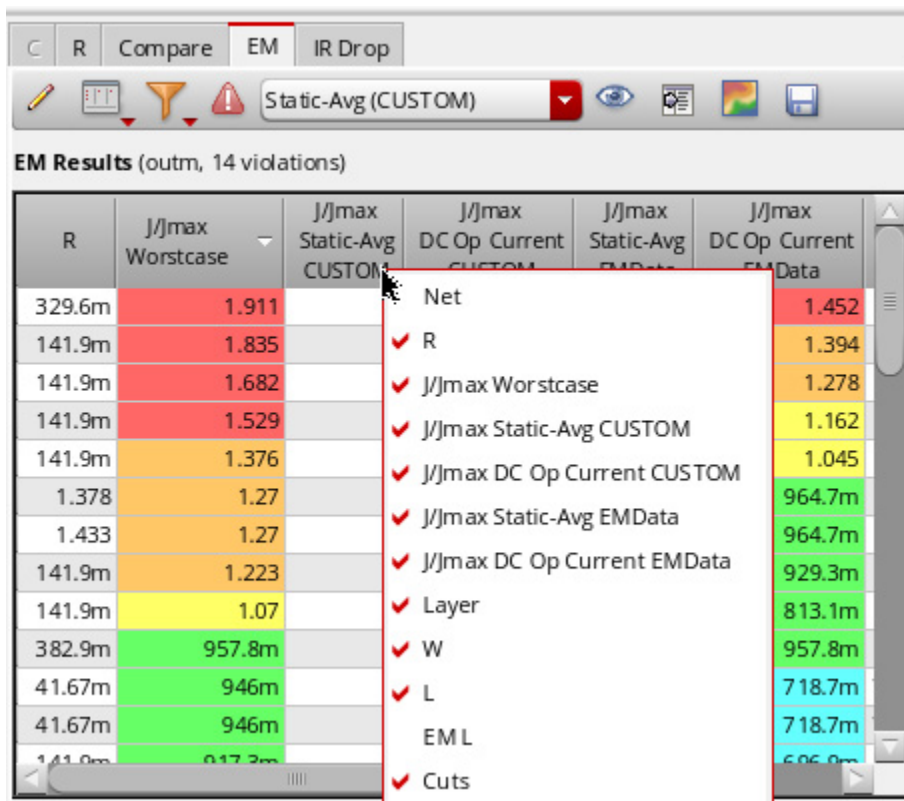
- ❑ Use the *Export Results to a CSV File* command to save the results from electromigration analysis for the selected net into a CSV file.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Note: This command saves all the EM related information. It is not limited to only the data displayed in the GUI. In order to save only the displayed information, switch to single-pane mode, keeping the *EM* tab active and then click *Export* on the EAD Browser toolbar.

- ❑ Change the columns shown in the EM results table by clicking the right mouse button in any column header, and selecting or clearing the check box next to the information you want to show or hide.

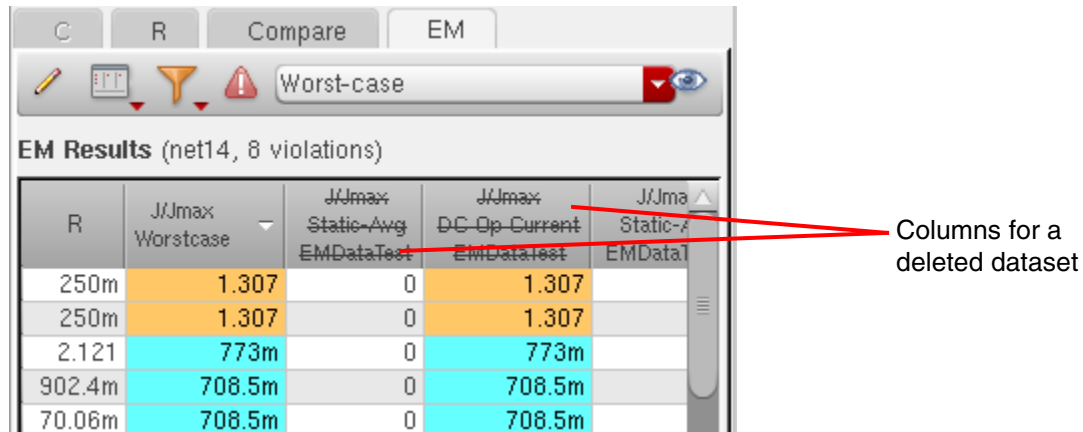


If any dataset has been deleted from the constraint view, the column header corresponding to that will appear in the ~~strikethrough~~ font style. You can remove

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

such datasets from this list by using the *Delete Electrical Data* command on the EAD toolbar.



- ❑ Move the pointer over any cell to see more information about the value shown in that cell.

J/Jmax DC Op Current	W	L	Blech L	Location
1.784	0.12	0.9325	19.465	48.1325:27.265 49.005:27.385
1.784	0.12	0.7925	13.315	43.03:23.025 43.8825:23.145
		0.53	13.315	44.615:23.025 44.735:23.675
		0.25	19.465	44.995:27.17 46.0975:27.29
		0.935	19.465	49.005:27.265 49.125:33.26
		0.325	13.315	44.615:23.555 44.9075:23.675
		0.325	19.465	47.14:27.265 48.1325:27.385
		0.025	19.465	46.0975:27.17 47.14:27.29
		0.325	13.315	44.9075:23.555 45.14:23.675

Check: DC Op Current
 EM Voltage Type: DC
 Dataset: dataset_0
 Net: net6
 J/Jmax: 1.784
 (J/Jmax)*W: 0.214
 Current: 337.7u
 Max current: 189.3u
 % of max: 178.4

For more information on using the *EM* tab, see [Viewing EM Information in the EAD Browser](#).

- The **IR Drop** tab shows the voltage drop details between different source and sink terminals connected to a net. The source and sink terminals are determined based on the direction of current for different terminals or instance terminals.

When you extract parasitics and run EM check, the maximum voltage drop value for each net is displayed in the *Max Drop* column of the summary pane. Click in a cell in this

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

column to view the IR Drop data across different parts of the selected net on the *IR Drop* tab in the details pane. The figure below shows an example of IR drop data for net14.

EAD Browser

Extraction Corner: nom_rcx

Electrical Dataset: All

Net	R Count	EM Viol	J/Jmax	Max Drop
AVDD	338	11	1.538	3.594m
AVSS	83	7	1.156	2.288m
Ibias	402	0	416.7m	543u
inm	32	0	0	0
inp	36			
net14	276	4	1.093	1.695m
net17	36	0	0	0

IR Drop

IR Drop Results (net14) - (From: |M6.1:D)

To	IR Drop Max IR Drop	IR Drop Static-Avg EMData	IR Drop DC Op Current EMData
M9.33:D	1.695m	1.695m	1.695m
M9.41:D	1.692m	1.692m	1.692m

Resistances

R	Layer	Static-Avg CUSTOM	DC Op Current CUSTOM	Static-Avg EMData	DC Op Current EMData
45	Cont	0	0	0	0
45	Cont	0	0	0	0
45	Cont	0	0	0	0

Move the mouse over any value to view the resistance path

The IR Drop tab shows two tables:

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- ❑ **IR Drop Results:** Shows for each dataset and current type the IR drop value between a pair of pins or instance terminals, or between a pin and an instance terminal. In this table, you can move the mouse over any value to highlight the resistance path on the layout canvas.
- ❑ **Resistances:** Shows IR drop for all resistor values for a path. When you click on a cell in the *Max Drop* column in the Summary pane, this table shows IR drop for all the resistances on the selected net. When you click on a particular row in the *IR Drop Results* table, this table shows IR drop for all the resistances between the instance terminals specified in that row. The header of the Resistances table is also changed to show the terminal names and total resistance between them, as shown in the figure below.

IR Drop Results (net14) - (From: |M6.1:D)

To	IR Drop Max IR Drop	IR Drop Static-Avg EMDData	IR Drop DC Op Current EMDData
M9.41:D	1.692m	1.692m	1.692m
M9.35:D	1.681m	1.681m	1.681m

Resistances (Total Resistance 17.34 between |M6.1:D and |M9.41:D)

R	Layer	IR Drop Static-Avg CUSTOM	IR Drop DC Op Current CUSTOM	IR Drop Static-Avg EMDData	IR Drop DC Op Current EMDData	W	L	Cuts	CutArea
1.602	Metal1	0	0	280.1u	280.1u	100m	2.505		
33.97m	Metal1	0	0	0	0	130m	60m		
250m	Via1	0	0	43.71u	43.71u			2	9.8
70.06m	Metal2	0	0	12.25u	12.25u	250m	290m		

The *SRP* (*Shortest Resistance Path*) icon on the *IR Drop* tab appears dim. This icon becomes available only when a resistance path is selected in the *Resistances* table.

You can click *SRP* to filter the *Resistances* table to selectively view only those resistors that are in the shortest resistance path.

Note: The *Resistances* table is visible only when the detailedParasiticTables environment variable is set to `t` or the *Detailed Parasitic Tables* check box on the EAD Options form is selected.

Note the following points for the *IR Drop Results* table:

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- IR drop from each available dataset is shown in a separate column, so the number of columns displayed may vary.
- When the *from* or *to* instance terminal or pin names are the same for all rows, the corresponding column is hidden and its static value is displayed on top of the table. For example, in the figure given above, the from instance terminal is the same for all rows, so it is displayed at the top (*From:* |M6:1:D). If required, you can show the *From* or *To* column by using the table header context menu.
- The *Max IR Drop* column shows the worst-case IR drop across all datasets for each pair. The rest of the columns show the IR drop data for each type of analysis from the selected datasets.
- By default, the results are displayed in decreasing order of max IR drop value across all datasets. However, you can sort the results in ascending or descending order of the values displayed in any column by clicking on the corresponding column header.
- Move the mouse over any path in the IR Drop table to highlight the path in the layout canvas.
- You can view the *From Location* and *To Location* columns to display the location co-ordinates of the *from* and *to* terminals of the resistance path, as well as the name of the layers enclosing them.
- These columns are hidden by default. Choose the *From Location* and *To Location* commands from the context-menu of the IR Drop table header to view these columns.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

When you select any cell in these two columns, EAD highlights the boundary of the enclosing conductor for the *from* or *to* terminal, as shown below.

The screenshot displays the EAD interface. On the left, the 'IR Drop Results' table for net14 is shown. The table has columns: From Location, To, To Location, and Drop. The 'To' column is highlighted, and a red arrow points from the selected cell 'M9.15:D' to the layout view on the right. The layout view shows a grid of conductors with a yellow circle highlighting the boundary of the conductor for the selected terminal.

Net	R Count	EM Viol	J/Jmax	Max Drop
AVDD	338	11	1.538	3.594m
AVSS	83	7	1.156	2.288m
Ibias	402	0	416.7m	543u
inm	32	0	0	0
inp	36			
net14	276	4	1.093	1.695m
net17	36	0	0	0
net019	39	0	946.5m	516u

From Location	To	To Location	Drop
(14.545:11.6),[Metal1]	M9.13:D	(6.385:16.5...	1.063m
(14.545:11.6),[Metal1]	M9.7:D	(7.925:15.5...	1.024m
(14.545:11.6),[Metal1]	M9.15:D	(7.925:16.5...	1.022m
(14.545:11.6),[Metal1]	M6.1:G	(14.11:11.5...	336u
(14.545:11.6),[Metal1]	M2.1:G	(14.255:8.7...	336u
(14.545:11.6),[Metal1]	M11.6:G	(31.625:8.7...	323.8u
(14.545:11.6),[Metal1]	M11.5:G	(30.755:8.7...	323.8u
(14.545:11.6),[Metal1]	M11.4:G	(29.885:8.7...	323.8u

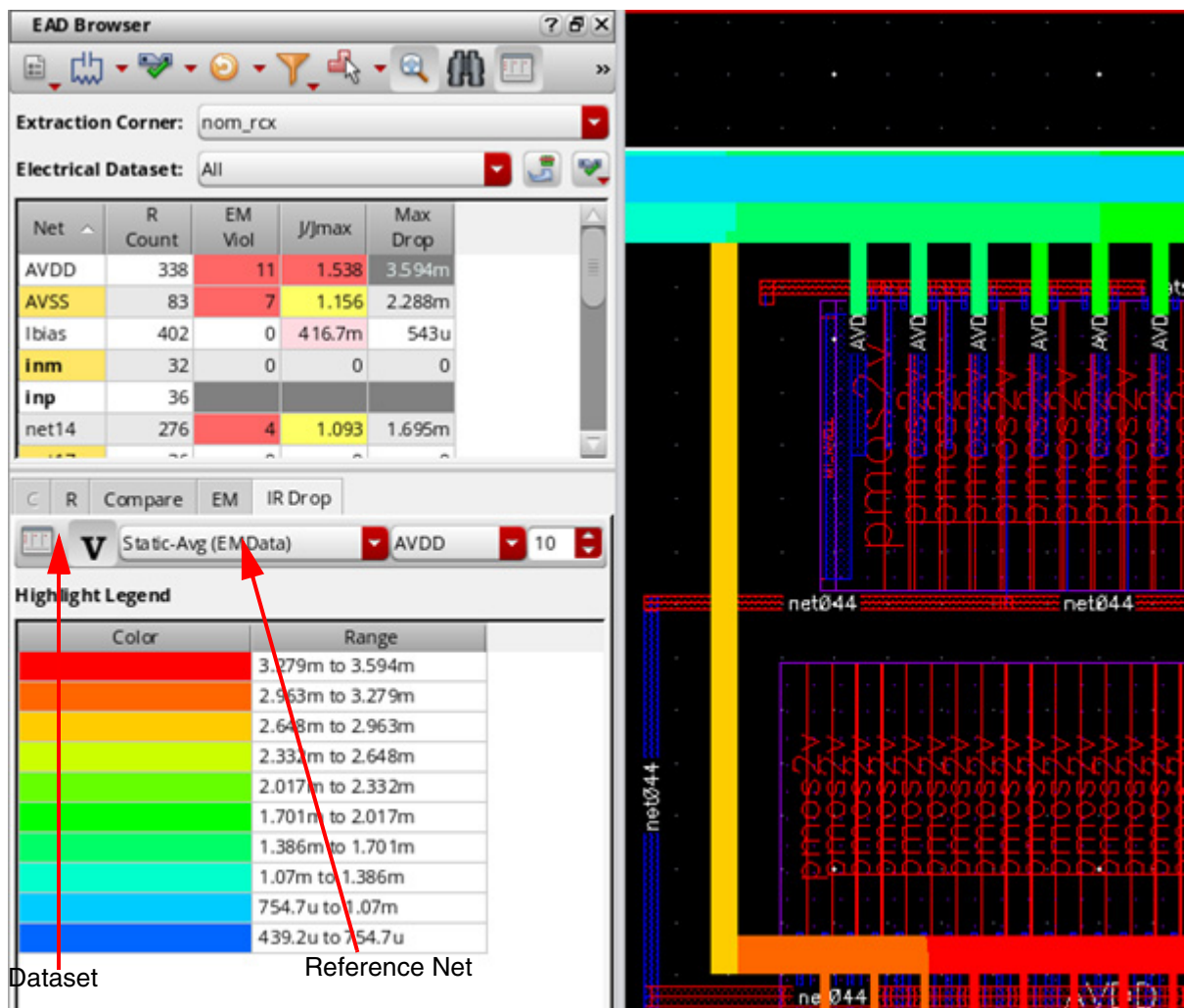
- You can search for a specific IR drop path on the net selected in the summary pane by using the *Find a specific IR Drop path* (🔍) command on the IR Drop toolbar. When you click this command button, the *Terminal Name* drop-down list is displayed on the *IR Drop* tab. This list shows the names of all the terminals on the selected net. Select a terminal from this list to view all the IR drop paths from or to that terminal.
- Use the *Toggle IR Drop Table* command on the toolbar of this tab to show or hide the *IR Drop Results* table.
- Use the options of the *Select which layers are displayed in the table* command on the toolbar to show or hide rows for selected layers. This command is visible when either the detailedParasiticTables environment

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

variable is set to t or the *Detailed Parasitic Tables* check box is selected on the Environment tab of the EAD Options form.

- When the IR Drop results are generated for Fast R extraction, you can overlay the IR drop data on the layout canvas. This gives a visual evaluation of voltage drop across the net in reference to the selected terminal for the given current type and dataset combination. To overlay the data use the *Highlight IR Drop on Canvas and Show Highlight Legend* () command on the toolbar of this tab. Layout EAD hides the tabular voltage drop data and displays a highlight legend. The color codes from this legend are used to show the heat map, that is, to overlay the voltage drop at different points on the selected net, as shown below.

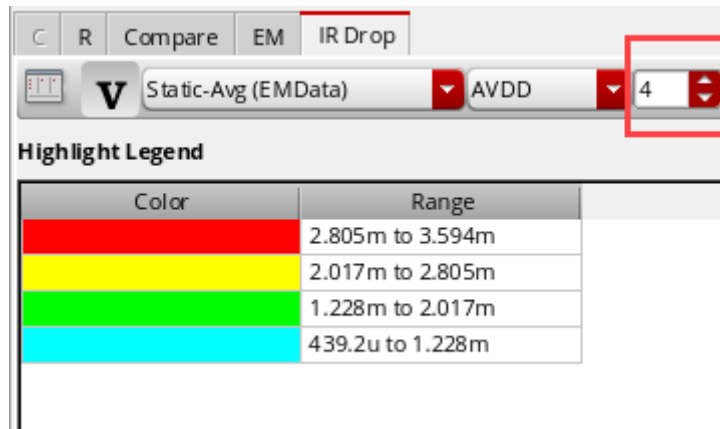


You can customize the numbers of bins of the range for the bins in which you need to categorize the IR drop data and view the heat map on the layout. For this, to edit the number of bins, edit the value in the *Number of bins* spin box.

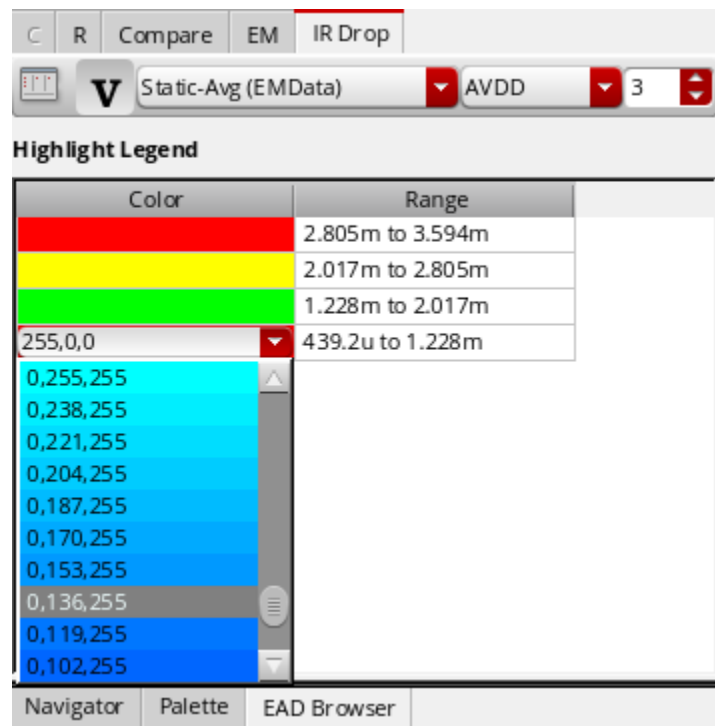
Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

The highlight legend is modified accordingly to reflect the required number of bins. For example, in the figure shown below, the data is categorized in only four ranges.



You can also change the color to be used for a particular range. For this, double-click a cell in the *Color* column of the *Highlight Legend* table, and choose a highlight color for the corresponding range, as shown below.



The overlay on the layout is also updated to reflect the change.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

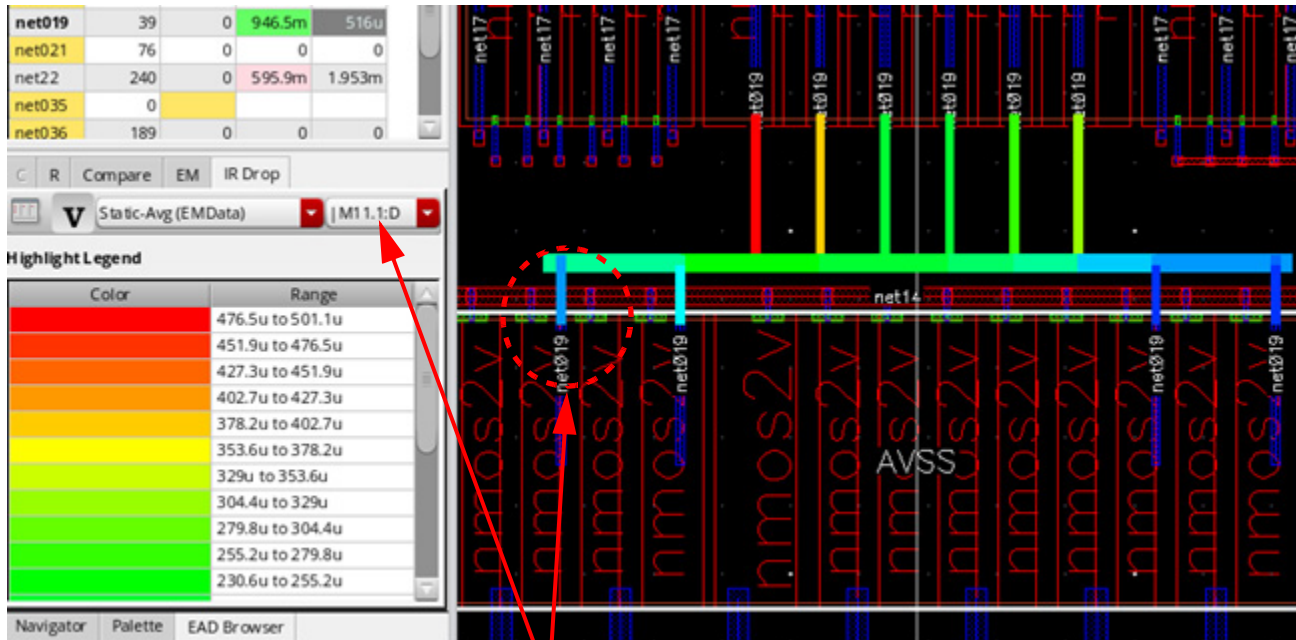
If multiple datasets are available, you can select any dataset for which you want to overlay the voltage drop information on canvas. You can also change the terminal to be used as reference for voltage drop calculation across the net. When you change the reference terminal or the dataset and current-type combination, the voltage drop values are recalculated and the overlay is changed. The highlight legend is also dynamically changed depending on the high and low range of voltage drop between the reference terminal and other terminals.

The following example shows the changes in IR drop overlay for `net19` according to the change in reference terminal.

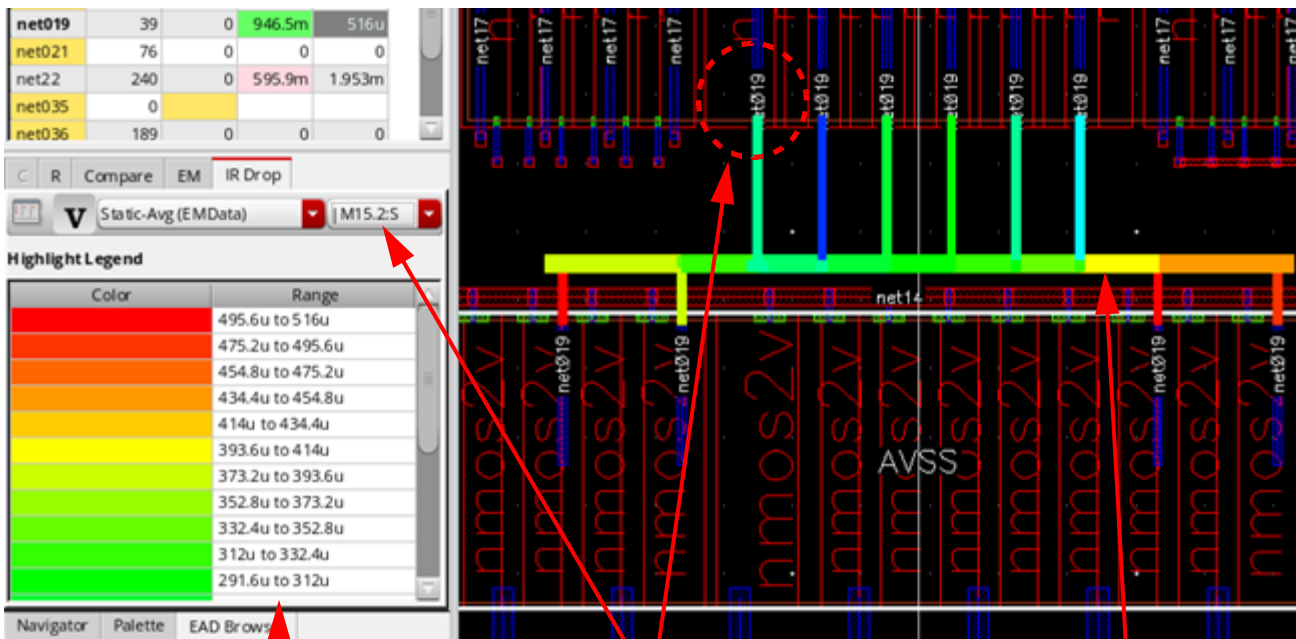
Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Change in IR drop overlay due to the change in reference terminal



M11.1:D as Reference



Note the change in range of voltage drop

M15.2:S as Reference terminal

Note the change in IR drop overlay

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- You can view the voltage drop overlay and tabular data at the same time. For this, click *Highlight IR Drop on Canvas and Show Highlight Legend* () on the toolbar to enable overlay, and then click *Show/Hide IR Drop Results Table* ()
- You can apply layer filters to view the voltage drop overlay on selected layers. For this, perform the following steps:

a. Set the `detailedParasiticTables` environment variable to `t`. This will display the *Selects which layers are displayed in the table* () icon on the toolbar of the *IR Drop* tab.

b. Set the `applyLayerFilterOnCanvasHighlight` environment variable to `t`.

c. Click *Highlight IR Drop on Canvas and Show Highlight Legend* () on the toolbar to enable overlay.

d. Click *Show/Hide IR Drop Results Table* () to view the other commands on the toolbar.

e. Click () and select a layer from the drop-down list of layer names.

The voltage drop overlay is displayed only for the selected layer. You can select multiple layer names to view the overlay for the together.

Note: The layer filter for voltage drop overlay can be applied only for conductor layers. It cannot be used for via layers.

- If the `Max Voltage Drop` constraint has been applied on the net selected in the summary pane and the maximum voltage drop value between any two nodes on that

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

net is more than the constraint value, the cell is highlighted in red to report a violation, as shown in the example below.

Extraction Corner: typical

Electrical Dataset: All

Net	R Count	EM Viol	J/Jmax	Max Drop
A	15	0	47.25m	2.032m
B	12	8	2.727	6.091m
C	45	28	22.83	46.98m
D	6	8	2.482	1.179m

Before selection, this cell was highlighted in red to indicate violation of constraint

Selected path highlighted on the layout canvas

IR Drop Results (C) - (From: |M0:D)

To	IR Drop Max IR Drop	IR Drop Static-Avg	IR Drop DC Op Current
M1:D	46.98m	46.98m	46.77m
C	25.14m	25.14m	25.03m

Highlight indicates violation of the Max Voltage Drop constraint

Constraint Manager

Show: All

Type (5)

route_em_test (5)

- Max Voltage...
- |M1:D
- |M0:D
- Alignment (2)
- Alignment (2)
- Alignment (2)
- Alignment (2)

Constraint (1) All Parameters

Name	Constr_0
Owner	t_testroute
Enabled	true
Status	not checked
Notes	
voltage	40m

Max Voltage Drop Constraint value

The rows with violations of the Max Voltage constraint are displayed on top, followed by other rows in the decreasing order of voltage drop.

Note: If you change the members of the Max Voltage constraint after extracting the parasitics, it is recommended to re-extract the parasitics to view the correct violations.

Also see: [Viewing IR Drop for high precision R extraction](#)

Virtuoso Electrically Aware Design Flow Guide

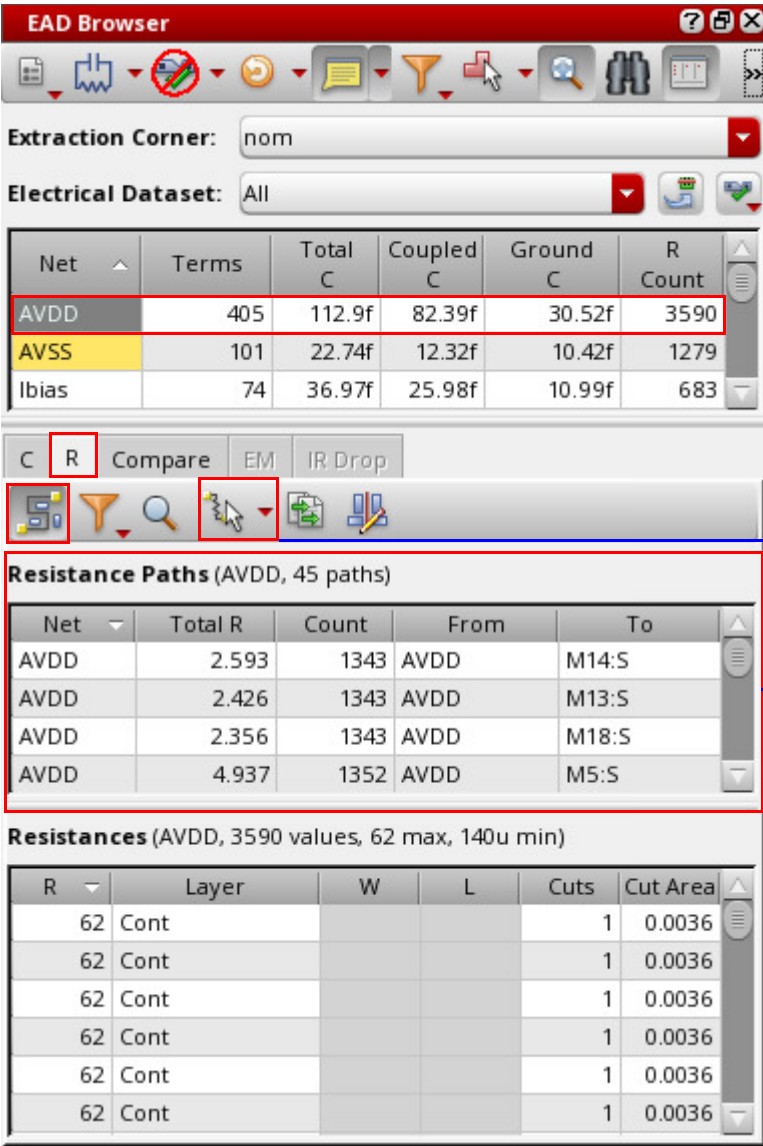
Extracting Parasitics in the Layout View

Viewing Resistance Paths between Terms

You can use the *Select pairs of resistance path terms* options to filter resistance paths between the selected terms in the *Resistance Paths* table.

To view resistance path between the selected terms:

1. Select a net in the *Summary* table and click the *R* tab.
2. Click the *Toggle resistance path table*  icon to display the *Resistance Paths* table.



The screenshot shows the EAD Browser interface. At the top, the 'Extraction Corner' is set to 'nom' and the 'Electrical Dataset' is 'All'. Below this is a table with columns: Net, Terms, Total C, Coupled C, Ground C, and R Count. The 'AVDD' row is selected, and the 'R' tab is active. Below the table, there are icons for 'C', 'R', 'Compare', 'EM', and 'IR Drop'. The 'R' icon is highlighted. Below the icons, the 'Resistance Paths (AVDD, 45 paths)' table is displayed. This table has columns: Net, Total R, Count, From, and To. The first four rows are highlighted. Below this table, the 'Resistances (AVDD, 3590 values, 62 max, 140u min)' table is displayed. This table has columns: R, Layer, W, L, Cuts, and Cut Area. The first six rows are highlighted. A red box highlights the 'R' tab and the 'Resistance Paths' table. A blue box highlights the 'Toggle resistance path table' icon. A blue line points from the text 'Select pairs of resistance path terms' to the 'Toggle resistance path table' icon. A blue line points from the text 'Resistance Paths table' to the 'Resistance Paths' table.

Net	Terms	Total C	Coupled C	Ground C	R Count
AVDD	405	112.9f	82.39f	30.52f	3590
AVSS	101	22.74f	12.32f	10.42f	1279
Ibias	74	36.97f	25.98f	10.99f	683

Net	Total R	Count	From	To
AVDD	2.593	1343	AVDD	M14:S
AVDD	2.426	1343	AVDD	M13:S
AVDD	2.356	1343	AVDD	M18:S
AVDD	4.937	1352	AVDD	M5:S

R	Layer	W	L	Cuts	Cut Area
62	Cont			1	0.0036
62	Cont			1	0.0036
62	Cont			1	0.0036
62	Cont			1	0.0036
62	Cont			1	0.0036
62	Cont			1	0.0036

3. Click the drop-down arrow in the *Select pairs of resistance path terms* icon and select one of the following options:

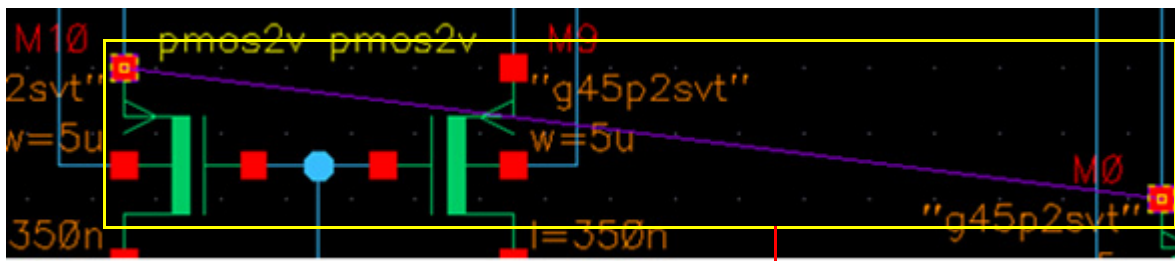
Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- ❑ **Select From-To Term Pairs:** To filter resistance paths between the *From* and *To* terms, first select the *From* term and then select the *To* term in the schematic.

The resistance path between the selected *From* and *To* terms is displayed in the *Resistance Paths* table and the path is highlighted in the schematic.

The following example shows the results when M10 : s and M0 : s are selected as the *From* and *To* terms in the schematic, respectively.



The resistance path between the From and To terms highlighted in the schematic

The Resistance Paths table is filtered to show the path between the selected From and To terms

Net	Total R	Count	From	To
AVDD	18.64	1377	M10:S	M0:S

You can now select different pair of *From* and *To* terms in the schematic to filter the resistance path in the *Resistance Paths* table and highlight the resistance path in the schematic. Note that when using the *Select From-To Term Pairs* option, both *From* and *To* terms are reselected. Press the *Esc* key to exit selection mode.

- ❑ **Select From Term and Multiple To Terms:** To filter resistance paths between a single *From* and multiple *To* terms, first select the *From* term and then select the *To* term in the schematic. If you again select a term in the schematic, only the *To* term changes. The first term you selected in the schematic remains selected as the *From* term. Press the *Esc* key to exit selection mode.
- ❑ **Select Path to Topmost Term:** To filter the resistance path from the selected instance term to the topmost term in the hierarchy. First, descend into a cell and then select the instance term that is connected to a term present in the higher level of hierarchy.

The resistance path between the selected instance term and the topmost term in the hierarchy is displayed in the *Resistance Paths* table and the path is highlighted in the schematic.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Changing Data Display in the EAD Browser

This section describes how to change the way data is displayed in the EAD Browser tables.

Toggling Dual Pane and Single Pane View

You use the *Toggle dual pane view* button on the EAD Browser toolbar to toggle between the default dual pane EAD Browser view and a single pane view.



Dual pane view is the default and features the summary table on top and the detail table underneath. Single pane view has all tabs listed in a single panel.

The screenshot shows the EAD Browser window in dual-pane view. The top section contains a summary table with columns: Net, Total C, Coupled C, Ground C, R Count, EM Viol, J/Jmax, and Max Drop. The bottom section contains a detail table titled 'EM Results (single_net, 23 violations)' with columns: R, J/Jmax Worstcase, J/Jmax Static-Avg CUSTOM, DC Op Current CUSTOM, J/Jmax Static-Avg dataset, and DC Op Current dataset. The 'EM Viol' column in the summary table is highlighted in red for the 'single...' row.

Net	Total C	Coupled C	Ground C	R Count	EM Viol	J/Jmax	Max Drop
IN	6.252f	1.525f	4.727f	43	0	38.64m	372.5u
cluste...	4.459f	2.358f	2.101f	30	42	435.7	290.3m
gndl...	4.758f	1.669f	3.089f	28	0	0	0
single...	10.33f	4.989f	5.338f	70	23	94.74	58.02m
vdd	7.597f	4.17f	3.427f	34	42	174.2	166.4m

R	J/Jmax Worstcase	J/Jmax Static-Avg CUSTOM	DC Op Current CUSTOM	J/Jmax Static-Avg dataset	DC Op Current dataset
1.47	94.74	0	0	94.74	16.18
2.352	37.9	0	0	37.9	3.235
2.352	37.9	0	0	37.9	3.235
248.2m	22.74	0	0	22.74	3.882
2.352	18.95	0	0	18.95	1.618
219.2m	11.37	0	0	11.37	970.6m
24.4m	11.37	0	0	11.37	970.6m
36m	11.37	0	0	11.37	1.941
83.6m	11.37	0	0	11.37	1.941
204.3m	10.42	0	0	10.42	889.7m
77.67m	10.42	0	0	10.42	889.7m

Dual pane view

The screenshot shows the EAD Browser window in single-pane view. The top section contains a summary table with columns: R, J/Jmax Worstcase, J/Jmax Static-Avg CUSTOM, DC Op Current CUSTOM, J/Jmax Static-Avg dataset, and DC Op Current dataset. The bottom section contains a detail table titled 'EM Results (single_net, 23 violations)' with columns: R, J/Jmax Worstcase, J/Jmax Static-Avg CUSTOM, DC Op Current CUSTOM, J/Jmax Static-Avg dataset, and DC Op Current dataset. The 'EM Viol' column in the summary table is highlighted in red for the 'single...' row.

R	J/Jmax Worstcase	J/Jmax Static-Avg CUSTOM	DC Op Current CUSTOM	J/Jmax Static-Avg dataset	DC Op Current dataset
1.47	94.74	0	0	94.74	16.18
2.352	37.9	0	0	37.9	3.235
2.352	37.9	0	0	37.9	3.235
248.2m	22.74	0	0	22.74	3.882
2.352	18.95	0	0	18.95	1.618
219.2m	11.37	0	0	11.37	970.6m
24.4m	11.37	0	0	11.37	970.6m
36m	11.37	0	0	11.37	1.941
83.6m	11.37	0	0	11.37	1.941
204.3m	10.42	0	0	10.42	889.7m
77.67m	10.42	0	0	10.42	889.7m

Single pane view

In dual pane view, when you click a value cell in the summary table, the relevant corresponding tab is opened in the detail table. For example, if you click the *EM Viol* cell for

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

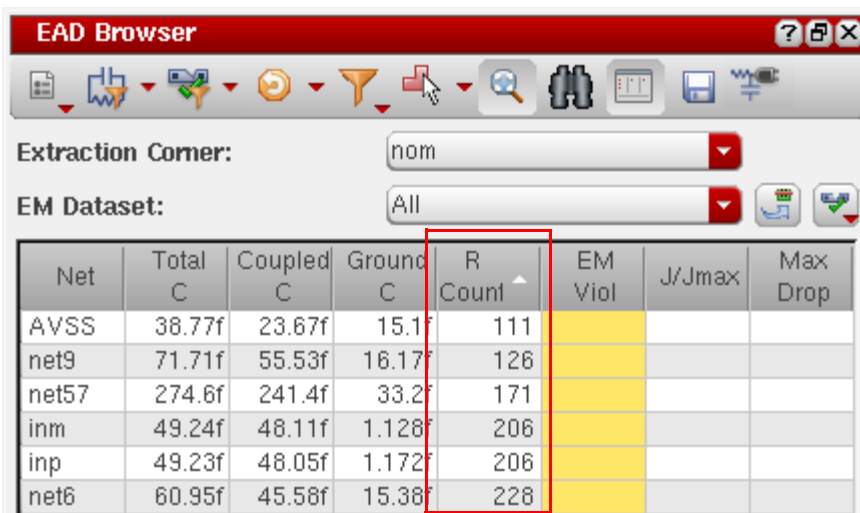
`vdd` in the summary table, the *EM* tab is opened in the detail table and the violation details for `vdd` are shown. Similarly, if you press the `Ctrl` key and select two nets in the summary table, the *Compare* tab is opened in the detail table.

Use single pane view when you need to focus on the results of a single net and need more space to view the results more comfortably.

Sorting Columns

You can change how data is sorted in columns and also in which position in the table a particular column appears.

1. Click once in a column header (for example *R Count*) to sort the column in ascending order.



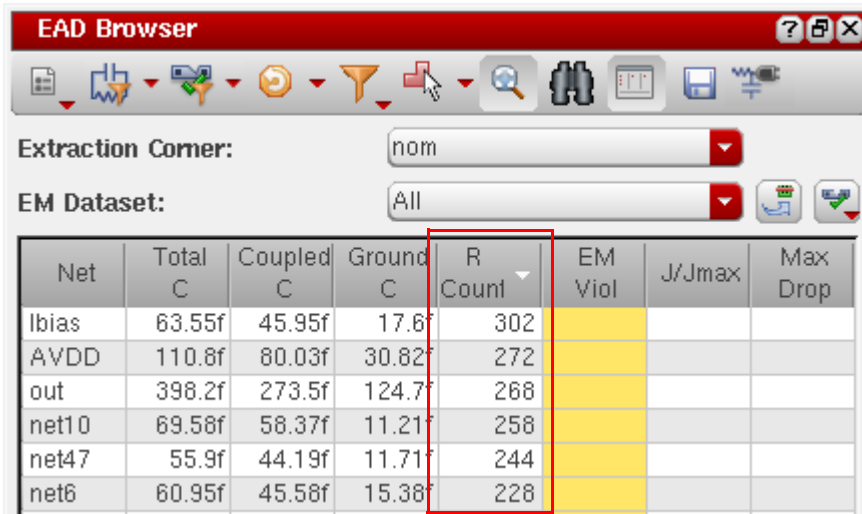
The screenshot shows the EAD Browser window with the following data:

Net	Total C	Coupled C	Ground C	R Count	EM Viol	J/Jmax	Max Drop
AVSS	38.77f	23.67f	15.17f	111			
net9	71.71f	55.53f	16.17f	126			
net57	274.6f	241.4f	33.27f	171			
inm	49.24f	48.11f	1.128f	206			
inp	49.23f	48.05f	1.172f	206			
net6	60.95f	45.58f	15.38f	228			

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- Click again to sort the column in descending order.



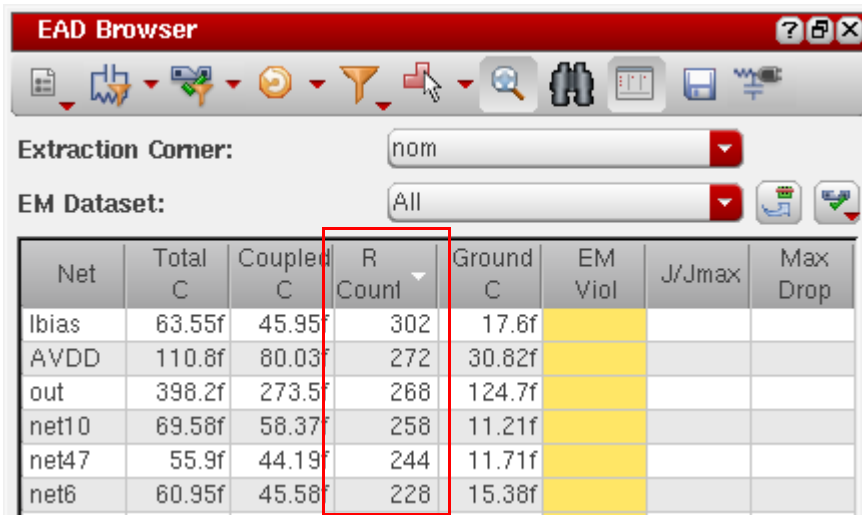
EAD Browser

Extraction Corner: nom

EM Dataset: All

Net	Total C	Coupled C	Ground C	R Count	EM Viol	J/Jmax	Max Drop
lbias	63.55f	45.95f	17.6f	302			
AVDD	110.8f	80.03f	30.82f	272			
out	398.2f	273.5f	124.7f	268			
net10	69.58f	58.37f	11.21f	258			
net47	55.9f	44.19f	11.71f	244			
net6	60.95f	45.58f	15.38f	228			

- Click-and-hold the column header and move your mouse to drag the column to a new location.
- Release the mouse button to drop the column at its new location.



EAD Browser

Extraction Corner: nom

EM Dataset: All

Net	Total C	Coupled C	R Count	Ground C	EM Viol	J/Jmax	Max Drop
lbias	63.55f	45.95f	302	17.6f			
AVDD	110.8f	80.03f	272	30.82f			
out	398.2f	273.5f	268	124.7f			
net10	69.58f	58.37f	258	11.21f			
net47	55.9f	44.19f	244	11.71f			
net6	60.95f	45.58f	228	15.38f			

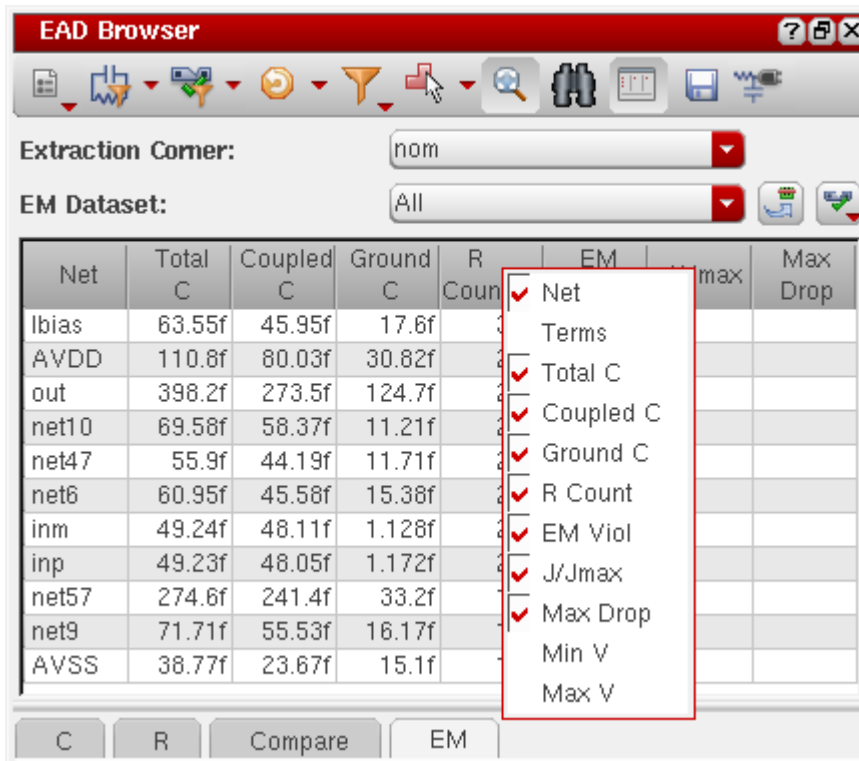
Showing and Hiding Columns

To specify which columns are visible in a table,

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

1. Right-click in any column header.



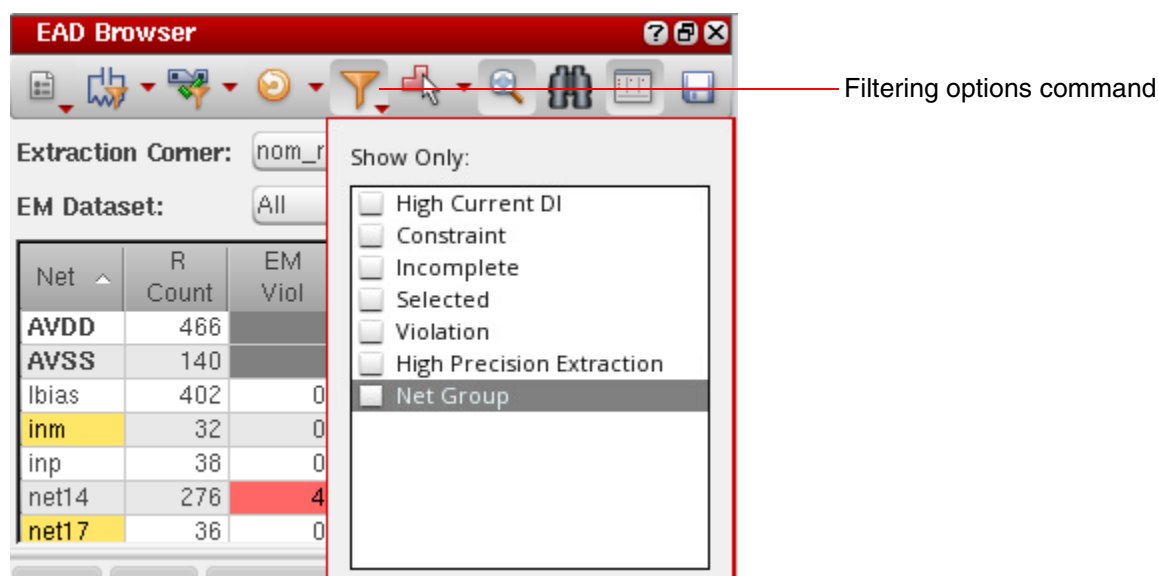
2. Select or deselect the check box next to the column you want to show or hide.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Filtering Rows

By default, all available nets are displayed in the summary pane. You can use the *Filtering options* to show only the nets for which you need to analyze parasitic details and EM violations.



Click *Filtering options* on the toolbar and select one or more of the following filtering options to show only those nets that meet the selected criteria.

- *High Current DI* shows information about the nets on which you have defined the HighCurrent design intent. Also see [Creating a HighCurrent Design Intent](#).
- *Constraint* shows information about the nets on which you have defined electrical constraints other than the High Current design intent. Also see [Extracting Parasitics in Layout EAD](#).
- *Incomplete* shows information only for the nets with incomplete parasitic network.
- *Selected* shows information about the nets selected on the layout canvas. When this filtering option is selected, click the nets in the layout canvas to show their details in the summary pane.
- *Violation* shows information about the nets that violate electrical constraints and/or EM limits.
- *High Precision Extraction* shows information about only those nets for which you have enabled *High Precision R Extraction* or *High Precision C Extraction*. Also see [Extracting Parasitics in Layout EAD](#).

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- *Net Group* shows information only about those nets that you have included in the specified net groups. When selecting this filter, you also need to select the net groups for which you need to view the information. Also see [Creating Net Groups](#).

The following example shows how the *Violation* filter is used to display the nets with violations.

The screenshot shows the EAD Browser interface. The 'Show Only' dropdown menu is open, and the 'Violation' option is selected. Below the menu, a table of net data is displayed. The table has columns: Net, R Count, EM Viol, J/Jmax, and Max Drop. The rows are: net14, AVSS, outp, AVDD, net044, and outm. The 'EM Viol' column is highlighted in red for all rows, indicating violations. A red arrow points from the 'Violation' filter selection to the table.

The Violation filter displays only those nets that show EM violations.

Net	R Count	EM Viol	J/Jmax	Max Drop
net14	312	4	1.093	1.703m
AVSS	89	7	1.156	2.288m
outp	125	2	1.159	3.523m
AVDD	340	11	1.538	3.596m
net044	293	9	1.19	3.911m
outm	126	14	1.911	4.355m

Creating Net Groups

You can create net groups to group similar types of nets together. This also helps in filtering nets that belong to a particular net group.

To create a net group, perform the following steps:

Virtuoso Electrically Aware Design Flow Guide

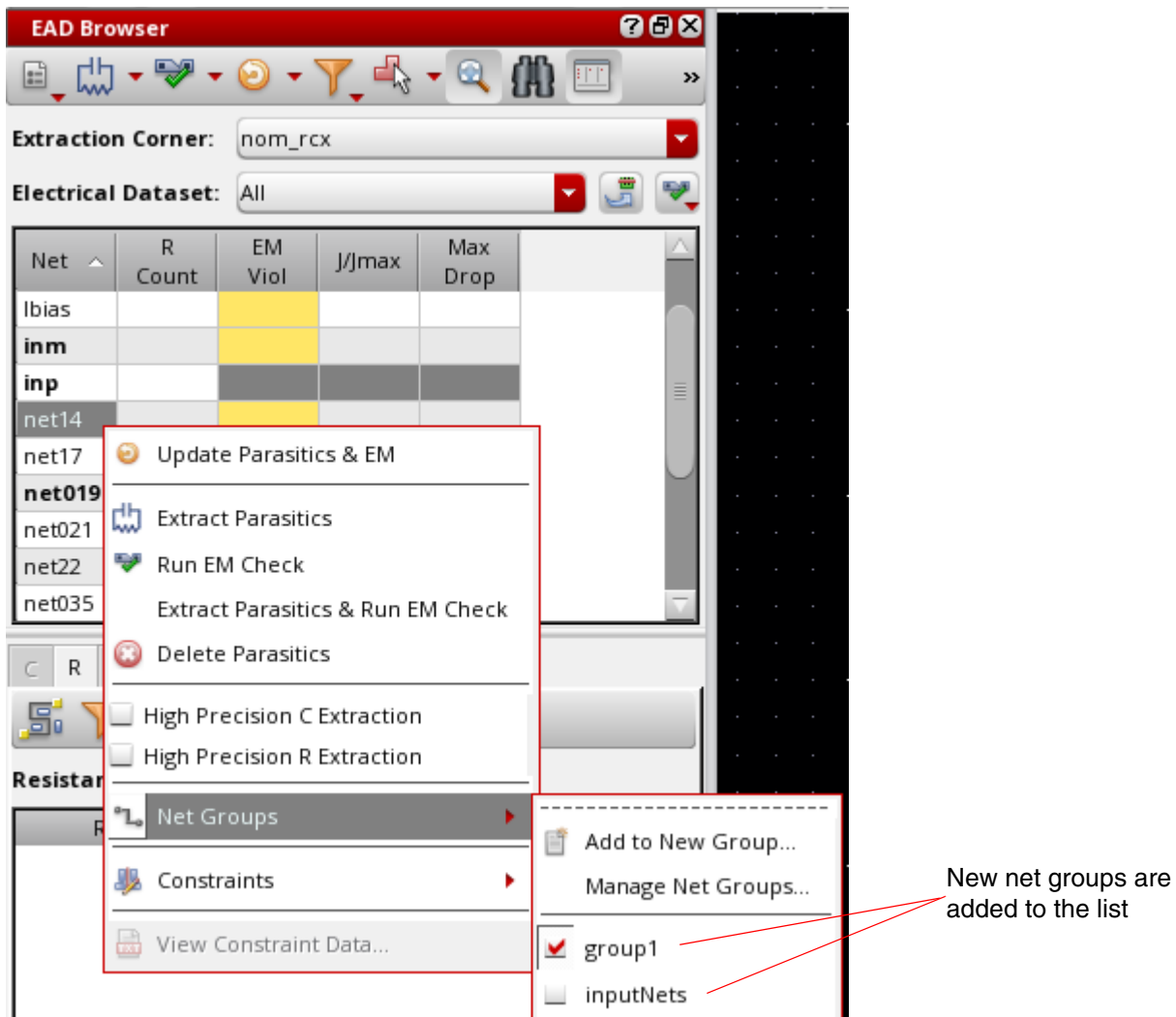
Extracting Parasitics in the Layout View

1. Right-click one or more nets in the summary pane of the EAD Browser, and choose *Net Groups – Add to New Group*.

The EAD – New Net Group form is displayed.

2. Enter a name for the new group in the *Net Group Name* field.
3. Click *OK* to close the form.

A new net group is created with the given name and the selected net(s) are added to the specified net group. The name of the net group is added to the *Net Groups* menu in the context-sensitive menu for the nets.



To add a net to an existing net group, select a net and choose *Net Groups – <name of the existing net group>*.

Note: A net can be included in more than one net group.

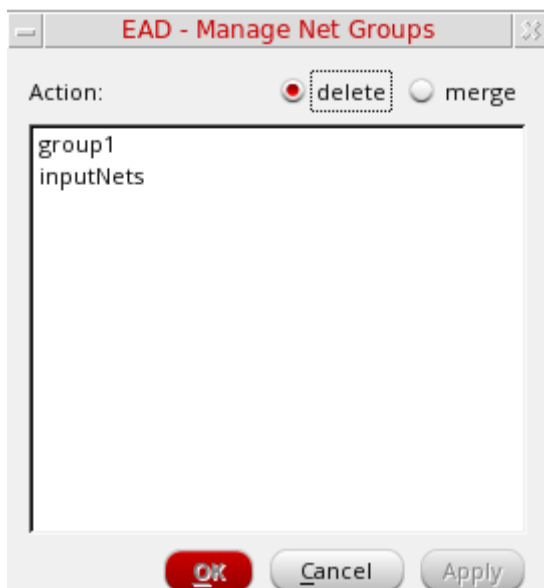
Managing Net Groups

You can delete or merge net groups by using the *Manage Net Groups* command.

Deleting Net Groups

To delete a net group, perform the following steps:

1. Right-click in the summary pane of the EAD Browser and choose *Manage Net Groups*. The EAD - Manage Net Groups form is displayed.



2. Select a group name from the list of groups.
3. Ensure that the *delete* action is selected.
4. Click *OK*.

The selected net group is deleted.

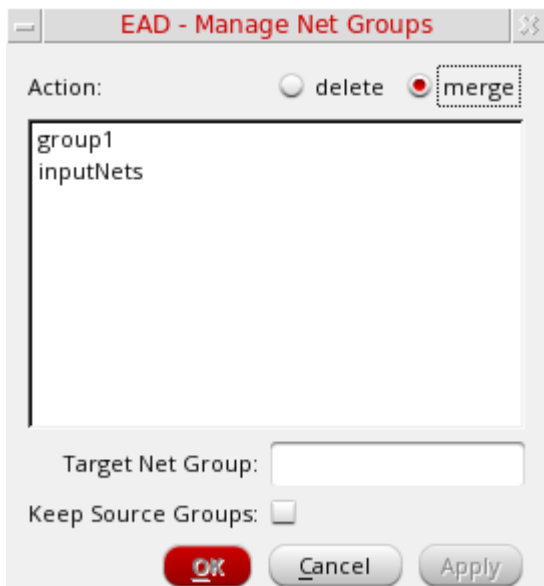
Merging Net Groups

To merge multiple net groups, perform the following steps:

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

1. Right-click in the summary pane of the EAD Browser and choose *Manage Net Groups*. The EAD - Manage Net Groups form is displayed.



2. Select two or more group names from the list of groups.
3. Select that the *merge* action.
4. In the *Target Net Group* field, specify a name to be given to the new or target group that will contain the nets from the merged net groups.
5. If you need to retain the existing groups that are being merged, select the *Keep Source Groups* check box.
6. Click *OK*.

A new net group is created with the nets from the selected net groups.

Searching Nets

To search for a particular net and to view results, use *Search*.

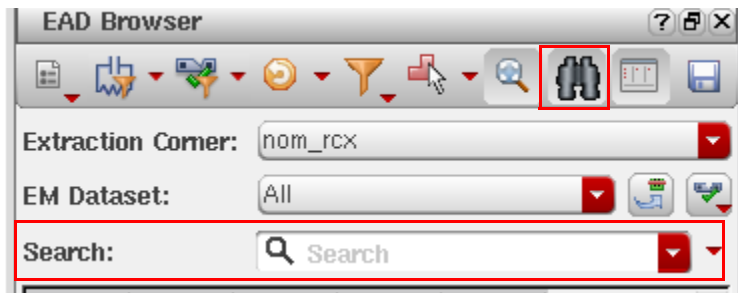


1. Click *Search* in the EAD Browser toolbar.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

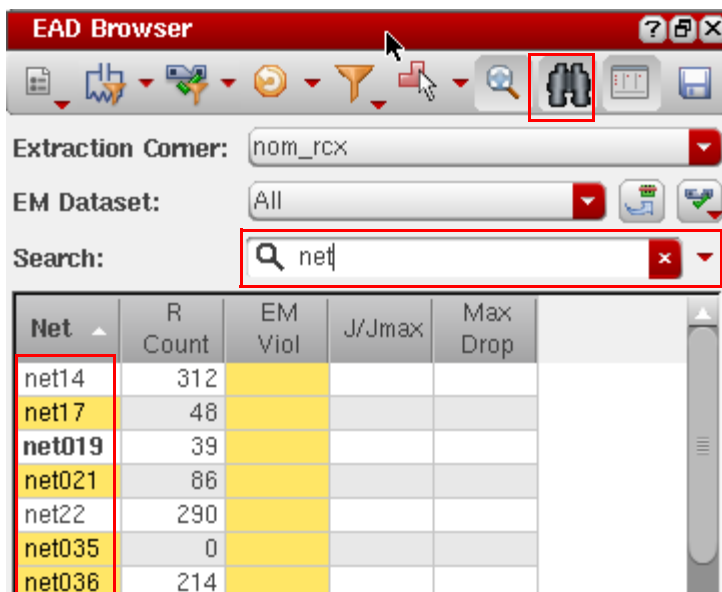
The *Search* drop-down list is added to the panel above the summary pane.



2. Type in the name of the net for which you are searching.

Alternatively, you can select a net name from the search history in the drop-down list, if available.

While you type in a name, all the net names that contain the search string get listed in the summary pane dynamically.

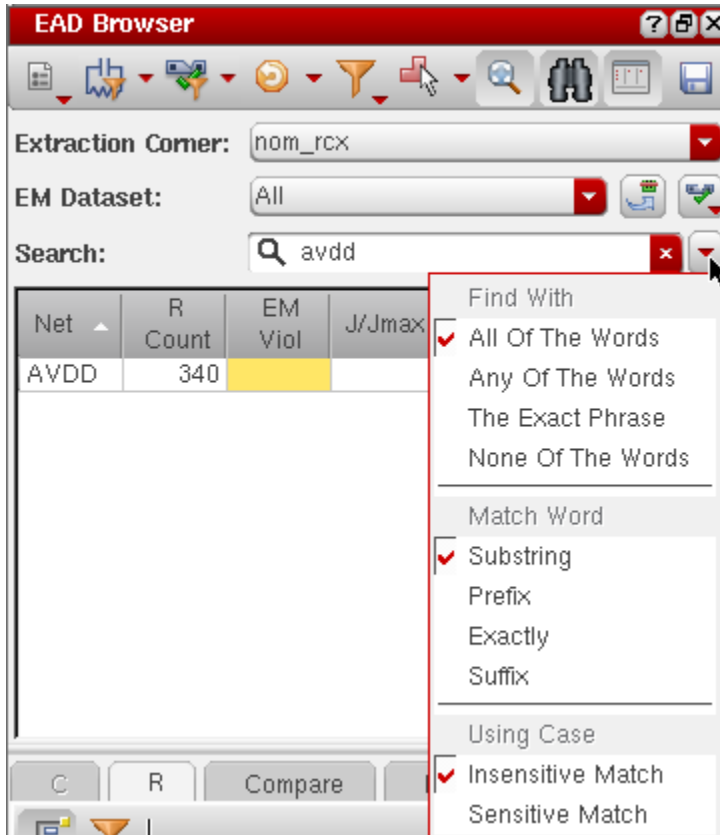


3. Select the desired net name in the summary pane and view the results in the details pane.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

4. You can customize the search by using the search settings available in the drop-down menu next to the *Search* drop-down list, as shown below.



Tip

When you are not searching for any net, you can toggle *Search* on the EAD Browser toolbar to hide the *Search* drop-down list.

Managing EAD Setups

You use the **Layout EAD Options** form to define the setups required to drive parasitic extraction and EM analysis in Layout EAD. A setup includes a pointer to the process settings required for parasitic extraction and EM checking, as well as user-customizable tool options. When you have specified the options you require, you can use the controls in the *Setups* section at the bottom of the form to save them in the system for later use.

Layout EAD Options - gpd090

General Extraction EM Environment

Process

Process Settings: gpd090

Temperature: 25 °C

Network

Excluded Nets: Select

Exclude Incomplete Nets: ☐

Primitives & Excluded Cells

Lib	Cell	View

Add Delete

Setups

Location: /.cadence/dfl/EAD/1/setup

Name: gpd090

Save To Load From Delete

OK Cancel Defaults Apply Help

Set options on each tab

Save settings as a Setup file

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

You might typically define two EAD setups; one for use in EM analysis and one for parasitic extraction to be used in resimulation:

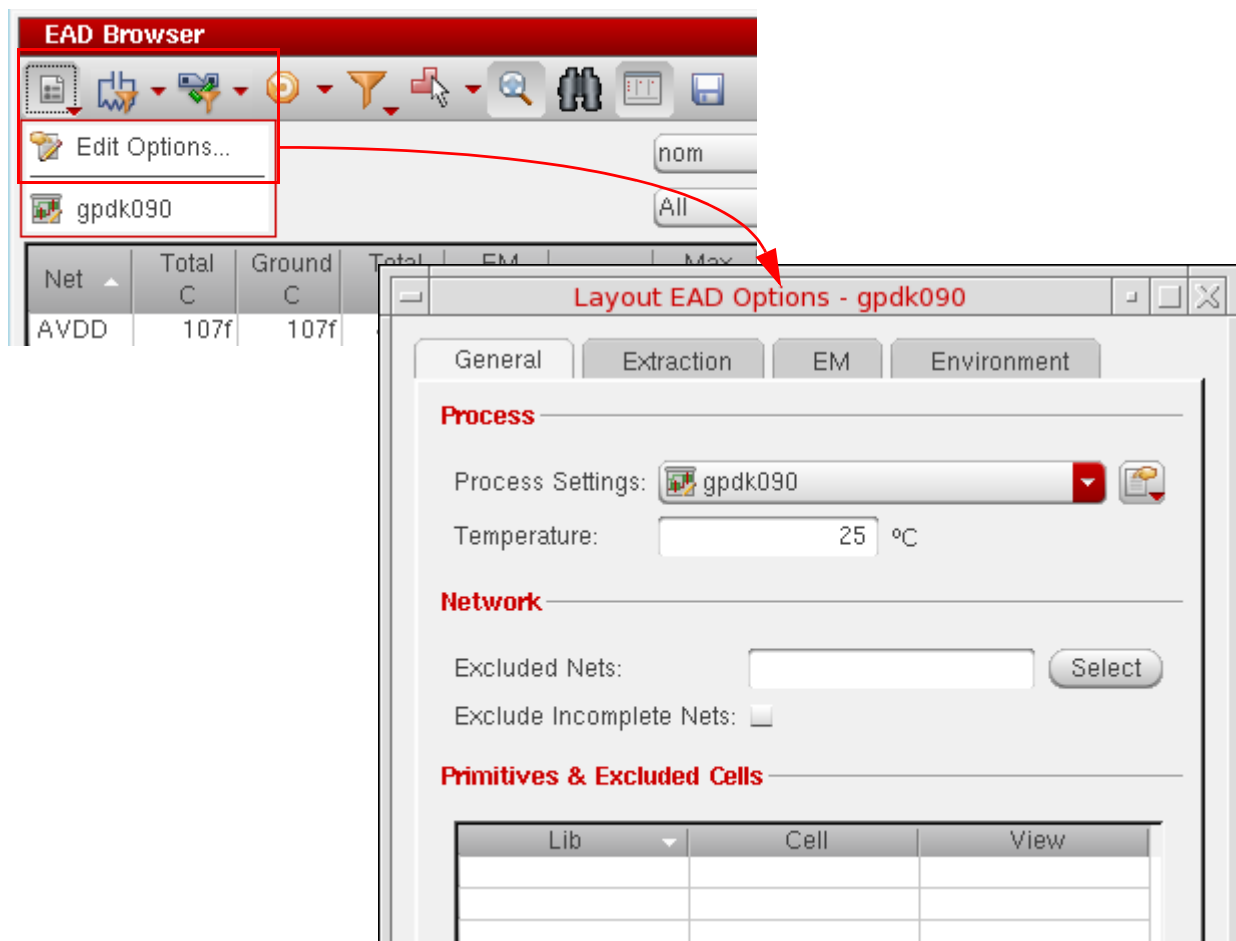
- For EM analysis, you need extract only R parasitics and include the shapes inside devices and Pcells
- For resimulation, you would typically perform both R and C extraction, and extract device and Pcell shapes (because these are already modeled in simulator models)

Preparing an EAD Setup

To prepare an EAD Setup,

1. Choose *Options – Edit Options* from the EAD Browser toolbar to open the Layout EAD Options form.

Alternatively, choose *EAD – Options* from the Layout EXL menu bar.



Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Note: You can also open the form by choosing *Options – Layout EAD* from the layout window menu bar.

2. Set the options on each of the four tabs to meet your requirements.

- ❑ *General* lets you specify the process settings to be used in the current session, the extraction temperature to be used, and the nets and cells to be excluded from parasitic extraction. See [General](#) for more details.

If you need to change the process settings for your design, see [Editing Process Settings](#).

- ❑ *Extraction* lets you specify the main extraction engine controls, resistance and capacitance thresholds, and high precision C and R extractor settings. See [Extraction](#) for more details.
- ❑ *EM* lets you specify the settings to be used for electromigration checking, including temperature, default and minimum current values, scaling factor, and lifetime values. See [EM](#) for more details.
- ❑ *Environment* lets you specify how violations are highlighted and control whether nets highlighted and selected in the EAD Browser are also highlighted and selected in the layout view. See [Environment](#) for more details.

Note: You can specify default values for each of the options on the form using the environment variable associated with the option in question. See [Appendix A, “Form Descriptions”](#) to see the environment variables for each option and [Appendix B, “Environment Variables”](#) for a complete list of all EAD-related environment variables.

Saving an EAD Setup

When you have set the options to meet your requirements, save the setup so that you can use it during your analysis of the current design.

Saving a Setup

To save the current settings to the currently loaded setup file,

1. Choose the required *Location* and *Name* from the pull-downs in the *Setups* section at the bottom of the form.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

2. Click *Save To*.



The setup file is saved with a `.ini` file extension to the specified location; for example:

`/.cadence/dfII/EAD/1/setup/gpdK090.ini`

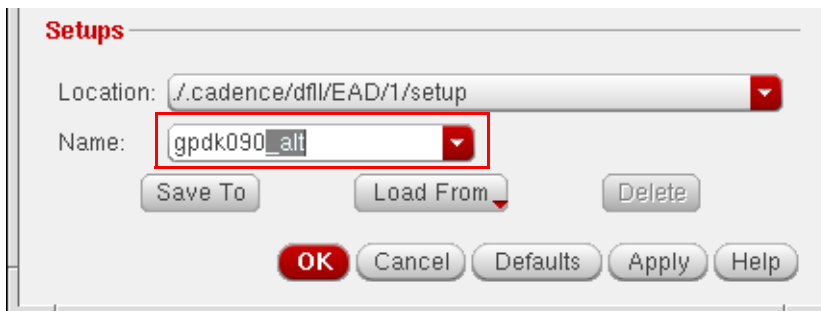
Important

Clicking *OK* or *Apply* in the form applies the current settings to the design but does not save the settings to the specified setup file. You must click *Save To* to save the settings to the setup file.

Creating a New Setup

To create a new setup file,

1. Type the new *Name* into the editable field.

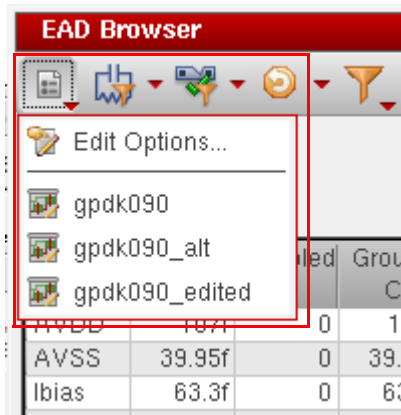


2. Click *Save To*.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

The new setup file is saved to the *Location* specified in the form and added to the list in the *Options* menu in the EAD Browser toolbar.



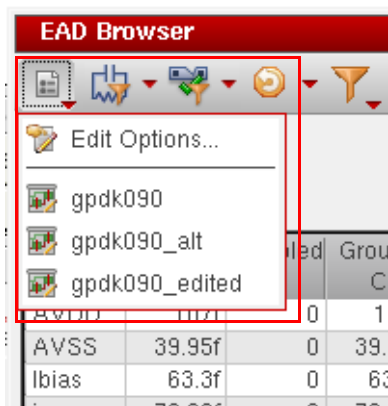
Important

Clicking *OK* or *Apply* in the form applies the current settings to the design but does not save the settings to the specified setup file. You must click *Save To* to save the settings to the setup file.

Referencing a New Setup

To use the new setup in the current design session,

- ➔ Click the *Options* button in the EAD Browser toolbar and choose the setup from the list.



Loading an EAD Setup

You can also load settings from an existing setup file into the Layout EAD Options form, edit them to meet your requirements, and then save them as a new setup.

When you load a new setup that maps to the same extraction corner (ICT file mapping), then all of the nets in the design are marked as modified and requiring re-extraction. If you load a new setup that maps to a different ICT file, then the parasitics are read and loaded into the EAD Browser and the graphical user interface is reset to its default state.

To load a different EAD setup,

1. Choose the *Location* and *Name* of the setup you want to load.

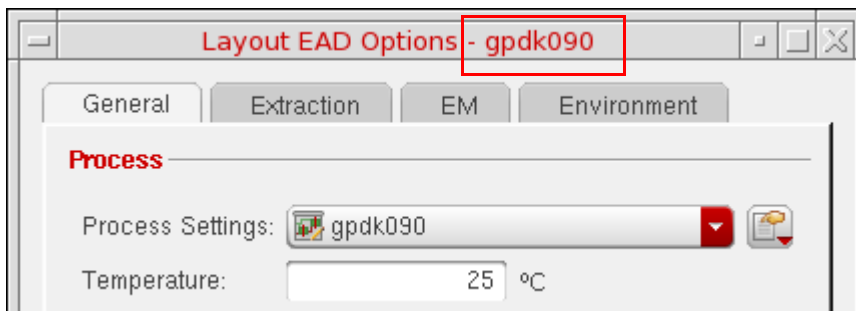


The list of locations includes all the `.cadence` directories found in the paths specified in the `Cadence setup.loc` file and in any additional directories specified by the `auxSearchPaths` environment variable. Also listed are any technology-dependent setup files stored in a subdirectory with the same name as the technology library for the design.

2. Click *Load From* to update the form with the values from the specified setup file or hold down the button to see a drop-down list of all the setup files stored for the design.

The specified setup is loaded and any stored parasitic information is retrieved and loaded. No extraction or EM checking is run by default, however, if the data exists, it is automatically loaded as well.

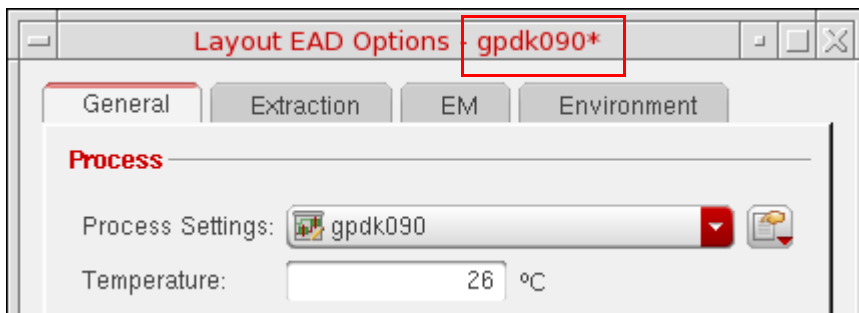
The form title updates to show the name of the loaded setup file.



Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

As soon as you make a change to the loaded setup, the form title displays an asterisk to show that there are unsaved changes.



3. Click *Save To* to save the changes in the setup file.

When EAD is initialized using the setup information, it checks whether the specified EM data source file (EM data file, ICT file, ICT-EM file, or iRCX-EM file) exists and is readable. If the file is not found or is not readable, it displays an appropriate warning message. To disable the display of this warning message, either click *Do not show again* on the message box or set the warnWhenEMDataSourcesInvalid environment variable to `nil`.



Tip

You can define a custom callback procedure and register it using the eadRegTrigger SKILL function to perform any customized tasks before loading any EAD-related setup.

Related Topics:

- [Editing Process Settings](#)
- [Checking Layer Mapping](#)

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Editing Process Settings

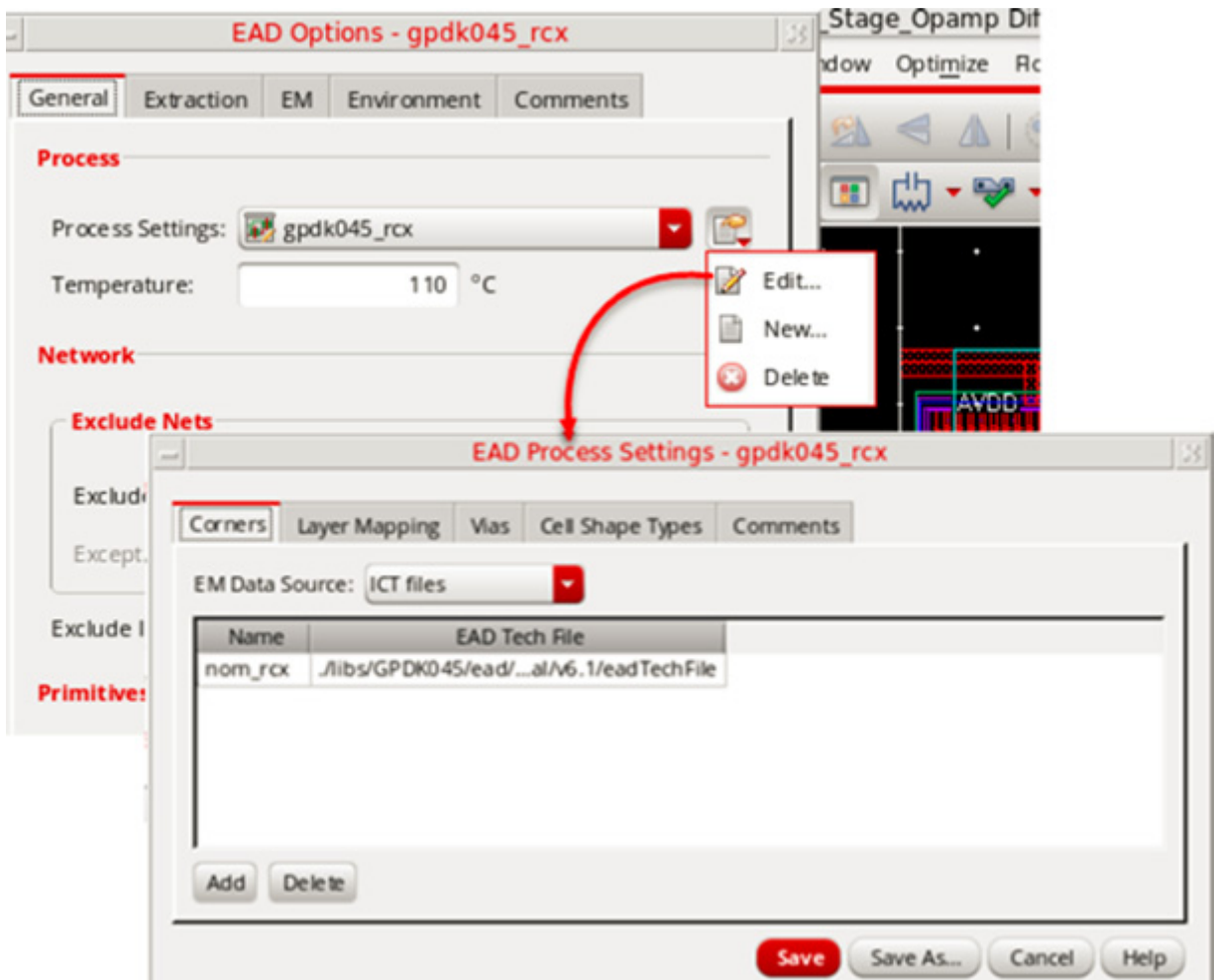
Process settings contain information about the extraction corners and the models to be used for each during EM checking, the mappings between layers in the ICT file and those in the Virtuoso technology file, and how vias and shapes in sub-cells are handled during extraction.

For details about the process information contained in the ICT technology file and the syntax of the commands, see *Chapter 3* in the [Quantus QRC Techgen Reference Manual](#).

Note: The Virtuoso release is independent of the Quantus release. Therefore, it is possible to see some mismatch in the syntax of the commands, especially for the advanced nodes, supported by the two tools at a given point of time.

To edit the process settings for a design,

1. Click *Process Settings* menu () and choose *Edit* to open the **EAD Process Settings** form.



Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Note: The *Save* button on the EAD Process Settings form is disabled when the process settings file cannot be edited.

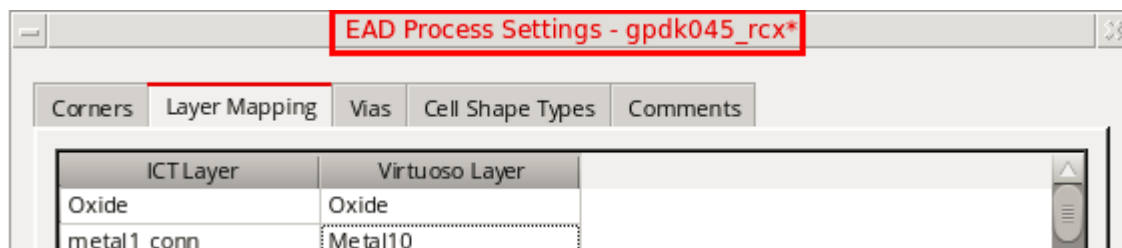
2. On the *Corners* tab of the EAD Process Settings form, choose an EM data source from the *EM Data Source* drop-down list. See [EAD Process Settings – Corners](#) for more details.

Also see: [Using Custom Variables in ICT-EM Files](#)

3. Edit the process settings on each tab to meet your requirements:

- ☐ *Corners* is where you associate your extraction corners with the ICT file or eadTechFile to be used for extraction. The EM Data Source field lets you specify the source of the EM models and rules; either a Virtuoso technology file, an ICT-EM file, or an EM data file. See [EAD Process Settings – Corners](#) for more details.
- ☐ *Layer Mapping* lets you specify how the layers used in the ICT file map to the layers defined in the Virtuoso technology file. See [EAD Process Settings – Layer Mapping](#) for more details.
- ☐ *Vias* lets you specify how vias are flattened and clustered. See [EAD Process Settings – Vias](#) for more details.
- ☐ *Cell Shape Types* lets you control how shapes in sub-cells contribute to parasitic extraction. See [Overview of Parasitic Extraction in Layout EAD](#) for more information.
- ☐ *Comments* lets you view and modify the comments saved in the process settings file.

As soon as you make a change to the process settings, the forms title displays an asterisk to show that there are unsaved changes.



4. When you have finished, do one of the following:

- ☐ Click *Save* to save the settings under the current *Process Settings* file name.
Note: The *Save* button is not enabled if the process file is read-only.
- ☐ Click *Save As* to save the settings under a different *Process Settings* name

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

The process settings file is saved is saved with a `.ini` file extension to the specified location; for example:

```
/.cadence/dfII/EAD/1/process/gpdk090.ini
```

You can then use these settings in your current design or in future EAD sessions.

Related Environment Variables

- ☐ [`viaInstanceMode`](#)
- ☐ [`viaShapesScanMode`](#)

Important Points to Note

- ☐ If multiple corners are specified, you must ensure that all of the technology files were consistently generated for R-only or RC extraction. A mix of R-only and RC technology files is not supported.
- ☐ If the EM data source is an EM data file, the EAD parser ignores the unsupported keywords and parameters. For each ignored keyword or parameter, a warning is displayed in the CIW.

Using Custom Variables in ICT-EM Files

If the EM calculations in the ICT-EM file are using certain variables that you would need to modify before running EM checks, you can register those variables with EAD by using the [`eadRegisterCustomEMVariable`](#) SKILL function. Later, you can modify those values by using the [`eadSetEMVariable`](#) SKILL function and then run EM extraction to use the revised variable value for EM calculation.

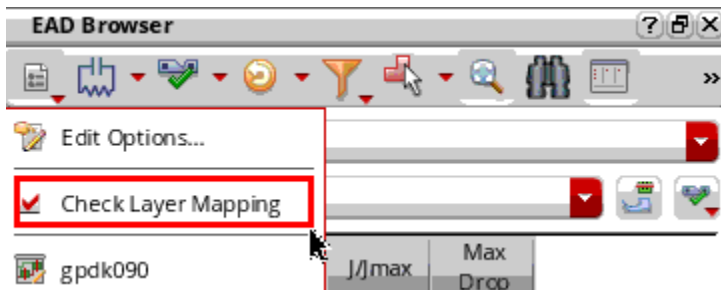
Checking Layer Mapping

Note: The Check Layer Mapping check box is hidden by default. Set the environment variable `layoutEAD.gui hideCheckLayerMapping` to `nil` to display this check box in the *Options* menu of the EAD Browser.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

After you load the setup from an EAD setup file, the *Check Layer Mapping* command is added to the *Options* menu on the EAD toolbar.



Use this command to check whether mapping of all the layers in Virtuoso technology file to the layers in the ICT file is complete. If any layer is found as unmapped, an error is displayed reporting a list of unmapped layers. In such a case, you can update layer mappings on the Layer Mapping tab of the EAD Process Settings form. While you are editing the layer mappings, you can use the Check command on the Layer Mapping tab to rerun the checks. Ensure that there are no warning messages.

Deleting an EAD Setup

To delete a setup that you no longer require:

1. Choose the *Location* and *Name* of the setup you want to delete.



2. Click *Delete* and confirm the action in the pop-up message that appears.

The setup file is deleted from the location on disk.

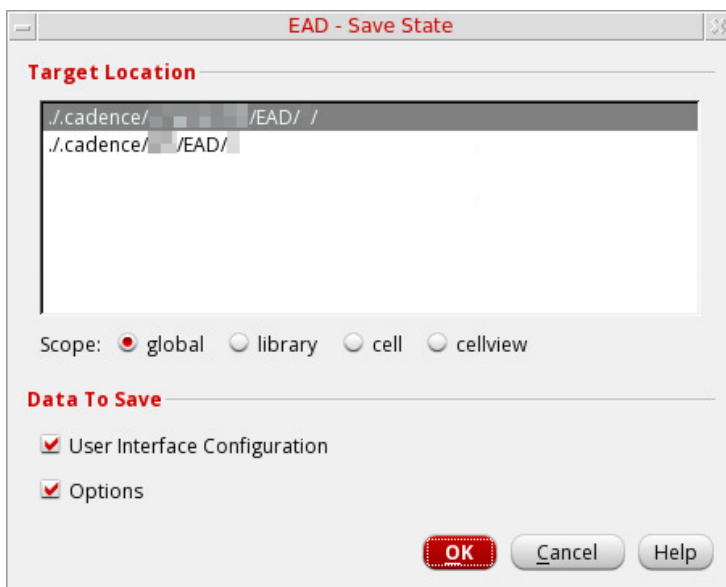
Working with EAD States

This section describes how to save, restore, and load an EAD state.

Saving an EAD State

Follow these steps to manually save the current EAD state:

1. On the *EAD Browser* toolbar, click *Save current EAD state*. The *EAD - Save State* form opens.



2. In the *Target Location* section, select a location where you want to save the state data.

The list of locations includes all the `.cadence` directories found in the paths specified in the `Cadence setup.loc` file.

Note: You can also specify your own directory to save or load an EAD state by setting the `statePath` environment variable. The directory specified by the environment variable is given priority over those specified in `setup.loc` file.

3. From the *Scope* options, select one of the following options tag the state being saved for the specific subsets of designs:
 - ☐ *global*: Saves the state that can be used for all designs.
 - ☐ *library*: Saves the state that can be used for all designs in library of the current design
 - ☐ *cell*: Saves the state that can be used for different views of the current design

- ☐ *cellview*: Saves the state that can be used only for the current design
- 4. In the *Data to Save* section, select the following options to save the specific parts of the EAD state:
 - ☐ *User Interface Configuration*: Save the GUI-specific data, such as toolbar buttons, toggle states, table column sizes, and so on.
 - ☐ *Options*: Saves the design-specific data, such as values of the current EAD options.
- 5. Click *OK* to save the current EAD state.

Restoring an EAD State Automatically

You can choose to automatically restore EAD to a state in which the design was last exited. Set the `saveStateOnClose` environment variable to `t`:

```
envSetVal("layoutEAD.gui" "saveStateOnClose" 'boolean t)
```

Loading an EAD State

You can load an EAD state, which has been saved earlier, by clicking *Load EAD state* on the *EAD Browser* toolbar. EAD searches for the state data based on the following parameters. The first instance of the state found is loaded.

- *Scope*: State data is searched by scope in the following order: `cellview`, `cell`, `library` and `global`.
- *Target Location (Path)*: State data is searched by paths in the order they are defined in the `setup.loc` file.

You can also load EAD state data automatically on startup. To do so, set the following environment variable to `t`:

```
envSetVal("layoutEAD.gui" "loadStateOnOpen" 'boolean t)
```

Extracting and Analyzing Parasitics

This section explains how to use Layout EAD to extract and analyze parasitics for a layout design; how to use the EAD Browser to find out more about the reported values; how to compare the values extracted for different nets of interest; and how to extract parasitics automatically while you are editing the layout in the canvas.

- [Overview of Parasitic Extraction in Layout EAD](#)
- [Customizing Extraction of Parasitics](#)
- [Extracting Parasitics in Layout EAD](#)
- [Viewing and Analyzing Parasitics](#)
- [Comparing Parasitic Values for Multiple Nets](#)
- [Using Automatic Update Mode](#)
- [Distributed Computation of Extraction Process](#)
- [Working with Job Policies](#)

Overview of Parasitic Extraction in Layout EAD

This section describes the default extraction behavior for various design components and explains the settings you can make to change how the extraction is performed.

- [Extracting Parasitics on Interconnect Wires Between Devices](#)
- [Extracting Parasitics inside Devices](#)
- [Extracting Parasitics for Devices and Interconnect](#)
- [Extracting Parasitics for Devices and Overlapping Interconnect](#)
- [Extracting Parasitics for Local Interconnect Layers](#)
- [Extracting Parasitics from Strong, Weak, or Must Connections](#)
- [Assigning Shapes for Extraction](#)
- [Preventing Extraction Inside Devices](#)
- [Stopping Extraction at the Device Boundary](#)
- [Specifying the Hierarchy Levels and Hierarchy Shape Type](#)

Note: For advanced nodes, EAD supports color aware RC extraction. As shapes are drawn on different colors, EAD can extract and display RC values based on the color of the wires.

Layout EAD defines the following types of default capacitance extraction:

- *Coupling capacitance* is the capacitance extracted between two shapes with a single net assigned to them; for example, `netA` and `netB` next to each other on `Metal1`.

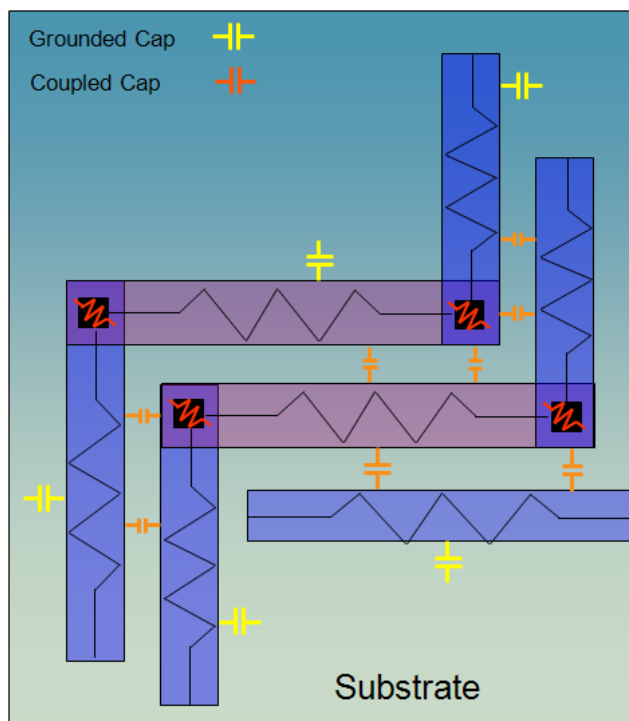
Net assignment is performed inside Layout EAD by the Layout XL connectivity extractor.

- *Grounded capacitance* is the capacitance extracted between a shape with a net assignment and a shape that does not have a net; for example:

- ☐ `netA` over substrate
- ☐ `netA` over a hierarchical cell that is not being extracted

Extracting Parasitics on Interconnect Wires Between Devices

The figure given below illustrates the default extraction behavior for wires used to connect devices:



Note the following points:

Virtuoso Electrically Aware Design Flow Guide

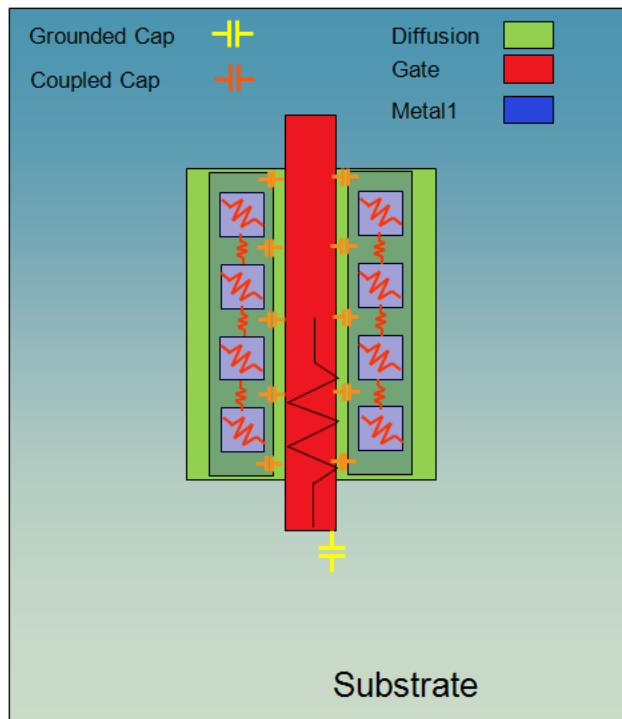
Extracting Parasitics in the Layout View

- Resistance is extracted for all layers defined as `conductor` layers in the ICT file and for vias/contacts (typically poly, contact, and metal layers and vias)

If the `layer_type` is define as `diffusion`, resistance is not extracted for that layer.
- Coupling capacitance is extracted between all `conductor` layers (i.e. Metal1 to Metal1, Metal1 to Metal2, and so on)
- Grounded capacitance is extracted between conductor layers and substrate. During resimulation the grounded node is specified in the *Parasitics – Setup* form.

Extracting Parasitics inside Devices

The figure given below illustrates the default extraction behavior inside devices:



Note the following points:

- Resistance is extracted for the **terminals** on layers specified as `conductor` in the ICT file and for vias/contacts (typically, metal and contacts, but excluding diffusion)
- Coupling capacitance is extracted between all conductor layers (e.g. Metal1 to Poly, Metal1 to Metal1, and so on)

Note: Coupling capacitance is **not** extracted for

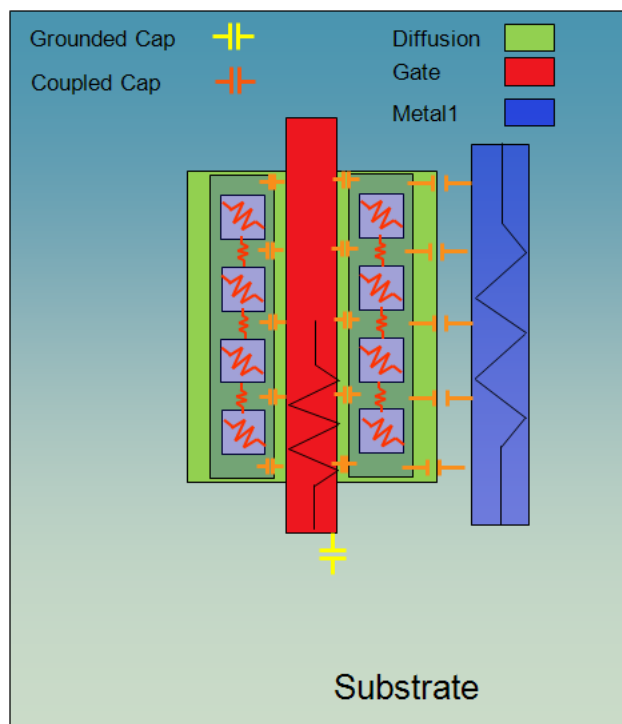
Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- ☐ gate-to-contact
- ☐ metal-to-contact
- ☐ contact-to-contact
- ☐ gate-to-diffusion fringing cap
- Grounded capacitance is extracted between conductor layers and substrate (poly to substrate). During resimulation, the grounded node is specified in the *Parasitics – Setup* form.

Extracting Parasitics for Devices and Interconnect

When extracting parasitics between devices and interconnect wires, the coupling capacitance is extracted between interconnect outside the device and interconnect inside the device, as shown in the figure given below.

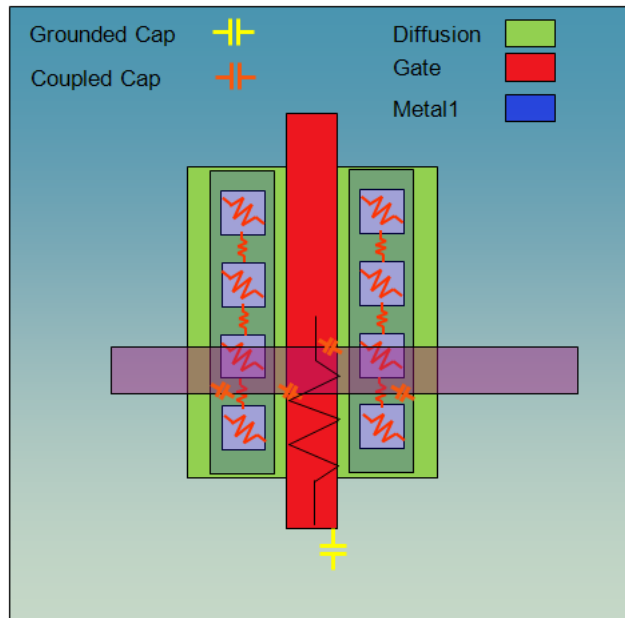


Extracting Parasitics for Devices and Overlapping Interconnect

The figure given below illustrates the default extraction behavior for devices and overlapping interconnect:

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View



Note the following points:

- Coupling capacitance is extracted between interconnect outside the device (defined as `conductor` in the ICT file) and interconnect inside the device (defined as `conductor` in ICT file).
- Grounded capacitance is extracted to any shape that has a net, but where the technology file defines no RC extraction on that shape; for example, substrate or diffusion.

Extracting Parasitics for Local Interconnect Layers

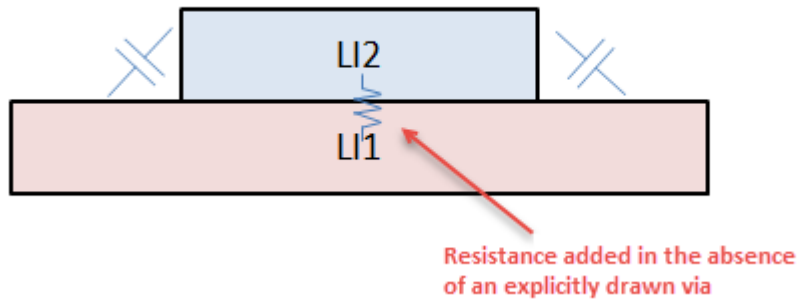
When two shapes on different local interconnect layers overlap and also touch by abutment in the vertical dimension, that is, top-to-bottom, the EAD extractor can compute this vertical resistance. Definition for this via is provided in the ICT file. Using the [Layer Mapping](#) tab of the EAD Process Settings form, you can map the via layer in the Virtuoso technology file to the via in the ICT file and then, draw a shape on that layer. While extracting parasitics, Layout EAD extracts via resistance from that shape and displays the R value in the *Resistances* table.

In no via definition is explicitly drawn, you can leave the ICT via layer unmapped and EAD will compute the resistance from an implied via between the overlapping layers, as shown in the

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

figure below, and calculates a resistance value depending on the area of contact between the two layers.



The value of R is displayed in the *Resistances* table. Since there is no design layer for the via, EAD uses the name of the local interconnect layer for the resistance displayed on the R tab.

Extracting Parasitics from Strong, Weak, or Must Connections

EAD extracts resistance paths between terminals and instance terminals. Depending on the strong, weak, or must pin connectivity, EAD performs R extraction as explained below.

- When different pin shapes are modeled as strong connect, there are multiple figs on a single pin that is further connected to a single instance terminal. In this case, EAD assigns the terminal node to one of the figs and extracts resistance paths to that fig. Multi-finger transistors with internal source-drain strapping should be modeled as strong connect.
- When different pin shapes are modeled as must connect, multiple instance terminals are connected externally, and each instance terminal has a pin and a fig associated with it. In this case, EAD extracts resistance paths to each instance terminal. Multi-finger transistors with no internal source-drain strapping should be modeled as must connect.
- When a pin shape is modeled as weak connect, there are multiple pins on a single terminal. In this case, EAD assigns the terminal node to one of the pins and extracts resistance paths to that pin.

For details on how to set pin connectivity models in the layout, refer to [Setting the Pin Connectivity Model](#) in *Virtuoso Layout Suite XL User Guide*.

Customizing Extraction of Parasitics

The following topics describe how you can customize the extraction of parasitics for devices in your layout:

- [Assigning Shapes for Extraction](#)
- [Preventing Extraction Inside Devices](#)
- [Stopping Extraction at the Device Boundary](#)
- [Specifying the Hierarchy Levels and Hierarchy Shape Type](#)
- [Identifying Instances to be Ignored during Extraction](#)
- [Modifying EM Analysis for Wide Pins](#)

Assigning Shapes for Extraction

Layout EAD can treat cells, views, terminals, and layers as different shape types for extraction purposes. This lets you perform a differentiated extraction of terminals on Pcells by defining how non-terminal shapes in sub-cells contribute to the extraction.

Layout EAD recognizes three different shape types:

- *regular* – shapes that have a net assignment.
Coupling capacitance is calculated to these shapes.
- *background* – shapes that do not have a net assignment.

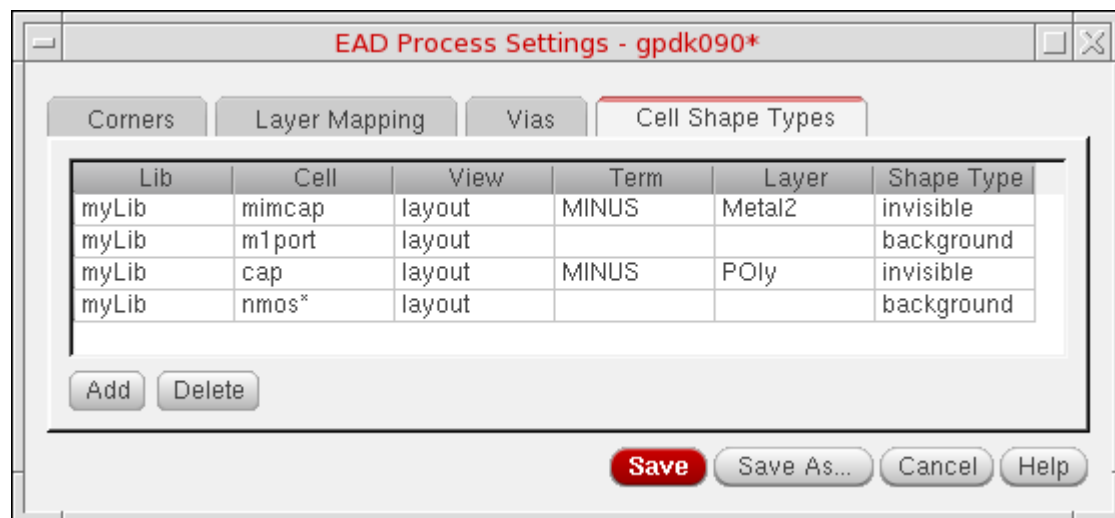
In general, grounded capacitance is calculated to these shapes; however, there is an exception for background shapes on device or instance terminals for which coupling capacitance is also calculated.

- *invisible* – shapes are not sent to the extraction engine.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

You specify shape types for the Pcells and instances in your design in the EAD Process Settings form during process setup:



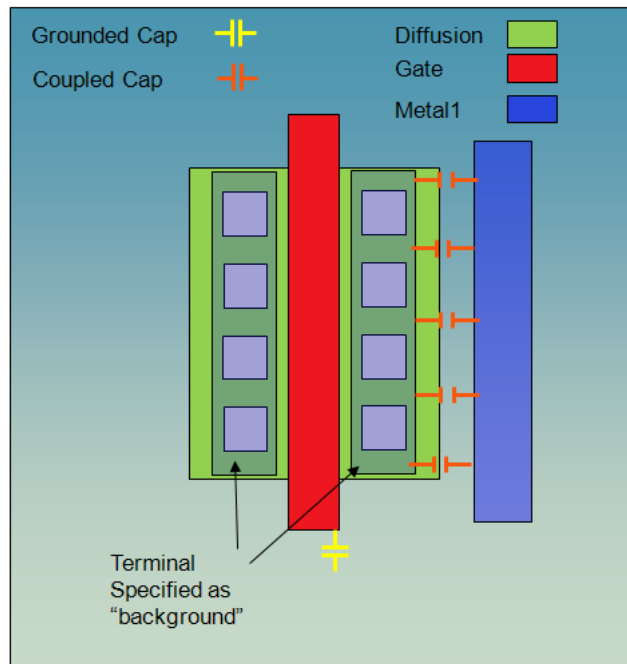
- **Row 1:** EAD will not pass the Metal2 shape for the MINUS terminal on myLib/mimcap/layout to the extractor because the shape is set to invisible
- **Row 2:** EAD will label all terminal shapes in myLib/m1port/layout as background so that no resistance is extracted to those shapes and capacitance is extracted from other shapes to the terminal shapes
- **Row 3:** EAD will make the Poly layer on the MINUS terminal in myLib/cap/layout invisible to the extractor
- **Row 4:** Similar to Row 2, but in this case, EAD will label all terminals on all the nmos* devices in the design as background

Example: Metal1 specified as background

To prevent RC extraction on Metal1 shapes in a Pcell, but still generate coupling capacitance to nets outside the Pcell, you can define Metal1 on source/drain terminals as background:

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View



Even though a terminal is defined as `background`, coupling capacitance is extracted to that net, but no `Metal1` to `Poly` or `Metal1` to `Metal1` coupling for terminal shapes in devices is extracted.

For more information, see [EAD Process Settings – Cell Shape Types](#) and [Editing Process Settings](#).

Preventing Extraction Inside Devices

You can stop Layout EAD from extracting parasitics inside devices using the *Primitives & Excluded Cells* table on the [General](#) tab.

Primitives & Excluded Cells		
Lib	Cell	View
Two_Stage_Opamp	OpAmp	layout_new
ether	inv_2x_hv	
em_test		

Parasitics are extracted up to and including the terminals of the excluded cells, but no extraction is performed inside the specified devices.

Use this option to exclude devices such as capacitors and resistors, where you do not want to extract resistance and capacitance twice, or for polygonal standard cells or other

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

instances that are not XL compliant.

Device terminals are extracted by default, which means that the resistance on the terminals is computed and all terminal shapes are passed to the extraction engine for capacitance extraction as well (allowing calculation of the coupling capacitance from the terminals to nets outside the Pcell).

See also: [extractionPrimitives](#)

Stopping Extraction at the Device Boundary

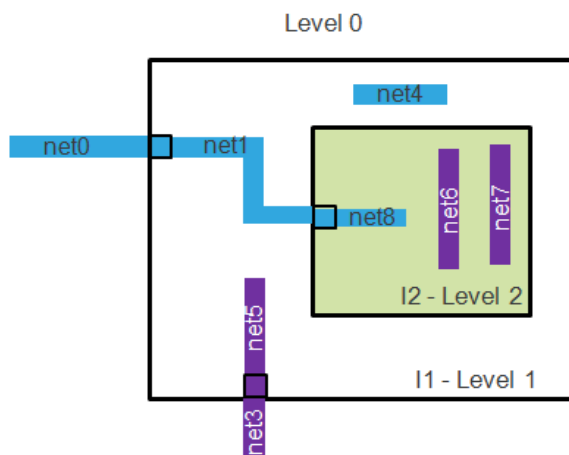
If, for a given technology, you want to specify that extraction should stop at the boundary of a Pcell or an instance, check the *Stop at Device Boundary option* in the [Extraction](#) tab. By default, EAD extracts parasitics for such shapes. But when this option is switched on, extraction stops at the device boundary, which means that terminals on Pcells and instances are treated as background shapes and are **not** extracted.

This works in conjunction with the *Hierarchy Levels* option for instances.

Specifying the Hierarchy Levels and Hierarchy Shape Type

The *Hierarchy Levels* option on the [Extraction](#) tab controls how many levels to chase a net down through the hierarchy and how many levels should be extracted for resistance.

Note: Decoupled capacitance is extracted to shapes for levels greater than the *Hierarchy Levels* setting. This option also works in conjunction with the *Stop at Device Boundary* option.



Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

For example, with *Hierarchy Levels* set to 1 (as shown in the above figure), the following nets are shown in the EAD Browser with R and C extracted:

- net0, net3, I1 | net4
- net1, net5 are shown as part of net0, net3 respectively

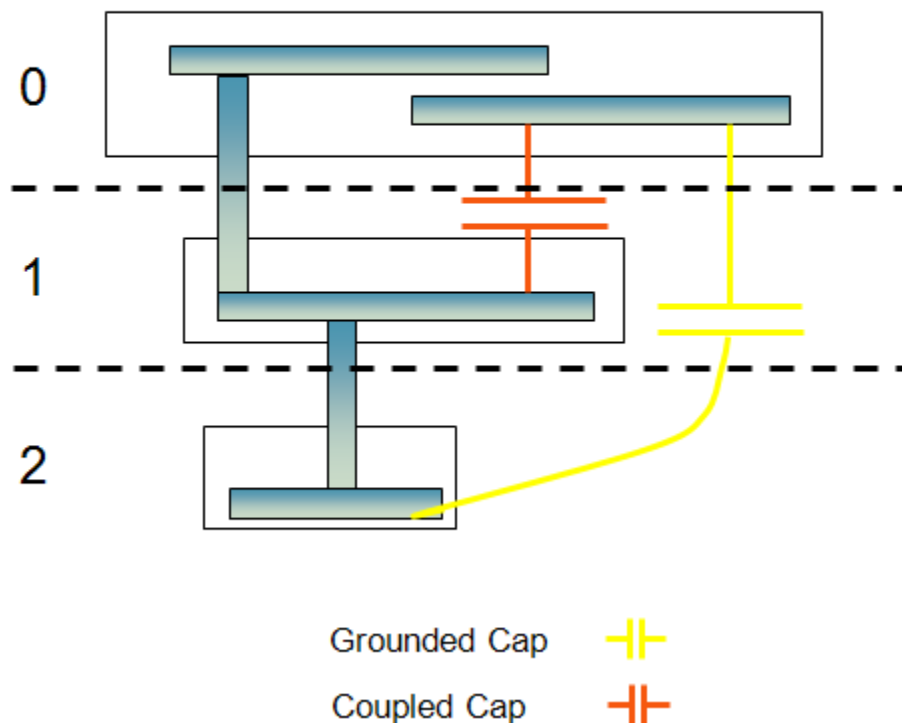
Decoupled capacitance is extracted to shapes on:

- I2 | net8, I2 | net6, I2 | net7

You can also use *Hierarchy Levels* in conjunction with the *hierarchyShapeType* environment variable to specify how geometry is treated during extraction. An example is shown below.

Hierarchy Levels = 1

hierarchyShapeType = "background"



Identifying Instances to be Ignored during Extraction

EAD uses the `propsUsedToIgnoreObjs` variable to look for the properties that identify the pins and instances to be ignored from EM checks. If your layout contains some more instances that are not being ignored by the properties defined by `propsUsedToIgnoreObjs`, you can apply the `eadIgnore` property on such instances. This helps in eliminating the irrelevant terminals and shows only relevant parasitic data after extraction.

You can apply the `eadIgnore` property on a layout cellview to eliminate extraction of all instances of that cellview in your layout. Alternatively, if you are unable to edit the cellview, you can select the instances of that cell in your layout and selectively apply the property.

Applying eadIgnore on a Master Layout Cellview

To apply the `eadIgnore` property on a master layout cellview:

1. Open the master layout cellview in the edit mode.
2. Type the following commands in the CIW:

```
cv = geGetEditCellView()  
dbCreateProp(cv "eadIgnore" 'boolean t)
```

3. Save the changes and close the cellview.

After you apply this setting, all instances of this layout are ignored during extraction.

Applying eadIgnore on Specific Instances in the Design Layout

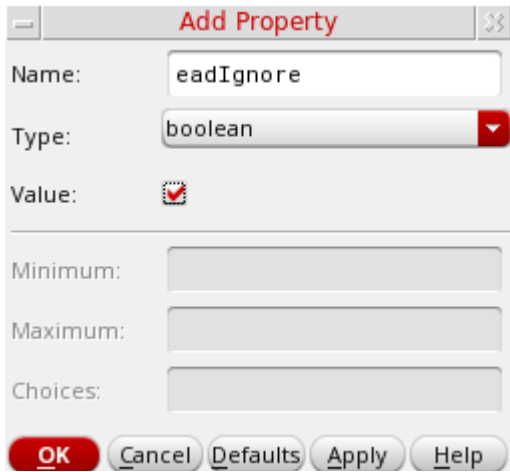
To apply the `eadIgnore` property on specific instances:

1. Open the layout of your design in the edit mode.
2. Open the Property Editor assistant.
3. Select the instance to which you need to apply this property.
4. Right-click in the Property Editor assistant and choose *Add Property to <inst-name>*.
5. In the Add Property form that is displayed, specify the property name in the *Name* field as `eadIgnore`.
6. In the *Type* field, select the type as `boolean`.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

7. Select the *Value* check box to set the property value to t .

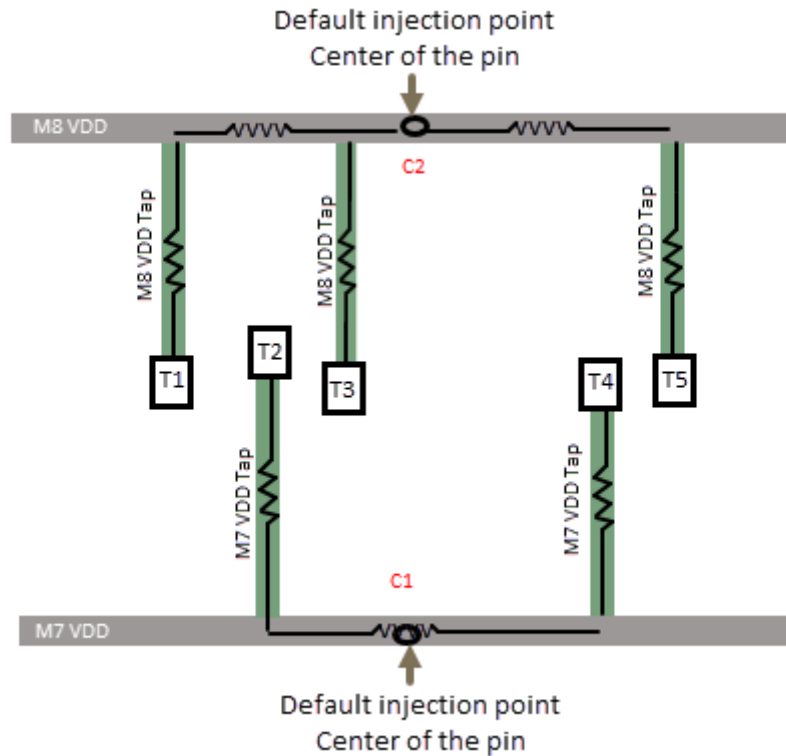


The screenshot shows the 'Add Property' dialog box. The 'Name' field contains 'eadIgnore'. The 'Type' dropdown menu is set to 'boolean'. The 'Value' checkbox is checked, indicating the value is set to true. Below these fields are empty input boxes for 'Minimum', 'Maximum', and 'Choices'. At the bottom, there are five buttons: 'OK' (highlighted in red), 'Cancel', 'Defaults', 'Apply', and 'Help'.

The required property is now set on the instance for which you want to ignore the instance terminals while extracting parasitics.

Modifying EM Analysis for Wide Pins

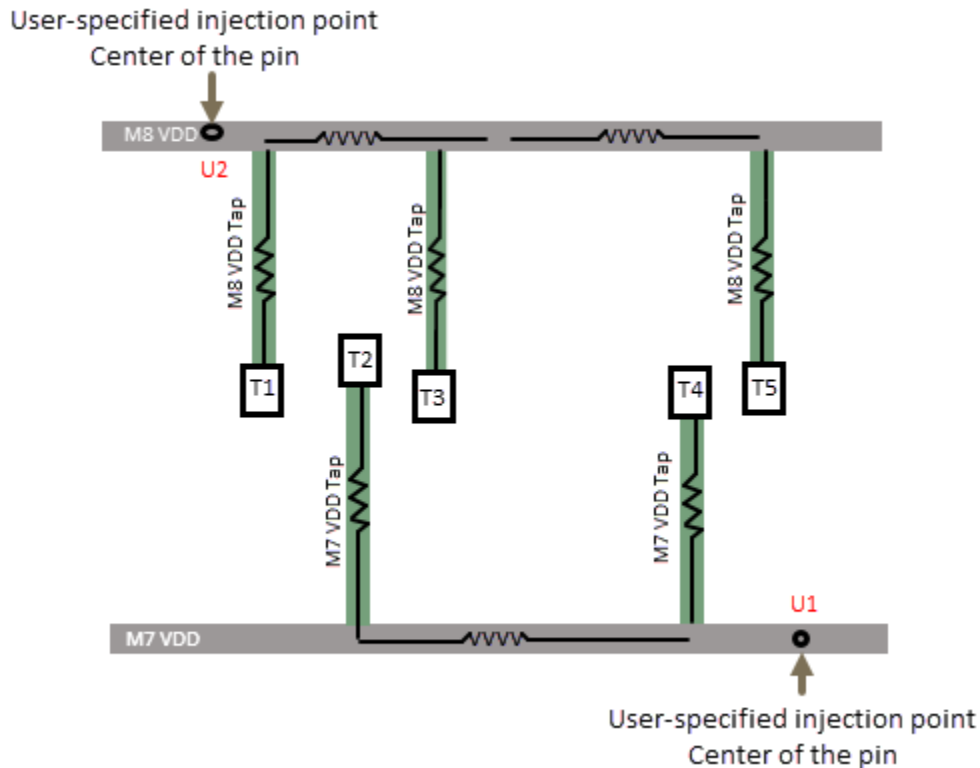
By default, EAD considers center of the pin as the default current injection point. EAD recalculates the current flowing through the pins, and while doing that considers the resistance on the pin to be zero. That is, the pins are considered to be super conductors.



Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

However, for wide pins, you can consider any other current injection point or multiple injection points at specific points that would overlap with the sources of current.



For each injection point, EAD inserts an additional resistor or node on the pin. It then calculates the total current flowing through the pin by considering the values of all resistors on the pin. Therefore, the resistance on pins also impacts the EM violation and IR drop calculations.

You can enable multiple point injection for wide pins in the following ways:

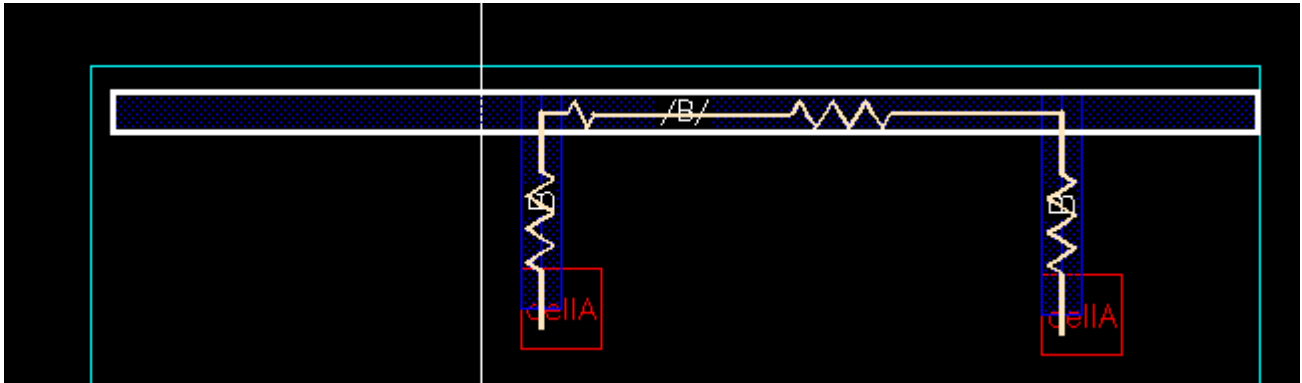
- By editing the layout view to add labels at the injection points
- By providing the location of injection points or nodes in a file

After creating labels or specifying a location file for nodes, run R extraction. New resistances that were added at the nodes are extracted along with other resistance values and their details are displayed on the *R* tab in the EAD Browser.

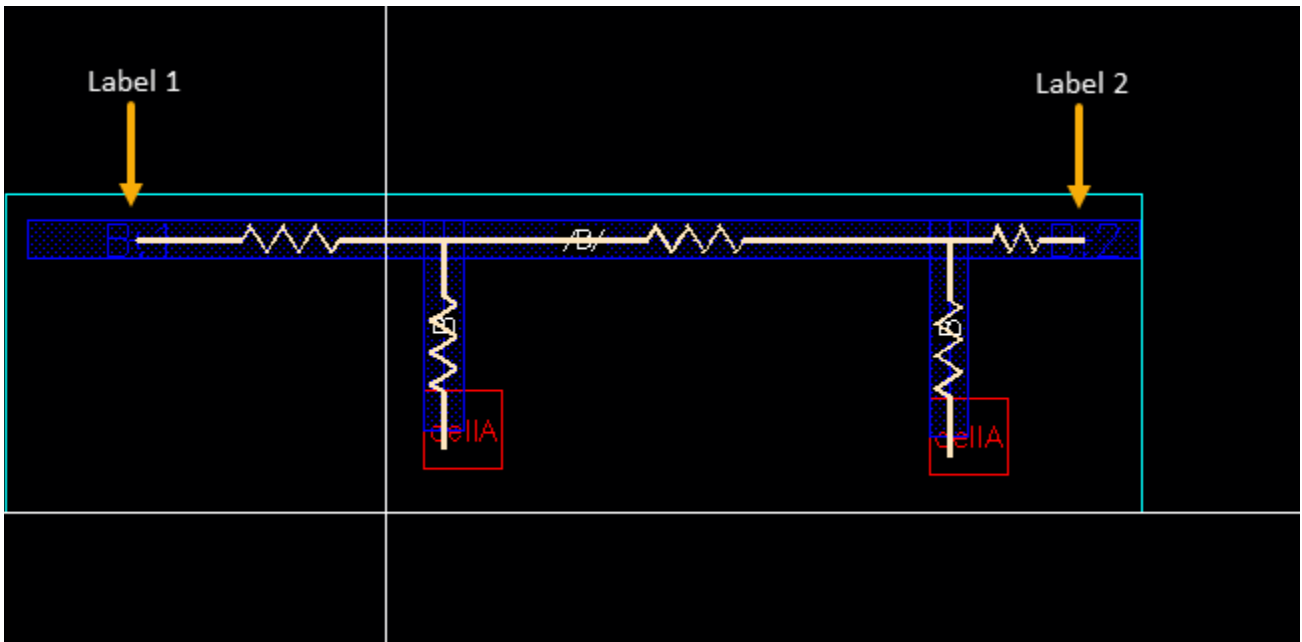
Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Consider the following example. By default, four resistors are extracted for pin B, as shown below.



Two labels are added to this pin to specify the current injection points. Running extraction after inserting nodes, returns five resistors, as shown below.

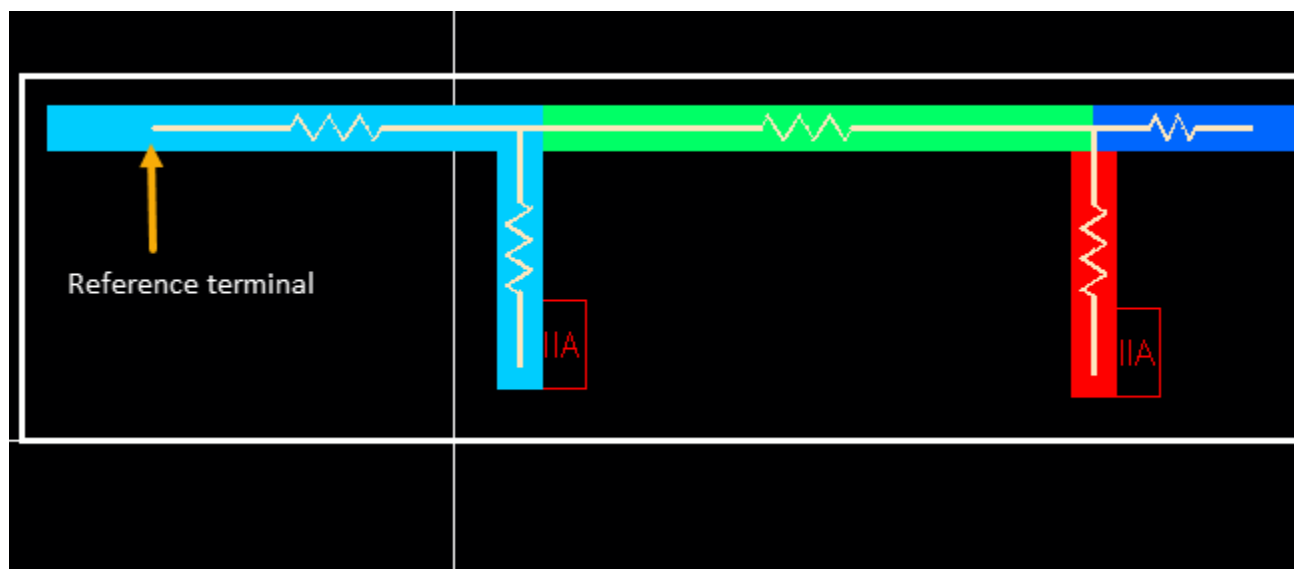


These resistances are considered in EM analysis and reflected in the EM violations reported on the *EM* tab and IR drop data displayed on the *IR Drop* tab.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

The heat map or IR drop violations for the above example also considers the additional resistance added for the new current injection point.



Editing Layout to Add Labels for Current Injection Point Creation

To add injection points or nodes on a layout:

1. Before launching Virtuoso, set the following two variables in the `.cdsinit` file:
 - ☐ Set `insertResNodeLabelPurpose` to specify the name of the purpose on which you plan to create labels to be used for the insertion of node injection points.
 - ☐ Set `stackViaArrayHandling` to "none" to allow the evaluation of EM violations at pins. By default, the pin resistance is considered as zero and EM violations are not calculated for pins.
2. Launch Virtuoso and open the layout view.
3. Choose *Create – Label* to open the Create Label form.
4. On the Create Label form,
 - a. Specify a label pattern in the *Label (Pattern)* field.

The label pattern must be in the `<pin-name>.<positive-integer>` format. For example, if you need to create labels for pin A, you can enter the value in the *Label (Pattern)* field as `A.1`.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- b. Select the *Use same layer as shape, select purpose* option and select a purpose name from the drop-down list next to this field.
- c. Set the *Height* label option to a small value, for example, 0.05, so that labels can fit on the pins.

Create Label

Mode ☒ Manual ☐ Auto

Label (Pattern) AVDD.1

Label Layer/Purpose

☐ Use current entry layer

☒ Use same layer as shape, select purpose label

☐ Auto (Using stampLabelLayers section from technology file)

☐ Select layer Nwell drawing

Label Options

Font stick Height 0.05 Justification centerCenter

☒ Drafting ☒ Attach

☐ Overbar ☒ Expand

☒ Keep Label

Scan Line

☐ Label Selected Objects ☒ Auto Zoom Out

Auto Step

☐ Step Custom X 0 Y 0

Snap Mode orthogonal

Rotate Sideways Upside Down

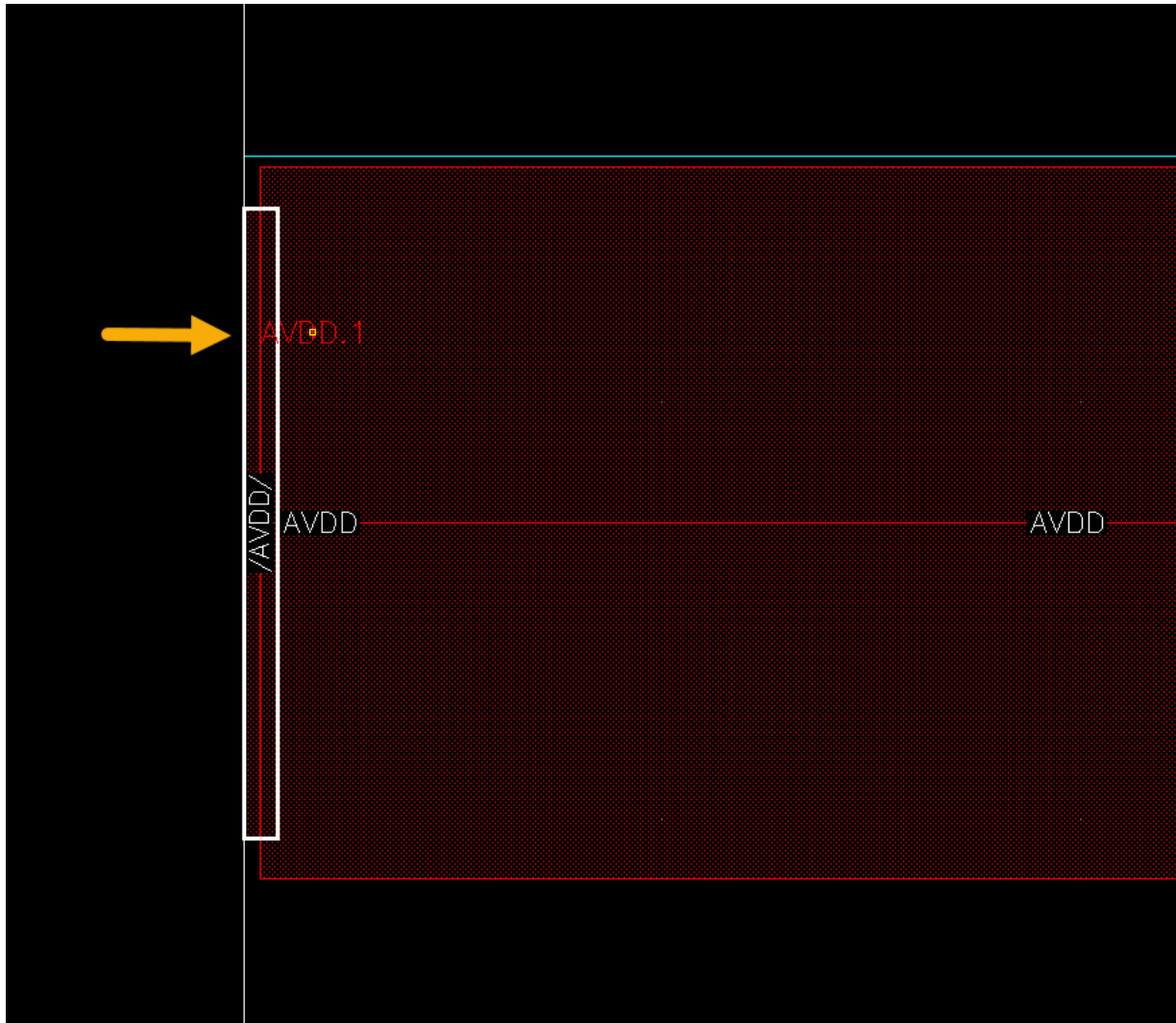
Hide Cancel Help

- d. Click *Hide* to close the form and to continue with label creation.

A label is attached to the pointer.

5. Bring the desired pin in focus and zoom in.

6. Click at the location where you want to add a label on the pin. A label is added with the name mentioned on the Create Label form.



7. Create more labels as required.

Adding Current Injection Points Using a File

If you have the current injection point locations available in a text file, you can use that to create injection points on the pins.

The text file should contain the location details in the following format.

```
#-----  
#source_name #loc_x #loc_y #layer_name #P/G  
<pinName>.<wholeNumber> <x-coordinate> <y-coordinate> <layerName> <POWER | GROUND>
```

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

An example of the text file is given below.

```
#source_name #loc_x #loc_y #layer_name #P/G
vdd1.0 14.04 3.264 metal2 POWER
vdd1.1 15.12 3.264 metal2 POWER
vdd1.2 6.48 3.264 metal2 POWER
vdd1.3 7.56 3.264 metal2 POWER
```

To use this file for the insertion of injection points, before launching Virtuoso, set the following environment variables in the `.cdsinit` file:

- ❑ Set `insertResNodes` to specify the name of the file that provides the location details for nodes to be created for EM analysis of wide pins.
- ❑ Set `stackViaArrayHandling` to "none" to allow the evaluation of EM violations at pins. By default, the pin resistance is considered as zero and EM violations are not calculated for pins.

After this, launch Virtuoso and open the layout view for EAD. Run extraction and EM analysis for the net. Layout EAD creates nodes at the specified injection points.

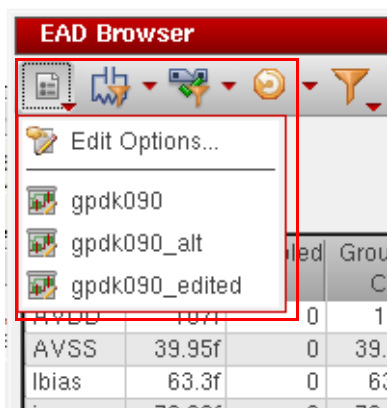
Extracting Parasitics in Layout EAD

Layout EAD extracts parasitics on a net-by-net basis, which makes the tool incremental and able to handle partial layouts.

Before extracting parasitics, you can configure the settings to customize the way EAD extracts parasitics for your layout. For more details, refer to [Customizing Extraction of Parasitics](#).

You can extract parasitics using the Extract Parasitics command in the [EAD Browser toolbar](#). For this, perform the following steps:

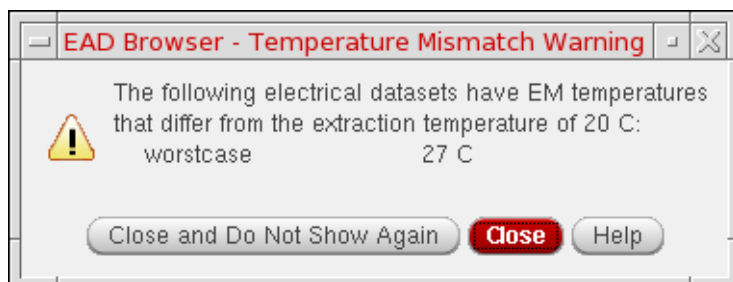
1. Choose the setup you require from the *Options* menu in the EAD Browser toolbar.



The specified setup is loaded. See [Managing EAD Setups](#) for more information.

Note: No parasitic extraction or EM checking is run by default; however, any stored parasitic information is retrieved and displayed in the summary pane. EM violations and J/Jmax values are not stored in the OpenAccess database and can only be displayed after you have run the EM check.

Any mismatches in the temperature values set for EM checking, extraction, and simulation are reported in a warning dialog.

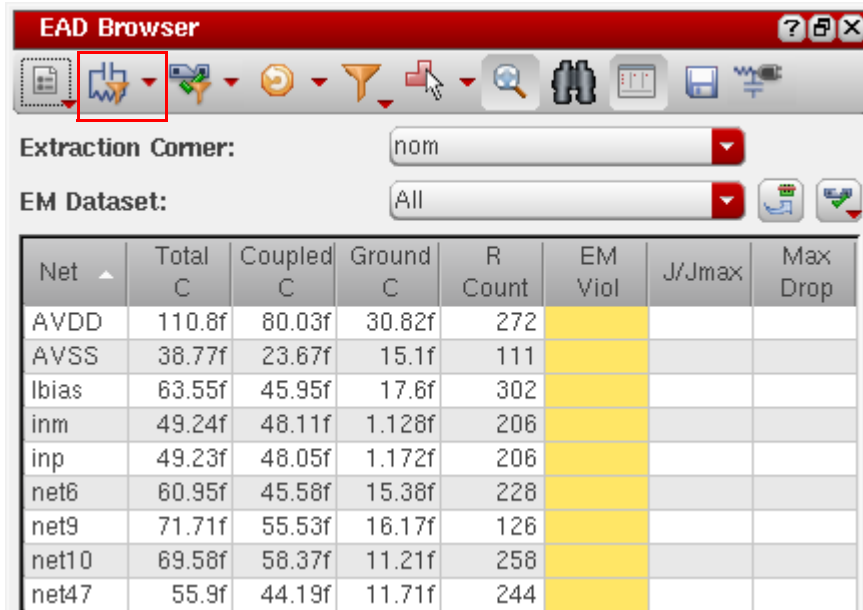


Virtuoso Electrically Aware Design Flow Guide

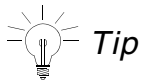
Extracting Parasitics in the Layout View

It is important that you use a consistent temperature throughout your flow; if the temperature used during simulation in ADE Assembler does not match that set for extraction and EM checking in Layout EAD, then your EM analysis is likely to be invalid.

2. Click *Extract Parasitics* on the toolbar of EAD Browser to extract parasitic information for the design.



Net	Total C	Coupled C	Ground C	R Count	EM Viol	J/Jmax	Max Drop
AVDD	110.8f	80.03f	30.82f	272			
AVSS	38.77f	23.67f	15.1f	111			
lbias	63.55f	45.95f	17.6f	302			
inm	49.24f	48.11f	1.128f	206			
inp	49.23f	48.05f	1.172f	206			
net6	60.95f	45.58f	15.38f	228			
net9	71.71f	55.53f	16.17f	126			
net10	69.58f	58.37f	11.21f	258			
net47	55.9f	44.19f	11.71f	244			



Tip

To extract parasitics for one or more selected nets, select their rows in the summary tab, right-click and choose *Extract Parasitics* from the context-sensitive menu.

The summary table is updated to show the parasitics (total, coupled, and grounded capacitance values and total number of resistors) for the listed nets.

- ❑ The extraction runs across all the extraction corners specified for a given design. This means that if the same design is open in two different layout windows and the EAD Browser is set up with different extraction corners in each, then extracting or updating in one window will extract and update results for both corners in both windows.
- ❑ By default, capacitance values in all tables are displayed in femtoFarad (fF). To vary the units depending on the magnitude of the capacitance values, disable the capacitanceFixedSuffix environment variable.
- ❑ By default, the tables show the maximum value of coupling capacitance between nets.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- ❑ By default, only terminals and instance terminals are considered as network terminal objects and a resistance path is computed to them. You can set the [extractToPins](#) environment variable to `t` so that pins are also considered as network terminal objects and a resistance path is computed to them.
- ❑ Violations of any of the supported constraints set in the Constraint Manager assistant are highlighted in red in the appropriate table cells. The supported constraints are:
 - [Max Resistance](#)
 - [Max Capacitance](#)
 - [Max Voltage Drop](#)
 - [Matched Capacitance](#)
 - [Max Coupling Capacitance](#)
 - [High Precision C Extraction](#)
 - [High Precision R Extraction](#)

For more information on these constraints, see [Appendix A, “Default Constraint Types”](#) in the [Virtuoso Unified Custom Constraints User Guide](#).

- ❑ You can also move the pointer over a net name to see a tooltip indicating whether the parasitic network for the net is complete or incomplete (incomplete networks are also highlighted in yellow).
3. Click one of the table cells to see more detailed information in the appropriate tab in the [Detail Pane](#). For example,
 - ❑ If you click a capacitance or resistance value, the *C* or *R* tab opens with a detailed breakdown of the selected value. See [Viewing and Analyzing Parasitics](#) for more information.
 - ❑ If you select capacitance values for two nets, the *Compare* tab opens allowing you to compare the information for the nets in question. See [Comparing Parasitic Values for Multiple Nets](#) for more information.
 4. If you make changes in the layout, you can update the parasitics at any time using the *Update Parasitics & EM* button or by pressing `Ctrl+u`.



Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

You can also right-click in a cell and update the parasitic and EM data, extract parasitics, and run the EM check for a selected net.

Net	Total C	Coupled C	Ground C	R Count	EM Viol	J/Jmax	Max Drop
lbias	63.55f	45.95f	17.6f	302	0	189.4m	1.133m
AVDD	110.8f	80.0f	30.8f	272	0	17.8m	1m
out	398.2f	273.5f	124.7f	111	82	20.78	6m
net10	69.58f	58.3f	11.28f	111	82	20.78	3m
net47	55.9f	44.19f	11.71f	111	82	20.78	3m
net6	60.95f	45.58f	15.37f	111	82	20.78	4m
inm	49.24f	48.1f	1.14f	111	82	20.78	06u
inp	49.23f	48.05f	1.18f	111	82	20.78	06u
net57	274.6f	241.4f	33.2f	111	82	20.78	1m
net9	71.71f	55.53f	16.18f	111	82	20.78	8m
AVSS	38.77f	23.67f	15.1f	111	82	20.78	38.94m

-  Update Parasitics & EM
-  Extract Parasitics
-  Run EM Check
-  Extract Parasitics & Run EM Check
- ☐ High Precision C Extraction
- ☐ High Precision R Extraction

For information on updating parasitics automatically while you edit, see [Using Automatic Update Mode](#).



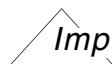
Tip

While the extraction process is running, you can use the `Ctrl + C` bindkey to interrupt the process. If you are using this command, ensure that the `checkInterrupt` environment variable is set to `t`.

High Precision C and R Extraction

For the critical nets in your design, you can selectively enable high precision C and R solvers. For this, select one or more nets in the Summary pane of the EAD Browser and select the *High Precision C Extraction* or *High Precision R Extraction*.

This creates a corresponding [High Precision C Extraction](#) or [High Precision R Extraction](#) constraint based on the options specified on the *Extraction* tab of EAD Options form. When you turn off high precision extraction, these constraints are deleted.



Important

High precision extraction performs a detailed analysis and therefore, is a time-consuming task. It is recommended to enable this only for critical nets.

When a port or a node is defined by a pin shape, the high precision R solver uses the geometry of the pin shape to construct a finite dimension port. This helps in correct emulation

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

of the current crowding or spreading at the current inlet or outlet for accurate resistance values. While doing this, the tool follows the rules given below to emulate the pin shape ports:

- If at least one of the edges of a pin shape is aligned with the boundary of layout shape, the current is emulated as flow in or flow out horizontally (in-plane with the layout layer) and perpendicular to the pin shape edge.
- If multiple edges of a pin are aligned with the shape boundary, the longest one is chosen as the port.
- If the pin shape is completely inside the layout shape, the current is emulated as flow in or flow out vertically (perpendicular to the layout layer) and the current flows through the whole area of pin shape.

In the following two cases, EAD marks the resistance network generated for a net by the high precision R solver as complete:

- When all extracted resistors are connected in only one network topology
- When all terminal nodes are included in the network

In all other cases, the resistance network of the net is marked as incomplete and the *Net Name* cell in the EAD Browser is highlighted in yellow.

Note: High precision R Solver provides better accuracy for a 45-degree wire resistance. Fast R extraction degenerates a 45-degree shape into small rectangular tiles in a step fashion, and then extracts these rectangular shapes. However, there occurs a loss of accuracy when using Fast R extraction for non-rectilinear shapes.

Customizing High precision R and C Extraction

Depending on your requirements, you can customize high precision R and C extraction by setting the fields in the *High Precision C Solver* and *High Precision R Solver* sections on the Extraction tab of the EAD Options form or by setting the environment variables described in this section.

- Each capacitance value, C (estimate), reported by the High Precision C solver is a statistical average with an associated standard error. This standard error is determined by the convergence target or goal set by the convergenceCThreshold environment variable or the *Default Convergence Target* field on the EAD Options form. The convergence target relates the statistical average to the actual capacitance value, C (actual).

For example, if the reported C (estimate) for a given net is 10.0ff with a convergence target of 4.0%, then the standard error is 4.0% of 10.0ff, which is 0.4ff. It can be said with 99.7% confidence that the difference between C (actual) and C (estimate) is not

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

more than 3 times of the standard error, which is $1.2ff$. This also means that the error in C (estimate) is not more than 3 times of the standard error.

The example in the table given below shows how different values of convergence target change the accuracy of the reported C (estimate) of $10.0ff$. With decreasing convergence targets, the maximum error in the C (estimate) value is reduced. That means, the reported C (estimate) becomes more accurate, but at an increased run time. If the convergence target is reduced to half, the run time increases four times (by 4x).

Convergence Target	Standard Error	Max Error in C (estimate) confidence	Run time (seconds)
4.0%	0.4ff	+/- 1.2ff	1s
2.0%	0.2ff	+/- 0.6ff	4s
1.0%	0.1ff	+/- 0.3ff	16s
0.5%	0.05ff	+/- 0.15ff	64s

Note: The table given above shows sample run time values. These values may differ for your designs.

- The default mesh density value used by the high precision R solver is 1.0 . If required, you can change this default by setting the `resistanceMeshDensity` environment variable or the *Default Mesh Density* field on the EAD Options form. A larger mesh density value creates a denser mesh, which also takes a longer time to run R extraction.

Note: You can use separate layer-wise mesh density values provided in a source text file by setting the `resistanceMeshDensityFileName` environment variable.

- You can use the `resistanceMesherTimeOut` environment variable to specify a threshold time to be compared with the estimated time to perform high precision R extraction for a net. If the estimated time is longer than the threshold time specified by this variable, high precision R extraction is stopped. A warning message is displayed and the remaining extraction for that net is not done.
- While performing high precision R extraction for a net, Layout EAD can save the mesh data, current density and voltage drop data in temporary binary files. These details can be later overlaid on the layout.

The file in which the mesh data is saved is specified by the `resistanceMeshFileName` environment variable. Similarly, the file in which the current density and voltage drop data is saved is specified by the `resistanceFieldFileName` environment variable. If required, you can set the values of these variables to a writable file location.

Also see: [Running EM Checks with High Precision R and C Extraction](#)

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- The width values displayed in the *W* column of the *Resistances* table are calculated by the mesh solver. They may not match the width drawn on the layout.
- By default, high precision R extraction performs via clustering and considers the rules defined on the Vias tab of the EAD Process Settings form. To disabled via clustering in high precision R extraction, set the hprDisableViaClustering environment variable to `t`.
- You can set the useNumberOfGPU environment variable to enable Graphical Processing Unit (GPU) computing for high precision C extraction and to specify the number of GPUs to be used.
- By default, high precision C extraction does not include vias in the modeling for capacitance extraction. You can include vias by setting the modelViaCapacitanceEffect environment variable to `t`, or by selecting the *Via Effects* check box on the Extraction tab of the Layout EAD Options form.
- You can include the non-gate forming poly shapes in the modeling for poly to diffusion capacitance extraction by setting the modelNonGateFormingPolyCap environment variable to `t`.
- By default, high precision C extraction models the actual metal fills that physically exist in the design. The fill metal modeling method can be selected from the *Metal Fill Modeling Method* drop-down list on *Extraction* tab of the EAD Option form.

By setting the hpcMetalFillDefFileName environment variable, you can enable the virtual metal fill modeling capability to study the impact of metal fills on capacitance values at an early design stage, even if the actual metal fills have not been added to the design database. When virtual metal fill modeling is enabled, high precision C extraction uses the available modeling information to add virtual metal fills into the geometry model for capacitance extraction. These metal fills are added only virtually and the flow does not change the design database, but the extraction process obtains the capacitance values by assuming that metal fills exist there physically.

If the design database contains actual metal fill shapes, they will be modeled along with the virtual metal fills as an effective way of incremental extraction. If there is an overlap of the actual and virtual metal fill, the actual metal fill will be considered for modeling, and the virtual fill will be ignored.

While setting the hpcMetalFillDefFileName environment variable, you can provide the virtual metal fill modeling information in the following ways:

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- ❑ In a model definition file in the following ASCII format:

<pre>num_of_blocks layer_name num_of_vertices_n X1 Y1 X2 Y2 ... Xn Yn min_net_fill_spacing min_X_fill_fill_spacing min_Y_fill_fill_spacing</pre>	} repeat num_of_blocks

where,

- **num_of_blocks** is an integer value that defines the number of custom virtual metal fill definition blocks given in this definition file;
- **layer_name** is a string value that defines the layer name in the technology file (ICT file)
- **num_of_vertices** is an integer value that defines the number of vertices used to define unit polygon metal fill shape. It was followed by num_of_vertices pair of floating point number to define the polygon, each represents X and Y coordinates of one vertex. These vertices can be listed either in clockwise or counter-clockwise order.
- **min_net_fill_spacing** is a float value that defines the minimum spacing between an edge of a net and a parallel edge of a metal fill shape;
- **min_X_fill_spacing** is a float value that defines the minimum spacing between any two horizontal edges of neighboring fill metal shapes
- **min_Y_fill_spacing** is a float value that defines the minimum spacing between any two vertical edges of neighboring fill metal shapes

A sample of the metal fill definition file is given below.

```
2
M1
4 0.0 0.0 0.12 0.0 0.12 0.12 0.0 0.12
0.3
0.3 0.3
M2
4 0.0 0.0 0.12 0.0 0.12 0.12 0.0 0.12
0.3
0.3 0.3
```

In this case, set the `hpcMetalFillDefFileName` environment variable to the path of the model definition file. EAD reads the information from this file to create virtual metal fills in the design.

- ❑ In the ICT file. If you want to use the virtual fill definition only from the ICT file, set the `hpcMetalFillDefFileName` environment variable to the name of a text file that contain only one line "0", which indicates that no custom virtual fill definition should be used.

For details about the format of virtual metal fill definition in ICT file, refer to the `metal_fill` command in the *Quantus QRC Techgen Reference Manual*.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- ❑ In both, the ICT file and the virtual metal fill definition file. In this case, you can set the `hpcMetalFillDefFileName` environment variable to the path of the text file that contains the modeling parameter definitions. The modeling parameters are read from both the sources. However, if the parameters for a particular layer are given in both the sources, the parameters given in the definition file have higher preference.
- If an ICT file contains via bias and spacing dependent contact bias, or VEEs (Via Edge Enlargements), high precision C extraction includes those in the capacitance models for extraction.

For via bias, the ICT file must include the `via_top_enlargement` and `via_bottom_enlargement` parameters in via definitions, as shown below.

```
via "ViA1" {  
    top_layer METAL_2  
    bottom_layer METAL_1  
    contact_resistance 7.9  
    min_top_encl 0.010  
    min_bot_encl 0.011  
    min_width 0.08  
    min_spacing 0.08  
    via_top_enlargement 0.1  
    via_bottom_enlargement 0.1  
}
```

For VEEs, the ICT file must include the `via_edge_enlargement` parameter in via definitions, as, shown below.

```
via_edge_enlargement  
vee_coco_spacings S1 S2 ... Sn  
vee_poco_spacings S1 S2 ... Sm  
vee_adjustments A11 A12 ... A1n  
                A21 A22 ... A2n  
                ...  
                Am1 Am2 ... Amn
```

In addition, a sub via mapping file in ASCII format that provides the mapping for layout via layer name to its corresponding tech file sub via layer name is also required. The file should be named as `subvia_layer_mapping.txt` and saved in the same directory that contains `eadTechFile`.

An example of the sub via mapping file is shown below.

```
-----  
CO DEMOS_odCont  
PO NMOS_odCont  
-----
```

In this example, CO is the layout layer name and DEMOS_odCont is the mapping sub via layer name.

While extracting capacitance for a layer in the layout, EAD uses the VEE table defined for the sub via mapped to it.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- By default, high precision capacitance extraction does not include modeling of spacing-dependent dielectric constant on metal interconnect layers. You can set the following SHELL environment variable before starting Virtuoso to enable variable dielectric effect:

```
setenv HPC_ENABLE_VARIABLE_EPSILON 1
```

finFET Support in High-Precision C Extraction

If your design contains multi-gate finFET devices, you can enable finFET modeling in high-precision C extraction by doing the following:

- Specify a fin width value using the finWidth environment variable and a fin spacing value using the finSpacing environment variable. These values should be greater than 0.0. The tool uses these two values to enable finFET modeling and to generate fin geometries.
- Set the modelViaCapacitanceEffect environment variable to `t`.
- Set the modelNonGateFormingPolyCap environment variable to `t`.

If the ICT file contains the `fin_width` and `fin_spacing` keywords for the active layer, the values defined by the finWidth and finSpacing variables will be overridden by the values of these two keywords. However, it is still required to set the values of these two variables to a non-zero value to enable finFET modeling.

iQuantus FS C Extraction (Advanced Nodes Layout EAD Only)

For sign-off level accuracy in extraction results, you can use iQuantus Field Solver (iQuantus FS) as the capacitance extraction engine. The C parasitic data extraction run by iQuantus FS is accurate.



iQuantus FS is supported only for foundry processes. To get details about the foundry processes for which this solver is supported, Contact Cadence Customer Support. For details on the iQuantus FS C extraction engine, refer to [Quantus QRC Extraction Users Manual](#).

Before using iQuantus FS engine for extraction, ensure that the following requirements are met:

- The following environment variables are set:
 - ❑ qrcTechFileName
 - ❑ qrcLayerSetupFileName

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

☐ qrcCapgenFileName

☐ hpcEngine

- The *Engine* field on the Extraction tab of the *EAD Options* form is set to `quantusFS`.

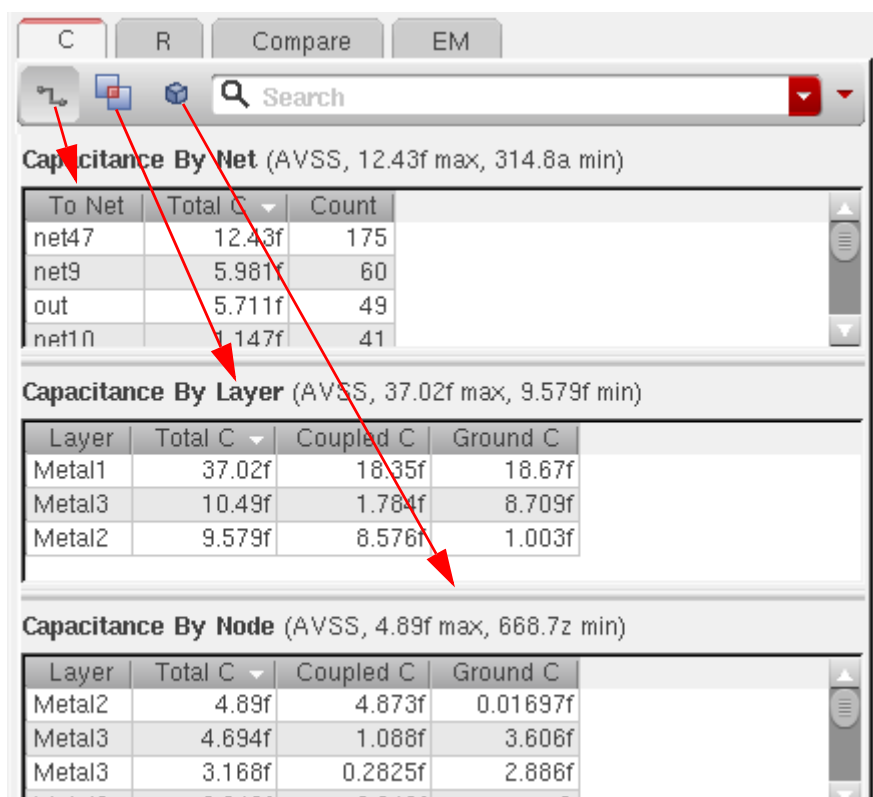
Follow these steps to use iQuantus FS for C extraction:

1. In the *High Precision C Solver* section on the *Extraction* tab of the *EAD Options* form, from the Engine drop-down list, select `quantusFS`.
2. To run extraction for all nets, click *Extract Parasitics for All Nets* from the *EAD Browser* toolbar.
3. To run extraction for the selected nets, do the following:
 - a. In the *Summary* pane of the *EAD Browser*, select one or more nets.
 - b. Right-click the selected nets, and then choose the *High Precision C Extraction* check box from the context-sensitive menu.
 - c. Click the *Extract Parasitics* command from the context menu.

Viewing and Analyzing Parasitics

This section describes how to use the EAD Browser to view and analyze the capacitance and resistance information for your design and explains what information is available for each.

1. Click a capacitance value in the summary pane to open the *C* tab in the detail pane.
2. Use the toolbar buttons to toggle capacitance information by net, by layer, or by node.



3. Click a resistance value in the summary pane to open the *R* tab and see the terminal-to-terminal resistance paths for the net in question, including the resistance value, resistor count, and from/to terminal names.

Note: If you have set the `resistanceMeshFileName` environment variable, the mesh data is saved while performing high precision R extraction for any net. You can later overlay that data on the net in the layout canvas. For more information, see [Viewing Mesh for Selected Nets](#).

4. Move the pointer over a *Total R* violation in the summary pane to see a tooltip detailing the violating resistance path, which is also highlighted in the *R* tab.

Note: If all the violations are not displayed in the tooltip, you can increase the size of tooltip by using the `maxTooltipLength` environment variable.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- Click a row in the *R* table to highlight the corresponding resistance path in the layout view.

The screenshot shows the 'Resistance Paths' window in the Virtuoso EDA tool. The window has tabs for 'C', 'R', 'Compare', and 'EM', with 'R' selected. The title bar indicates 'Resistance Paths (AVDD, 100 paths, 36.27 max, 10.2 min)'. The table below lists various resistance paths with columns for Total R, Count, From, and To. The first row, with a Total R of 36.27, is highlighted with a red box. A red arrow points from this row to a corresponding path in the layout view on the right. In the layout view, a white zigzag line represents the resistance symbol, and a red line points to it with the label 'Resistance symbol'.

Total R	Count	From	To
36.27	67	M3:S	M10:S
36.13	62	M3:S	M5:S_source...
36.07	66	M3:S_source...	M10:S
35.92	61	M3:S_source...	M5:S_source...
35.83	60	M3:S	M5:S_source...
35.82	65	M3:S	M10:S_source...
35.75	63	M3:S	M10:S_source...
35.74	59	M3:S	M5:S_source...
35.68	64	M3:S	M10:S_source...
35.66	58	M3:S	M5:S_source...
35.63	59	M3:S_source...	M5:S_source...
35.61	64	M3:S_source...	M10:S_source...
35.58	62	M3:S	M9:S
35.57	57	M3:S	M5:S_source...
35.55	63	M3:S	M10:S_source...
35.54	62	M3:S_source...	M10:S_source...
35.54	58	M3:S_source...	M5:S_source...
35.48	63	M3:S_source...	M10:S_source...
35.48	56	M3:S	M5:S_source...
35.45	57	M3:S_source...	M5:S_source...

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

If there is only one resistance path for a net, the resistance path is selected by default after you extract parasitics for that net. Only the resistors available on the resistance path are displayed in the Resistances table.

The screenshot shows the EAD Browser window with the following sections:

- Extraction Corner:** nom
- Electrical Dataset:** All
- Summary Table:**

Net	Total C	Coupled C	Ground C	R Count	EM Viol	J/Jmax	Max Drop
A	11.27f	0	11.27f	15			
AVDD	118.2f	76.26f	41.96f	250			
AVSS	57.55f	27.03f	30.52f	166			
- Resistance Paths (A, 1 paths, 356.8m max, 356.8m min):**

Total R	Count	From	To
356.8m	7	I1:PLUS	A
- Resistances (A, 7 values, 1.4 max, 1.413m min):**

R	Layer	W	L	Cuts	Cut Area
300.7m	Metal3	5.175	26.1265		
21.67m	Metal3	10.3	3.72		
33m	Metal3	4.545	3.0515		
1.413m	Metal3	10.3	0.244		
7.195m	Metal3	4.545	0.5455		
1.4	Via2			1	14.3295
1.4	Via2			1	0.0196

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

If you deselect the resistance path, all resistors that overlap with the net are displayed in the Resistances table.

EAD Browser

Extraction Corner: norm

Electrical Dataset: All

Net	Total C	Coupled C	Ground C	R Count	EM Viol	J/Jmax	Max Drop
A	11.27f	0	11.27f	15			
AVDD	118.2f	76.26f	41.96f	250			
AVSS	57.55f	27.03f	30.52f	166			

C R Compare EM IR Drop

Resistance Paths (A, 1 paths, 356.8m max, 356.8m min)

Total R	Count	From	To
356.8m	7	I1:PLUS	A

Resistances (A, 15 values, 1.4 max, 1.413m min)

R	Layer	W	L	Cuts	Cut Area
1.4	Via2			1	0.0196
1.4	Via2			1	0.0196
1.4	Via2			1	0.0196
1.4	Via2			1	14.3295
300.7m	Metal3	5.175	26.1265		
177.2m	Metal2	5.03	15.614		
33m	Metal3	4.545	3.0515		
32.69m	Metal2	5.58	3.39		
21.67m	Metal3	10.3	3.72		
18.91m	Metal2	5.03	1.8375		
14m	Via2			100	1.96
11.88m	Metal3	4.545	0.9405		
9.233m	Metal3	10.3	1.8375		
7.195m	Metal3	4.545	0.5455		
1.413m	Metal3	10.3	0.244		

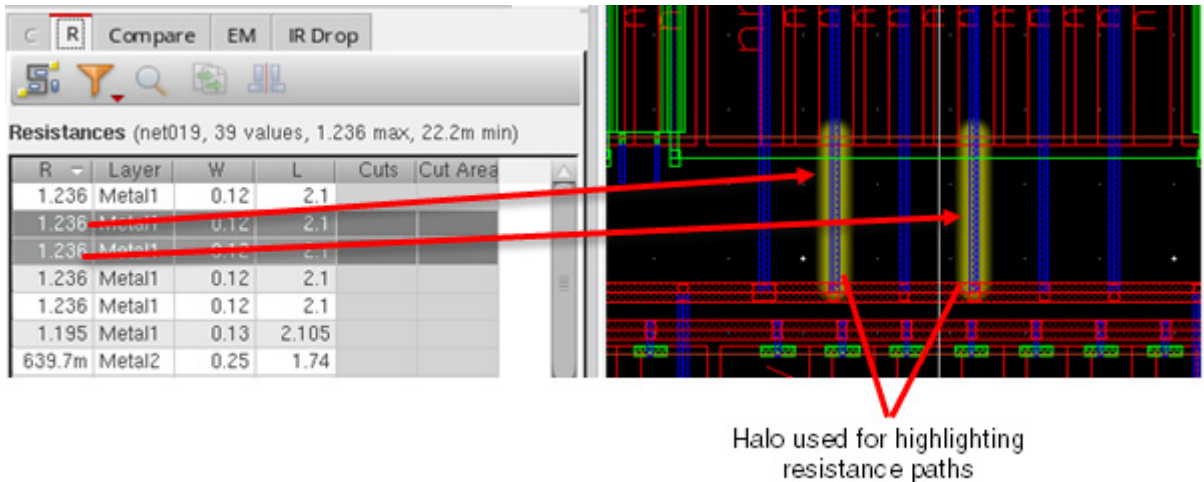
Resistance path deselected

Note: By default, Layout EAD uses a resistor symbol to highlight a resistance path on a net. You can set the `resistanceDisplay` environment variable to "halo" to use a halo

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

instead, as shown below.



Viewing Mesh for Selected Nets

If you have set the `resistanceMeshFileName` environment variable, while performing high precision R extraction for selected nets, mesh data is saved in the specified binary file. In addition, the *Highlight mesh* () icon also appears on the toolbar of the R table. The

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

names of nets for which high precision extraction was performed also appear as bold, as shown below.

EAD Browser

Extraction Corner: nom_rcx

Electrical Dataset: All

Net	R Count	EM Viol	J/Jmax	Max Drop
net17	36	0	0	0
net019	39	0	946.5m	516u
net021	76	0	0	0
net22	240	0	595.9m	1.953m
net035	0			
net036	189	0	0	0
net044	258			
outm	123	14	1.911	4.352m
outp	122	2	1.159	3.52m

Net for which high precision extraction is enabled appears in

Highlight mesh

To overlay the saved mesh data on a net in the layout canvas, do the following:

1. Click the resistance value for that net in the summary pane to open the *R* tab in the detail pane.
2. Click *Highlight mesh*.

A mesh is displayed on the selected net in the layout.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

3. Zoom in to a large level to view the mesh details clearly.

The screenshot shows the EAD Browser interface on the left and the Layout View on the right. The EAD Browser displays the 'Extraction Corner' as 'nom_rcx' and the 'Electrical Dataset' as 'EMData'. A table lists various nets with their R Count, EM Viol, J/Jmax, and Max Drop. 'net019' is highlighted with a red box. Below this, the 'Resistances' table for 'net019' is shown, listing values for R, Layer, W, L, Cuts, and Cut Area.

Net	R Count	EM Viol	J/Jmax	Max Drop
inm	26			
inp	36			
net14	276	4	1.093	1.695m
net17	36	0	0	0
net019	42			
net021	76	0	0	0
net22	240	0	595.9m	1.953m
net035	0			
net036	189	0	0	0

R	Layer	W	L	Cuts	Cut Area
824.9	Metal2	0	2.98		
227.3	Metal2	0.001	3.01		
220.1	Metal2	0.001	2.82		
201	Metal2	0.001	2.82		
171.5	Metal2	0.001	2.82		
15.17	Metal2	0.0115	2.87		
13.95	Metal2	0.0125	2.84		
12.64	Metal2	0.0095	2.04		
8.13	Metal2	0.0155	2.07		
7.350	Metal2	0.0155	1.88		

The Layout View on the right shows a zoomed-in view of the circuit layout. A yellow callout bubble points to a mesh overlaid on a selected net, labeled 'net019'. Other nets like 'net14' and 'net035' are also visible.

4. You can reset the layer filter to view the mesh on specific layers.

Important Points to Note

- The mesh data extracted during high precision extraction is saved in the binary files specified using the `resistanceMeshFileName` environment variable. After the analysis is complete, you can choose to delete the saved files.
- The *Highlight mesh* button on the toolbar of R tab is enabled only for those nets for which high precision extraction has been done and mesh data is available.
- Consider a case when you have displayed the mesh data for a net on the layout canvas. If you open any other layout cellview that contains a net with the same name for which mesh data is available, the *Highlight mesh* () button remains enabled because the binary file is available in the current working directory. However, when you click *Highlight mesh*, the tool detects that the mesh data file is not for the current cellview, and deletes it. To display mesh data for the net in the current layout cellview, you need to again extract the parasitics using high precision extraction.

Comparing Parasitic Values for Multiple Nets



You can view a video demonstration for this feature at [Comparing Parasitics and Resolving Electrical Violations](#).

If you select two or more values in the summary pane, the Compare tab opens comparing values for the selected nets. This can be useful if, for example, you have a pair of symmetric nets that need to be balanced in terms of parasitics. You can select the nets in the layout canvas and compare the coupled and grounded capacitances of the nets and the amount of coupling to other nets in the design.

To do this,

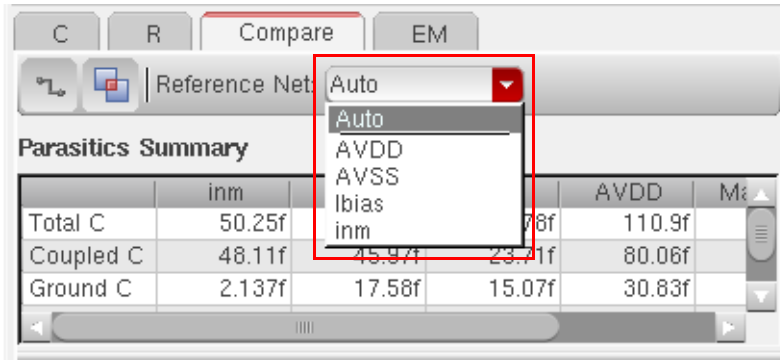
1. Select two or more values in the summary table.

The *Compare* tab opens showing the maximum difference and % error between the parasitic values on the selected nets.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

2. Use the *Reference Net* drop-down to choose which of the selected nets is the reference net for the percentage difference calculation.



The default is *Auto*, which means that the net with the smallest value is used as the reference for the difference calculation. You can change this as required, for example, when comparing the lengths of bus bits that are intended to be matched.

3. Use the two toggle buttons on the toolbar to show and hide tables comparing
 - ☐ Coupling capacitance values by net
 - ☐ Total capacitance values by layer

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- Click an entry in the *Coupling Capacitance by Net* table to highlight the net in the layout canvas.

The screenshot shows the EAD Browser interface with the 'Compare' tab selected. The 'Reference Net' is set to 'Auto'. The 'Parasitics Summary' table is displayed, followed by the 'Coupling Capacitance by Net' table. The 'Total Capacitance by Layer' table is also visible. The layout canvas on the right shows a circuit board with various nets highlighted in green and red, corresponding to the data in the tables. Red arrows indicate the flow of information from the tables to the layout canvas.

	inm	lbias	AVSS	Max Diff	% Error
Total C	45.58f	59.11f	57.09f	13.53f	29.68%
Coupled C	29.76f	44.86f	28.71f	16.16f	56.28%
Ground C	15.82f	14.25f	28.39f	14.14f	99.24%
R Count	38	431	191	419	2153.77%

Net	inm	lbias	AVSS	Max Diff	% Error
AVDD		25.88f	0.3148f	25.57f	8120.90%
AVSS	0.979f				
inm			0.979f		
inp	0.002882f		1.073f	1.07f	37117.81%

Layer	inm	lbias	AVSS	Max Diff	% Error
Metal1	1.179f	7.235f	37.02f	35.84f	3039.83%
Metal2	0.4653f	3.655f	9.579f	9.114f	1958.84%
Metal3			10.49f		0.00%
Poly	43.94f	48.22f		4.282f	9.74%

Coupling capacitance values between the nets in rows and columns. (when only a single coupling capacitance value is found in a row, Max Diff and % Error are blank)

Using Automatic Update Mode

By default, EAD does not extract dynamically as you update the design, but tracks the changes you make and highlights in red nets that need to be extracted. You then run the extraction by clicking the *Update Parasitics & EM* button in the EAD Browser toolbar (or by pressing **Ctrl+u**).

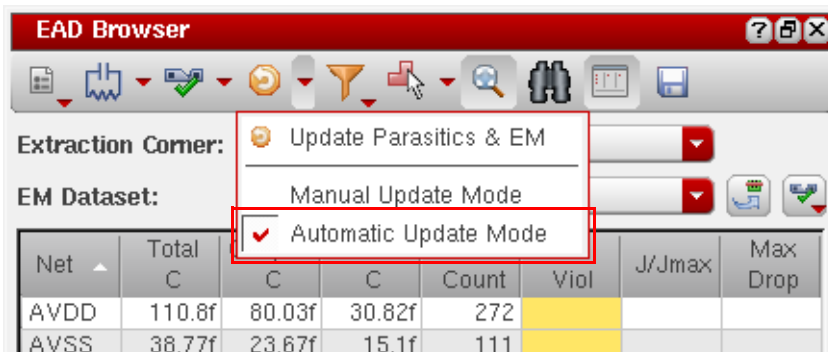
Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

You can use *Automatic Update Mode* to extract parasitic information while you are editing your design in the layout canvas. Any net that has previously been extracted is automatically updated in this mode, even if the parasitics have been deleted due to some other user action.

To update the parasitic information dynamically while you are editing the design,

- ➔ Choose *Automatic Update Mode* from the *Update Parasitics & EM* menu.



Now when you edit your design by, for example, routing a net in the canvas, you will see the values in the EAD Browser updating as you work.

This lets you see immediately the effect that your interactive changes are having on the parasitics for the design.

Note: If the same design is open in two different layout windows and the EAD Browser is set up with different extraction corners in each, then extracting or updating parasitics in one window with *Automatic Update Mode* enabled will extract and update results for both corners in both windows, even if the other window has update mode set to *Manual*.

Distributed Computation of Extraction Process

For large designs, you can distribute extraction on groups of nets to multiple batch jobs running on computer farms or dedicated servers.

Layout EAD provides the following SKILL functions to run and manage distributed jobs for extraction:

- [eadJobSubmit](#)
- [eadJobStatus](#)
- [eadJobClose](#)

Depending on your extraction requirements and the available resources, you can submit extraction jobs in any of the following ways:

- [Automatic submission of jobs for all nets to an LSF farm](#)
- [Automatic submission of jobs for all nets to dedicated servers](#)
- [Customized distributed computation of net groups using different job policies](#)

Automatic submission of jobs for all nets to an LSF farm

Preferred Scenario: Use this approach to submit jobs for extraction on all nets in your design to an LSF farm.

Steps:

1. Specify the following settings to enable the tool to automatically submit jobs:
 - ☐ Using the `ead_dp_net_group_number` variable, specify the number of net groups into which all the nets from the design are to be distributed equally.
 - ☐ Using the `ead_dp_batch_command` variable, specify the batch command to distribute a job to an LSF farm.
 - ☐ Using the `ead_dp_batch_command_option` variable, specify the option string used by the batch command defined by the `ead_dp_batch_command` variable.
2. Start the job submission using the [eadJobSubmit](#) SKILL function.

Note: As configuration information is generated automatically by the tool in this mode, specify the second argument of the [eadJobSubmit](#) SKILL function as "NULL".

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Layout EAD divides all the nets from the design into the specified number of net groups and distributes the extraction jobs using the given batch command and its options. The number of LSF jobs submitted is equal to the number of net groups.

After job submission, you can use `eadJobStatus` to check the status of running jobs. When `eadJobStatus` command returns `t`, which means all jobs have been successfully completed, you can run `eadJobClose` to collect the extraction results. The results from all the jobs will be combined into one OA database and displayed in the summary table of the EAD Browser.

Example:

Open a layout view in Virtuoso Layout EAD and set the following variables in the CIW:

```
ead_dp_net_group_number = 5
ead_dp_batch_command = "/grid/sfi/farm/bin/bsub"
ead_dp_batch_command_option = "-W 3:00 -q lnx64 -R \"OSREL==EE50 || OSREL==EE60\"
span[hosts=1] rusage[mem=8000]\" -P IC:6.1.6:RD:QUAL"
```

Run the following command in the CIW:

```
eadJobSubmit( hiGetCurrentWindow() "NULL" )
```

Layout EAD automatically divides all nets into five groups and submits the jobs using the given command and options. After that, run the following command to check job status:

```
eadJobStatus( hiGetCurrentWindow() "NULL" )
```

When all the jobs are complete, run the following command to collect the results from all jobs:

```
eadJobClose( hiGetCurrentWindow() "NULL" )
```

Automatic submission of jobs for all nets to dedicated servers

Preferred Scenario: Use this approach to submit jobs for extraction on all nets in your design to a set of dedicated servers in your setup.

Steps:

1. Specify the following settings to enable the tool to automatically submit jobs:

- ☐ Using the `ead_dp_server_names` variable, specify the list of available servers on which the distributed jobs can run.
- ☐ Using the `ead_dp_batch_command` variable, specify the batch command to distribute jobs to dedicated servers.
- ☐ Using the `ead_dp_batch_command_option` variable, specify the option string used by the batch command defined by the `ead_dp_batch_command` variable.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

2. Start the job submission using the `eadJobSubmit` SKILL function.

Note: Note: As configuration information is generated automatically by the tool in this mode, specify the second argument of `eadJobSubmit` as "NULL".

Layout EAD divides all nets from the design into a set of net groups where the number of groups is equal to the number of servers specified by the `ead_dp_server_names` variable and distributes the extraction jobs using the given batch command and its options.

After job submission, you then can use `eadJobStatus` to check the status of running jobs. When `eadJobStatus` returns `t`, which means all jobs have been successful completed, you can run `eadJobClose` to collect the extraction results. The results from all the jobs will be combined into one OA database and displayed in the summary table of the EAD Browser.

Example:

Open a layout view in Virtuoso Layout EAD and set the following variables in the CIW:

```
ead_dp_server_names = '("server1" "server2" "server3")  
ead_dp_batch_command = "/usr/bin/rsh"  
ead_dp_batch_command_option = "-l username"
```

Run the following command in the CIW:

```
eadJobSubmit(hiGetCurrentWindow() "NULL")
```

Layout EAD automatically divides all nets into three groups and submits the jobs using the given command, and options to servers `server1`, `server2`, and `server3`.

Run the following command in the CIW to check job status:

```
eadJobStatus(hiGetCurrentWindow() "NULL")
```

When all jobs finish and the above command returns `t`, run the following command to collect results from all jobs:

```
eadJobClose(hiGetCurrentWindow() "NULL")
```

Customized distributed computation of net groups using different job policies

Preferred Scenario: Use this approach when you want to submit the extraction jobs for a particular net group to a specific distributed job. In this case, you can have multiple customized job policies and specify which policy is to be used for each net group. You can specify these details in a job configuration file that is provided to the `eadJobSubmit` SKILL function.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Steps:

1. Create a **job configuration file** in the ASCII format that contains the layout cellview details of the layout cellview, and the names of files that define the net groups and job policies to be used for each net group.

The expected format of a job configuration file is given below.

```
<job_policy_definition_file_name>
<net_group_definition_file_name>
<oa_lib_top_level_path>
<oa_library_name>
<oa_cell_name>
<oa_view_name>
NULL          //reserved for future use
```

2. A job policy definition file in ASCII format that contains one default job policy and some custom job policies. The given number of custom policies is also specified in this file.

The expected format of a job policy definition file is given below.

```
batch_type batch_command      // batch type: 1=dedicated server, 2=LSF, 3=SGE
batch_command_option
setup_file_name
NULL
number_of_customized_job_policy // N – number of custom job policies
net_group_id batch_type batch_command
batch_command_option
setup_file_name
NULL
//reserved for future use
```

} default job policy definition block

} // reserved for future use

} Repeat N blocks. The net group with net_group_id will
the corresponding job policy defined here. Net groups
no customized job policy will use the default job policy
definition block.

3. A net group definition file in ASCII format that specifies the number of net groups, and the net details for each group.

The expected format of a job net group definition file is given below.

```
number_of_net_group           // M
net_group_id number_of_nets_in_this_group // P
net1_name
net2_name
.....
```

} Repeat P times

} Repeat M times

4. Start the job submission using the eadJobSubmit SKILL function.

Note: Specify the name of the configuration file created in step 1 in the second argument of the eadJobSubmit SKILL function.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Layout EAD reads the net group details from the net group definition file and submits the job for each net group as per the job policy definition attached to it.

Example:

Following are the examples of the three ASCII files.

Job Configuration File (*ead.control*)

```
./ead.option  
./ead.nets  
./Two_Stage_Opamp  
Two_Stage_Opamp  
DiffOpAmp  
routed_layout  
NULL
```

Job Policy Definition File (*ead.option*)

```
2 /grid/sfi/farm/bin/bsub ← Here, 2 specifies that this job policy uses an LSF farm  
-W 3:00 -q lnx64 -R "OSREL==EE50 || OSREL==EE60 span[hosts=1] rusage[mem=8000]" -P IC:6.1.6:RD:QUAL  
.cadence/dfil/EAD/1/setup/mySetup.ini  
NULL  
1 ← Here, 1 specifies that this job policy uses a dedicated server  
1 1 /usr/bin/rsh  
-l loginID server1  
.cadence/dfil/EAD/1/setup/mySetup.ini  
NULL
```

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Net Group Definition File (ead.nets)	
3	
1 2	
AVDD	
AVSS	
2 5	
lbias	
inm	
inp	
net14	
net019	
3 7	
net17	
net021	
net22	
net035	
net036	
net044	
outm	

Run the following commands in CIW to submit jobs, check job status, and collect job results:

```
eadJobSubmit( hiGetCurrentWindow() "ead.control" )
eadJobStatus( hiGetCurrentWindow() "ead.control" )
eadJobClose( hiGetCurrentWindow() "ead.control" )
```

As specified in the job policy definition, the tool runs extraction on net AVDD and AVSS on the dedicated server server1, whereas each one of the other two net groups is run as a separate job on the LSF farms.

Working with Job Policies

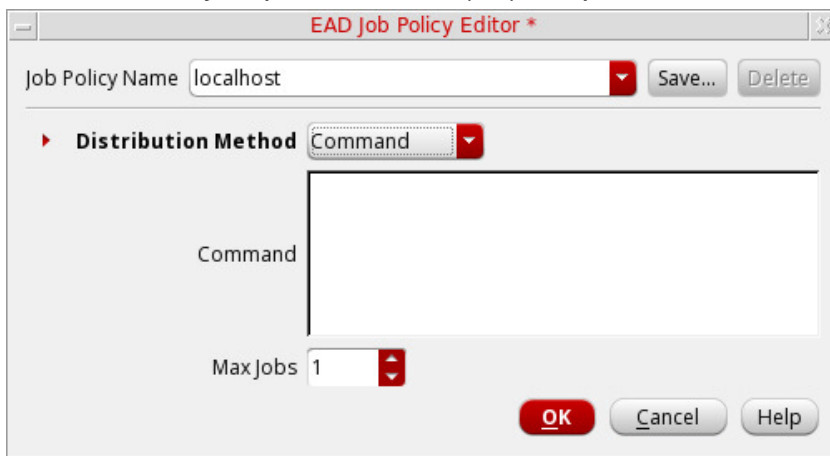
This section describes how to set up a job policy and define the methods of how distributed processing jobs are submitted to local or remote hosts. In EAD, you can use these jobs to extract parasitics and to run EM analysis.

- Setting Up a Job Policy
 - Editing a Job Policy
 - Deleting a Job Policy
- Using Distributed Processing Jobs

Setting Up a Job Policy

Follow these steps to set up a job policy:

1. On the *Environment* tab of the *EAD Options* form or on the *EAD Net Options* form, click the *Edit job policies* icon () to open the *EAD Job Policy Editor* form.



2. In the *Job Policy Name* field, specify a name for the job policy.

Note: If no job policy has been set up earlier, the *Job Policy Name* field displays `localhost` as the default job policy name, which can be specified by setting the `defaultJobPolicy` environment variable.

When using a job policy other than `localhost`, ensure that *Max Jobs* field is set to a value less than or equal to the number of available licenses.

3. From the *Distribution Method* drop-down list, choose one of the following methods:
 - *Command*: To submit the extraction job to a job distribution engine.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- ☐ *LBS*: To submit the extraction jobs to the Cadence load balancing software.
- ☐ *Local*: To run extractions on the local host.
- ☐ *Remote-Host*: To run extractions on a remote host you specify.

4. Depending upon the *Distribution Method* you selected, specify the fields described as follows:

Distribution Method	Fields	Description
<i>Command</i>	<i>Command</i>	<code>bsub</code> command to submit a job for batched execution by a distributed load-sharing batch system
	<i>Max Jobs</i>	Maximum number of simultaneous jobs that you want to run.
	<i>Queue</i>	Name of the queue to which the job is submitted
<i>LBS</i>	<i>Queue</i>	Name of the queue to which the job is submitted
	<i>Max Jobs</i>	Maximum number of simultaneous jobs that you want to run.
<i>Remote-Host</i>	<i>Hosts</i>	List of remote hosts

5. Click *Save* to save the changes. The *EAD Save Job Policy* form opens, as shown in the following figure:

6. The *Name* field displays the name of the job policy you specified in the *EAD Job Policy Editor* form.

7. From the *Location* list, select the directory where you want to save the job policy. You can save the job policy in one of the following directories:

- ☐ `.cadence` directory in your current directory

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

- ❑ .cadence directory in your home directory

8. Click *Save* to save the changes.

You can now extract parasitics and run EM analysis for the nets with the job policy you defined. See the [Using Distributed Processing Jobs](#) section for more information.

Editing a Job Policy

Follow these steps to edit a job policy:

1. On the *EAD Job Policy Editor* form, from the *Job Policy Name* drop-down list, select the job policy that you want to edit.
2. From the *Distribution Method* drop-down list, select a distribution method.

Note: The asterisk (*) displayed in the title of the *EAD Job Policy Editor* form indicates that the currently selected job policy is being edited.

3. Depending upon the *Distribution Method* you selected, update other fields, as required.

See the [Setting Up a Job Policy](#) section for the description of the fields in the *EAD Job Policy Editor* form.

4. Click *Save*. The *EAD Save Job Policy* form opens.
5. From the *Location* list, select the directory where you want to save the updated job policy.
6. Click *OK*.

Deleting a Job Policy

- ➔ In the *EAD Job Policy Editor* form, from the *Job Policy Name* drop-down list, select the job policy you want to delete, and then click *Delete*.

The selected job policy is deleted.

Note: You cannot delete the default job policy.

Using Distributed Processing Jobs

In EAD, you can use the distributed processing jobs to extract parasitics and to run EM analysis for all the nets or for the selected nets.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

Note: With a single EXL or MXL license, you can run four different jobs simultaneously. To run more than four jobs at a time, you need extra licenses.

To use distributed processing jobs for all the nets, do the following:

1. Click the *Environment* tab on the *EAD Options* form.
2. In the *Job Policy Setup* section, from the *Default Job Policy* drop-down list, select a job policy you have already defined. If you want to define a new job policy, click *Edit job policies*. See the [Setting Up a Job Policy](#) section for more information.
3. Click OK to close the form.
4. Click one of the following commands on the *EAD Browser* toolbar:
 - ☐ *Extract Parasitics For All Nets*: To use the jobs for extracting parasitics for all nets.
 - ☐ *Run EM Check On All Nets*: To use the jobs for running EM checks on all nets.
 - ☐ *Extract Parasitics & Run EM Check*: To use the jobs for both, extracting parasitics and running EM checks.

The jobs are submitted according to the methods defined in the selected job policy and results are updated in the table.

To use the distributed processing jobs for the selected nets, do the following:

1. In the *Summary* pane of the *EAD Browser*, select the nets for which you want to extract parasitics or run EM analysis.
2. Right-click the selected nets and then choose *Net Options*. The *EAD Net Options* form opens, as shown in the following figure:



3. From the *Job Policy* drop-down list, select the job policy that you want to use.

Note: You can also click the *Edit job policies* icon () to open the *EAD Job Policy Editor* form, where you can update the existing job policies or define a new job policy. See the [Setting Up a Job Policy](#) section for more information.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

4. Click *OK* to save the changes.
5. In the *Summary* pane of the *EAD Browser*, right-click the selected nets and then choose one of the following options from the context menu:
 - ☐ *Extract Parasitics*: To use the jobs only for extracting parasitics.
 - ☐ *Extract Parasitics With Policy*: To select another job policy to extract parasitics for the selected nets.
 - ☐ *Run EM Check*: To use the jobs only for running EM analysis.
 - ☐ *Extract Parasitics & Run EM Check*: To use the jobs for both, extracting parasitics and running EM analysis.

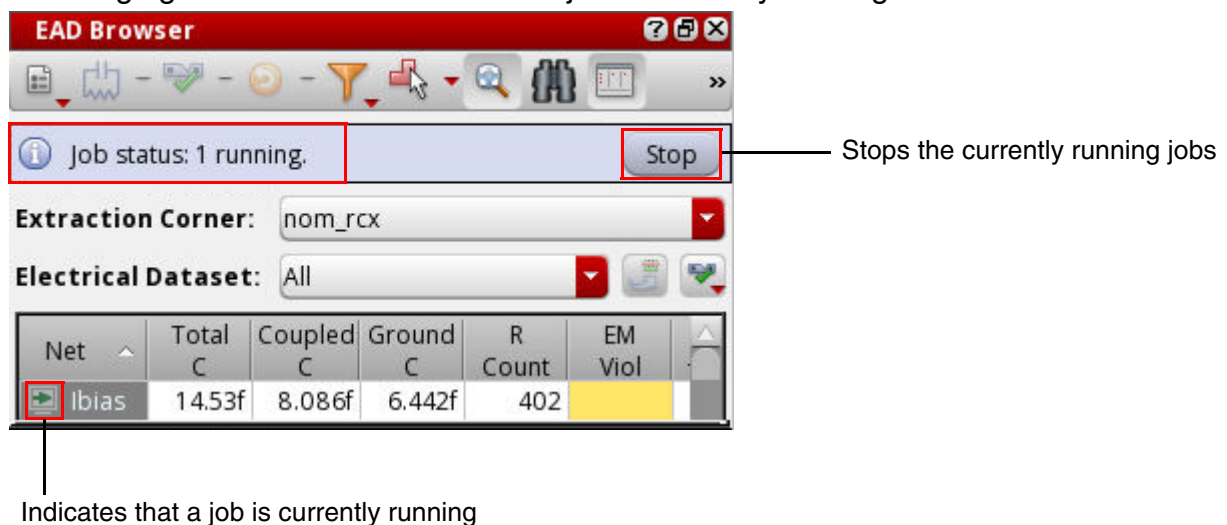
The jobs are submitted according to the methods defined in the selected job policy and results are updated in the table.

Important point to note:

- All the four engines are supported for distributing processing jobs.
- If you make any changes in the design after submitting the distributed processing jobs, the extraction or EM results may be inconsistent.

Stopping and Resubmitting a Job

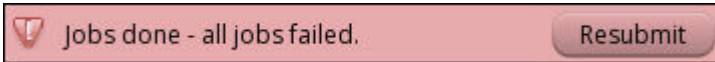
Status of the submitted jobs is shown below the *EAD Browser* toolbar. For example, the following figure shows that a submitted job is currently running:



Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

You can click *Stop* to stop the currently running jobs. Alternatively, you can right-click the selected nets, and then choose *Stop Job* from the context menu to stop the jobs. The jobs that have not been finished are stopped and the following status is displayed:

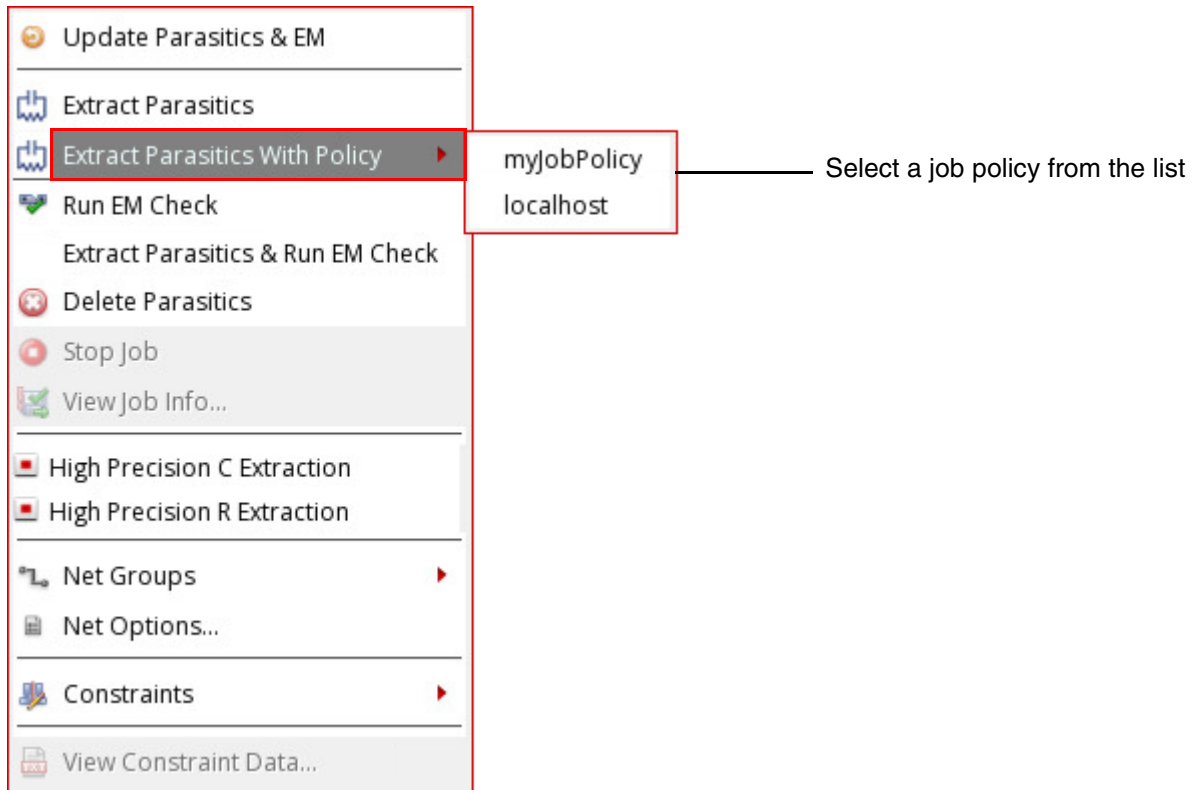


A 'cross' icon (☒) is displayed next to the nets for which the jobs have failed or stopped. You can click *Resubmit* to submit the unfinished jobs again. After all the jobs are executed successfully, the extraction results are updated in the *Detail* pane.

Extracting Parasitics With a Policy

You can also extract parasitics for the selected nets using the *Extract Parasitics With Policy* command:

1. In the *Summary* pane of the *EAD Browser*, select a net. To select multiple nets, hold down the *Ctrl* key while you click the other nets you want to select.
2. Right-click the selected nets to open the context-sensitive menu, and then point to *Extract Parasitics With Policy*. A list of job policies is displayed, as shown in the following figure:



Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

3. Select the job policy that you want to use to extract parasitics. The extraction results for the selected net are updated in the *Detail* pane.

Job Statuses

When you submit a distributed job, an icon is displayed next to the nets. This icon indicates the current status of the submitted jobs. The following table shows the different statuses of the jobs and the corresponding icons that are displayed next to the nets for which the jobs have been submitted.

Icon	Status of the Job
	Pending
	Running
	Failed

Stopping a Running Job

Do the following to stop a running job that has not finished yet:

- ➔ Right-click the nets for which the jobs are currently running, and then choose *Stop Job*.

Viewing Job Information

You can view the following information about a failed job:

- *Error Log*: Displays information about the failed job
- *Nets*: Displays a list of nets for which the jobs have failed
- *Output Log*: Displays information such as, who submitted the job, name of host to which the job was submitted, home and working directory, time when the job was submitted
- *Status*: Displays the status of the jobs
- *Virtuoso Log*: Displays the log file for the job's Virtuoso session

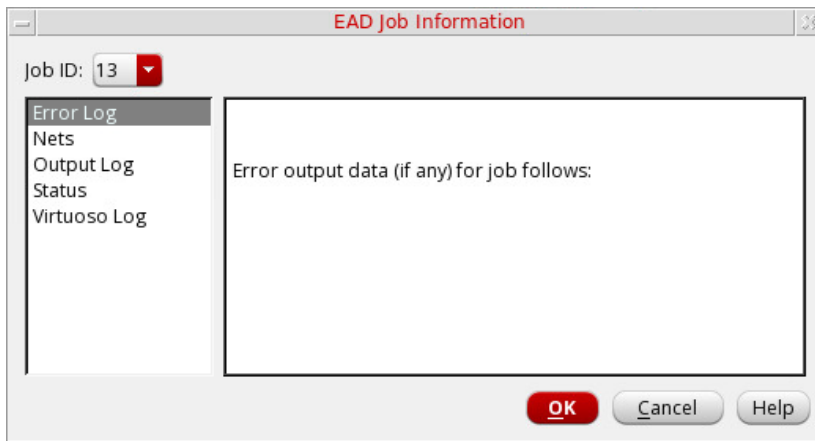
Follow these steps to view these information about the failed jobs:

1. In the *Summary* pane of the *EAD Browser*, right-click a net, and then choose *View Job Info* from the context menu.

Virtuoso Electrically Aware Design Flow Guide

Extracting Parasitics in the Layout View

The *EAD Job Information* form opens, as shown in the following figure.



Note: The *View Job Info* command is enabled in the context menu only for the nets for which the jobs have failed.

2. In the left pane, select the information that you want to view. The selected information is displayed in the right pane.

Performing Electromigration Checks and Fixing Violations

This section describes the electromigration (EM) checking capabilities in Layout EAD.

Electromigration is a general term used to describe failure mechanisms in the metal wires of a chip caused by the movement of metal atoms in a wire because of high current stress. As electrons move through a metal wire, they collide with the atoms in that wire. These collisions cause wires to become heated and, if enough electrons collide with a metal atom over a period of time, the metal atom can move in the direction of the electron flow, potentially creating

- An open circuit if the wire breaks
- A short to an adjacent metal wire if enough atoms move to the same location

EM checking is performed to analyze the power grid and signals of the design for any density violations of the DC and AC current through the metal conductor lines, vias, and contacts. These violations can then be corrected by limiting current density in the affected areas of the design.

You can use Layout EAD to

- Run EM checks
- Analyze the results and identify violations in the EAD Browser
- Highlight problem areas and display detailed EAD-related information in the layout canvas

You can then edit your design directly in the layout window to address the issues found and rerun the analysis either automatically or on-demand to ensure that the problems were resolved.

This section covers the following topics:

- [Running EM Analysis to Identify Violations](#)

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

- ☐ Verifying the Setup for EM Analysis
- ☐ Running an EM Check
- ☐ Running EM Checks with High Precision R and C Extraction
- ☐ Running EM Checks on a CUSTOM Dataset
- Viewing EM Information in the EAD Browser
- Viewing Electrical Information in the Layout Window
 - ☐ Using Info Balloons
 - ☐ Using Point-To-Point Info Balloons
 - ☐ Using Measurement Display
- Highlighting and Fixing Violations in the Layout Window
- Fixing Violations Using EM Driven Trunk Optimization
 - ☐ Optimizing EM Driven Trunks
- Running EM Analysis on Power Grids
 - ☐ Running EM Analysis on the Scratch Cellview
 - ☐ Running EM Analysis on the Original Grid
 - ☐ Modifying the Original Layout
- Generating EAD Dataset

Running EM Analysis to Identify Violations

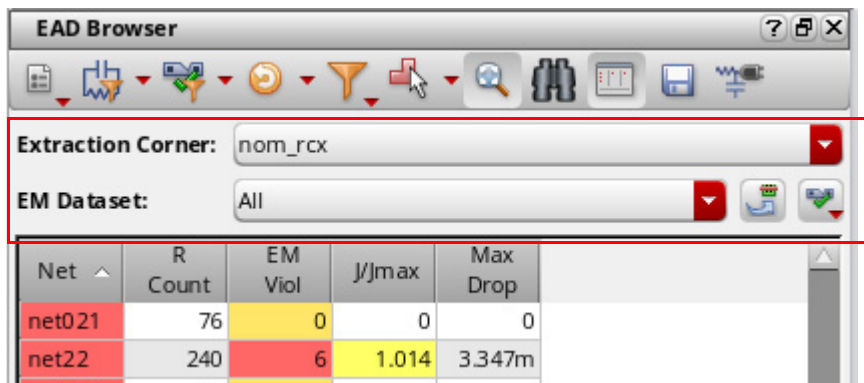
Layout EAD uses the electrical data produced by simulation in ADE Assembler and the parasitic information extracted from the layout view to analyze your design against the EM checks you have enabled and highlight any EM issue that exist.

This section describes the settings you need to make before running the EM check, how to run the EM check itself, and how to run an EM check when there is no simulation data available from ADE Assembler.

Verifying the Setup for EM Analysis

Use the *Extraction Corner* and *EM Dataset* fields at the top of the EAD Browser to specify the corner for which results are to be displayed and the dataset to be used to run the EM checks and display results.

You can also update the electrical data from ADE Assembler and enable or disable specific checks for the selected dataset.



To set these options,

1. Choose the *Extraction Corner* for which you want to display results from the drop-down list.

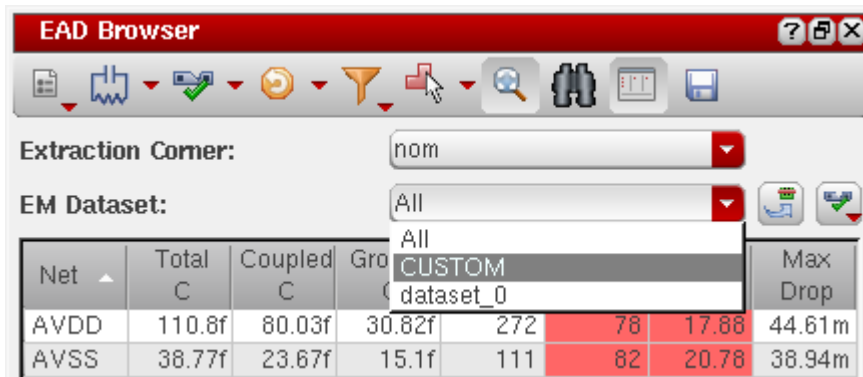
You define these extraction corners during EAD Setup in ADE Assembler. See [Setting up Testbenches and Corners](#) for more information.

Each corner must also be mapped to an ICT file containing the required process information. See the [EAD Process Settings – Corners](#) tab for more information.

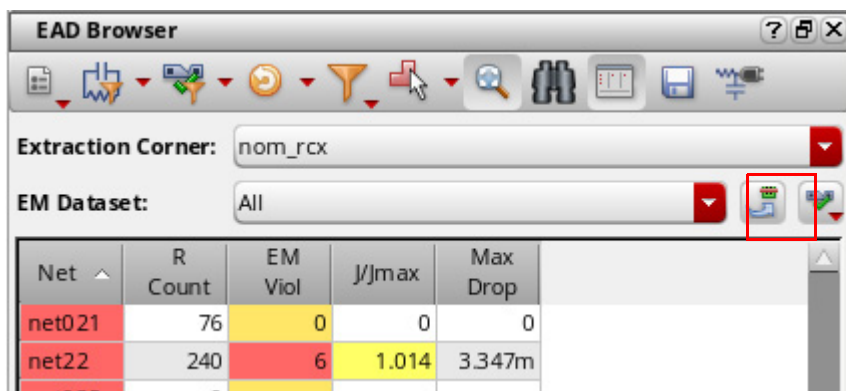
Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

2. Choose the *EM Dataset* to be used for the EM check.



- ☐ The default is *All*, which specifies that EM results are reported for all available datasets.
 - ☐ You can limit the results to a particular dataset by choosing that dataset from the drop-down list.
 - ☐ The *CUSTOM* dataset is a special dataset that lets you enter current values manually in cases where there is no current data available from ADE Assembler or if you want to enter your own currents in order to perform a what-if analysis. See [Running EM Checks on a CUSTOM Dataset](#) for more information.
3. Use the *Update with latest electrical data* button to reload the electrical data for the datasets that have changed in ADE Assembler since the datasets were last transferred to the layout.

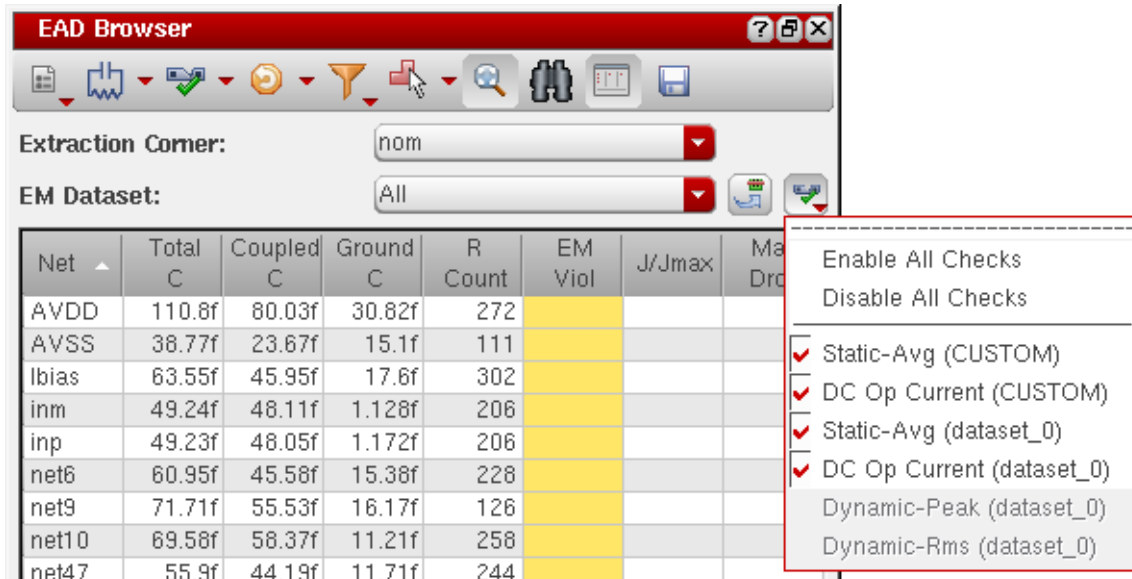


Use this if you have updated or created new datasets in ADE Assembler.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

- Use the *Select EM checks* drop down to enable or disable all or selected checks for the selected datasets.



Important

The check boxes for the *Dynamic-Peak* and *Dynamic-RMS* checks are enabled only when the *Dynamic Analysis* check box on the **EM** tab of the EAD Options form is checked.

The following table describes the available EM checks, the type of current that is checked and the EM rule that is applied for each check.

EM Check	Type of Current Used	EM Rule Applied in the Check
<i>DC Op</i>	Current from the DC Operating Point analysis	From ICT File: em_jmax_dc_avg From emDataFile: avgCurrentDensSpecList
<i>Static-Avg</i>	Average current from the Transient analysis	From ICT File: em_jmax_ac_avg From emDataFile: avgCurrentDensSpecList

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

<i>Dynamic-Peak</i>	Peak current flowing through the resistor for the specified time of Transient Analysis. You can specify time in the <i>Dynamic Analysis</i> section on the <i>EM</i> tab of the Layout EAD Options form.	From ICT File: <code>em_jmax_ac_peak</code> From emDataFile: <code>ACpeakCurrentDensSpecList</code> or <code>peakCurrentDensSpecList</code> Note: <code>peakCurrentDensSpecList</code> is same as <code>ACpeakCurrentDensSpecList</code> , but when both the rules are available, <code>ACpeakCurrentDensSpecList</code> is used for the nets with signal type not set to power or ground.
<i>Dynamic-Rms</i>	Rms current flowing through the resistor for the specified time of Transient Analysis. You can specify time in the <i>Dynamic Analysis</i> section on the <i>EM</i> tab of the Layout EAD Options form.	From ICT File: <code>em_jmax_ac_rms</code> From emDataFile: <code>rmsCurrentDensSpecList</code>

Note: Layout EAD does not support the `avgAbsCurrentDensSpecList`, `userDefinedCurrentDensSpecList`, and `DCrmsCurrentDensSpecList` EM rules from emDataFile.

If any dataset has been deleted from the constraints view, its name will appear in the ~~striketrough~~ font style. You can remove such datasets from this list by using the *Delete Electrical Data* command on the EAD toolbar.

Running an EM Check

You run the EM check using the *Run EM Check* button in the EAD Browser toolbar.



Click once to run the EM check for the selected extraction corner based on the values in the selected datasets and the parasitics extracted from the layout view. If there are no datasets available, you are prompted to load them from the schematic.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

Click the drop-down arrow to see the additional commands listed in the table below:

Command	Lets you...
<i>Extract Parasitics & Run EM Check</i>	Run parasitic extraction and then perform the EM check
<i>Update Electrical Data, Extract Parasitics, & Run EM Check</i>	Reload any updated electrical data from ADE Assembler for the selected datasets, run parasitic extraction, and then perform the EM check
<i>Update Electrical Data</i>	Transfers the electrical datasets from ADE Assembler or schematic to the layout. While doing this, if EAD finds that the name of the dataset being transferred matches with the name of an existing dataset at the layout side, it compares the creation timestamps of the two. If their timestamp is same, no action is taken. If their timestamp is different, the tool shows a dialog box and prompts you to confirm whether the dataset at the layout side should be overwritten. If you confirm, it deletes the existing dataset and copies the newer version to the layout. Before deleting the dataset at the layout side, EAD ensures that the ADE Assembler results database is accessible. If, due to some reason, the ADE Assembler results database is not accessible, the transfer of that dataset is skipped, and the version already saved with the layout is retained.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

Delete Electrical Data

Delete any terminal currents stored in the layout view for selected datasets.

This command displays the Dataset Deletion Dialog that shows a list of all the available electrical datasets. Choose the datasets you want to delete in the and click *Delete* to remove them from the design. EM checks and results are updated automatically.

Note: A dataset name displayed in ~~striketrough~~ font style implies that it has already been deleted from the constraints view. You can safely delete it from Layout EAD too.

Disable EM

Disable EM checking for the current Layout EAD session.

Selecting this option removes all EM-related data from the browser tables and disables the *EM* tab.

Important

(Advanced nodes Only) EAD supports color-based EM rules and can perform checks based on the color of shapes.

To run an EM check and view the results in the EAD Browser,

1. Click the *Run EM Check* button in the EAD Browser toolbar.

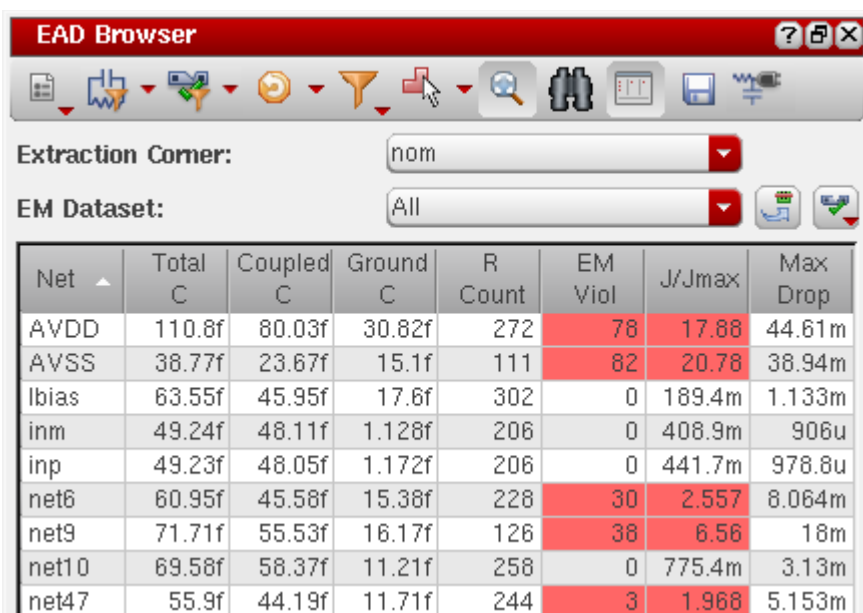
Layout EAD runs the check, matching EM statements in the order specified in the ICT file for the given layer, with the following exceptions:

- ☐ For vias where the current direction is known, those statements with matching direction are processed first
- ☐ For single vias only, an EM statement matching a via with the keyword SINGLE will be selected over other matching statements regardless of the order
- ☐ For terminals that have multiple pins associated with them, each pin is currently treated as a voltage source (at 0V) instead of a current source

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

When the check is finished, EAD updates the summary table to show color-coded electromigration information for the nets in the design. The color coding is based on the settings in the [Environment](#) form.



EAD Browser

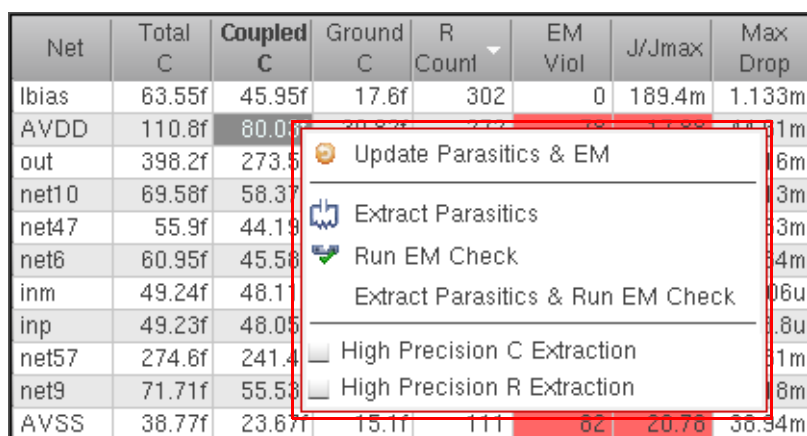
Extraction Corner: nom

EM Dataset: All

Net	Total C	Coupled C	Ground C	R Count	EM Viol	J/Jmax	Max Drop
AVDD	110.8f	80.03f	30.82f	272	78	17.88	44.61m
AVSS	38.77f	23.67f	15.1f	111	82	20.78	38.94m
lbias	63.55f	45.95f	17.6f	302	0	189.4m	1.133m
inm	49.24f	48.11f	1.128f	206	0	408.9m	906u
inp	49.23f	48.05f	1.172f	206	0	441.7m	978.8u
net6	60.95f	45.58f	15.38f	228	30	2.557	8.064m
net9	71.71f	55.53f	16.17f	126	38	6.56	18m
net10	69.58f	58.37f	11.21f	258	0	775.4m	3.13m
net47	55.9f	44.19f	11.71f	244	3	1.968	5.153m

- ☐ *EM Viol* shows the number of electromigration violations for each net.
- ☐ *J/Jmax* shows the worst case ratio of current to maximum current. A value greater than or equal to 1.0 is a violation. The higher the value, the more serious the violation.
- ☐ *Max Drop* shows the maximum IR drop between two terminals on the net.

2. Right-click in one of the cells to open the context menu.



Net	Total C	Coupled C	Ground C	R Count	EM Viol	J/Jmax	Max Drop
lbias	63.55f	45.95f	17.6f	302	0	189.4m	1.133m
AVDD	110.8f	80.03f	30.82f	272	78	17.88	44.61m
out	398.2f	273.5f	124.7f	206	0	408.9m	906u
net10	69.58f	58.37f	11.21f	258	0	775.4m	3.13m
net47	55.9f	44.19f	11.71f	244	3	1.968	5.153m
net6	60.95f	45.58f	15.38f	228	30	2.557	8.064m
inm	49.24f	48.11f	1.128f	206	0	408.9m	906u
inp	49.23f	48.05f	1.172f	206	0	441.7m	978.8u
net57	274.6f	241.4f	33.2f	111	82	20.78	38.94m
net9	71.71f	55.53f	16.17f	126	38	6.56	18m
AVSS	38.77f	23.67f	15.1f	111	82	20.78	38.94m

- Update Parasitics & EM
- Extract Parasitics
- Run EM Check
- Extract Parasitics & Run EM Check
- ☐ High Precision C Extraction
- ☐ High Precision R Extraction

The commands here let you perform the following actions for the selected net:

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

- ❑ Update parasitic information and EM data
- ❑ Re-extract the parasitics and rerun the EM check (either separately or with a single click)
- ❑ Enable the high precision C and R extraction, which creates constraints to extract capacitance and resistance values with a higher degree of accuracy. See [Running EM Checks with High Precision R and C Extraction](#) for more information on high precision R and C extraction.

3. Click one of the EM table cells in the summary pane to open the EM tab in the [Detail Pane](#).

See [Viewing EM Information in the EAD Browser](#) for more information.

Important

By default, the EM results are not saved when you save the layout view. To ensure that the EM results are saved into the OA database when you save the layout, set the [writeEMResultsToOA](#) environment variable to `t` before running the EM checks. If the results are saved into the OA database, next time when you open the layout view, the EM results will be displayed in the Summary pane of the EAD Browser.

Related Environment Variables

- [distributeTermCurrent](#)
- [printAdjustedCurrents](#)

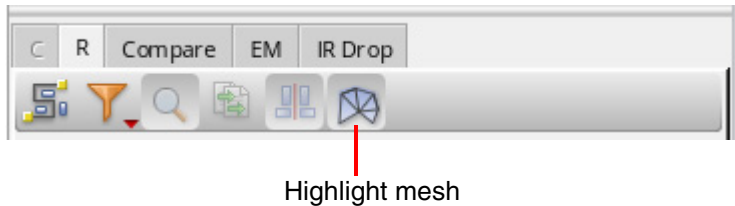
Running EM Checks with High Precision R and C Extraction

For critical nets, you can perform high precision R and C extraction to get the detailed resistance or capacitance data that can be used to further perform EM checks for those nets. Therefore, before performing high precision R extraction, set the [resistanceMeshFileName](#) and [resistanceFieldFileName](#) environment variables to save the resistance mesh and field data in binary files. Later, you can use this saved data to run EM checks and to overlay this data on the layout canvas and identify high current density areas and EM hot spots. If you need to extract data from multiple cellviews, you can use the [hprStampDataFileName](#) environment variable to add cell identification information to the names of mesh and data files.

Virtuoso Electrically Aware Design Flow Guide

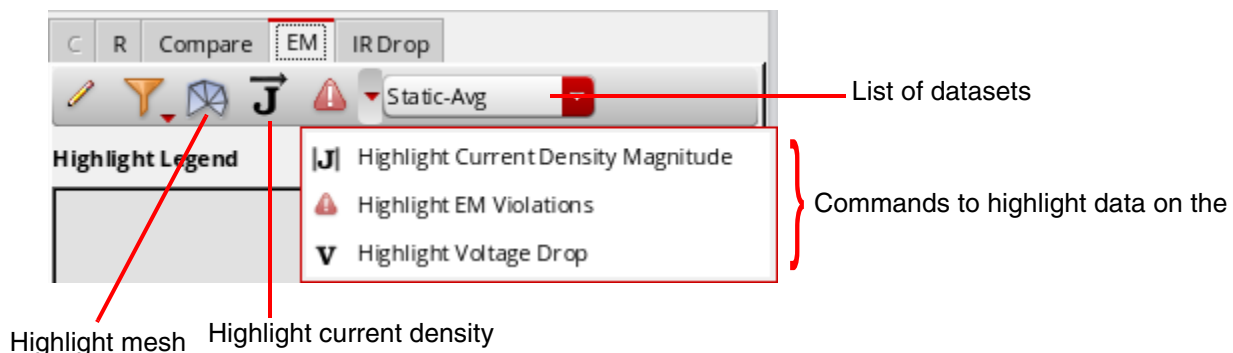
Performing Electromigration Checks and Fixing Violations

When you perform high precision R extraction, the *Highlight mesh* command appears on the toolbar of the *R* tab displayed for that net:



In addition, the following additional commands appears on the toolbar of the *EM* tab displayed for that net:

- *Highlight current density*
- A drop-down list with the names of datasets



You can use these commands to run EM checks for the nets.

For more details, refer to the following topics:

- [Viewing Current Density and Vectors](#)
- [Viewing Voltage Drop](#)
- [Identifying EM Hot Spots](#)

Viewing Current Density and Vectors

If you have performed high precision resistance analysis for a net and saved the data in binary files, you can run EM checks and overlay its current density and magnitude on the layout canvas.

To run EM checks and overlay the current density for a net, do the following:

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

1. Click the resistance value for that net in the summary pane of the EAD Browser to open the *R* tab in the details pane.
2. Using the current dataset drop-down list, select a dataset for which you need to view the current density.
3. On the toolbar of the *EM* tab, click *Highlight current density*.

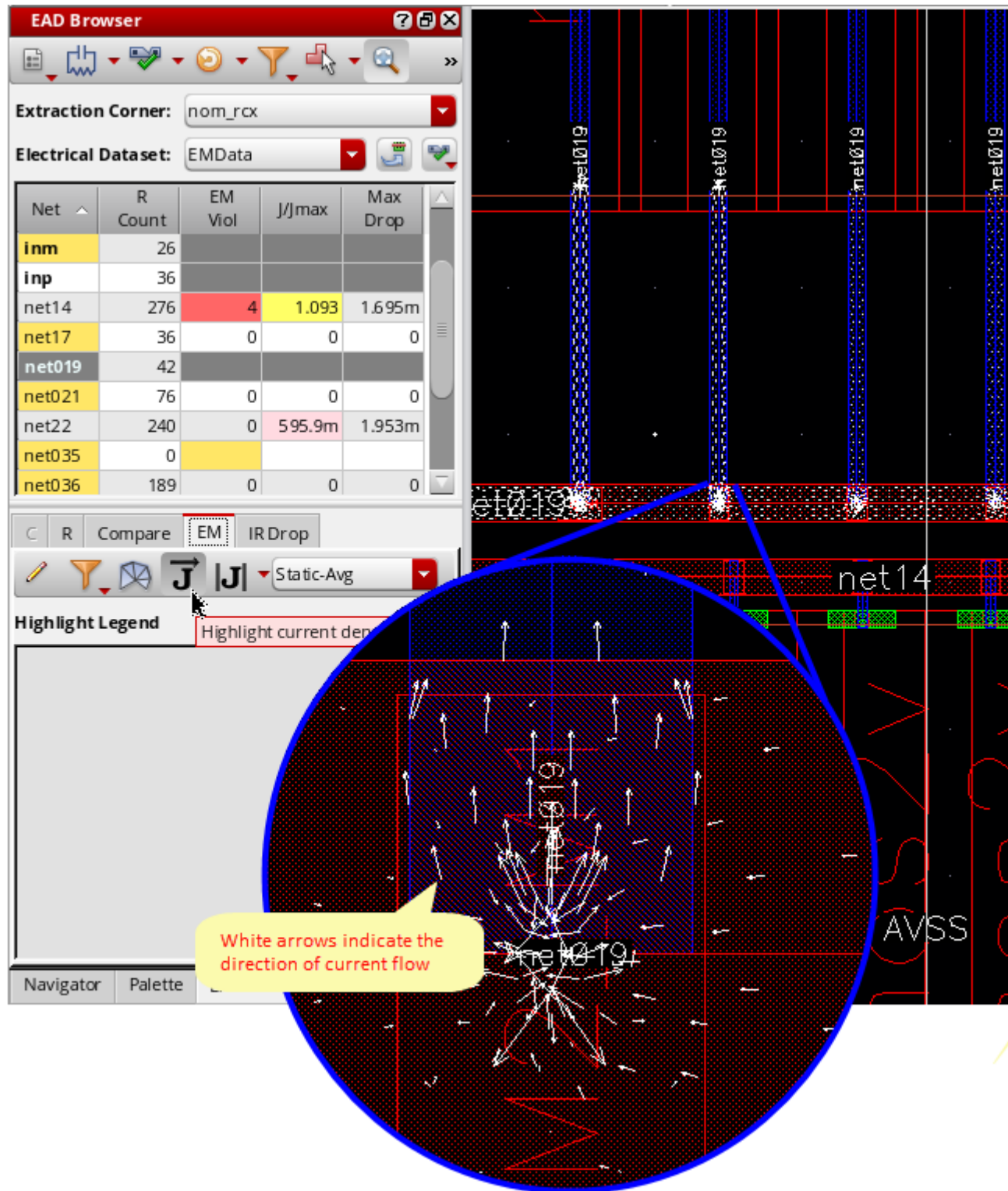
Layout EAD runs EM checks for the net to identify the current density and overlays the data in the layout.

Note: The unit for current density is $A/\mu m^2$.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

4. Zoom in to a large level to view the current density clearly.



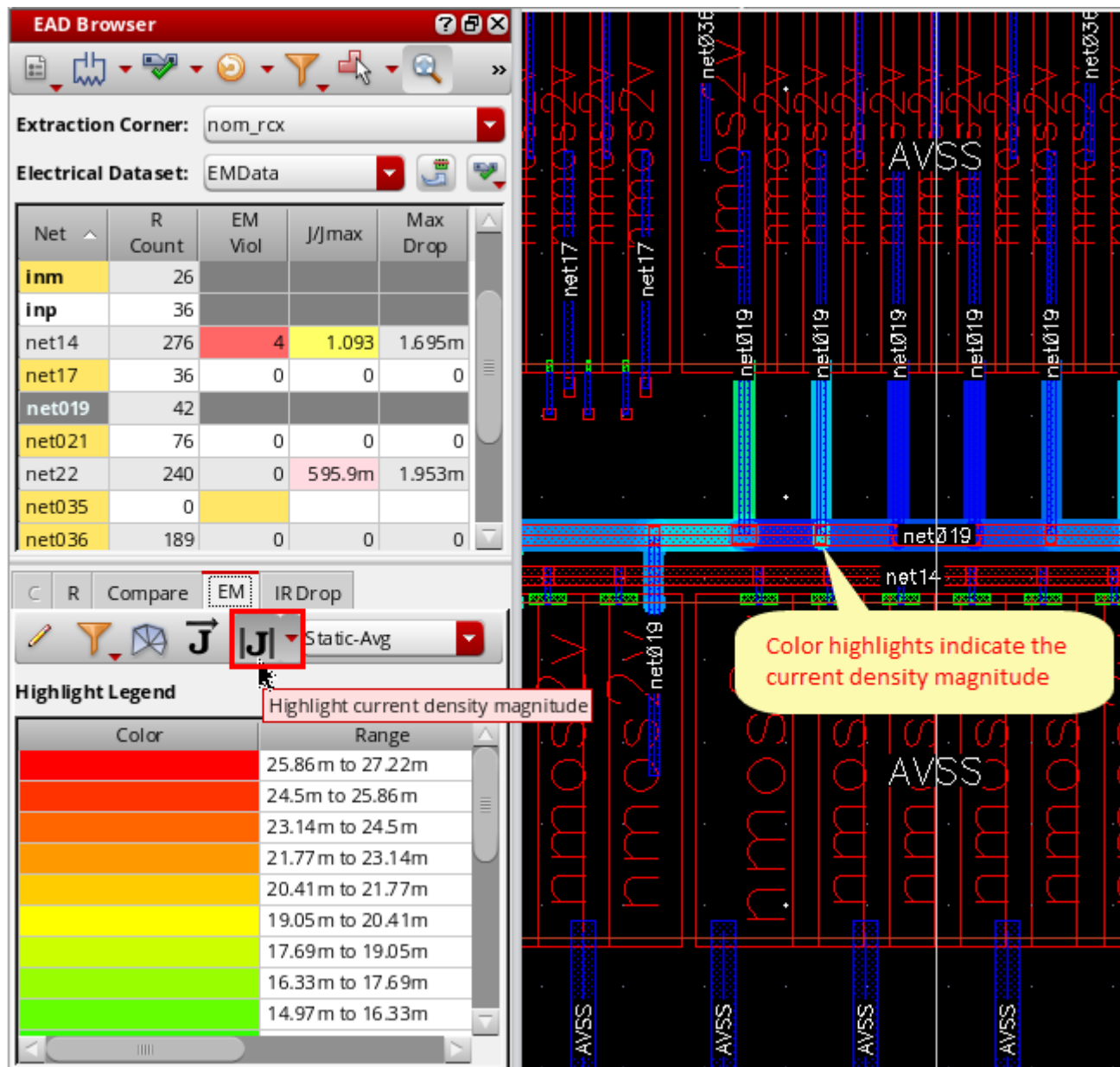
Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

5. You can reset the layer filter to view the current density on specific layers.

To view the current density magnitude, select *Highlight current density magnitude* from the drop-down menu on the toolbar of the *EM* tab.

Layout EAD uses highlights of different colors to mark the magnitude of current, as shown in the figure given below.



The colors at the red end represent higher current density and the colors at the blue end represent lower current density.

Viewing Voltage Drop

If you have performed high precision resistance analysis for a net and saved the data in binary files, you can run current analysis to identify the IR drop or voltage drop (in volts) for that net. By default, the tool displays an overlay as per the real voltage drop values at different points on the net. However, when you set the `showVoltageDropViolation` environment variable to `t` and provide the voltage drop tolerance data for critical nets, Layout EAD uses the reference tolerance levels to display the violation level in percentage. The violation level is the voltage drop percentage allowed by the design specification. This value is calculated as

$$(|V - \text{refV}| / \text{vTol}) * 100.$$

You can provide the voltage drop tolerance data for critical nets in one of the following two ways:

- ➔ By defining the Max Voltage Drop constraint in the Constraint Manager
- ➔ By defining the voltage drop tolerance data in a reference file specified by using the `resistanceFieldFileName` environment variable. The reference file should be an ASCII text file written in the format shown below.

```
<netName> <layerName> <refX> <refY> <vTol> <refV>
```

An example reference file is shown below.

```
AVDD Metal3 0.8875 67.5400 0.10 2.0
AVSS Metal3 0.8875 1.0000 0.05 0.0
```

The above sample file defines that for the net AVDD, reference voltage is 2.0 volts, reference point is at x=0.8875um and y=67.5400um on layer Metal3, and the voltage drop tolerance is +/-0.10 volts. Similarly, reference values are given for the net AVSS.

Important

If, for a critical net, the IR drop tolerance is defined using both the Max Drop constraint as well as a reference file, the tolerance value given in the constraint is used to calculate violation. If multiple constraints are defined for a net, EAD considers the constraint with the smallest IR drop tolerance to identify violations.

To view the voltage drop for a critical net for which high precision R extraction has been run, perform the following steps:

1. Click the net in the summary pane of the EAD Browser.
2. Open the *EM* tab in the detail pane.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

3. Using the current dataset drop-down list, select a dataset for which you need to view the voltage drop.
4. In the drop-down command list on the toolbar of the *EM* table, click *Highlight voltage drop*.

The voltage drop is highlighted on that net, as shown below.

The screenshot displays the EAD Browser interface. The 'Extraction Corner' is set to 'nom_rcx' and the 'Electrical Dataset' is 'EMData'. The EM table lists various nets, with 'net019' highlighted in red. A yellow callout bubble points to the 'Highlight voltage drop' button in the toolbar, stating 'Highlights show voltage drop'. The 'Highlight Legend' table shows color ranges for voltage drop, with red indicating the highest range (-25.21u to 1.516u) and green indicating the lowest range (-239u to -212.3u). The circuit diagram on the right shows the layout with voltage drop highlights on the nets, including 'net019' and 'net036'.

Net	R Count	EM Viol	J/Jmax	Max Drop
inm	26			
inp	36			
net14	276	4	1.093	1.695m
net17	36	0	0	0
net019	42			
net021	76	0	0	
net22	240	0	595.9m	1.
net035	0			
net036	189	0	0	0

Color	Range
Red	-25.21u to 1.516u
Orange	-51.94u to -25.21u
Yellow	-78.67u to -51.94u
Light Green	-105.4u to -78.67u
Green	-132.1u to -105.4u
Light Blue	-158.9u to -132.1u
Blue	-185.6u to -158.9u
Dark Blue	-212.3u to -185.6u
Black	-239u to -212.3u

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

Note: If the `showVoltageDropViolation` environment variable is set to `t`, the voltage drop overlay shows the voltage drop level calculated by using the reference details. If no reference information was provided, the value of voltage drop is displayed.

The colors at the red end represent higher voltage drop and the colors at the blue end represent lower voltage drop levels. The range of voltage drop values represented by each color are listed in the *Highlight Legend* displayed in the details section of the EAD Browser.

Important

The resistance network, which is used to compute current density and voltage drop, is extracted using HPR and the corresponding data is saved in binary files specified by `resistanceMeshFileName` and `resistanceFieldFileName` variables. After the analysis is complete, you can choose to delete these binary files.

Also see: [Viewing IR drop for fast R extraction](#)

Identifying EM Hot Spots

If you have performed high precision resistance analysis for a net and saved the data in binary files, you can run EM checks to identify the EM hot spots for that net.

Layout EAD compares J with the $J_{\max}$ value specified for that net in the EM rules and calculates the EM violation ($J/J_{\max}\%$). These EM violations can be overlaid on the layout canvas to identify the EM hot spots that need correction.

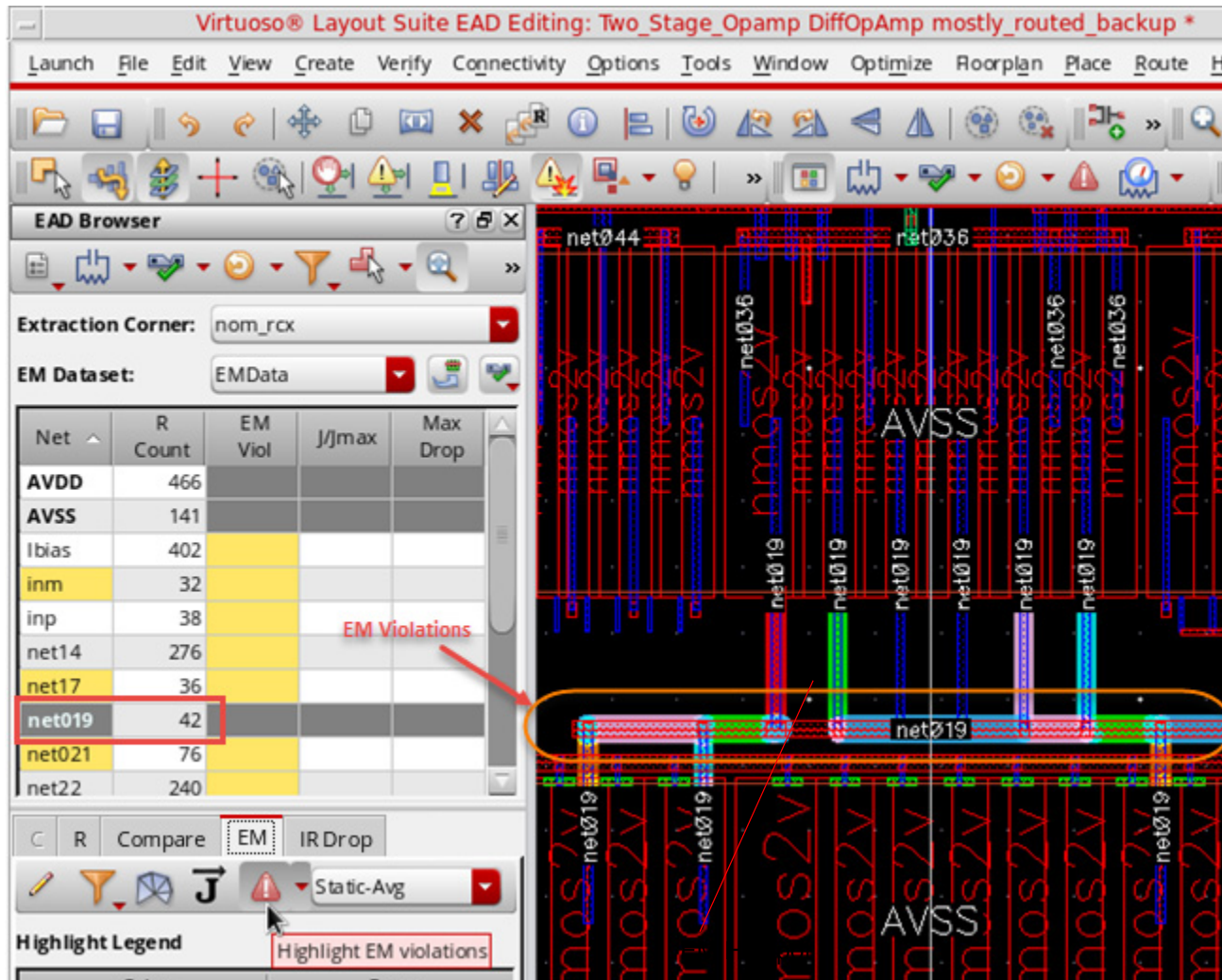
To run the EM checks and to overlay the EM violations for a net on the layout, do the following:

1. Click the net in the summary pane to view its details in the detail pane.
2. Open the *EM* tab in the detail pane.
3. Using the current dataset drop-down list, select a dataset to use the current data.
4. Click *Highlight EM Violations*.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

The violating EM hot spots are highlighted in different colors on the selected net in the layout. The colors correspond to the color options set in the *EM Violation Highlighting* section on the *Environment* tab of the *EAD Options* form.



By default, the Jmax values are taken from the EM rules specified in the *eadTechFile* or *ICT* file. However, for a quick check of potential EM violations, you can provide a custom Jmax definition file with *.jmax* extension. Name of the file before *.jmax* should be same as the name set using the *resistanceFieldFileName* environment variable. This custom Jmax file should be an ASCII file written in the format shown below.

```
<layer-name>      <Jmax-value-for-the-layer>
```

An example Jmax file is shown below.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

fp	0.02
wm	0.01

Note: The unit for J_{max} is $A/\mu m^2$.

For layers other than those specified in the `.jmax` file, the J_{max} values are taken from the `eadTechFile` or `ICT` file. If multiple EM rules are found for a layer, Layout EAD uses the rules with the worst or smallest J_{max} value.

Important

The high precision resistance extraction for identification of EM hot spots is slow. Therefore, it is recommended to use this feature for validation of critical nets only.

Highlighting EM Violations

Layout EAD compares J with the J_{max} value specified for that net in the EM rules and calculates the EM violation ($J/J_{max}\%$). These EM violations can be overlaid on the layout canvas to identify the EM hot spots that need correction.

To highlight EM violations for a net:

1. Click the net in the summary pane to view its details in the detail pane.
2. Open the *EM* tab in the detail pane.
3. Click Show Legend Table.

The *Highlight Legend : EM* table is displayed. The table displays the EM violation ranges used to highlight violations in the EAD. These are the same ranges that are

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

defined in the *Environment* tab of the EAD Options form. If you want to add another range, click *Add* anywhere in the table.

The screenshot shows the EAD Browser window with the following data:

Net	R Count	EM Viol	J/Jmax	Max Drop
outm	123	14	1.911	4.352m
AVDD	338	11	1.538	3.594m
net044	243	9	1.19	3.903m
AVSS	83	7	1.156	2.288m

The legend table is titled "Highlight Legend : EM" and contains the following data:

% of Limit	Color	Visibility
20	Teal	✓
40	Pink	✓
60	Cyan	✓
80	Green	✓
100	Yellow	✓
120	Orange	✓
140	Red	✓

Annotations in the image point to the "Show Legend Table" button (a legend icon) and the "Select layers" button (a funnel icon) in the toolbar.

If you have enabled layer-based filtering for EM violations by setting the environment variable `applyLayerFilterOnCanvasHighlight` to `t`, you can select the layers for which you want to highlight EM violations from the *Select which layers are displayed in the table* drop-down list.

4. Click *Highlight EM violations*.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

The EM violations are highlighted on the canvas.

The screenshot displays the EAD Browser interface on the left and a circuit canvas on the right. The EAD Browser shows the 'Extraction Corner' set to 'nom_rcx' and the 'Electrical Dataset' set to 'EMData'. A table lists net data:

Net	R Count	EM Viol	J/Jmax	Max Drop
AVSS	83	7	1.156	2.288m
net14	276	4	1.093	1.695m

Below the table, the 'Highlight Legend : EM' is shown with a table of visibility settings:

% of Limit	Color	Visibility
20	Blue	☒
40	Light Blue	☒
60	Cyan	☒
80	Green	☒
100	Yellow	☒
120	Orange	☒
140	Red	☒

The circuit canvas on the right shows a layout with various components like 'AVDD', 'AVSS', 'cap', and 'outp'. A yellow box highlights a specific area on the canvas, corresponding to the data in the EAD Browser.

You can clear the check box in the *Visibility* column against a range if you do not want to highlight violations for that range.

Running EM Checks on a CUSTOM Dataset

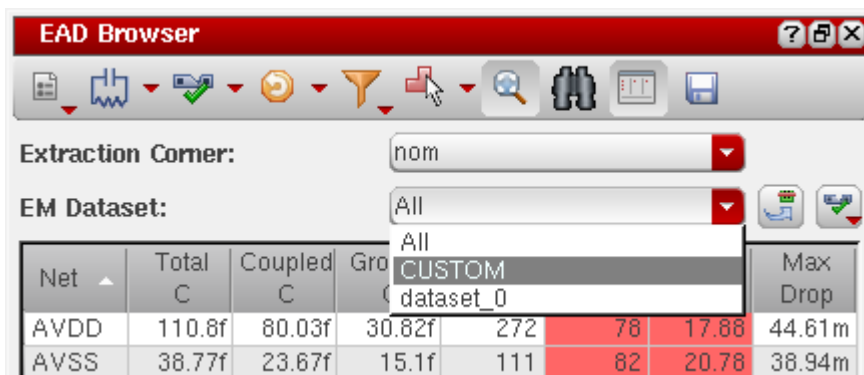
You can use *CUSTOM* datasets if you do not have access to simulated current data from ADE Assembler or if you want to perform a what-if analysis based on current values entered manually in the table. If you are trying multiple combinations of current values for nets, you can save these combinations in different user-defined datasets. These user-defined datasets can be later used to run EM checks. You can also update the electrical values in these datasets.

- [Using Custom Datasets for EM Checks](#)
- [Creating User-Defined Custom Datasets](#)
- [Saving CSV Templates for User-Defined Custom Datasets](#)
- [Reloading and Updating User-Defined Custom Datasets](#)

Using Custom Datasets for EM Checks

To use a custom dataset, do this,

1. Choose *CUSTOM* from the *EM Dataset* drop-down.



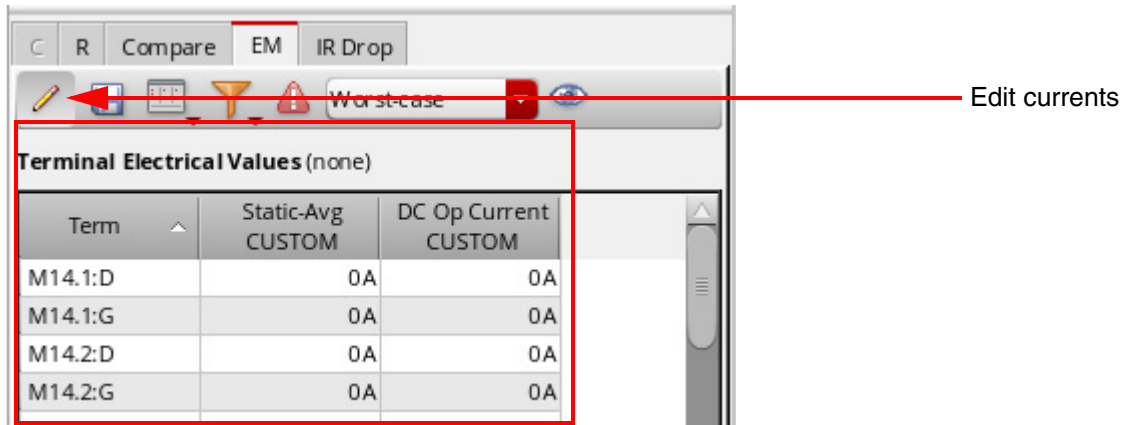
All EM information is removed from the summary and detail panes.

2. Select a net in the summary pane and click *Edit currents* on the EM tab toolbar.

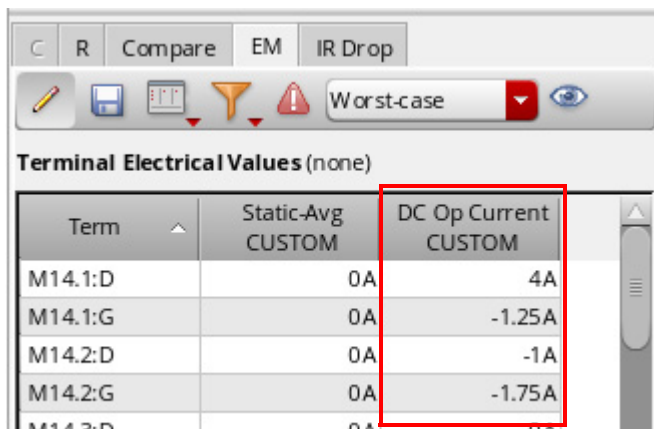
Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

The *EM* table in the detail pane lists all the terminals on the selected net but with no current values set.



3. Type in current values for the terminals on the selected net.



While editing the terminal current data in this table, you can copy data from one or more rows, columns, or cells, and paste it to other rows, columns, or cells. For this, you can use the **Ctrl+C** and **Ctrl+V** bindkeys. However, it is important to note that the following points with respect to copying data from selected cells in a row or column:

- ❑ You can paste the cells copied from the selected columns of a row to the same columns of other rows. For example, cells copied from columns A and B of row1 can be pasted to columns A and B of row 2 or row3. It cannot be copied to columns D and E of row 2.

Similarly, you can paste the cells copied from the selected rows of a column to the same rows of other columns.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

- ❑ You can copy the selected cells either from a single row or from a single column at a time. Copying a set of cells from a combination of multiple rows and columns is not supported.

Note: For an accurate EM check, the values you enter must be KCL-compliant (that is, the sum of all the values entered for the net must be 0).

4. Right-click the net in the summary pane and choose *Run EM Check*.

Layout EAD performs the EM check based on the current values you entered and updates the summary table only for the selected net.

The screenshot shows the EAD Browser window. At the top, the 'Extraction Corner' is set to 'nom_rcx' and the 'Electrical Dataset' is 'CUSTOM'. Below this is a summary table with columns: Net, R Count, EM Viol, J/Jmax, and Max Drop. The row for 'net036' is highlighted with a red border, showing 189 R Count, 8 EM Viol, 18.75K J/Jmax, and 81.34 Max Drop. Below the summary table are tabs for 'C', 'R', 'Compare', 'EM', and 'IR Drop'. The 'EM' tab is selected, and the 'Worst-case' button is visible. Below the tabs is a section titled 'Terminal Electrical Values (net036)' containing a table with columns: Term, Static-Avg CUSTOM, and DC Op Current CUSTOM.

Net	R Count	EM Viol	J/Jmax	Max Drop
net035	0			
net036	189	8	18.75K	81.34
net044	243			
outm	123			
outp	122			

Term	Static-Avg CUSTOM	DC Op Current CUSTOM
M14.1:D	0A	4A
M14.1:G	0A	-1.25A
M14.2:D	0A	-1A
M14.2:G	0A	-1.75A

Creating User-Defined Custom Datasets

While performing what-if analysis, you can create user-defined datasets with the manually entered current values. These datasets can be saved later as a part of the design and then reloaded to run EM checks.

To create a user-defined dataset:

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

1. Select *CUSTOM* from the *EM Dataset* drop-down.
2. Select a net in the summary pane and click *Edit currents* on the EM tab toolbar.
3. Edit the electrical values in the *Terminal Electrical Values* table on the EM tab in the details pane of the EAD Browser.

The screenshot shows the EAD Browser window with the following components:

- Extraction Corner:** nom_rcx
- Electrical Dataset:** CUSTOM (indicated by a red arrow and the text: "When creating a dataset, first choose the CUSTOM dataset.")
- Summary Table:**

Net	R Count	EM Viol	J/Jmax	Max Drop
net035	0			
net036	189			
net044	243			
outm	123			
outp	122			

A red arrow points to the "Edit currents" button on the EM tab toolbar (indicated by the text: "Edit currents").

Below the toolbar, the "Terminal Electrical Values (none)" table is shown. The "Static-Avg" and "DC Op Current" columns are both labeled "CUSTOM". A red arrow points to the "CUSTOM" label in the "Static-Avg" column header (indicated by the text: "Note the name of dataset in the column header is CUSTOM").

Term	Static-Avg CUSTOM	DC Op Current CUSTOM
M15.3:D	0A	0A
M16.1:D	25fA	654mA
M16.2:D	25fA	750mA
M16.3:D	0A	0A

Two red arrows point to the "0A" and "654mA" cells in the "M16.1:D" row, with the text: "Enter custom electrical values."

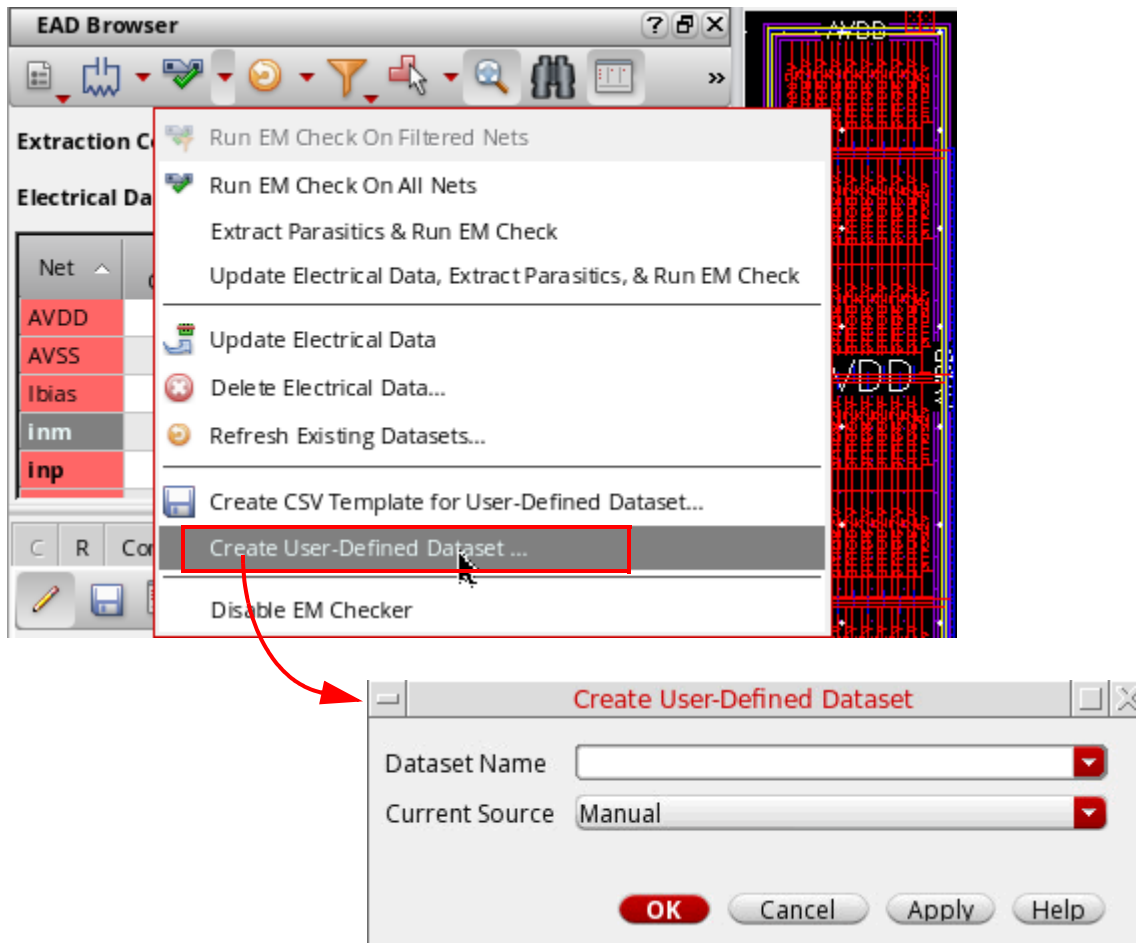
While editing the terminal current data in this table, you can copy data from one or more rows, columns, or cells, and paste it to other rows, columns, or cells. For this, you can use the **Ctrl+C** and **Ctrl+V** bindkeys.

4. Select *Create User-Defined dataset* from the *Run EM Check* drop-down menu on the EAD Browser toolbar.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

The Create User-Defined Dataset form is displayed.



5. In the *Dataset Name* field, specify the name for the custom dataset. For example, myDataset1.

Note: You can use any alphanumeric character in a dataset name. CUSTOM and All are reserved names that cannot be used. Also, the names of any simulation datasets saved from ADE Assembler cannot be used.

As you have edited the current values in the details pane, you can keep the *Current Source* as Manual. For details on the different sources of current that can be used while creating a user-defined dataset, refer to [Sources of Current Values for a User-Defined Dataset](#).

6. Click *Apply* and then click *OK* to complete the creation of dataset.

The electrical values that you changed for the selected nets are stored in the specified user-defined dataset.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

Note: When you save the layout view of the design, the user-defined datasets are also saved with it in the design database.

Sources of Current Values for a User-Defined Dataset

As listed in the *Current Source* drop-down list of the Create User-Defined Dataset form, a user-defined dataset can get the current data from any one of the following sources:

- Manual
- Simulation Results
- Schematic CSV
- CSV
- Spectre EM/IR Results

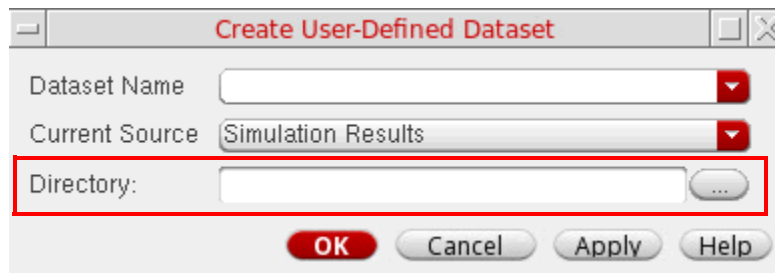
The following sections explain these sources of current data:

Manual

You can directly enter the current values for a net in the *Terminal Electrical Values* table on the details pane of the EAD Browser. This method has been used in the steps describing the creation of a custom dataset.

Simulation Results

You can directly import the current values from the simulation results saved in a `psf` directory. When you choose this mode, the *Directory* field is displayed on the form, as shown below.



The screenshot shows the 'Create User-Defined Dataset' dialog box. It has a title bar with the text 'Create User-Defined Dataset'. Inside, there are three main fields: 'Dataset Name' with a dropdown arrow, 'Current Source' with a dropdown arrow showing 'Simulation Results', and 'Directory:' with a text input field and a browse button (three dots). The 'Directory:' field and its browse button are enclosed in a red rectangular box. At the bottom, there are four buttons: 'OK' (highlighted in red), 'Cancel', 'Apply', and 'Help'.

Click the browse button next to the *Directory* field and browse to the `psf` directory where the simulation results are saved. The path to the `psf` directory is displayed in the *Directory* field. Click *Apply* and then click *OK* to close this form.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

The current values for the selected net are displayed in the *Terminal Electrical Values* table on the *EM* tab on the details pane.

The screenshot shows the EM tool interface. At the top, the 'Extraction Corner' is set to 'nom' and the 'EM Dataset' is 'simDTFromLay'. Below this is a table with columns: Net, Total C, Coupled C, Ground C, R Count, EM Viol, J/Jmax, and Max Drop. The table lists various nets, with AVDD highlighted in red. Below the table are tabs for C, R, Compare, and EM. The EM tab is selected, and a 'Worst-case' button is visible. Below the tabs is the 'Terminal Electrical Values (AVDD)' table, which has two columns: Term and Static-Avg DC Op Current. The table lists various terms and their current values. A red bracket on the right side of the table points to the 'Static-Avg DC Op Current' column, with a label 'Current values imported from simulation results'.

Net	Total C	Coupled C	Ground C	R Count	EM Viol	J/Jmax	Max Drop
AVDD	117.2f	75.74f	41.43f	250	246	16.27	13.76m
AVSS	56.97f	27.21f	29.76f	166	309	1.685K	43.83m
Ibias	53.23f	39.6f	13.63f	239	3	1.478	883.7u
inm	39.45f	23.84f	15.61f	67	49	358.2	495.4u
inp	39.45f	23.88f	15.57f	66	90	11.79K	535.8u
net6	30.14f	19.77f	10.37f	69	47	3.789	2.881m
net9	52.52f	37.34f	15.18f	121	174	1.413K	6.673m
net10	79.26f	59.28f	19.98f	197	185	91.35	6.316m
net47	65.62f	41.59f	24.04f	217	147	145.4	1.627m
net57	68.34f	57.4f	10.94f	80	34	5.415	1.554m
out	138.6f	98.11f	40.52f	159	233	427.3	48.99m

Term	Static-Avg simDTFromLay	DC Op Current simDTFromLay
AVDD	-3.329mA	-2.739mA
M10:B	103.1pA	52.07fA
M10:S	50uA	50uA
M3:B	2.095uA	411.5fA
M3:S	328.1uA	337.7uA
M4:B	2.139uA	29.95fA
M4:S	328.2uA	333.2uA
M5:B	4.814uA	98.73pA
M5:S	2.36mA	1.764mA
M9:B	-20.1nA	6.079pA
M9:S	253.7uA	254uA

Current values imported from simulation results

Note: You may not see the terminal current values for the instances that have been modified in schematic.

You can edit the values in this table. The data is saved in your dataset, which can be reloaded later.

Schematic CSV

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

If the current values for schematic objects are available in a CSV file, you can import the data from that file to create a dataset. This option is particularly useful when either you cannot run simulations to get the electrical data in results or the electrical data saved in the simulation results is incomplete. In such cases, you can save the estimated current or voltage details in a CSV, and import the file into Layout EAD to create a user-defined custom dataset.

The CSV file must be in the following format:

```
#TERM_DELIMITER: /
#RESULT_TYPES(SCALE_FACTOR): Idc(<scale-factor-Idc>), Iavg(<scale-factor-Iavg>)

<inst-name>/<instance-terminal-name> , , ,
<inst-name>/<instance-terminal-name> , , ,
...
```

Note the following points for this format:

- In the rows where the electrical data is saved, you can include or exclude placeholders for the current data depending on your requirements.
- You need to provide the scaling factor only for those result types for which you are saving the current data in this file. For example, if you saving only the direct current data, add `Idc(<scale-factor-Idc>)` in the file header. You do not need to add the scaling factor for `Iavg`.

An example CSV file is shown below.

```
#TERM_DELIMITER: /
#RESULT_TYPES(SCALE_FACTOR): Idc(1)

AVDD,0.00393418
AVSS,-0.00369249
M1/B,6.88283e-11
M1/D,0.000239754
M1/G,6.14897e-11
M1/S,-0.000239754
M10/B,1.6815e-11
M10/D,-5e-05
....
```

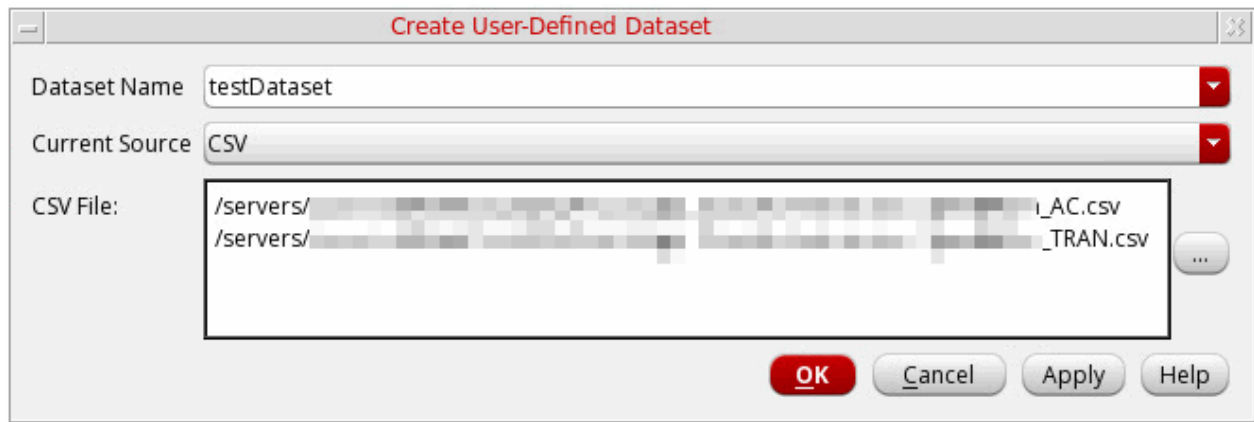
See Also: [eadCreateDataSetFromSchCSV](#)

CSV

If the current values for a net are available in a CSV file, you can import that to create a dataset. When you choose this mode, the *CSV File* field is displayed on the form. Click the browse button next to the *CSV File* field and browse to the directory where the CSV files is saved. To import multiple CSV files simultaneously, hold down the *Ctrl* key while you select the CSV files from which you want to import electrical data. The path to the CSV files are displayed in the *CSV File* field.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations



Click *Apply* and then click *OK* to close the form.

New columns for the newly-created dataset are added in the *Terminal Electrical Values* table and the current values are displayed in them, as shown below.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

The screenshot shows the EAD Browser window with the following components:

- Extraction Corner:** nom
- EM Dataset:** All
- EM Dataset Table:**

Net	Total C	Coupled C	Ground C	R Count	EM Viol	J/Jmax	Max Drop
AVDD	59.18f	30.56f	28.62f	195			
AVSS	29.62f	10.62f	19f	160			
IN	21.97f	5.938f	16.04f	121			
OUT	17.19f	9.834f	7.351f	72			
net4	39.2f	15.74f	23.46f	198			
I0/net19	35.49f	15.5f	20f	149			
I1/net19	35.49f	15.51f	19.98f	149			
- Buttons:** C, R, Compare, EM (selected)
- Worst-case:** Worst-case
- Terminal Electrical Values (none):**

Term	Static-Avg CUSTOM	DC Op Current CUSTOM	Static-Avg testDataset	DC Op Current testDataset
I0/NM0:D	0	0	420u	513.2n
I0/NM1:G	0	0	-14.88u	564.4n
I0/PM0:D	0	0	-380.7u	-1.078u
I0/PM1:G	0	0	-24.37u	594.2p

Red arrows indicate the following:

- EM tab
- Columns added for the new dataset, testDataset
- Current values imported from the CSV file

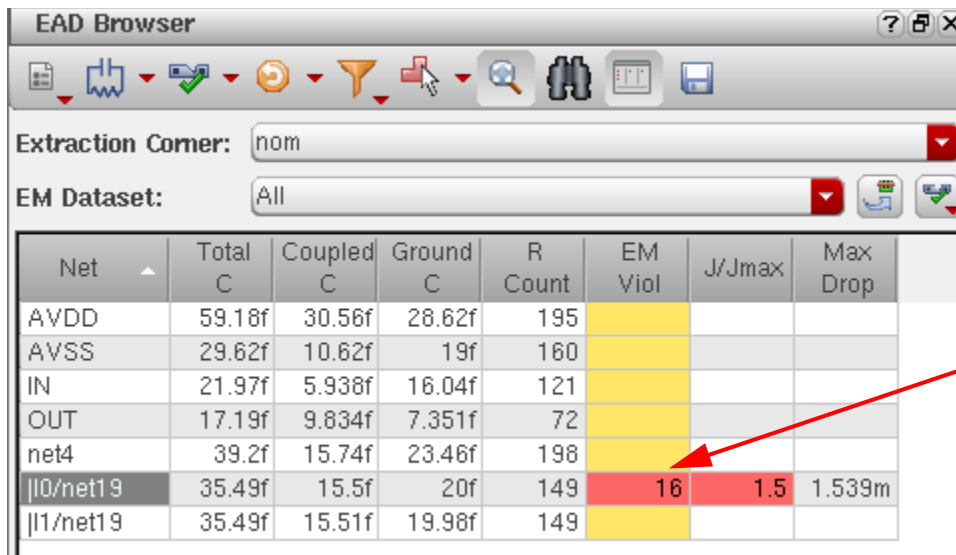
The current values for the nets mentioned in the CSV files are imported and displayed in the *Terminal Electrical Values* table. If required, you can edit the values in this table. The updated values are then saved in the your dataset.

Note: A CSV file can contain only the static average or DC op current values for one or more nets. It cannot contain dynamic current data for a net.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

After importing data from the CSV file, run EM checks to view EM violations.



Net	Total C	Coupled C	Ground C	R Count	EM Viol	J/Jmax	Max Drop
AVDD	59.18f	30.56f	28.62f	195			
AVSS	29.62f	10.62f	19f	160			
IN	21.97f	5.938f	16.04f	121			
OUT	17.19f	9.834f	7.351f	72			
net4	39.2f	15.74f	23.46f	198			
I0/net19	35.49f	15.5f	20f	149	16	1.5	1.539m
I11/net19	35.49f	15.51f	19.98f	149			

EM violation for I0/net19 after importing current values from the CSV file

A sample csv file is shown below.

```
#Current Types: "Static-Avg","DC Op Current"
#Net Name: |I0/net19
I0/|NM0:D,420uA,513.2nA
I0/|NM1:G,-14.88uA,564.4nA
I0/|PM0:D,-380.7uA,-1.078uA
I0/|PM1:G,-24.37uA,594.2pA
```

The first two lines in the file provide the header information. It is also possible that the source CSV file contains either one of the static average or DC Op current data for one or more nets. In that case, the current values are loaded only for the column name specified in the CSV header.



Tip

You can use the template files saved from the EAD Browser to create a source CSV file with current information. For more details, refer to [Saving CSV Templates for User-Defined Custom Datasets](#).

Spectre EM/IR Results

If the current values for a net are available in the results database saved for a Spectre EM/IR analysis, you can import that to create a dataset. When you choose this mode, the *Directory* field is displayed on the form. Click the browse button next to the *Directory* field and browse to the directory where the results are saved. Click *Apply* and then click *OK* to save the dataset.

Saving CSV Templates for User-Defined Custom Datasets

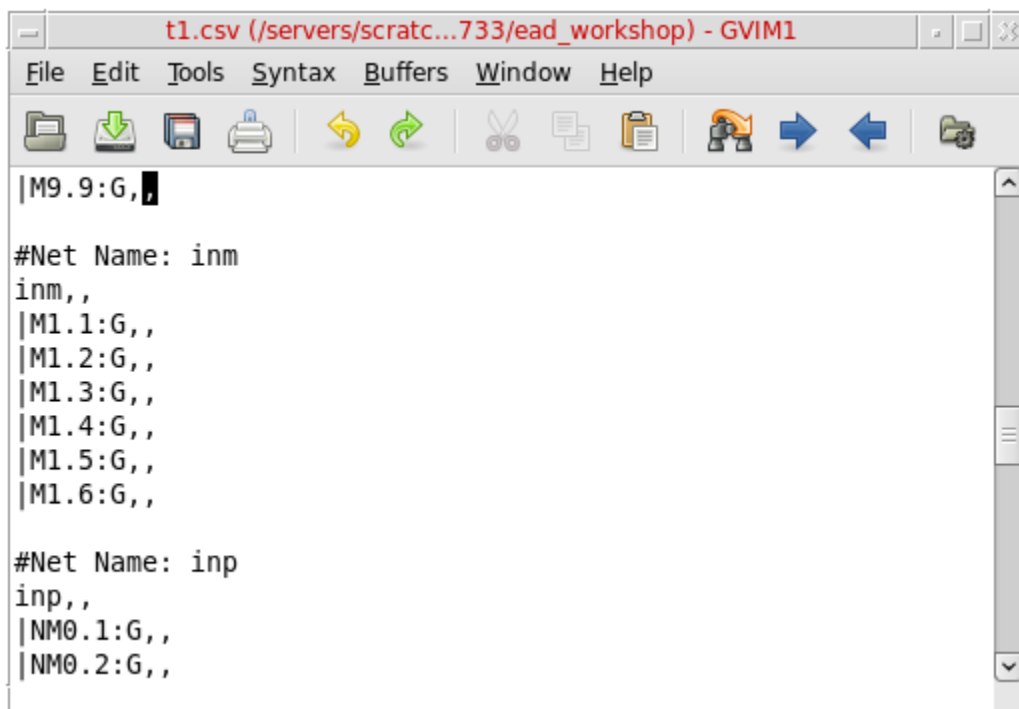
When you import the current information from CSV files to create custom datasets, it is important to create the CSV file in the required format. If the format is incorrect, the current information is not imported.

To ensure that the current details are written in the correct format in the CSV file, you can save template from the EAD browser and edit the current information. These templates have some pre-filled information that you can modify or complete while keeping the format unchanged.

You can save templates from EAD Browser in the following ways:

- By saving the details of all the nets displayed in the Summary pane of the EAD Browser. For this, click *Run EM Checks* to open the drop-down list and choose *Create CSV Template for User-Defined Dataset*. Provide a file name and path for the template file in the EAD Save CSV For EM Dataset form and click *OK*.

A template file in the CSV format is saved, as shown in the following example:



The template file saved in this way contains the following information:

- ❑ A section for each net listed in the Summary pane
- ❑ Each net name is followed by the names of terminals connected to it
- ❑ A comma-separated blank format where you can enter the current details

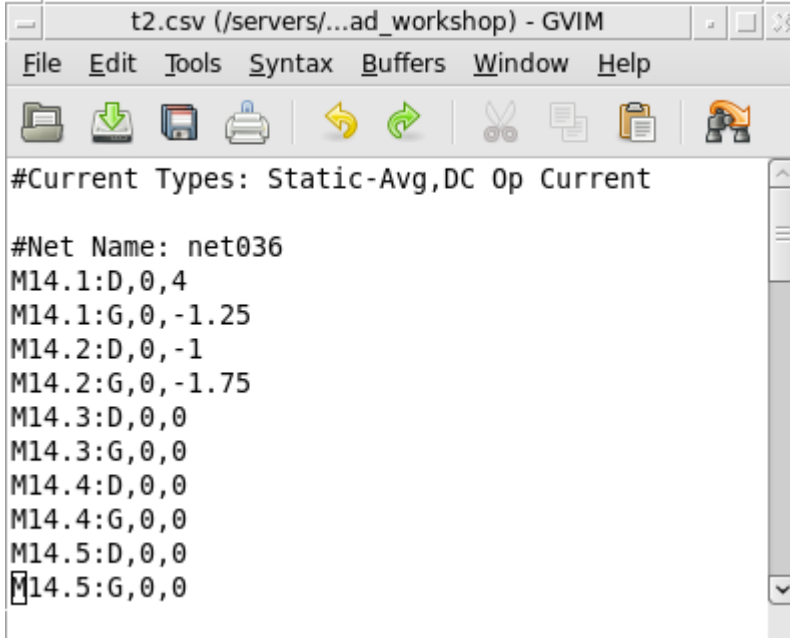
Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

Note: This template file does not get any information from the datasets.

- By saving the terminal current information for the nets selected in the Summary pane. For this, perform these steps:
 - a. In the Summary pane, select one or more nets for which you need to provide the current information.
 - b. Click *EM* in the Details pane to open the EM tab.
 - c. Click *Edit currents* () on the EM tab toolbar to display the *Terminal Electrical Values* table.
 - d. The *Export electrical Data to a CSV file* () command is also displayed to the right of *Edit currents*.
 - e. Click *Export electrical Data to a CSV file*.
 - f. Provide a file name and path for the template file in the EAD Save CSV For EM Dataset form and click *OK*.

A template file in the CSV format is saved, as shown in the following example:



The template file saved in this way contains the following information:

- ☐ A section for each net selected in the Summary pane
- ☐ Each net name is followed by the names of terminals connected to it

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

- ❑ A comma-separated list of current values exported from the *Terminal Electrical Values* table

You can save this template to dump the current details for all or a particular dataset selected in the *EM Dataset* drop-down list on the Summary pane.

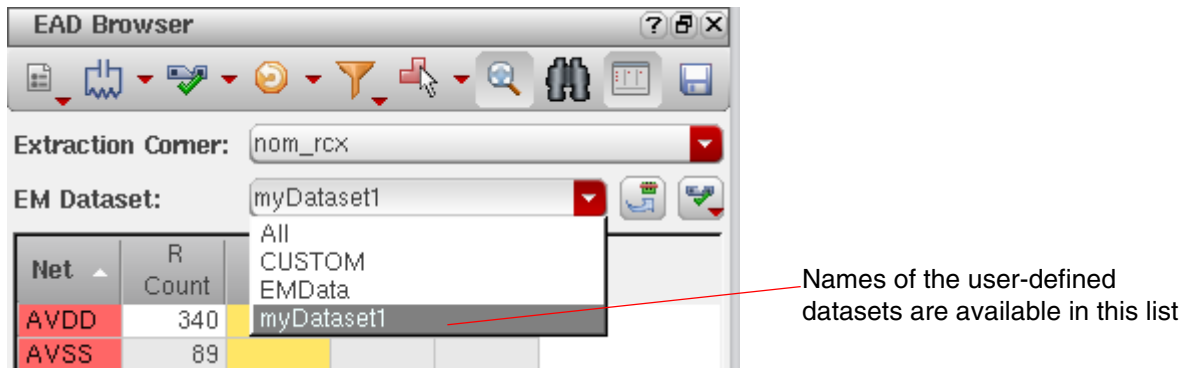
Note: If the *Terminal Electrical Values* table shows information for multiple datasets, a separate template file is saved for each dataset. The dataset name is added to the name of the CSV file saved for it.

After modifying the current details in the template files, you can use those to import details and create custom dataset by using the *Create User-Defined Dataset* command. For more details, refer to [Creating User-Defined Custom Datasets](#).

Reloading and Updating User-Defined Custom Datasets

To reload a user-defined dataset for EM check:

1. Choose the name of a user-defined dataset from the *EM Dataset* drop-down.



All EM information is removed from the summary and detail panes.

2. In the summary pane, select the net for which you had saved the electrical currents and click *Edit currents* on the EM tab toolbar.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

- The *Terminal Electrical Values* table in the detail pane lists all the terminals on the selected net with the electrical values from the dataset.

The screenshot shows the EAD Browser window. At the top, the 'Extraction Corner' is set to 'nom_rcx' and the 'EM Dataset' is 'myDataset1'. Below this is a table of nets with columns: Net, R Count, EM Viol, J/Jmax, and Max Drop. The 'net14' row is highlighted in red. Below the net table are tabs for 'C', 'R', 'Compare', and 'EM'. The 'EM' tab is active, showing a 'Worst-case' dropdown. Below this is the 'Terminal Electrical Values (net14)' table.

Net	R Count	EM Viol	J/Jmax	Max Drop
AVDD	340			
AVSS	89			
Ibias	274			
inm	39			
inp	38			
net14	312			
net17	48			

Term	Static-Avg myDataset1	DC Op Current myDataset1
M17.1:G	25fA	700uA
M17.2:G	25fA	700uA
M17.3:G	0A	0A

Annotations in the image:

- Selected net:** Points to the 'net14' row in the net table.
- The name of the user-defined dataset that has been loaded appears in the column headers:** Points to the 'myDataset1' header in the 'Terminal Electrical Values' table.
- Electrical values that were specified earlier. These can be edited here:** Points to the '700uA' value in the 'DC Op Current' column.
- Values for other nets can also be specified in the same dataset:** Points to the '0A' value in the 'DC Op Current' column for 'M17.3:G'.

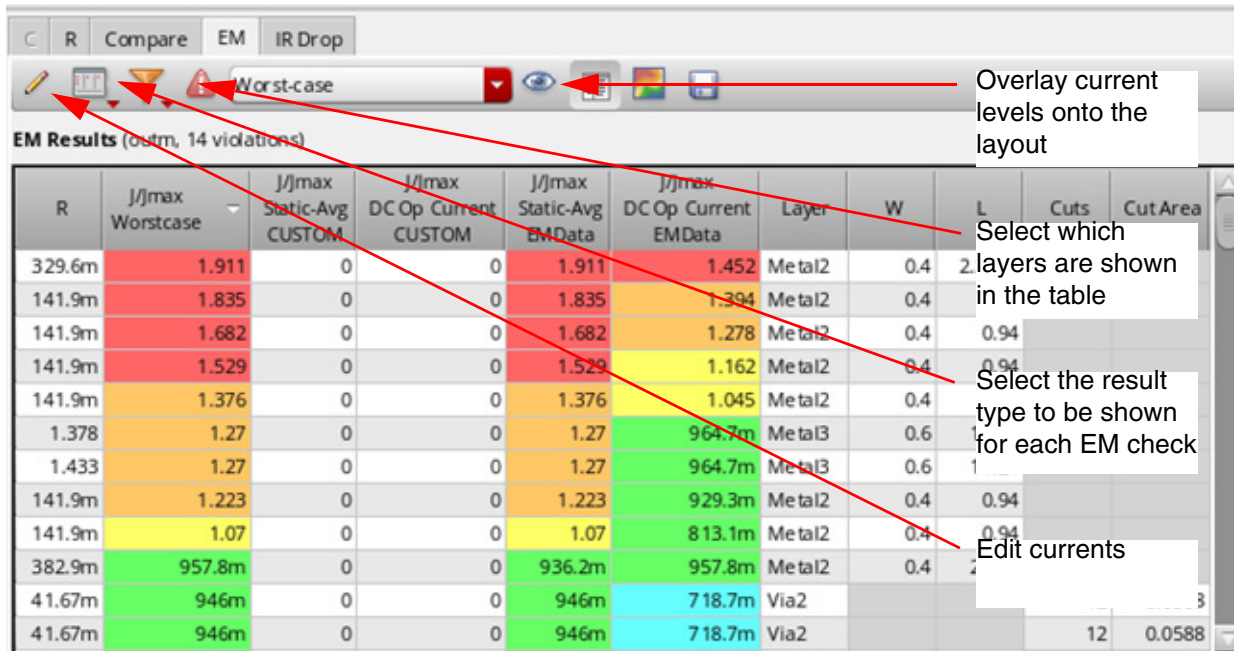
Now, you can edit the current values for the same or other nets, and run EM checks. The edited values are again saved with the dataset when you save the design.

Note: You can also choose to show all the datasets together by selecting **ALL** from the *EM Dataset* drop-down list, and then edit electrical values for any of the user-defined datasets.

Viewing EM Information in the EAD Browser

This section describes how to analyze the results of an EM check in the EAD Browser's Detail Pane. To do this,

1. Click one of the nets in the summary pane to open the *EM* tab in the detail pane.



The screenshot shows the EAD Browser interface with the 'EM' tab selected. The 'EM Results' table displays 14 violations. Red arrows point from text boxes to specific UI elements: 'Overlay current levels onto the layout' points to the 'Worst-case' button; 'Select which layers are shown in the table' points to the 'Layer' column; 'Select the result type to be shown for each EM check' points to the 'J/Jmax' column headers; and 'Edit currents' points to the 'J/Jmax Worstcase' column.

R	J/Jmax Worstcase	J/Jmax Static-Avg CUSTOM	DC Op Current CUSTOM	J/Jmax Static-Avg EMData	DC Op Current EMData	Layer	W	L	Cuts	CutArea
329.6m	1.911	0	0	1.911	1.452	Metal2	0.4	2		
141.9m	1.835	0	0	1.835	1.394	Metal2	0.4			
141.9m	1.682	0	0	1.682	1.278	Metal2	0.4	0.94		
141.9m	1.529	0	0	1.529	1.162	Metal2	0.4	0.94		
141.9m	1.376	0	0	1.376	1.045	Metal2	0.4			
1.378	1.27	0	0	1.27	964.7m	Metal3	0.6	1		
1.433	1.27	0	0	1.27	964.7m	Metal3	0.6	1		
141.9m	1.223	0	0	1.223	929.3m	Metal2	0.4	0.94		
141.9m	1.07	0	0	1.07	813.1m	Metal2	0.4	0.94		
382.9m	957.8m	0	0	936.2m	957.8m	Metal2	0.4			
41.67m	946m	0	0	946m	718.7m	Via2				
41.67m	946m	0	0	946m	718.7m	Via2			12	0.0588

You see a detailed breakdown of the EM violations present for the net selected in the summary pane. If there are no violations for the selected net, the table is empty.

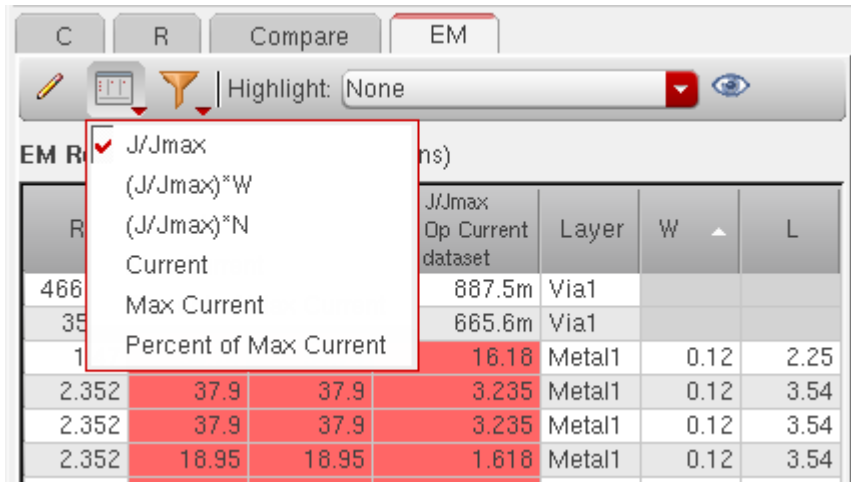
Each row represents a resistor that was extracted in EAD. The color coding is controlled by the settings in the Environment tab.

For information on how to customize the table display, see Changing Data Display in the EAD Browser.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

2. Use the *Result type* menu to choose which values are displayed in the table.



- ☐ J/J_{max} shows the ratio of current to maximum current. A value greater than or equal to 1.0 is a violation. The higher the value, the more serious the violation.
- ☐ $(J/J_{max}) * W$ shows the ratio of current to maximum current multiplied by the resistor width. In some cases, this value can give an indication of how wide the wire would need to be in order to remove the violation.

This value is shown only for the resistors extracted from conductor layers.

Note: Width that is equal to conductor's drawn width multiplied by conductor's J/J_{max} ratio is calculated. J_{max} is then recalculated at the new width. This step is repeated iteratively to find a recommended width at which the calculated J_{max} is closer to the J value by scaling up or down the previously calculated width based on ratio of currents.

- ☐ $(J/J_{max}) * N$ shows the ratio of current to maximum current multiplied by the existing number of via cuts. This gives an estimate of the number of via cuts that could possibly fix the EM violation reported for the via resistor.

This value is shown only for the resistors extracted from via layers.

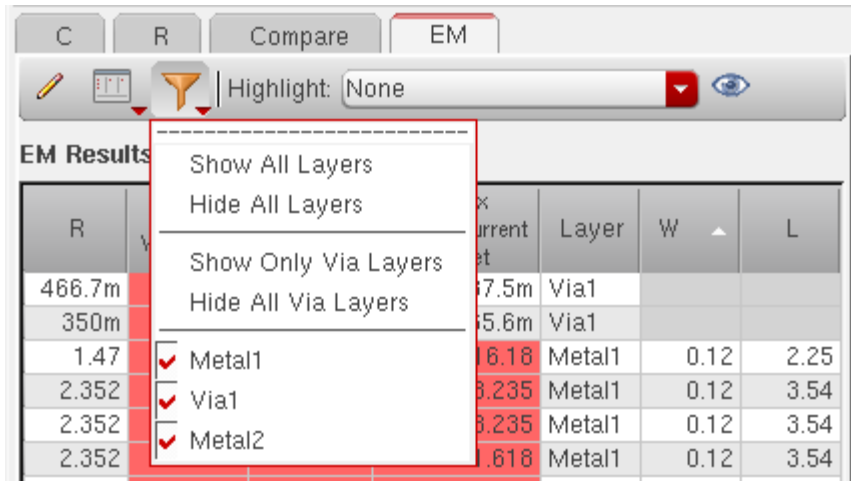
- ☐ *Current* shows the current from simulation results
- ☐ *Max Current* shows the maximum current that can flow through this resistor
- ☐ *Percent of Max Current* shows the value as a percentage of the maximum current
- ☐ *Delta T* shows additional columns to display the delta T results generated when the self-heating effects are enabled for electromigration, that is, when the *Enable Self Heating Effects* check box on the *EM* tab of the *EAD Options* form is selected. For

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

more details on how to view delta T results on the *EM* tab in EAD Browser, refer to [Viewing Delta T Information](#).

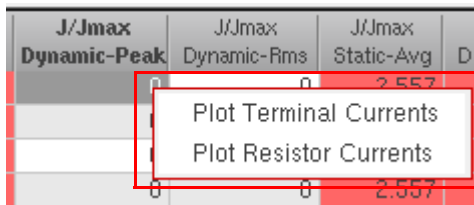
3. Use the *Select layers* menu to control which layers are shown in the table.



4. Use the *Highlight* drop-down to choose an overlay in the layout canvas for the selected net. An overlay color-codes the select net on the canvas based on the EM information available for that net. Use the *Highlight Transparency* slider to change the transparency of the overlay to let you see the underlying design geometry.

For more information on using overlays to highlight violations in the canvas, see [Highlighting and Fixing Violations in the Layout Window](#).

5. When *Dynamic Analysis* is enabled in the **EM** tab, you can right-click in any *J/Jmax Dynamic-Peak* or *J/Jmax Dynamic-Rms* cell and
 - ☐ Choose *Plot Terminal Currents* to view the currents associated with the terminals of the selected net.
 - ☐ Choose *Plot Resistor Currents* to view the intermediate currents through the selected resistor. This menu command is available only when the `preserveResistorCurrents` environment variable is set to `t`.



Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

- Change the columns shown in the EM results table by clicking the right mouse button in any column header and selecting or clearing the check box next to the information you want to show or hide.

EM Results (outm, 14 violations)

R	J/Jmax Worstcase	J/Jmax Static-Avg CUSTOM	J/Jmax DC Op Current CUSTOM	J/Jmax Static-Avg EMData	J/Jmax DC Op Current EMData
329.6m	1.911		Net		1.452
141.9m	1.835		✓ R		1.394
141.9m	1.682		✓ J/Jmax Worstcase		1.278
141.9m	1.529		✓ J/Jmax Static-Avg CUSTOM		1.162
141.9m	1.376		✓ J/Jmax DC Op Current CUSTOM		1.045
1.378	1.27		✓ J/Jmax Static-Avg EMData		964.7m
1.433	1.27		✓ J/Jmax DC Op Current EMData		964.7m
141.9m	1.223		✓ Layer		813.1m
141.9m	1.07		✓ W		957.8m
382.9m	957.8m		✓ L		718.7m
41.67m	946m		✓ EML		718.7m
41.67m	946m		✓ Cuts		606.0m

- Move your mouse pointer over any cell to see more information about the value shown in that cell.

For example, the terminal electrical values table shows multi-pin terminals as 0 volts ("0V"). This is because the terminal with multiple pins, is treated as a reference terminal and each pin is treated as a voltage source (at 0V) instead of a current source. If you

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

position the mouse pointer over the value, you see a tooltip displaying the current at the pin computed during the EM check.

P5:s	0A
P6:b	785.5fA
P6:d	2.14nA
P6:g	2.254zA
P7:b	0A
P7:d	0A
P7:g	0A
P7:s	0A
VDD	0V
VDD	0V
VDD	0V

Pin current: 7.915nA

In addition, when you click a terminal row in the table, the display zooms to the terminal shapes.

Similarly, if you position the mouse pointer over the length (L) value, you see a tooltip displaying the drawn length and the EM length.

EM Results (outm, 14 violations)

R	J/Jmax Worstcase	J/Jmax Op Current JSTOM	J/Jmax Static-Avg EMData	J/Jmax DC Op Current EMData	Layer	W	L	EM L	Cuts
329.6m	1.911	0	1.911	1.452	Metal2	0.4	2.2395	14.895	
141.9m	1.835	0	1.835	1.394	Metal2	0.4	0.94	14.895	
141.9m	1.682	0	1.682	1.278	Metal2	0.4	0.94	14.895	
141.9m	1.529	0	1.529	1.162	Metal2	0.4	0.94	14.895	
141.9m	1.376	0	1.376	1.045	Metal2	0.4	0.94	14.895	
1.378	1.27	0	1.27	964.7m	Metal3	0.6	13.69	28.6	
1.433	1.27	0	1.27	964.7m	Metal3	0.6	14.24	28.6	
141.9m	1.223	0	1.223	929.3m	Metal2	0.4	0.94	14.895	
141.9m	1.07	0	1.07	813.1m	Metal2	0.4	0.94	14.895	
382.9m	957.8m	0	936.2m	957.8m	Metal2	0.4	2.58		
41.67m	946m	0	946m	718.7m	Via2				12

Drawn length: 14.24 microns
EM length: 28.6 microns

Note: EM length is the effective length used to determine which EM rule in the ICT file is to be applied. The default method for EM length calculation is `centerline`, but you can change it by using the `segmentLength` environment variable.

- Click *Export table data* (📄) in the EAD Browser toolbar to export the data currently displayed in the browser tables in either CSV or HTML format.

Note: Data contained in columns that are currently hidden is not exported.

Viewing Delta T Information

When you enable the self-heating effects by setting the *Enable Self Heating Effects* check box on the *EM* tab of the EAD Options form, Layout EAD saves the delta T values in electromigration results.

$\text{delta T} = \text{Coupling Delta T} + \text{RMS Delta T}$

You can view the RMS Delta T and Coupling Delta T values for the corresponding resistance by hovering the mouse pointer over any cell in the *Delta T* columns on the *EM* tab of the EAD Browser. The cells that contain a *delta T* value greater than the value of *Delta T* specified on the *EM* tab of the EAD Options form are highlighted in red.

These delta T values are used in the calculation of the temperature derating factor for a net. The calculation of delta T is dependent on the current flowing through metal conductor associated with the routing of the net and any device underneath the metal layer. While checking the current values to be used for the calculation of delta T, Layout EAD considers the values set for the metalRegionSearchExtent and odRegionSearchExtent environment variables.

To view delta T results, do this:

1. On the *EM* tab, use the *Result type* menu to choose *Delta T*.

Note: The *Delta T* command appears in the *Result type* menu only when Delta T values are saved in electromigration results.

Note the following changes in the EM results table displayed on the EM tab.

- The columns that displayed J/Jmax values are hidden.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

- ❑ Columns to display the delta T values are displayed in the EM results table, as shown below.

EM Results (OUT, 0 violations)

R	Delta T Worstcase	Delta T Dynamic-Rms user_dataset	Delta T Dynamic-Rms she_dataset	Layer
4.499	1.933n	1.933n	1.933n	M2
2.867				
5.915	1.821n	1.821n	1.821n	M1
1.91	1.821n	1.821n	1.821n	M1
1.865	1.821n	1.821n	1.821n	M1
7.676	1.821n	1.821n	1.821n	M1
852.3m	1.298n	1.298n	1.298n	M1
1.137	1.298n	1.298n	1.298n	M1
2.772		0	0	M1

Note: Delta T values are collected due to the flow of RMS currents in the metal resistors. Therefore, to display delta T values, one column is added for each dataset that contains RMS currents.

- ❑ Two new commands are displayed in the *EM* tab toolbar:
 - The *Highlight delta T Values on Canvas and show Highlight Legend*  command button to display a heat map of delta T on the layout canvas
 - The *Electrical dataset to be used for delta T highlighting* drop-down list that contains the name of electrical datasets containing delta T values

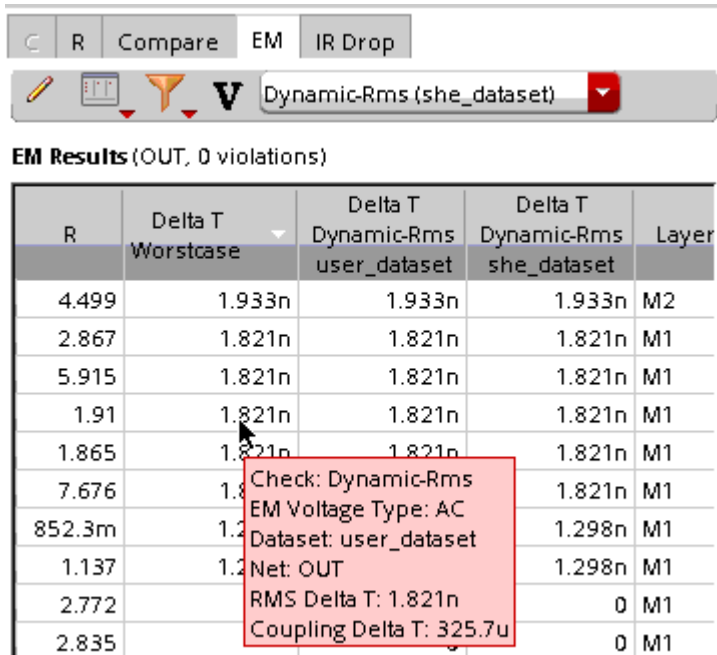
You can use these commands to highlight the delta T values on the layout canvas.

2. To view the RMS Delta T and Coupling Delta T values for the corresponding resistance, move the pointer over any cell in the Delta T columns.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

The data is displayed in a tooltip, as shown below.



EM Results (OUT, 0 violations)

R	Delta T Worstcase	Delta T Dynamic-Rms user_dataset	Delta T Dynamic-Rms she_dataset	Layer
4.499	1.933n	1.933n	1.933n	M2
2.867	1.821n	1.821n	1.821n	M1
5.915	1.821n	1.821n	1.821n	M1
1.91	1.821n	1.821n	1.821n	M1
1.865	1.821n	1.821n	1.821n	M1
7.676	1.821n	1.821n	1.821n	M1
852.3m	1.298n	1.298n	1.298n	M1
1.137	1.298n	1.298n	1.298n	M1
2.772	0	0	0	M1
2.835	0	0	0	M1

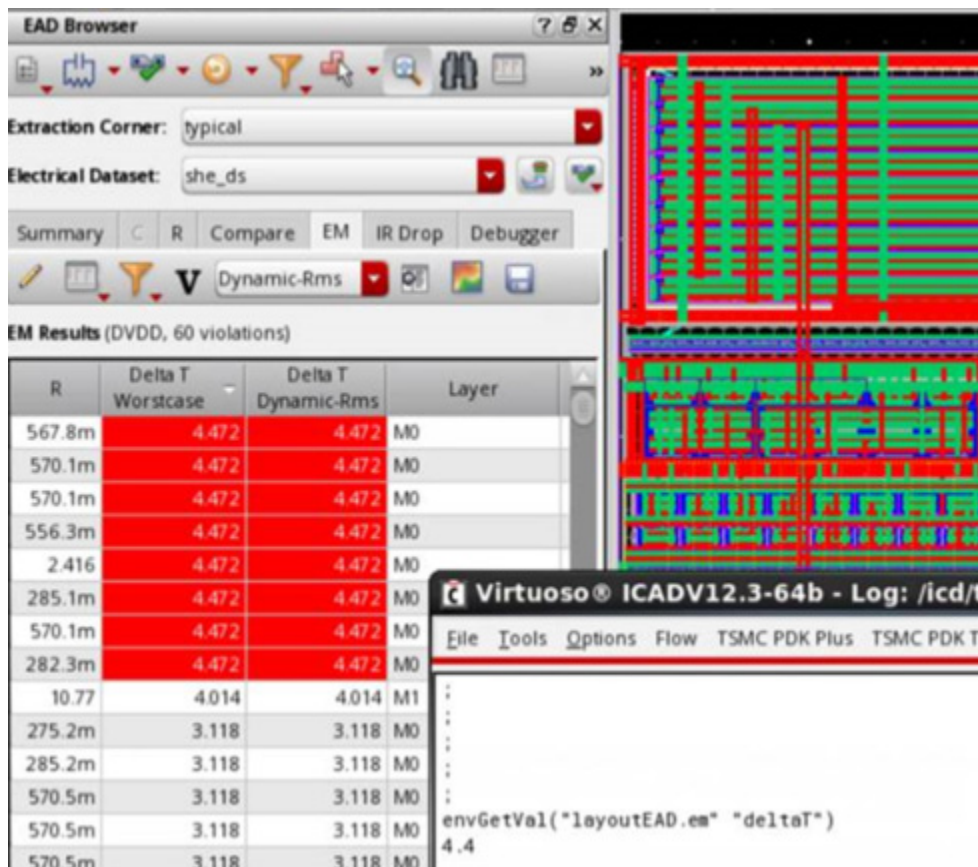
Check: Dynamic-Rms
 EM Voltage Type: AC
 Dataset: user_dataset
 Net: OUT
 RMS Delta T: 1.821n
 Coupling Delta T: 325.7u

Note: The cells that contain a delta T value greater than the one specified by the layoutEAD.em deltaT environment variable or the *Delta T* on the *EM* tab of EAD Options

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

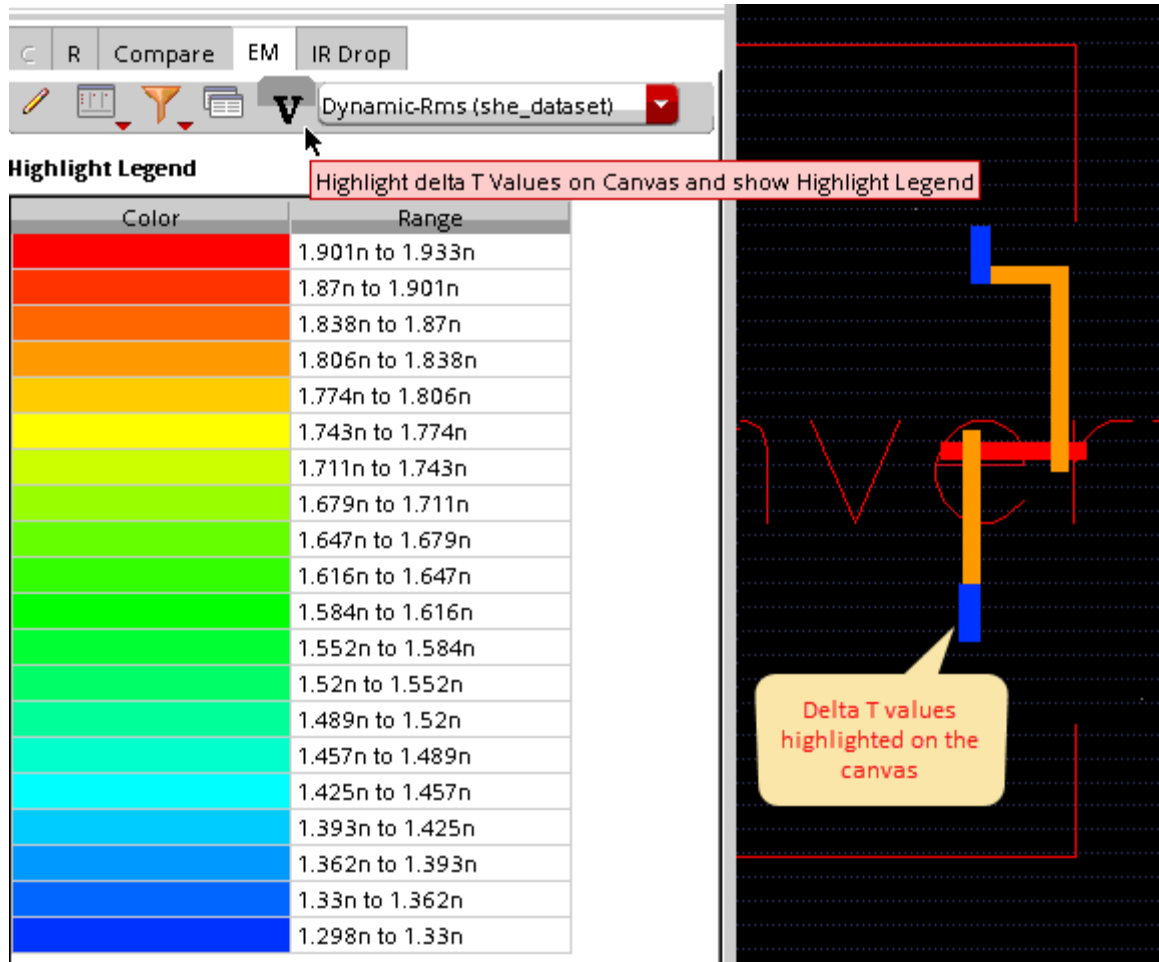
form are highlighted in red, as shown in the example given below.



Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

3. To view the delta T values highlighted on the canvas, select an electrical dataset from the drop-down list and click **V** on the toolbar.

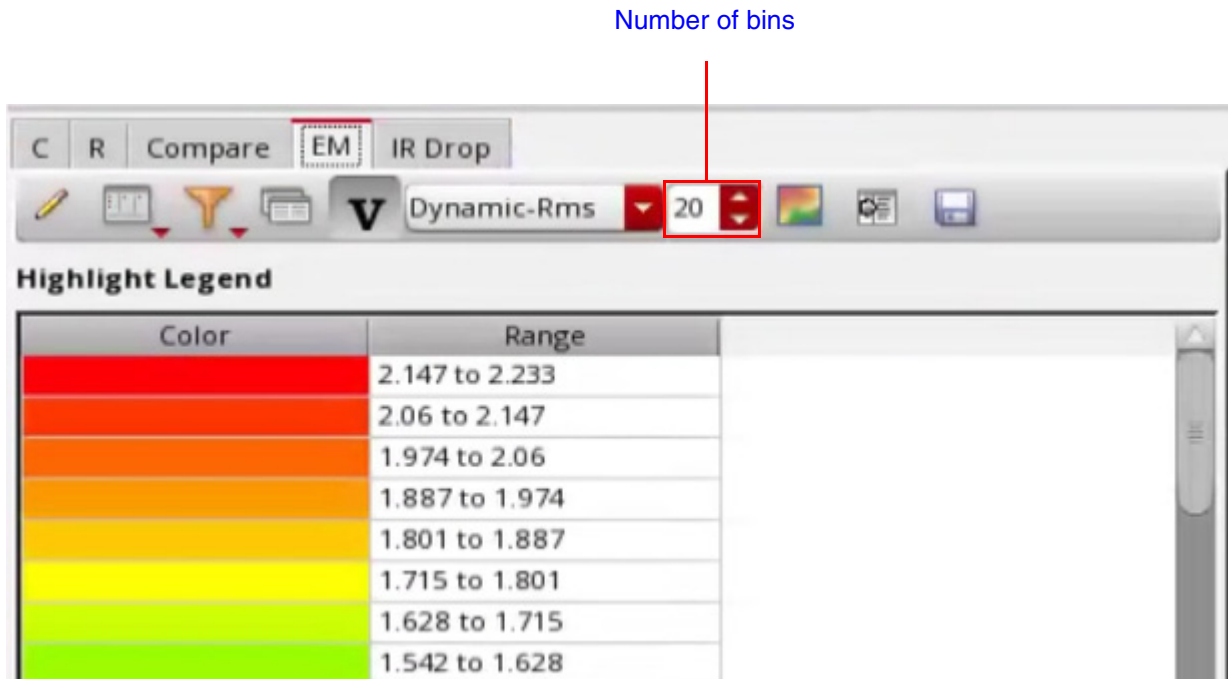


A highlight legend showing the color map to the value ranges is also displayed.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

You can control the number of rows displayed in the *Highlight Legend* table using the *Number of Bins* spin box. The default value set in this spin box is 20. You can specify any integer value between 1–60 in this spin box.

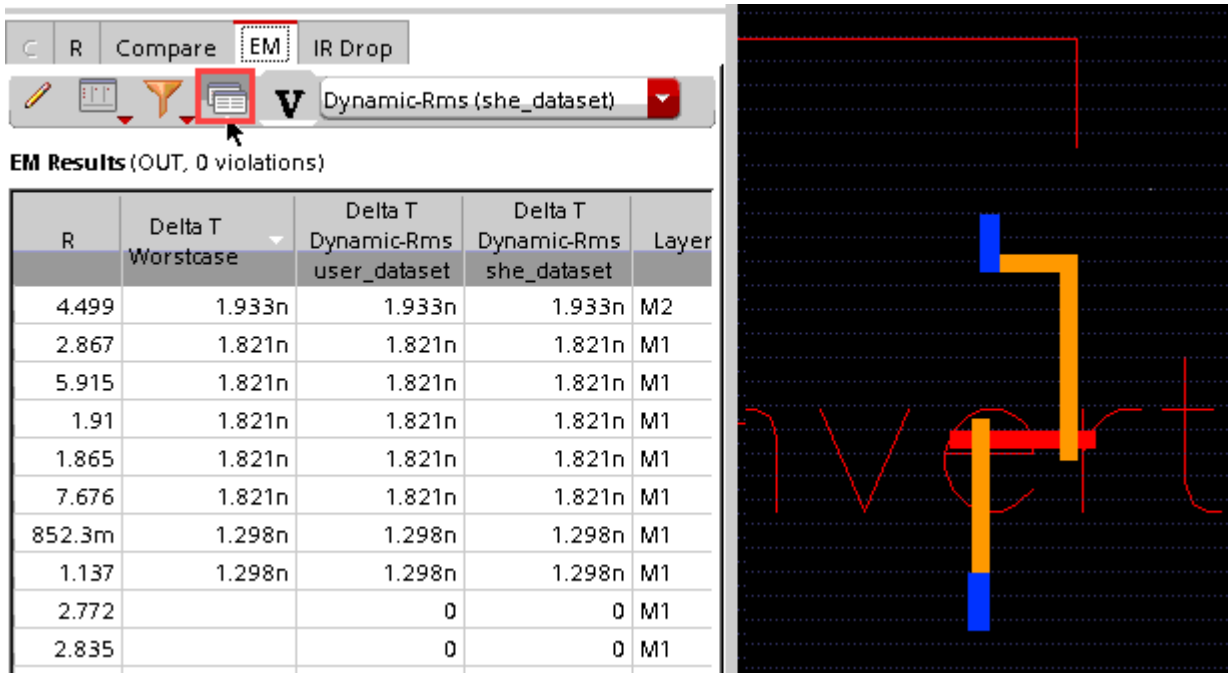


4. For better analysis of the highlighted values, it is better to open the EM results table along with it. For this, click *Show/Hide Delta T values in the EM Results table*  on the toolbar. This command is visible only when you click .

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

The EM results table are displayed and the highlight is also maintained on the canvas, as shown below.



EM Results (OUT, 0 violations)

R	Delta T Worstcase	Delta T Dynamic-Rms user_dataset	Delta T Dynamic-Rms she_dataset	Layer
4.499	1.933n	1.933n	1.933n	M2
2.867	1.821n	1.821n	1.821n	M1
5.915	1.821n	1.821n	1.821n	M1
1.91	1.821n	1.821n	1.821n	M1
1.865	1.821n	1.821n	1.821n	M1
7.676	1.821n	1.821n	1.821n	M1
852.3m	1.298n	1.298n	1.298n	M1
1.137	1.298n	1.298n	1.298n	M1
2.772		0	0	M1
2.835		0	0	M1

Viewing Electrical Information in the Layout Window

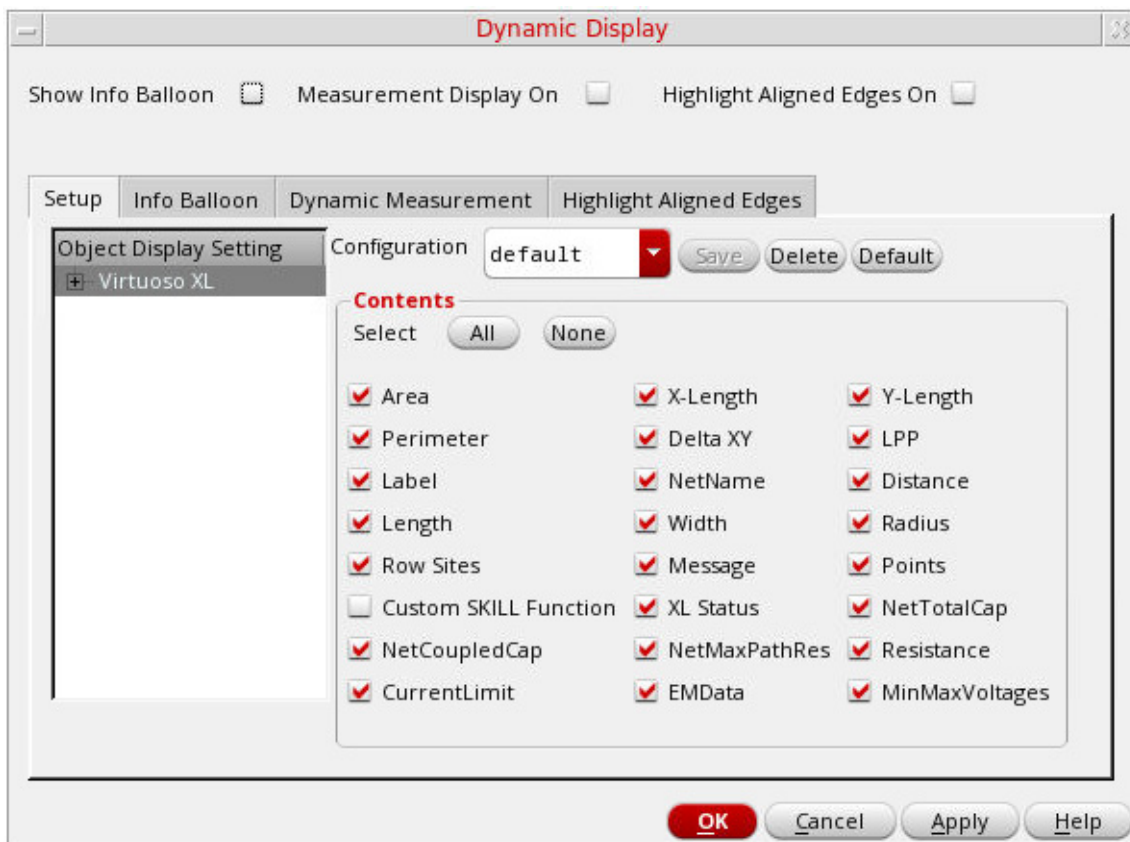
EAD provides various features to let you view electrical information in the layout canvas.

- [Using Info Balloons](#)
- [Using Point-To-Point Info Balloons](#)
- [Using Measurement Display](#)

Using Info Balloons

You can use the layout editor's *Show Info Balloon* feature to see the details of EAD parameters when you move the mouse pointer over nets in the layout canvas. To do this,

1. From the layout window menu bar, choose *Options – Dynamic Display* and enable *Show Info Balloon* at the top of the form.

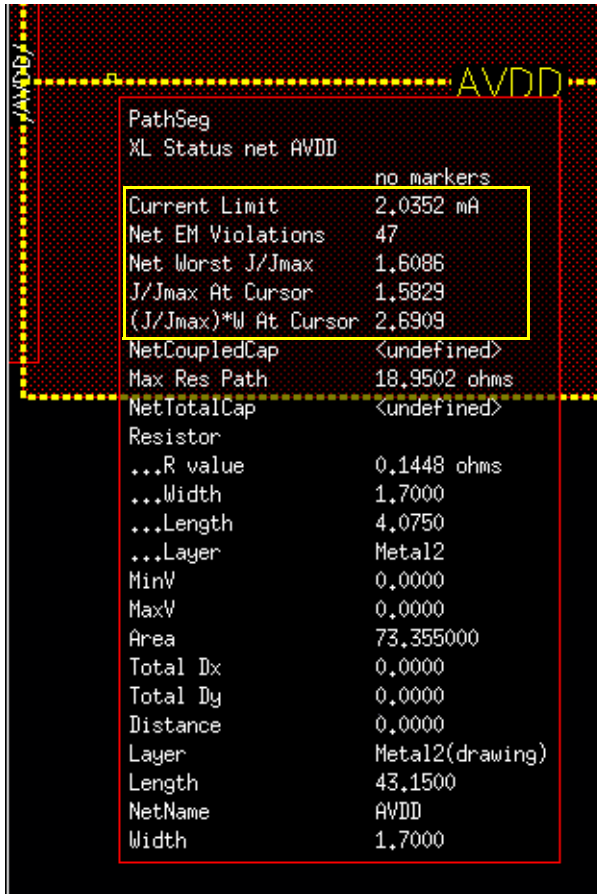


2. Choose the information you want to display from the *Contents* pane and click *OK*.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

3. Move your mouse pointer over a net in your design to see an info balloon containing the EM information for the highlighted segment.



Note: Recommended width information is displayed along with the EM violations only when the *EM Data* parameter is selected in the Dynamic Display form because recommended width is calculated only if EM analysis is performed.

The precision of the values displayed in the Info Balloon is controlled by the `displaySigDigits` environment variable. The default is 4 and the maximum value is 10.

To change the precision, type the following in the CIW:

```
envSetVal("layout" "displaySigDigits" 'int integer_value)
```

4. Move your cursor over a different net.

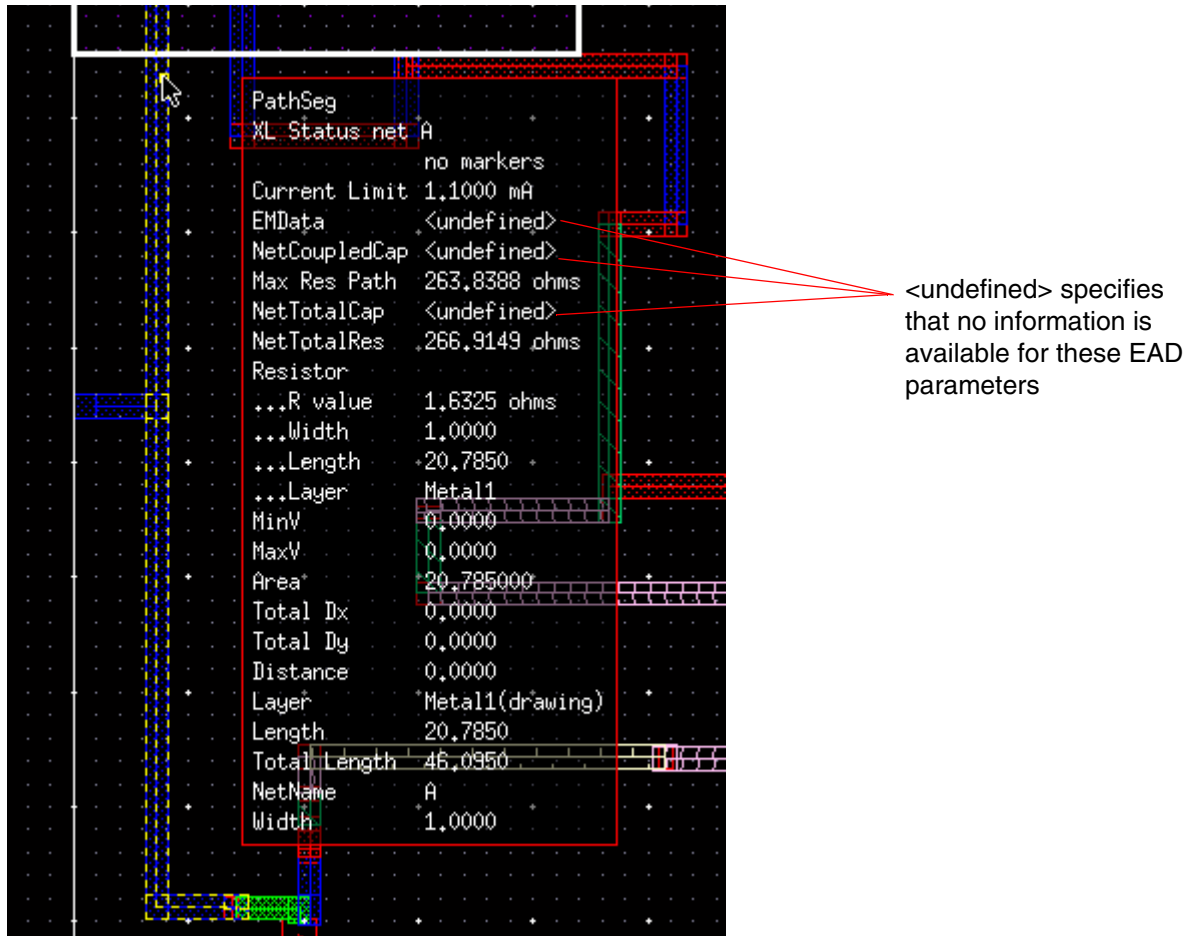
The original info balloon disappears and a new one opens containing information on the new net.

Note: If the calculated information for any EAD parameter is not available for a net, the Info Balloon shows the corresponding value as `<undefined>`. For example, you

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

performed an R-only extraction (that is, *Extraction Mode* on the *Extraction* tab of the Layout EAD Options form is set to R), but on the Dynamic Display form, you have selected the options to display both the resistance and capacitance values. In this case, capacitance values will be displayed as `<undefined>`, as shown in the figure below.

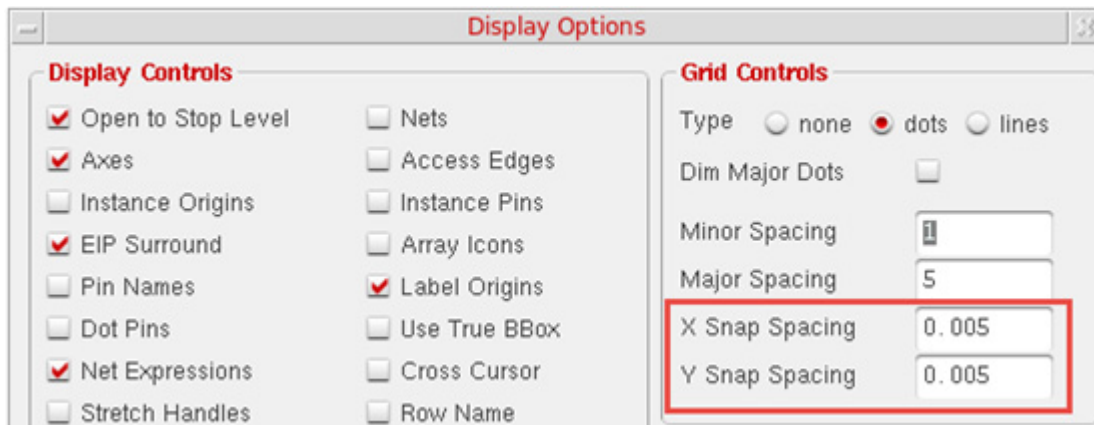


If you find any inconsistency between the mouse location or the information being displayed in the info balloon; for example, the current limit and J/Jmax are reported as `Undefined` and 0, respectively, or the values of some other object as shown; it could be because the point is getting snapped outside the bounds of the DB object while displaying the info balloon for that object. In this case, you can set either or both of the following two settings in Layout EAD:

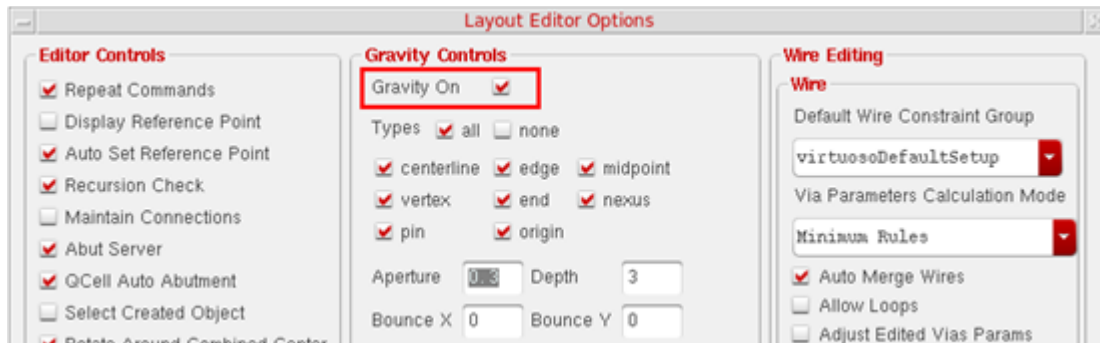
Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

- ❑ In the Layout EAD window, choose *Options – Display* to open the Display Options form, and reduce the values in the *X Snap Spacing* and *Y Snap Spacing* fields.



- ❑ In the Layout EAD window, choose *Options – Editor* to open the Layout Editor Options form, and select the *Gravity On* check box.

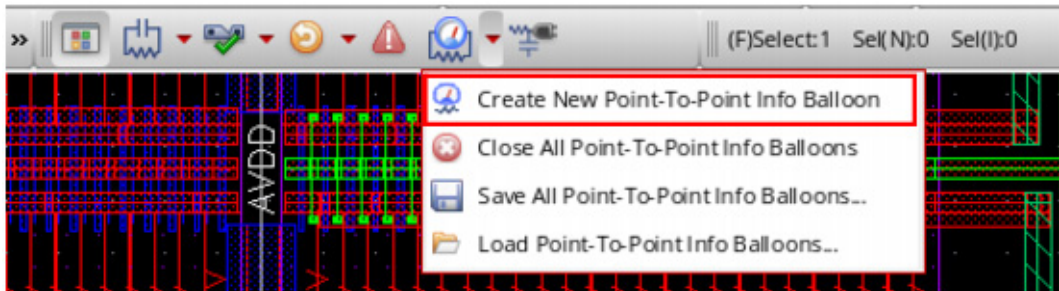


Using Point-To-Point Info Balloons

You can use the Point-To-Point Info Balloon feature to interactively view EAD parameters for the segment between any two points in your design. Point-to-point info balloons remain visible on the canvas until you close them either individually by clicking the 'x' in the top-right corner or all at once using the *Close All Point-To-Point Info Balloons* command from the main EAD toolbar or from the EAD Browser toolbar. They are also closed automatically when you exit Layout EAD.

To show point-to-point info balloons,

1. Click *Create New Point-To-Point Info Balloon* from the main EAD toolbar or from the EAD Browser toolbar.



2. (Optional) Change the snapping mode for the points by pressing the *s* bindkey. For more details, refer to [Snapping Start and End Points to Existing Nodes](#).

Note: The snapping mode is also set by the [P2PSnapping](#) environment variable, which is by default set to "node". Use the *s* bindkey to use a snapping mode other than the one set by this variable.

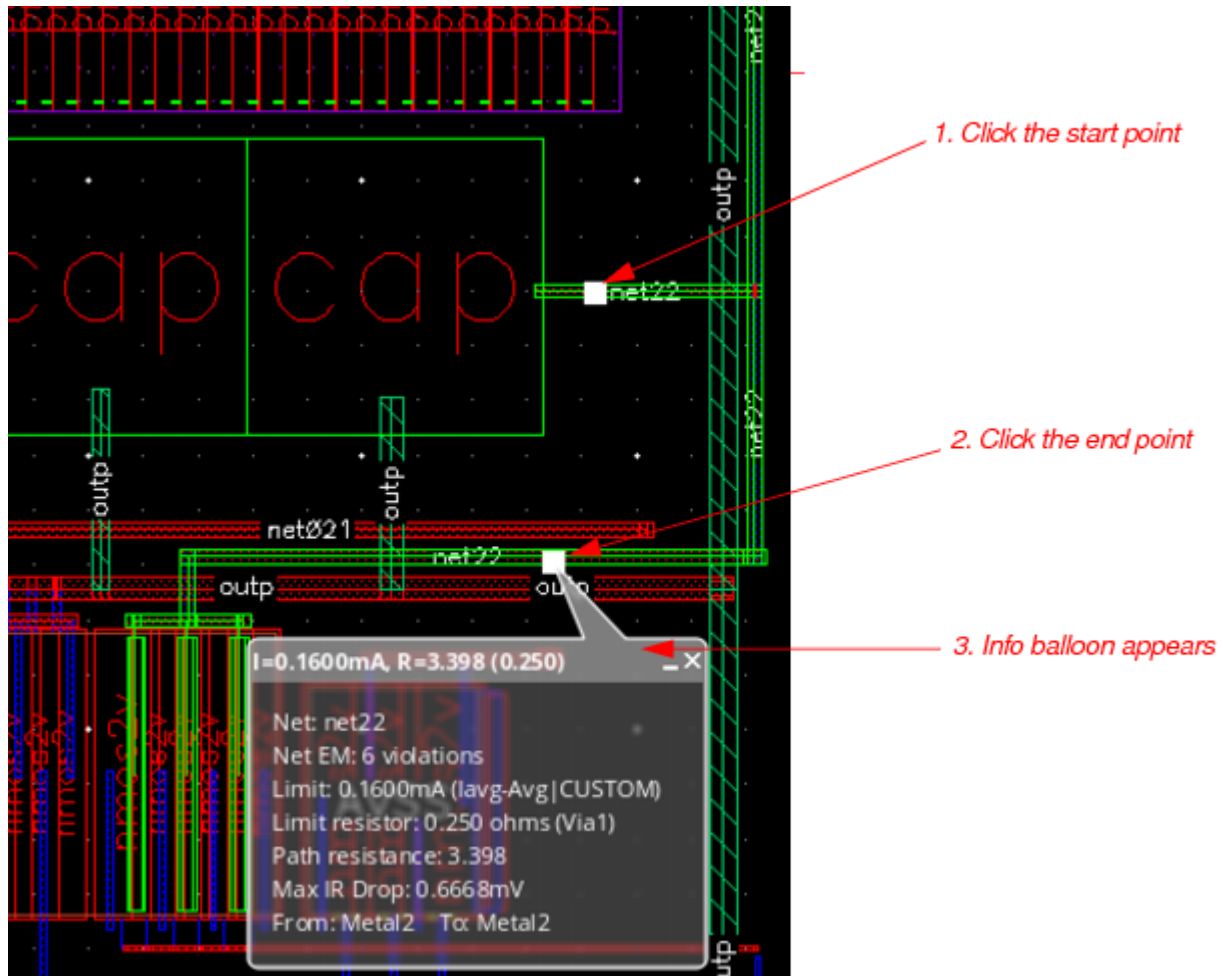
3. Click a start and end point on any net.

Layout EAD displays an info balloon showing the EM violation information for the net (across all datasets), the current limit for the path, the resistance value of the limiting resistor, and the total resistance of the path. The limiting resistor refers to the resistor in the point-to-point path that can carry the least amount of current as reported by the current limit information on the info balloon. The title of the info balloon displays a summary of this information, which includes the current limit, point to point path resistance, and the resistance value of the limiting resistor (in parenthesis).

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

Layout EAD also **highlights in red** the **first segment** of the net that contains the **first limiting resistor**.



Important Points to Note

- If the from and to points of an info balloon are within $\pm \text{DBUPerUU}/10$, Layout EAD considers them as the same point and therefore, the path resistance value is 0.
- While you are dragging mouse from the first point to the second point, if the `updateP2POnDrag` environment variable is set to `t`, the resistance details are dynamically updated to show the resistance up to the current location of mouse. In case of large nets, frequent dynamic updates might make the info balloon look unresponsive because the tool is taking time to calculate the resistance values from the first point to the current location. Therefore, it is recommended to delay the update frequency by changing the value of the `clickClickUpdateDelay` environment variable from the default value (0.05) to a higher value.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

- If you do not want to update the info balloons while you are dragging the mouse from the first point to the second, set the `updateP2POnDrag` environment variable to `nil`.
- If the layout contains mosaic instances corresponding to the instances in the schematic, you need to convert those mosaic instances to regular instances so that the resistance can be extracted from the paths connected to them and added to the R value on the point-to-point info balloon. However, if the mosaic instances are only routing shapes and do not correspond to any instance in the schematic, you can press the `Ctrl + N` bindkey after selecting the first point for the info balloon to find the nets connected to them and extract resistance.
- By default, Layout EAD processes and extracts all the shapes present on a net while you select points for point-to-point info balloons. However, when the number of shapes on a net is large, this can impact performance. In such a case, you can set the `clickClickUseNetCache` environment variable to `nil`.

Snapping Start and End Points to Existing Nodes

When you select points for info balloons, Layout EAD inserts user nodes at the nodes closest to the selected points and re-extracts parasitics. As re-extraction takes time, you can choose to snap the points to the existing nodes or terminals on the same net and reuse the already extracted data. This helps in improving the performance of the point-to-point info balloon display command by avoiding an additional extraction.

Important

Though, in general, snapping improves the performance of the point-to-point info balloon display, you may still observe some delay if the net on which you are creating the balloon is highlighted in red in the *Summary* pane. This is because the red highlight indicates that the parasitics for that net need to be re-extracted due to a change in the design or the extraction setup, which is done when the info balloon is created.

By default, snapping is off and the points selected by the user are considered as the start and end point for an info balloon. You can use any one of the following two snapping modes:

- **Snap to closest node:** In this mode, the start and end points are snapped to the closest nodes, and resistance between those nodes is shown.
- **Snap to closest terminal:** In this mode, the start and end points are snapped to the closest terminals, and resistance between those terminals is shown.

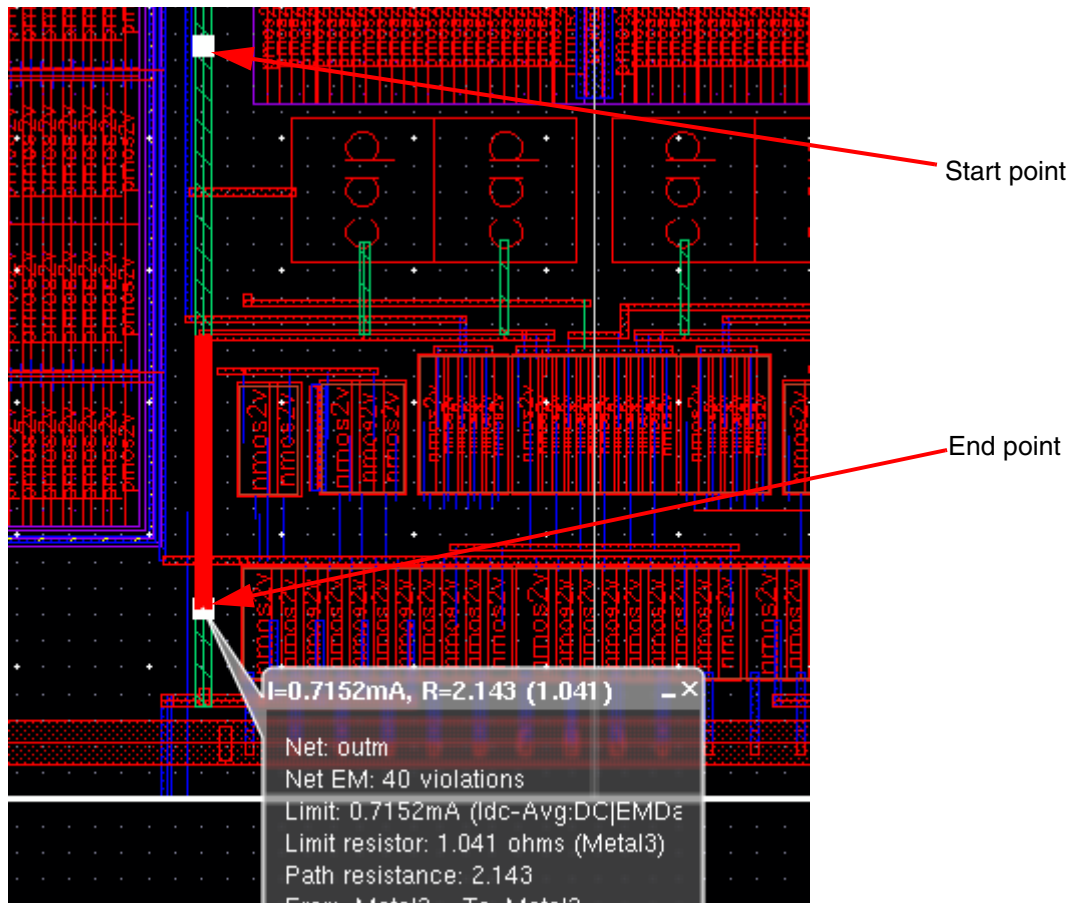
To switch between the snapping modes in [step 2](#), press the `s` bindkey.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

The following figures show how the end points for an info balloon change for different snapping modes:

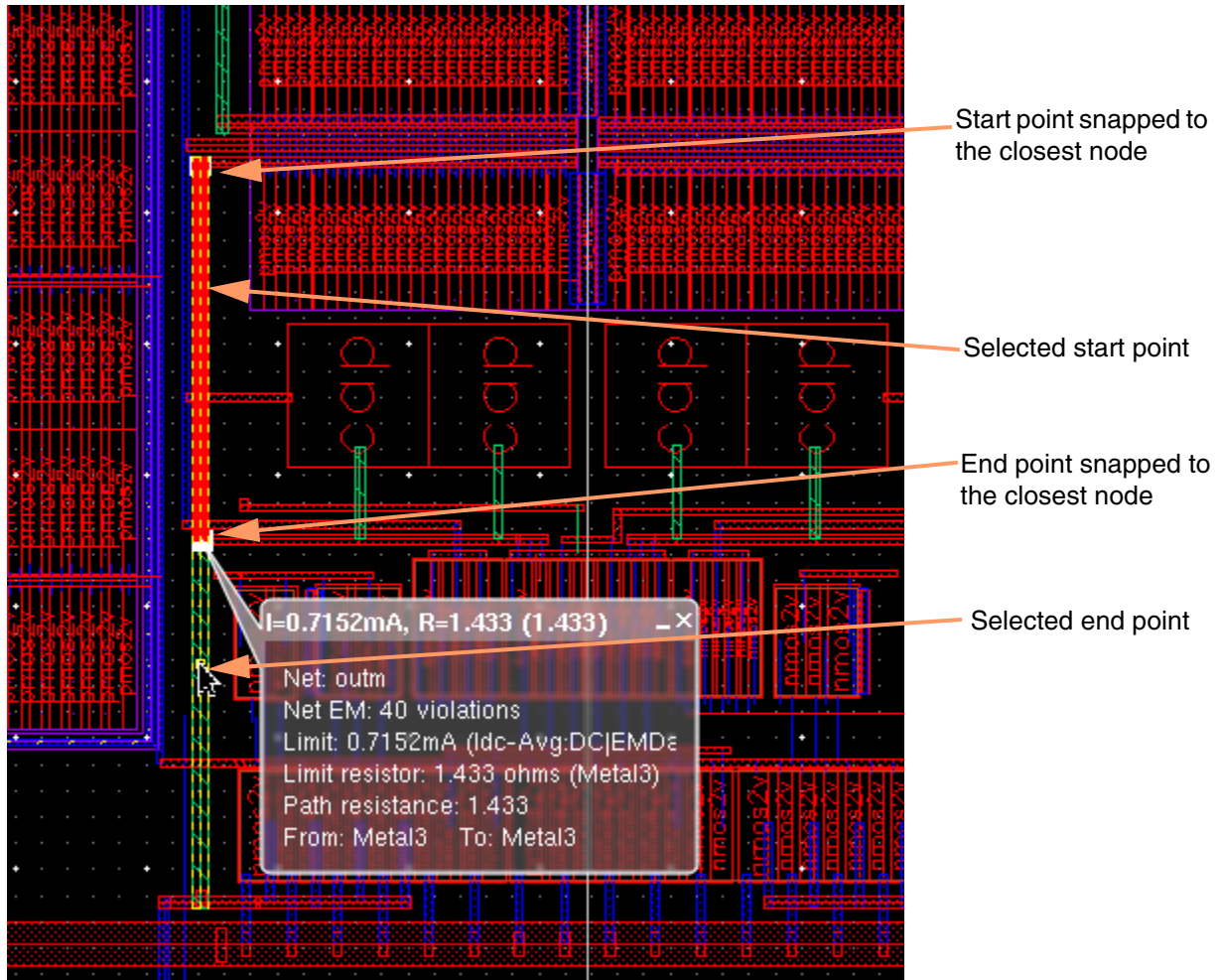
Info balloon with snapping off



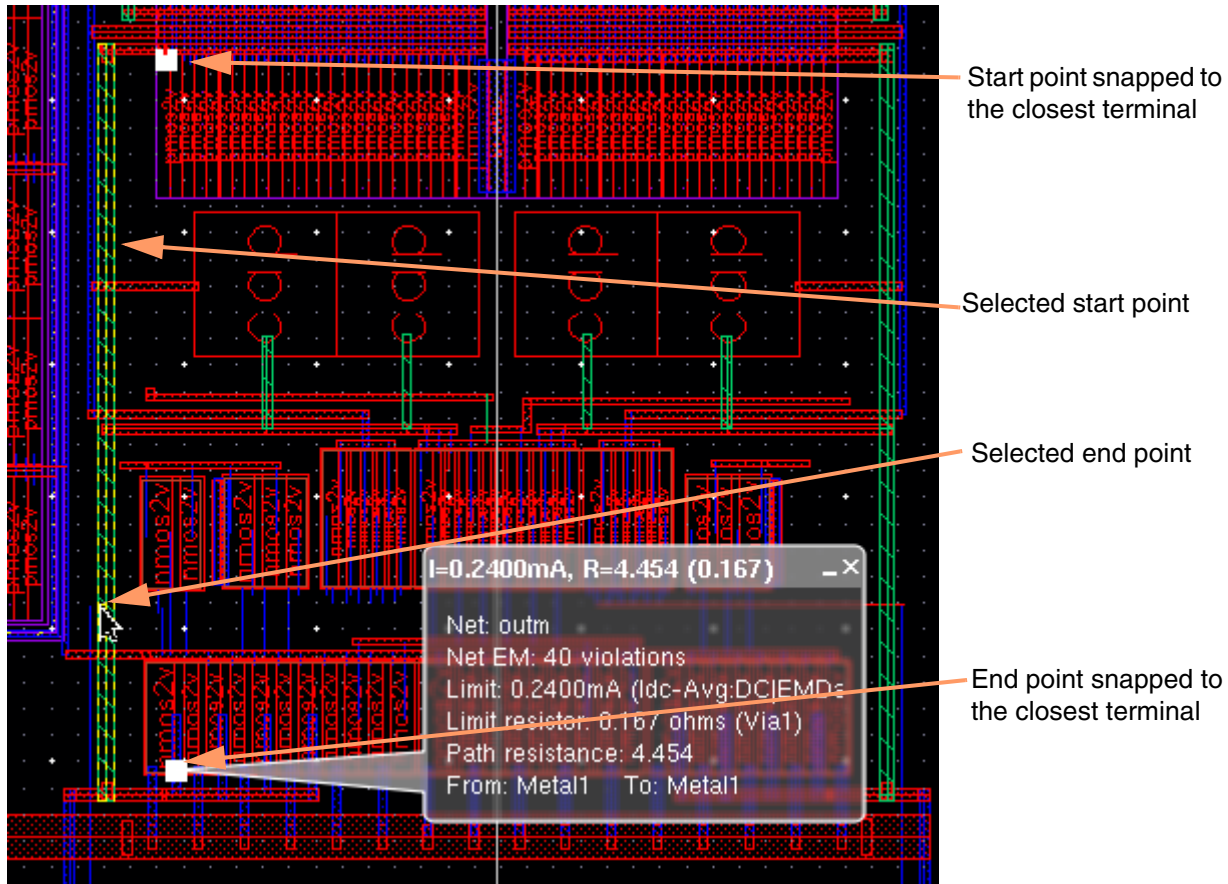
Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

Info balloon with snapping to the closest node



Info balloon with snapping to the closest terminal



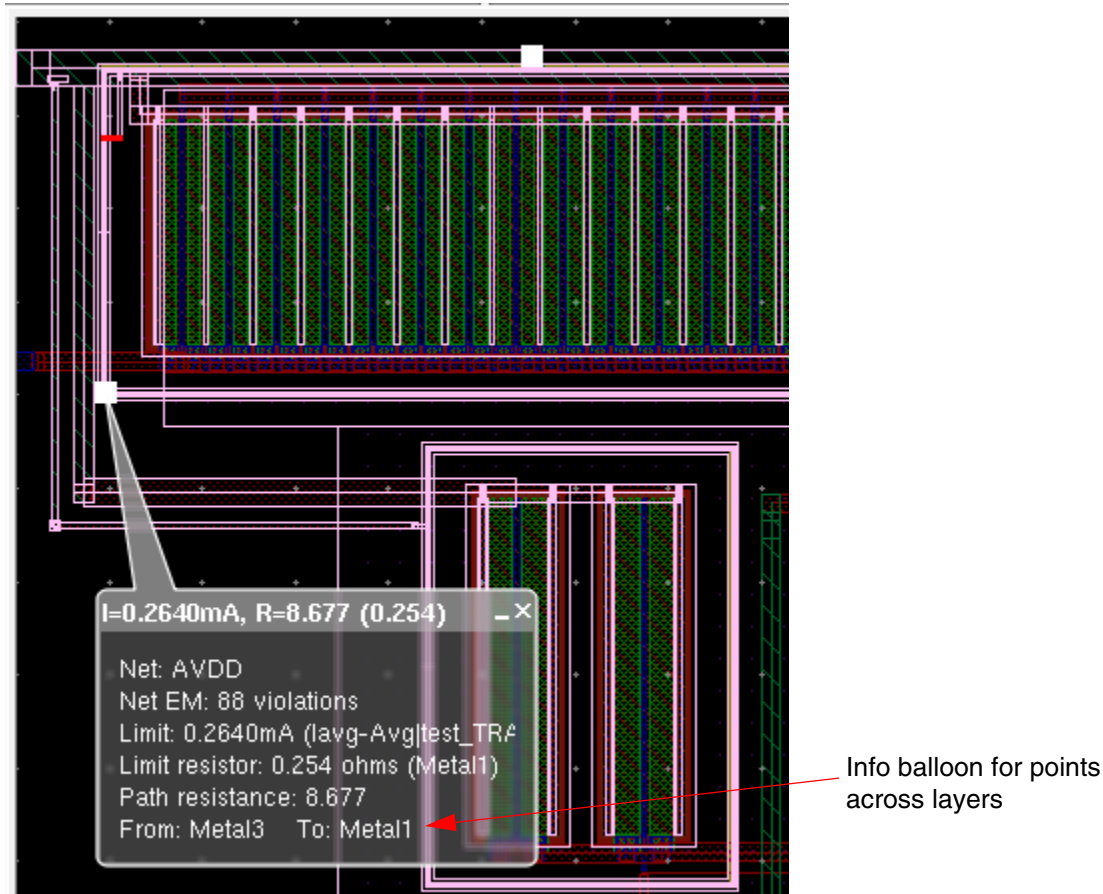
Displaying Info Balloons with Start and End Points from Different Layers

While selecting the start and end points for info balloons, press the **1** key to traverse through the overlapping interconnect layers. The name of the current layer is displayed on the info

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

balloon to help you choose the correct layer. The CIW also shows the target layer name. This helps in selecting two points across different layers to view the electrical information.



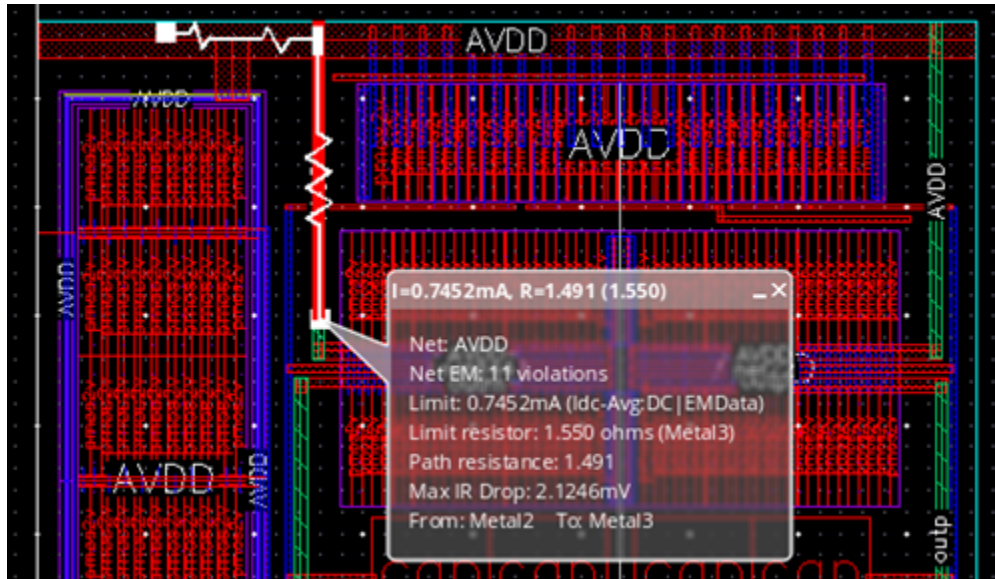
Displaying Resistance Paths Between the Start and End Points of an Info Balloon

You can use the `displayP2PResistancePath` environment variable to specify your requirements to display resistance paths between the start and end points of an info balloon. The resistance paths are displayed in addition to the halo that highlights the path between the selected points.

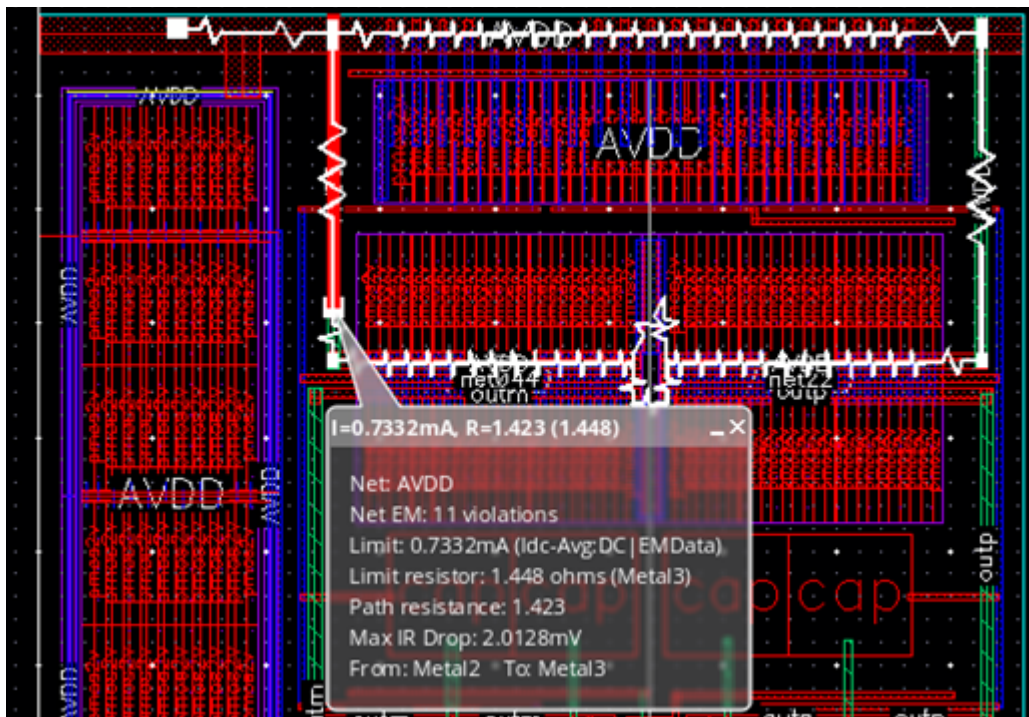
Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

By default, this variable is set to "none" and EAD does not show any resistance path. Set it to "singlePath" to show the shortest path containing the limiting resistance, between the two points, as shown below.



If you want to display all the resistance paths between the points, as shown below, set the variable to "allPaths".



If you do not want to display resistance paths, set this variable to "none".

Updating Info Balloons to Reflect Layout Edits

You can edit any segment between the two points directly on the layout canvas. You can set the update mode to specify whether to update the information on an info balloons as per the layout changes.

- If *Automatic Update Mode* is set, EAD automatically updates the layout overlay as well as the information shown in the balloon.
- If *Manual Update Mode* is set, you must click the *Update Parasitics & EM* button on either of the EAD toolbars (or press **Ctrl+u**) to update the display and information shown in the info balloon.

Saving Info Balloons to XML Files

You can use the *Save All Point-to-Point Info Balloons* command from the EAD toolbar to save the displayed info balloons in an XML file. This helps in saving the details of critical nets that you identified and would need to mark again while reviewing the design.

To save the info-balloons already displayed on the layout canvas, choose *Save Point-to-Point Info Balloons* from the EAD toolbar. The Save Point-to-Point Info Balloons form is displayed. Specify the location and name for the XML file in which you want to save the info balloons, and click *Save*.

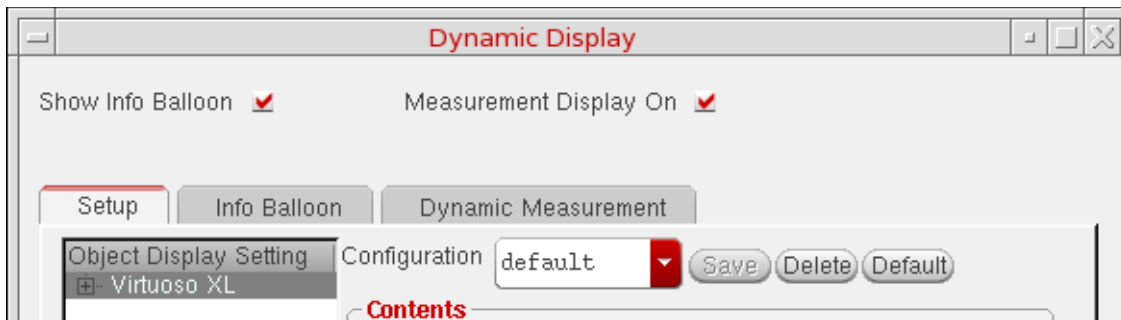
To load the saved info balloons, choose *Load Point-to-Point Info Balloons* from the EAD toolbar. The Load Point-to-Point Info Balloons form is displayed. Specify the XML file and click *Load*. The info balloons are displayed at their original locations. The titles of info balloons loaded from the saved XML file are marked with a *. If the design is changed after saving the info balloons, their title is highlighted with a yellow background to indicate that they are not up-to-date. In this case, you need to click *Update Parasitics & EM* on the EAD Browser toolbar to update the display in information on the info balloon. However, If *Automatic Update Mode* is set, EAD updates the info balloons whenever a design is changed.

Note: The cellview information is also saved along with info balloons in the XML file. This prevents loading info balloons into a cellview other than the one for which it was saved.

Using Measurement Display

Use the layout editor's *Measurement Display On* feature to see real-time current information while you are routing your design, for example, using the *Create Wire* command. To do this,

1. Choose *Options – Dynamic Display* to open the Dynamic Display form and enable *Measurement Display On* at the top of the form.



2. Enable the EM-related information you want to see; for example, *Resistance*, *CurrentLimit*, and *EMData* as shown above.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

During *Create Wire*, you now see the real-time information about the resistance and current limit of the wire being drawn, along with the number of EM violations being generated as you draw the net.



Highlighting and Fixing Violations in the Layout Window

To see a visual representation of the EM violations for a selected net in the layout window,

1. Dim the display using the *Dim* button in the Options toolbar on the VLS XL window.

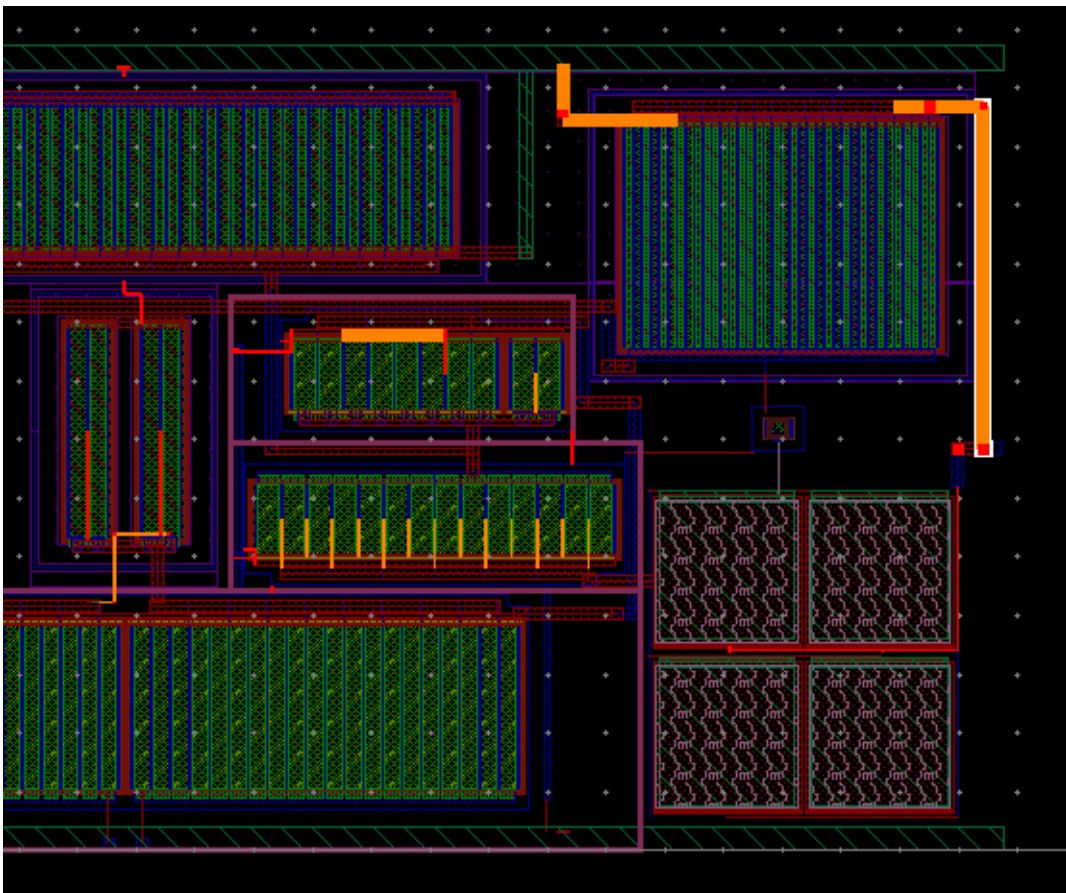


This dims the display of the cellview components making it easier to see the overlay in the canvas.

2. Use the *Highlight Worst-Case Violations for All Nets* button in the main EAD toolbar to see the worst case violations in the canvas.



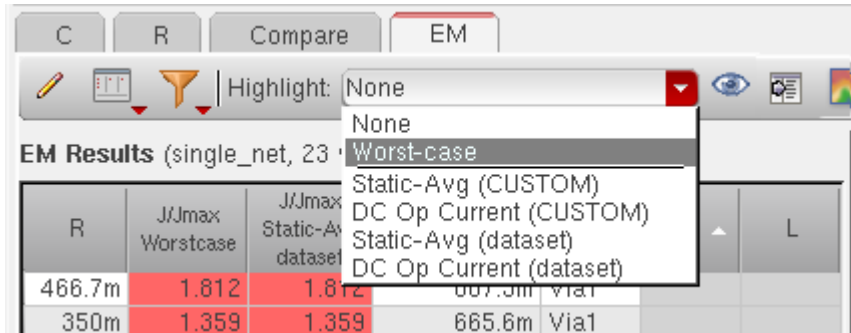
The worst case violations for all nets are highlighted.



Virtuoso Electrically Aware Design Flow Guide

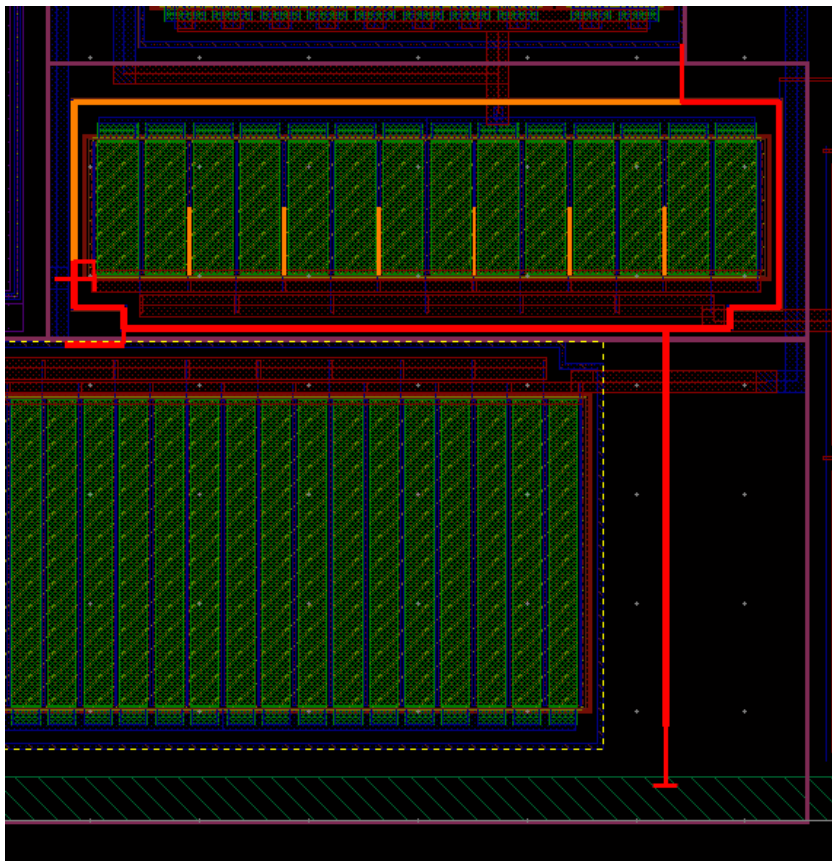
Performing Electromigration Checks and Fixing Violations

3. To analyze the violations for a specific net in more detail, go to the *EM* tab and choose an overlay to apply in the layout canvas.



4. Select a net in the summary table to highlight the violations on that net in the layout canvas.

The layout editor display automatically pans and zooms to the area of the design in question.



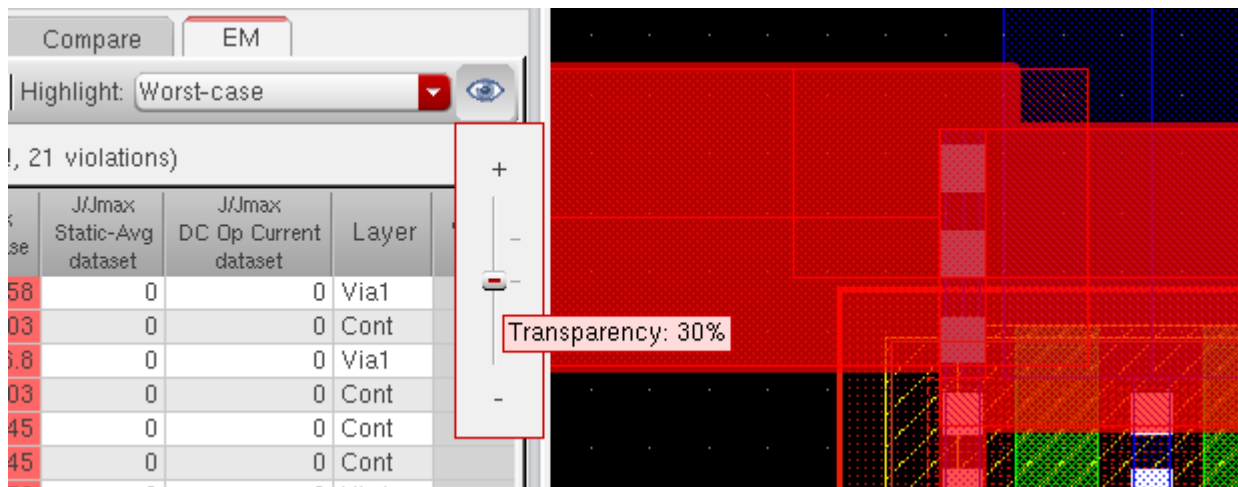
The color-coding used for the overlay is based on the settings in the [Environment](#) form.

Virtuoso Electrically Aware Design Flow Guide

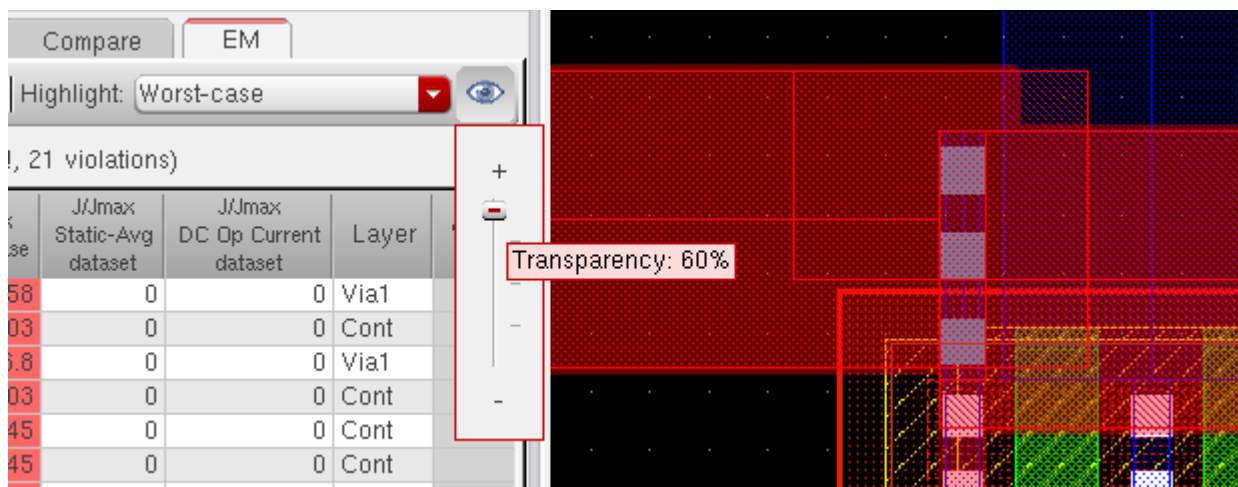
Performing Electromigration Checks and Fixing Violations

5. Use the *Highlight Transparency* slider to change the transparency of the overlay highlighting.

The default is 30%. You can change this using the `violationTransparency` environment variable.



Slide towards the '+' sign to increase the transparency and see more of the underlying geometry.



6. Switch on *Automatic Update Mode* so that parasitic and EM information is updated automatically as you edit your design in the canvas.

Note: If you have *Manual Update Mode* enabled, you must click the *Update Parasitics & EM* button on either of the EAD toolbars (or press `Ctrl-u`) to update the display.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

7. Select a net segment with an EM violation and perform some interactive edits to resolve the issues. For example:

- ❑ Use *Edit – Basic – Properties* to open the Edit Properties form and widen the wire
- ❑ *Stretch* or *Move* individual segments to better balance the current flowing through them
- ❑ Use *Create Wire* to change the way the net is routed or connected to neighboring components

With each change you make, EAD re-extracts parasitics for the changed net in real time and reruns the EM check. The color of the net changes in the layout window to reflect the status of any remaining violations on the net.

8. When you have finished fixing violations, use *File – Save* to save the partial layout.

The parasitic information for the design is updated and saved in the layout view, which can be used to rerun simulations in ADE Assembler to check if the specifications are still met.

For more information, see [Resimulating Designs with Extracted Parasitics](#).

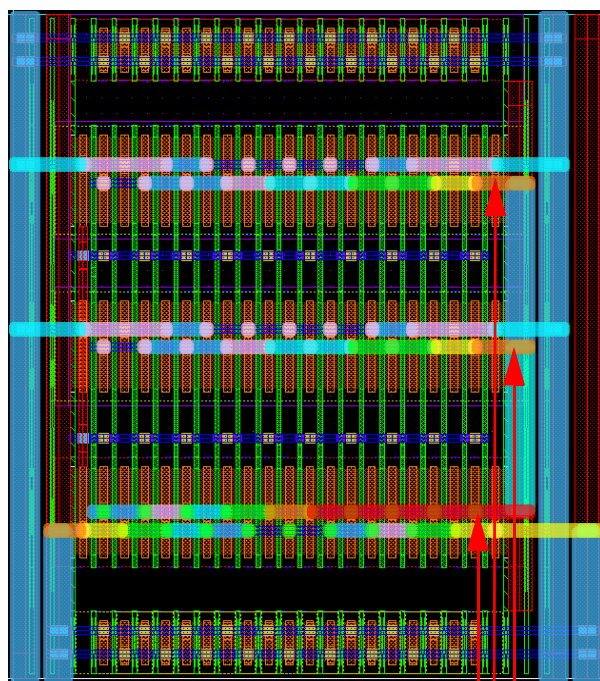
Fixing Violations Using EM Driven Trunk Optimization

EM Trunk Optimization is an automatic post-route EM-driven Pin to Trunk optimization feature. This feature automatically extracts the hint data generated by EAD and uses it to fix the EM violations on trunks, twigs, and vias of a completely routed net. The EM Trunk Optimization feature lets you quickly resize trunks to meet EM requirements. The extent of EM Trunk Optimization is controlled by selecting trunks or nets in the design. If no trunk or net is selected, then all the trunks in the design are optimized at once.

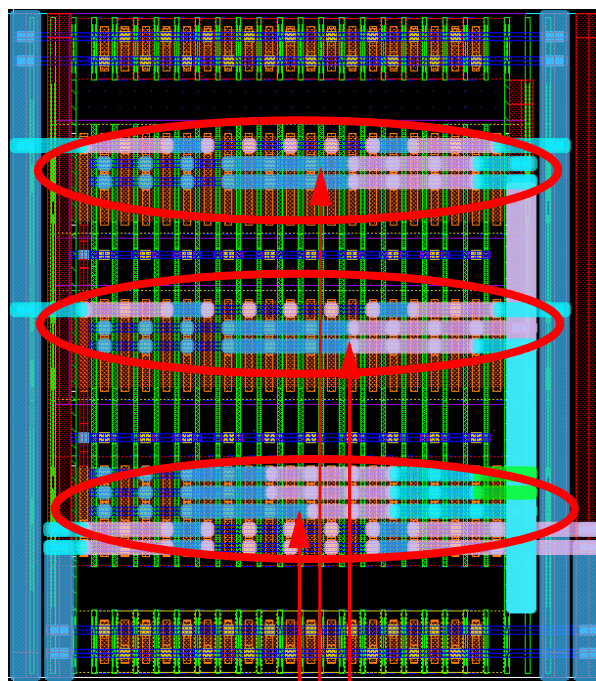
EAD analyzes the design and calculates hints that are used by EM Trunk Optimization to resize the trunk. Pin to Trunk routing is required to create the trunks. EM Trunk Optimization only works on trunks that were created by the Pin to Trunk Router.

Important

The EM Pin to Trunk Optimization feature is only available in Virtuoso Layout EAD.



EM violations on trunks reported by EAD



Trunks are optimized to meet EM rules

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

The Layout EAD toolbar has the following buttons for automatic EM-driven Pin to Trunk optimization:



■ Optimize Trunk Width for Complete Network

Starts the *Optimize Trunk Width for Complete Network* feature. Click the drop-down arrow to see the EM optimization modes listed in the table below.

Command	Let's you
<i>Expand to Fix EM Violations</i>	Expand the width of trunks and increases the number of cuts in via arrays according to the hints provided by EAD.
<i>Shrink Oversized Trunks</i>	Reduce the width of the oversized trunks and the number of cuts of oversized via arrays.
<i>Shrink and Expand</i>	This option comprises of two steps: ■ Shrink trunks and via arrays by running the <i>Shrink Oversized Trunks</i> command. ■ Expand trunks and vias by running the <i>Expand To Fix EM Violations</i> command.
<i>Options</i>	Display the P2T EM Optimization form. For more information about the options in this form, see P2T EM Optimization .

■ Select Trunk Optimization Style

Selects the style of optimizing a trunk. Click the drop-down arrow to see the styles for optimizing a trunk to resolve EM violations.

Command	Lets you...
<i>Widen - No Stranding</i>	Widen the trunks to fix EM violations.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

<i>Parallel Stranding</i>	Insert additional parallel trunks to fix EM violations.
<i>Stacked Stranding</i>	Stack the trunks to fix EM violations.

■ Toggle Trunk Tapering for Complete Network



Enables trunk tapering during trunk widening. This button turns on or off trunk tapering. By default, trunk tapering is off. Click the button to turn on trunk tapering.

Note: The *Toggle Trunk Tapering for Complete Network* button is disabled when the *Trunk Optimization Style* is selected either as *Parallel Stranding* or *Stacked Stranding*. Tapering feature is only supported for widening of trunks and not supported for stacking and stranding of trunks.

Note: The *EM Trunk Optimization*, *Select Trunk Optimization Style*, and *Toggle Trunk Tapering for Complete Network* buttons in all open cellviews are automatically synchronized; that is, if any of these buttons is reset in one cellview, the corresponding button is automatically reset to the same value in all other open cellviews.

Important Point to Note

When no net is selected on the layout canvas or in the *Navigator Assistant*, the EM trunk optimization is applied to all the nets. However, when one or more nets are selected in the layout canvas or in the *Navigator Assistant*, the EM trunk optimization is applied only to the selected nets.

Optimizing EM Driven Trunks

This section describes the various ways in which you can optimize trunks to fix the EM violations in your design.

- [Expanding Trunks to Fix EM Violations](#)
- [Shrinking Oversized Trunks](#)
- [Shrinking and Expanding Trunks](#)

Expanding Trunks to Fix EM Violations

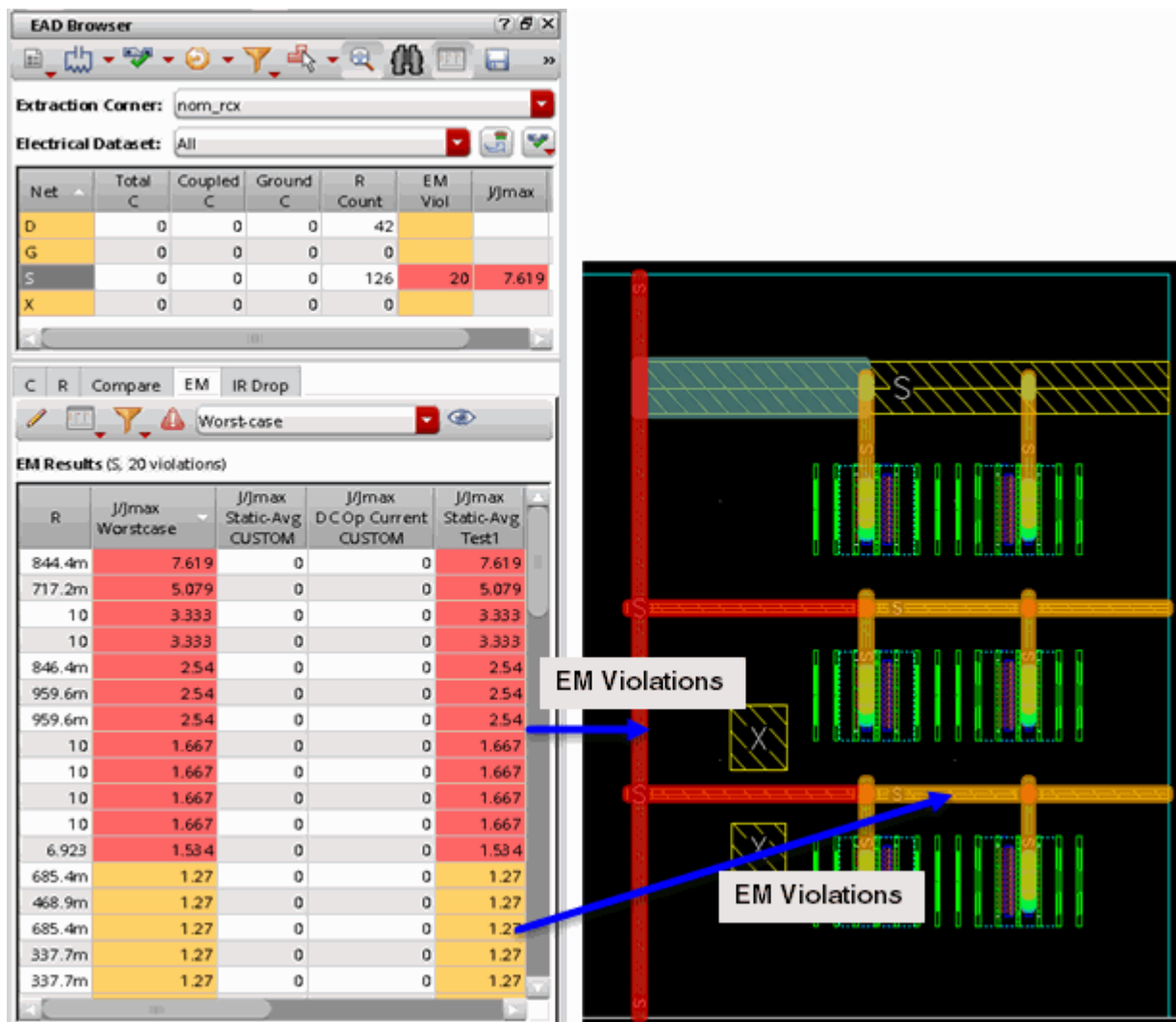
This section covers how the width of the trunks are expanded and the cuts of vias are increased according to the hint values provided by EAD. You will also see how the width of a trunk is expanded according to the trunk optimization style that is selected.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

To do this:

1. Choose a dataset from the *EM Dataset* drop-down list.
2. Click the *Extract Parasitics* button in the EAD toolbar.
3. Click the *Run EM Check* button in the EAD toolbar.
4. Click *Highlight Worst-Case Violations for All Nets* to view and highlight the violations in the layout canvas.
5. Select a net in the summary table to highlight the violations on that net in the layout canvas. You will see a detailed breakdown of the EM violations present for the net selected in the summary pane. If there are no violations for the selected net, the table is empty.



Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

6. Click the *Select Trunk Optimization Style* button in the EAD toolbar. You can select the trunk optimization style as: *Widen - No Stranding*, *Parallel Stranding*, and *Stacked Stranding*. The result of trunk optimization varies depending upon the trunk optimization style that is selected. To see how the trunk optimization varies depending on the selected trunk optimization style, see [Expand with Widen - No Stranding](#), [Expand with Parallel Stranding](#), and [Expand with Stacked Stranding](#).
7. Click the *Optimize Trunk Width for Complete Network* button with the default option of *Expand to Fix EM Violations*.

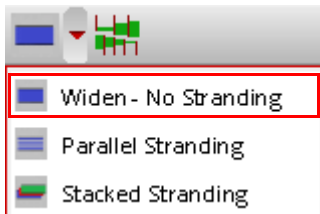
Expand with Widen - No Stranding

When the trunk optimization style is selected as *Widen - No Stranding*, the width of the trunk is widened according to the width hint value provided by EAD. If the widened trunk is too close to the neighboring trunks, the neighboring trunks are pushed away to meet the spacing requirement.

Note: The neighboring rectangles and instances are not pushed.

To do this:

1. Click the *Select Trunk Optimization Style* button in the EAD toolbar and select *Widen - No Stranding*.

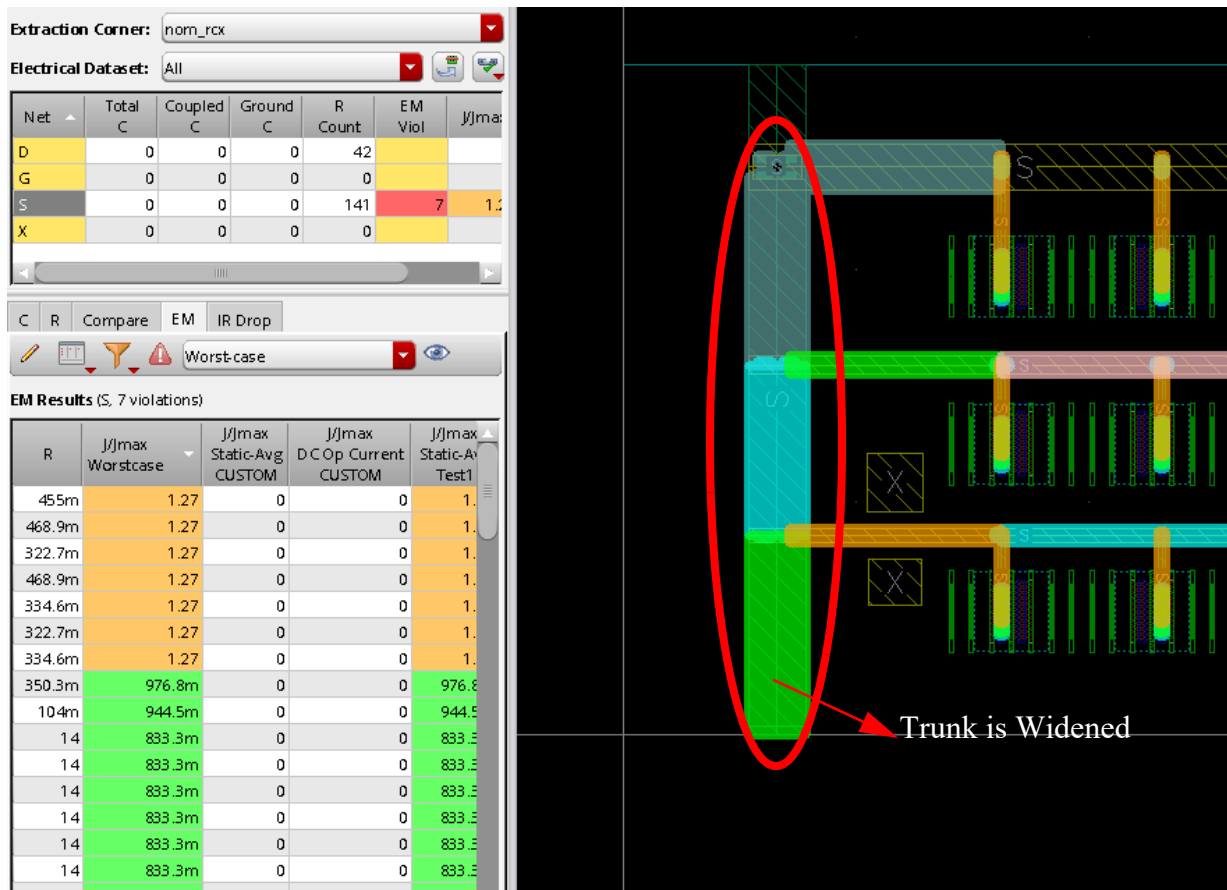


2. Click the *Optimize Trunk Width for Complete Network* () button with the default option of *Expand to Fix EM Violations*. As a result, the top trunk is widened. However,

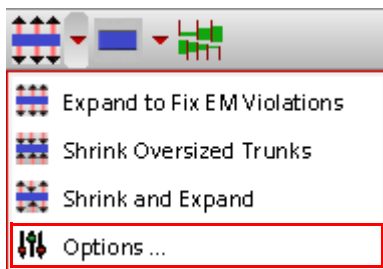
Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

the bottom trunk is not completely widened to avoid DRC spacing violations. The EM violations on the bottom trunk and twigs are highlighted in orange.



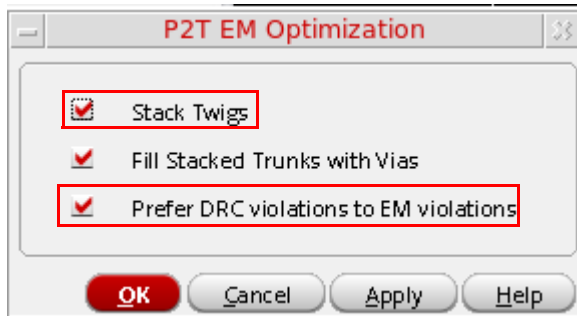
- To fix the remaining EM violations in the design, click *Options* from the *Optimize Trunk Width for Complete Network* drop-down list.



Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

4. The P2T EM Optimization form is displayed. From this dialog box, select *Stack Twigs* and *Prefer DRC Violations to EM Violations* options.



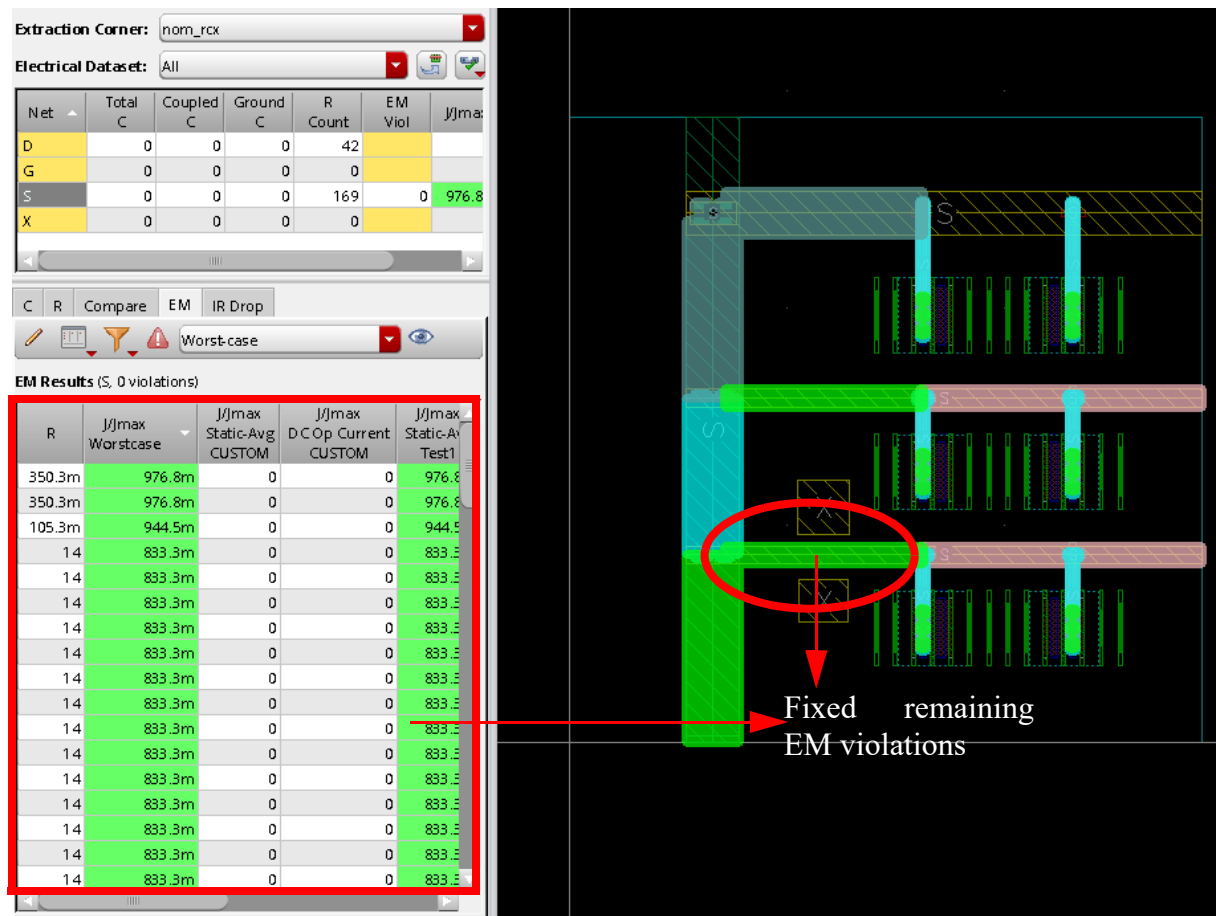
When there is insufficient space for a trunk to expand, the trunk is expanded depending on the setting of the *Prefer DRC Violations to EM violations* option. By default, the *Prefer DRC violations to EM Violations* option is deselected. In such a case, the width of the trunk is expanded to a value less than the hint value provided by the EAD analysis engine to keep the layout DRC-error free. When the *Prefer DRC Violations to EM Violations* option is selected, the width of the trunk is expanded according to the hint value generated by EAD, leaving a short or a spacing violation between the widened trunk and a neighboring shape.

5. Click the *Optimize Trunk Width for Complete Network* button. The following figure shows that there are no more EM violations. All the violations on the twigs and the bottom

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

trunk are resolved. The following figure displays the result of how the trunks are resized in such a case.



Expand with Parallel Stranding

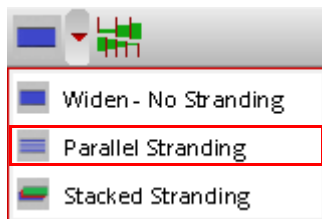
When the trunk optimization style is selected as *Parallel Stranding*, the number of strands are calculated based on the hint width provided by EAD divided by the original width of the trunk. The final number of strands are then rounded up to the nearest integer. Also, the strands are arranged in parallel on the same layer. The length of each strand is identical to the length of the original trunk. In addition, vias are added between each strand and all intersecting segments. The adjacent trunks parallel to the strands are pushed away to avoid shorts and the spacing violations.

To do this:

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

1. Click the *Select Trunk Optimization Style* button in the EAD toolbar and select *Parallel Stranding*.



2. Click the *Optimize Trunk Width for Complete Network* () button with the default option of *Expand to Fix EM Violations*. As a result, additional trunks are added and the EM violations on the selected net are fixed, as shown in the following figure.

Extraction Corner: norm_rcx

Electrical Dataset: All

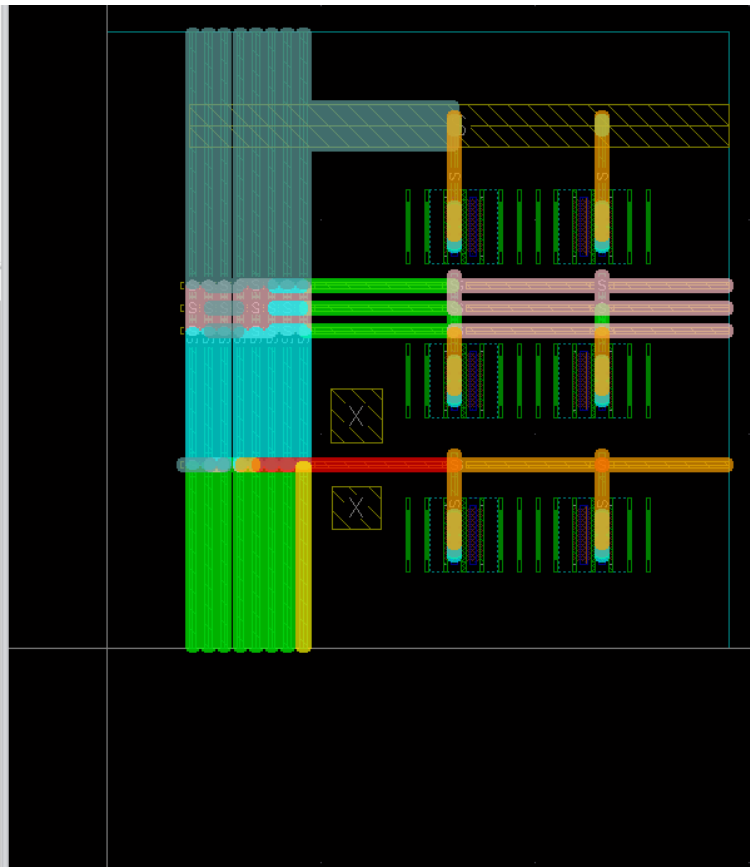
Net	Total C	Coupled C	Ground C	R Count	EM Viol	J/Jmax
D	0	0	0	42		
G	0	0	0	0		
S	0	0	0	247	15	2.5
X	0	0	0	0		

C R Compare EM IR Drop

Worst-case

EM Results (S: 15 violations)

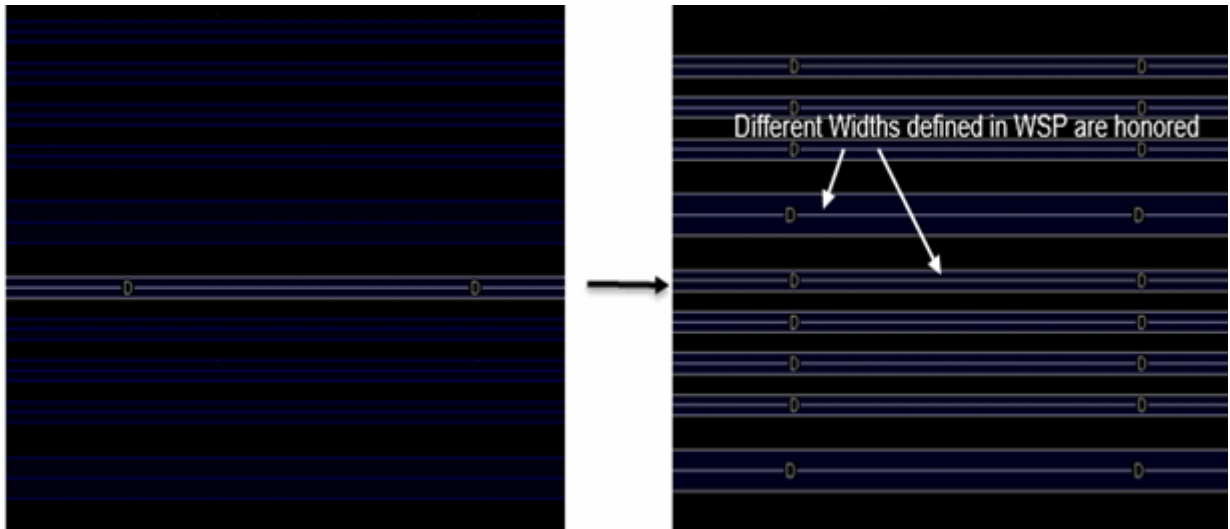
R	J/Jmax Worstcase	J/Jmax Static-Avg CUSTOM	J/Jmax D C Op Current CUSTOM	J/Jmax Static-Avg Test1
702.3m	2.54	0	0	2.5
73.51m	2.187	0	0	2.1
73.51m	1.849	0	0	1.8
10	1.667	0	0	1.6
10	1.667	0	0	1.6
73.51m	1.523	0	0	1.5
232.5m	1.27	0	0	1.2
685.4m	1.27	0	0	1.2
232.5m	1.27	0	0	1.2
325.8m	1.27	0	0	1.2
468.9m	1.27	0	0	1.2
468.9m	1.27	0	0	1.2
325.8m	1.27	0	0	1.2
73.51m	1.207	0	0	1.2
834.4m	1.009	0	0	1.0
834.4m	985.9m	0	0	985.9
834.4m	966.5m	0	0	966.5



Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

In case the original trunk is on a track pattern of WSP, the parallel strands are snapped to adjacent tracks. The width of each strand is identical to the width of the original trunk or WSP track width.



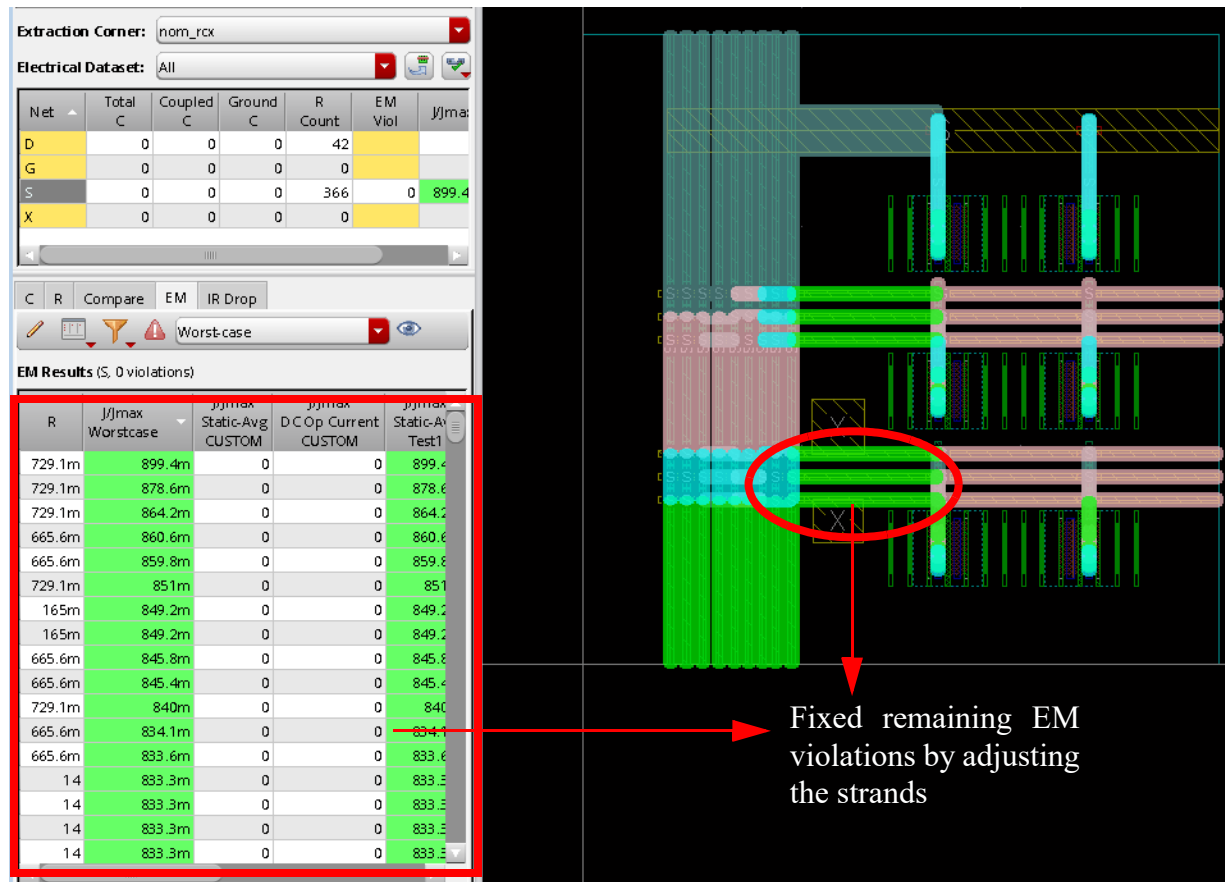
3. To fix the remaining EM violations in the design, click *Options* from the *Optimize Trunk Width for Complete Network* drop-down list.
4. The P2T EM Optimization form is displayed. From this form, select *Stack Twigs* and *Prefer DRC Violations to EM Violations* options. For more information on why should you select stack twigs, see [Stacking Twigs to Fix EM Violations](#).

When there is insufficient space for all the strands to adjust, the number of strands are adjusted depending on the setting of the *Prefer DRC Violations to EM violations* option. By default, the *Prefer DRC violations to EM Violations* option is deselected. In such a case, the number of strands are reduced to keep the layout DRC-error free. When the *Prefer DRC Violations to EM Violations* option is selected, the inserted number of strands are kept as is, leaving shorts and spacing violations with neighboring shapes.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

- Click the *Optimize Trunk Width for Complete Network* button. The following figure shows how the number of strands are adjusted to fix the remaining EM violations.



Expand with Stacked Stranding

When the trunk optimization style is selected as *Stacked Stranding*, the EM violations are fixed by creating additional trunks on the layers above or below the original trunk layer.

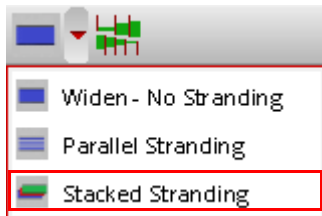
Stacking starts from the first layer above the original layer in the same routing direction and stops when the total width of the stacked strands is equal to or greater than the width of the hint value provided by EAD. Also, the width of each strand is equal to the width of the original trunk or the `minWidth` of the layer of the strand, whichever is larger. The length of each strand is equal to the length of the original trunk. In addition, the strands are connected together by the new via stacks that get added at the locations of existing vias on the original trunks.

To do this:

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

1. Click the *Select Trunk Optimization Style* button in the EAD toolbar and select *Stacked Stranding*.

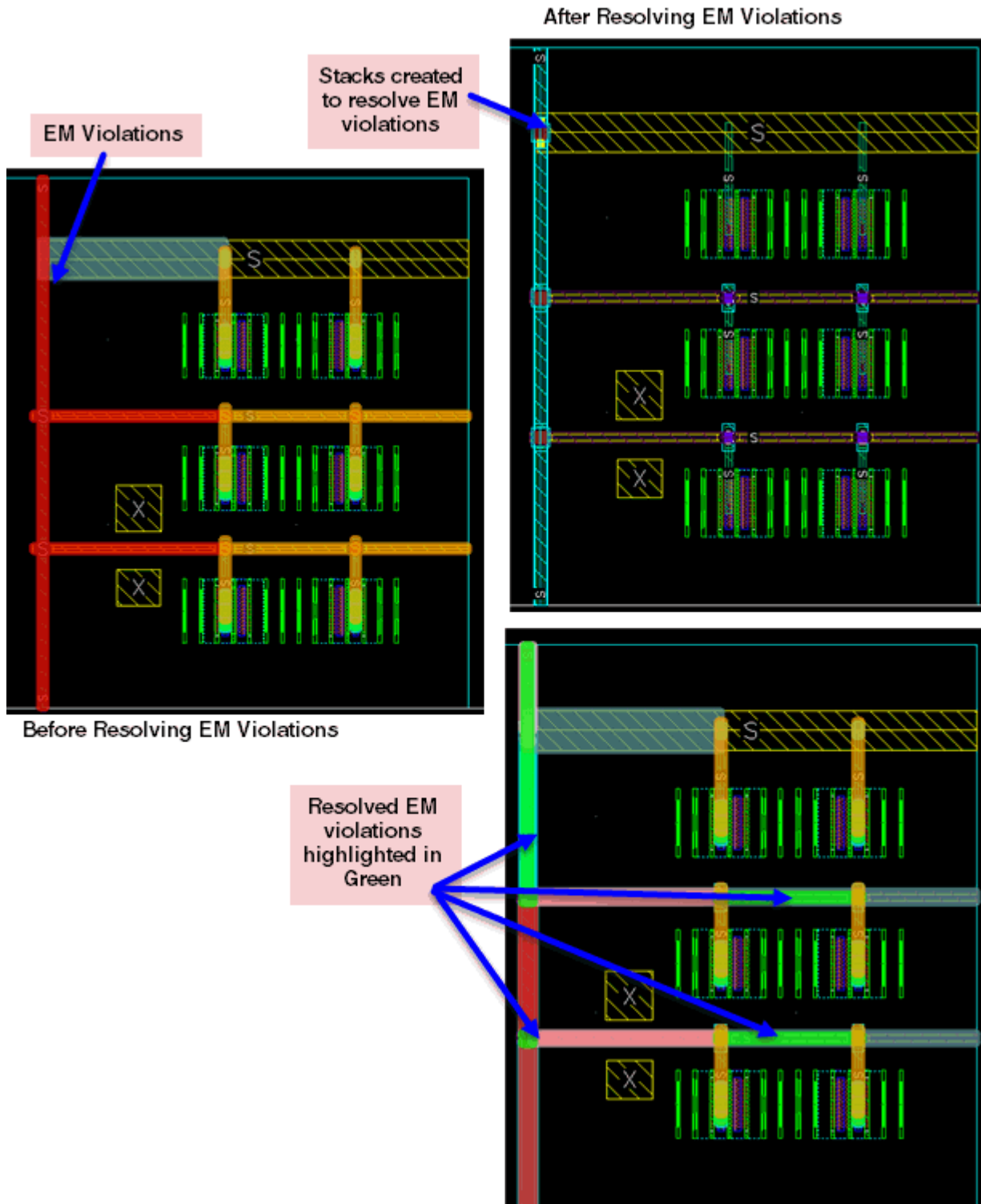


2. Click the *Optimize Trunk Width for Complete Network* () button with the default option of *Expand to Fix EM Violations*. If the top-most layer is attained and total width

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

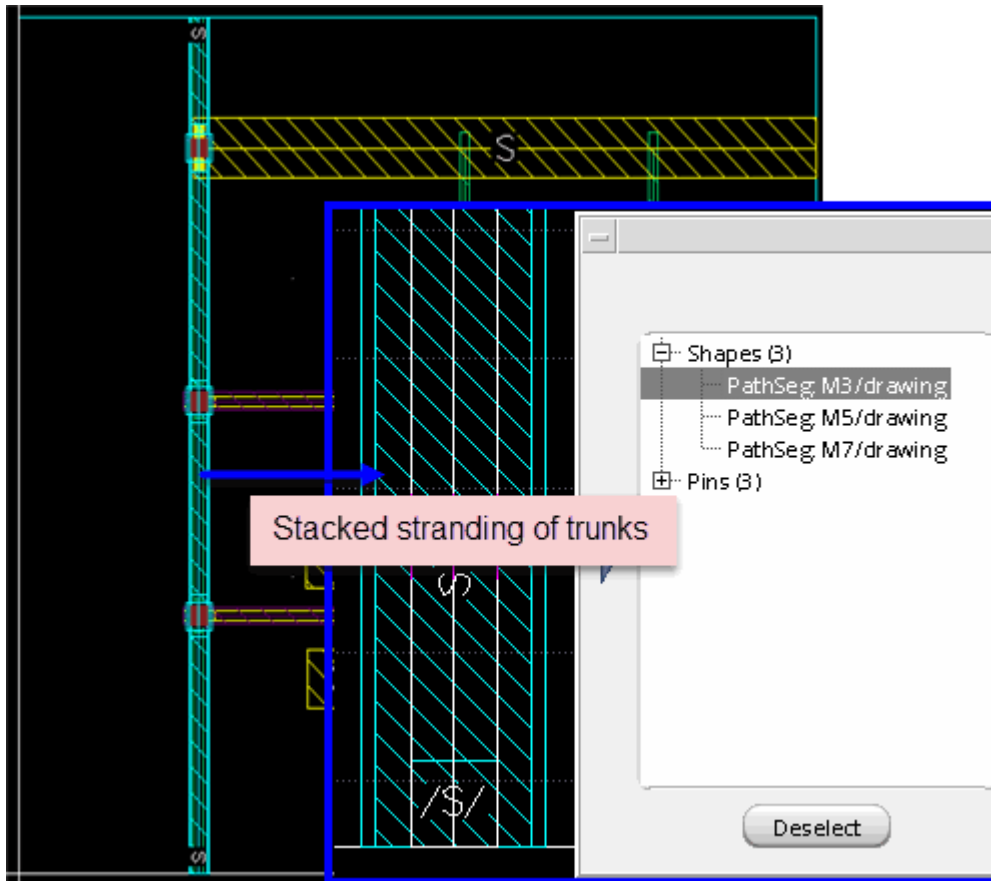
of the strands is still less than the hint width value provided by EAD, additional strands are added to layers below the original layer.



Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

By default, stacking in each layer is skipped if an obstacle is found at the location of the original trunk. If there is a WSP on a layer but there is no track at the location of the original trunk, the new trunk is snapped to the nearest WSP track. However, if the track does not yield a feasible solution, the command skips the layer and proceeds to the next layer.

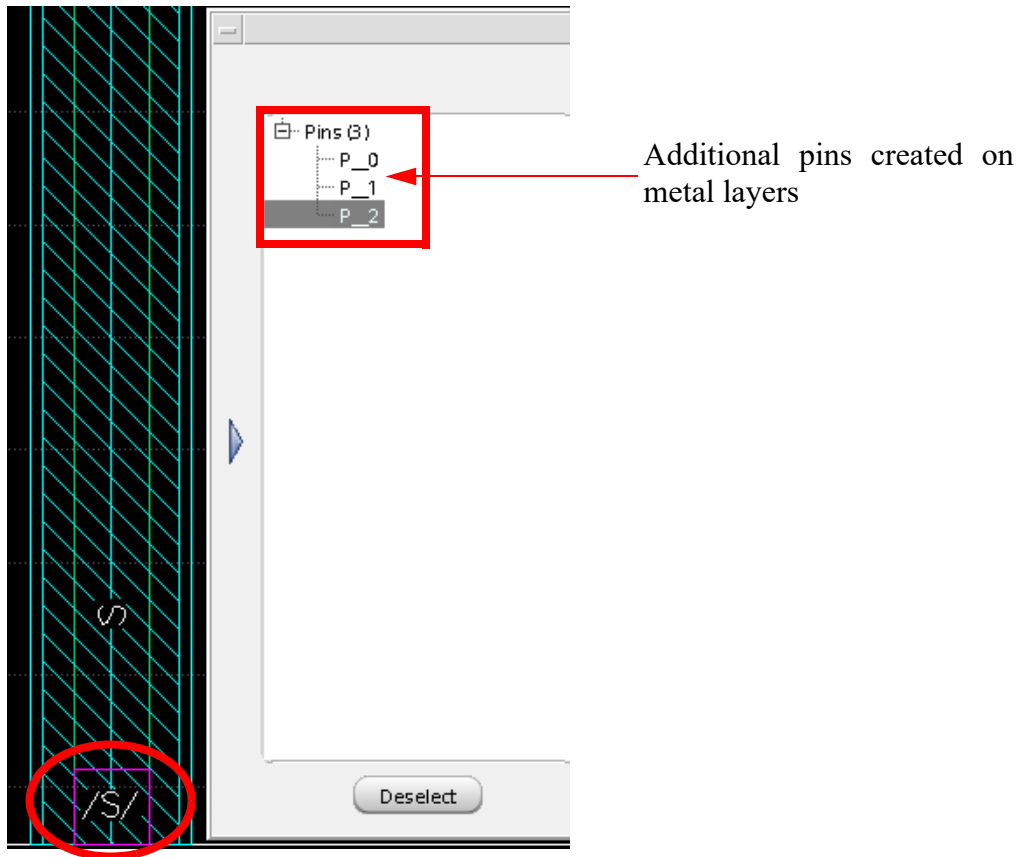


Note: To ensure that the stranded trunks are placed on tracks in a design that consists WSPs, select *Snap Pattern* from the *Snap Wires To* drop-down list in the Wire Assistant, before clicking *Optimize Trunk Width for Complete Network*.

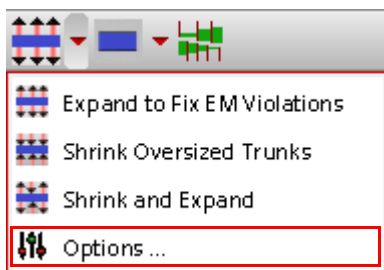
Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

When the original trunk is connected to a top-level pin shape, additional pin shapes are created to which the new strands are connected.



3. Next, click *Options* from the *Optimize Trunk Width for Complete Network* drop-down list.

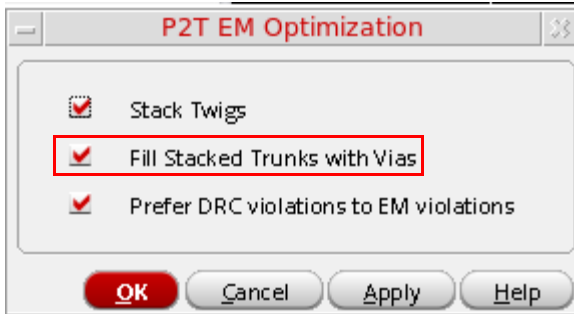


4. The P2T EM Optimization form is displayed. To continue stacking until the total width of the trunk satisfies the width hint value provided by EAD, select *Prefer DRC Violations to EM Violations* option from the form.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

5. While the trunks are stacked, you can fill the stacked trunks with vias for fixing violations. To do this, select the *Fill Stacked Trunks with Vias* option from the P2T EM Optimization form.

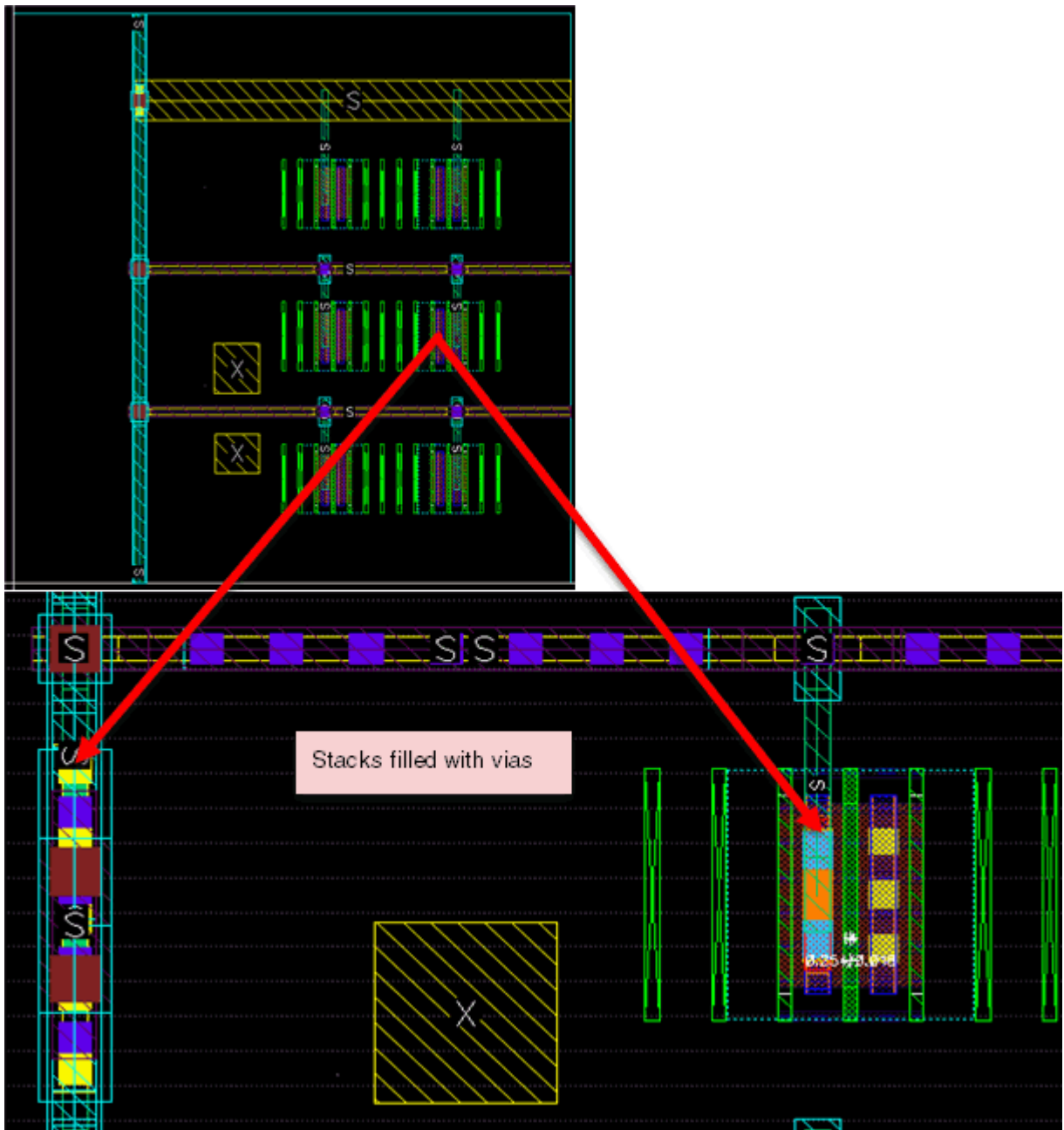


6. Click the *Optimize Trunk Width for Complete Network* button. In the following figure, you can see that vertical stacked strands are created on Metal15 and Metal17 because

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

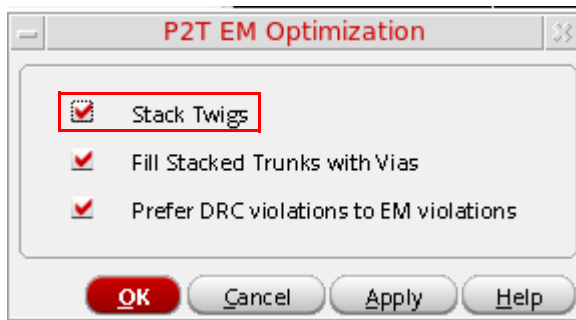
these layers have a vertical routing direction and a horizontal stacked strand is created on Metal6 layer because it has a horizontal layer direction.



Stacking Twigs to Fix EM Violations

This section describes how to fix the EM violations that exist due to the twigs. It also describes how a trunk is widened and how the twigs are stacked to fix the EM violations. To do this:

1. Select a net in the summary table to highlight the violations on that net in the layout canvas.
2. From the drop-down list, click *Options*. The P2T EM Optimization form displays.
3. Select the *Stack Twigs* option from the form.

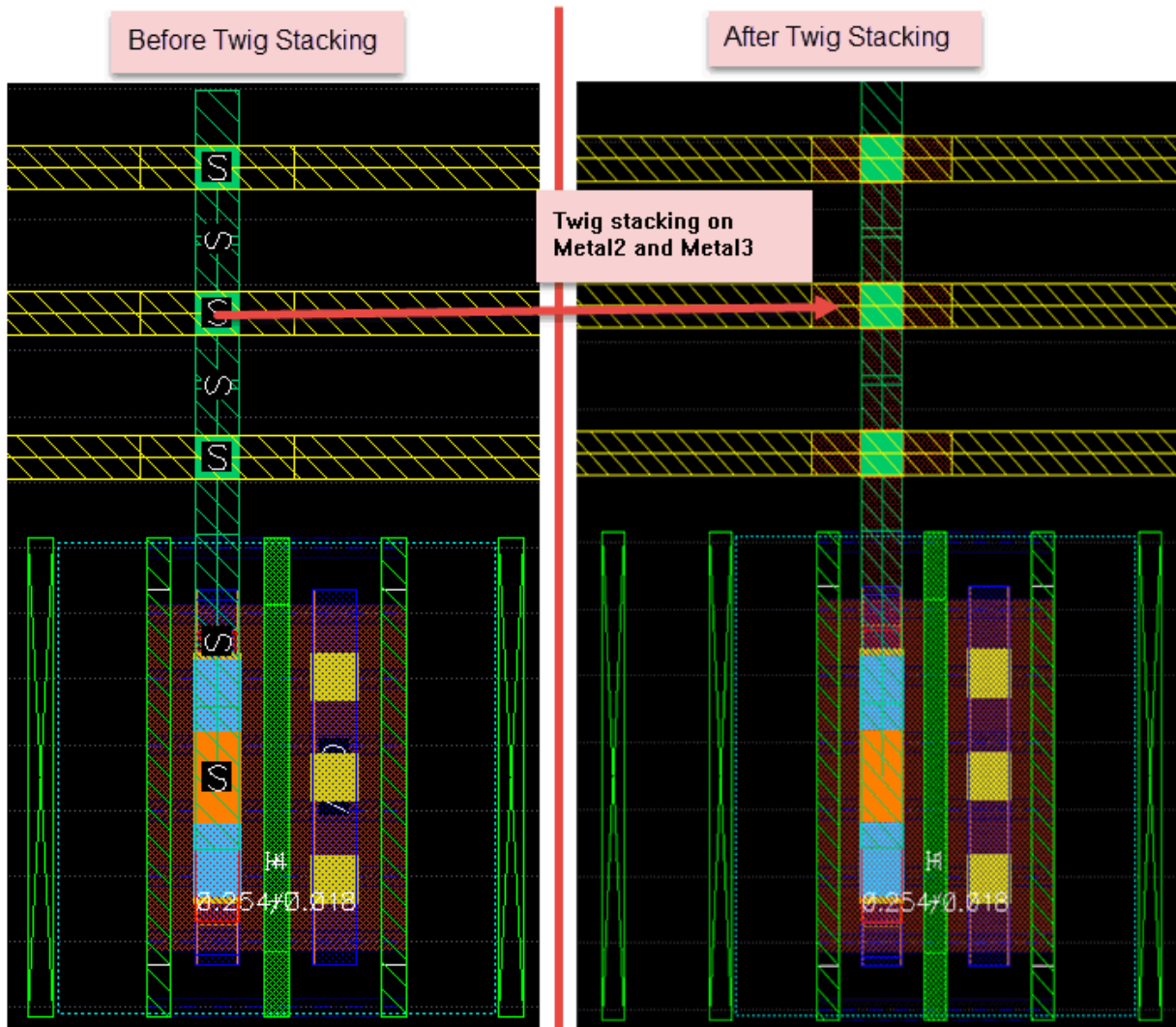


When the *Stack Twigs* option is selected, stacked stranding is performed on each twig segment with an EM violation. The stacking of twigs is similar to that of the trunk stacked stranding except that the stacking of twig segments starts from the first layer below the layer of the original twig segment. The layer for stack follows the routing direction of the original twig segment. In addition, vias are added to stitch the stacked segments together.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

4. In the EAD toolbar, click the *Optimize Trunk Width for Complete Network* () button. The following figure displays how all the violations on the twigs and trunks were resolved.



Tapering of Trunks to Fix EM Violations

This section describes how to taper down the width of the trunks when expanding trunks to fix EM violations.

To do this:

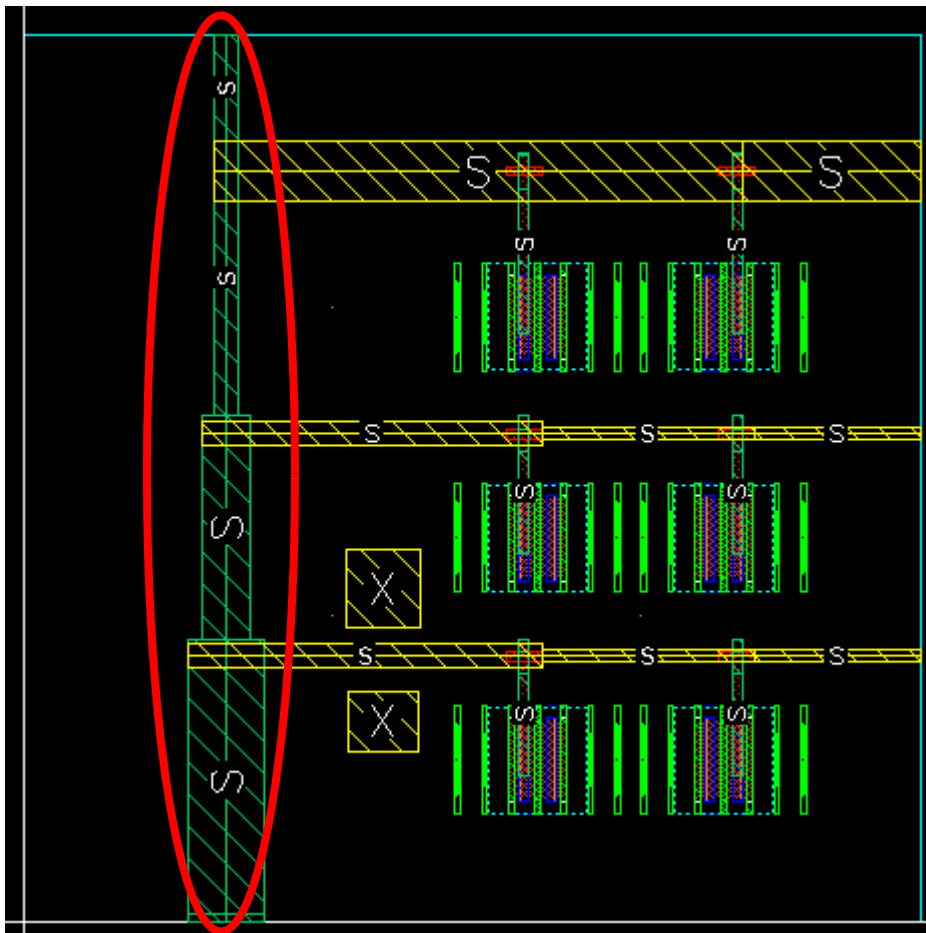
1. Select a net in the summary table to highlight the violations on that net in the layout canvas.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

2. In the EAD toolbar, click the *Toggle Trunk Tapering for Complete Network* button. When trunk Tapering is enabled, the original trunk is split into multiple trunks. The trunk splitting occurs at each twig or trunk intersection.
3. Click the *Optimize Trunk Width for Complete Network* button.

The width of the individual trunks are widened separately according to EAD hints and vias between the splitted trunks are also accordingly adjusted. The tapered trunks are aligned along their centerline. However, the centerline alignment may not be possible if the tapered trunks are pushed as a result of a neighboring trunk being expanded. In such a case, each trunk in a pair of adjacent trunks is allowed to slide up and down to provide space to the neighbor trunk above or below the tapered trunk. The common edge between the adjacent tapered trunks remains equal to the size of the narrower trunks. The following figure shows an example of how the trunks are tapered during trunk optimization.

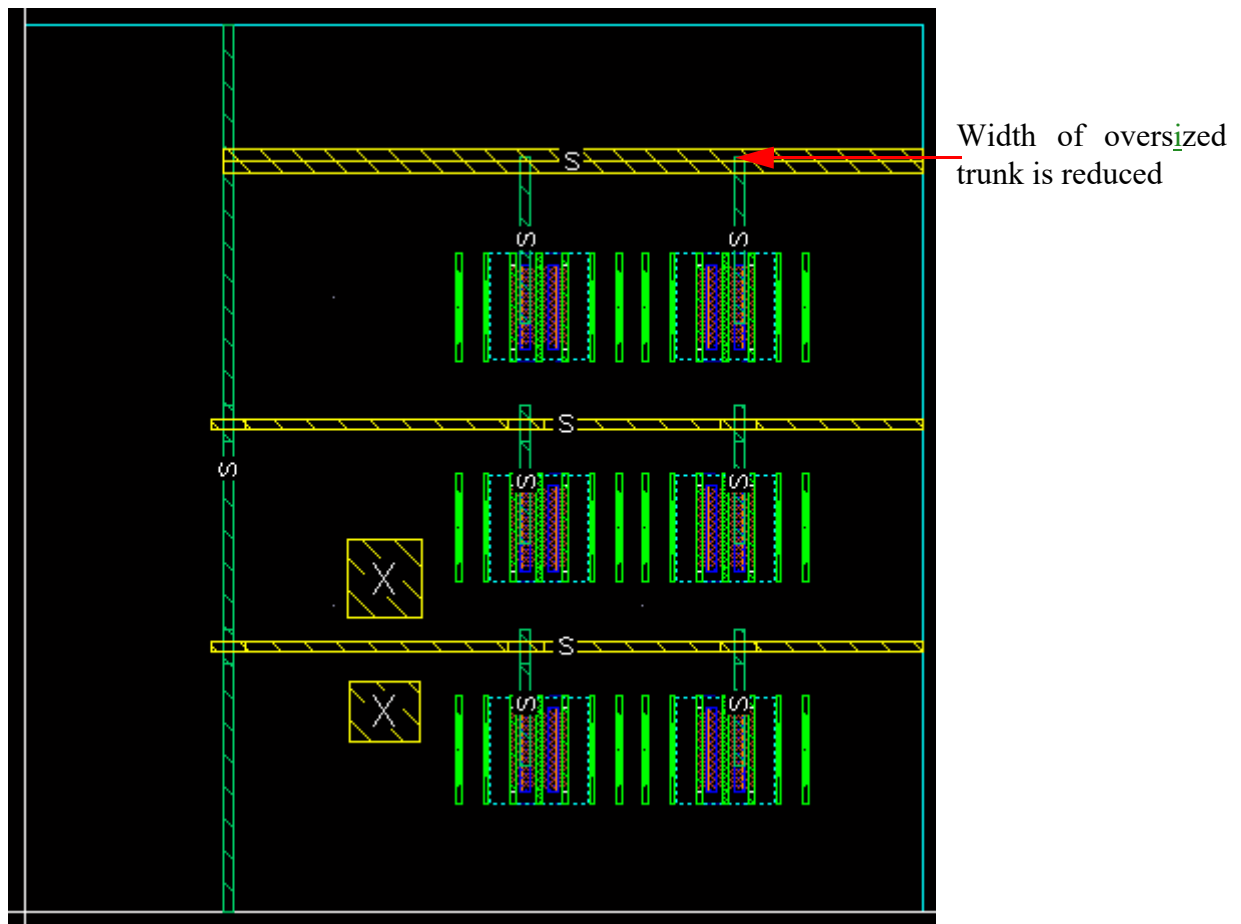


Shrinking Oversized Trunks

This section covers how the width of the oversized trunks and the number of cuts in oversized via arrays in the design are reduced according to the hint values provided by EAD.

1. Select a net in the summary table to highlight the violations on that net in the layout canvas.
2. In the EAD toolbar, click the *Optimize Trunk Width for Complete Network* button.
3. From the drop-down list, select the *Shrink Oversized Trunks* command. When this command is run, the width of the oversized trunk is reduced and the cuts of the vias on the trunk are also reduced. It does not reduce the number of strands. Also, no compaction is done after width of trunks is reduced and the width of twigs is not reduced.

Here, you can see an example where the wide top trunk that was oversized for the defined EM rules is reduced to the ideal width to meet the EM rules.



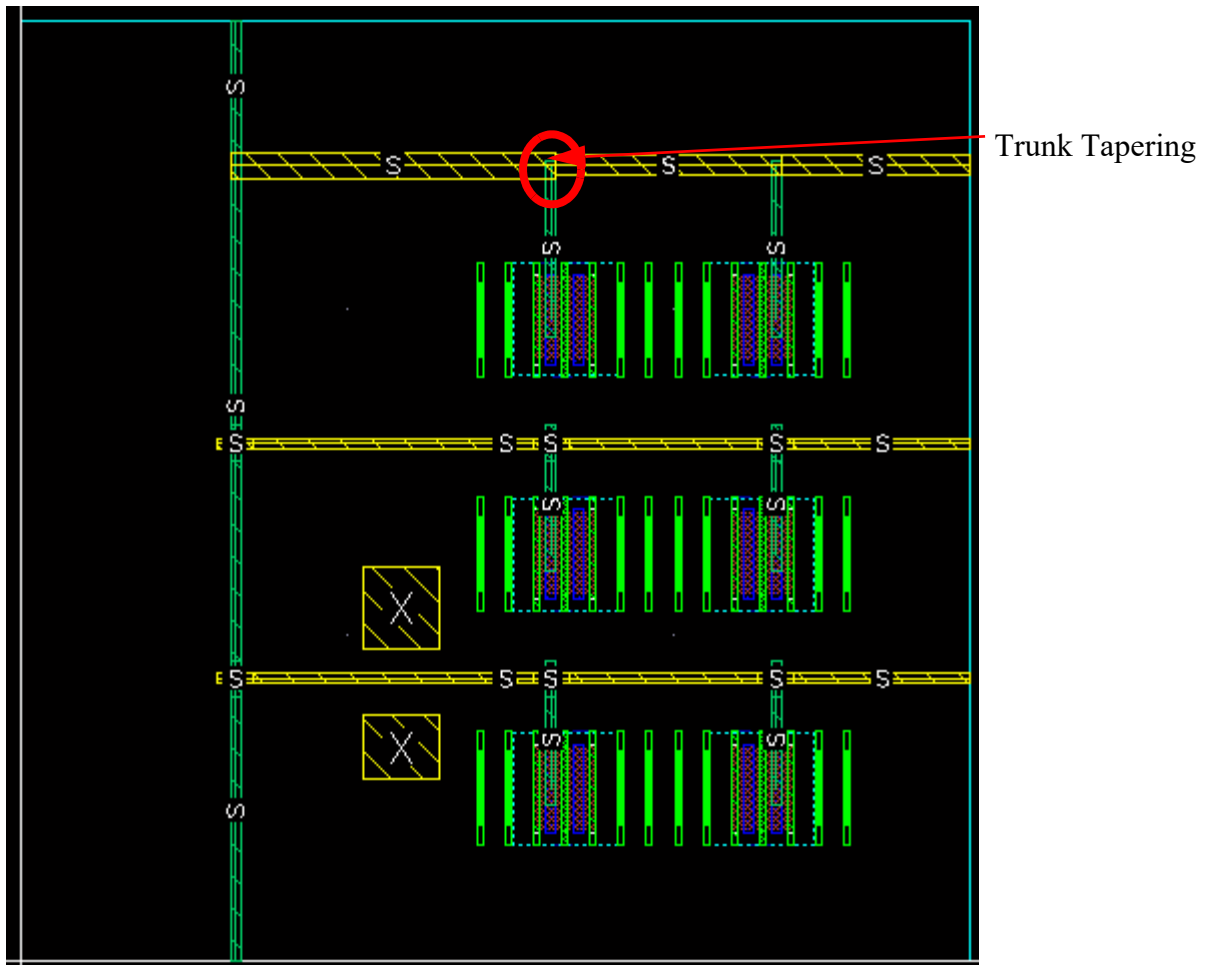
Note: The *Shrink Oversized Trunks* command only works when the optimization style

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

is set to *Widen - No Stranding*. The command does not work when the optimization style is set to *Parallel Stranding* or *Stacked Stranding*.

4. Now, to taper down the width of the optimized trunk, click the *Trunk Tapering for Complete Network* button.
5. Again, click the *Optimize Trunk Width for Complete Network* button in the EAD toolbar. The following figure displays tapering of the optimized trunk.



Shrinking and Expanding Trunks

This section covers how the width of the oversized trunks can be reduced and at the same time the width of the undersized trunks can be expanded using a single command. To do this:

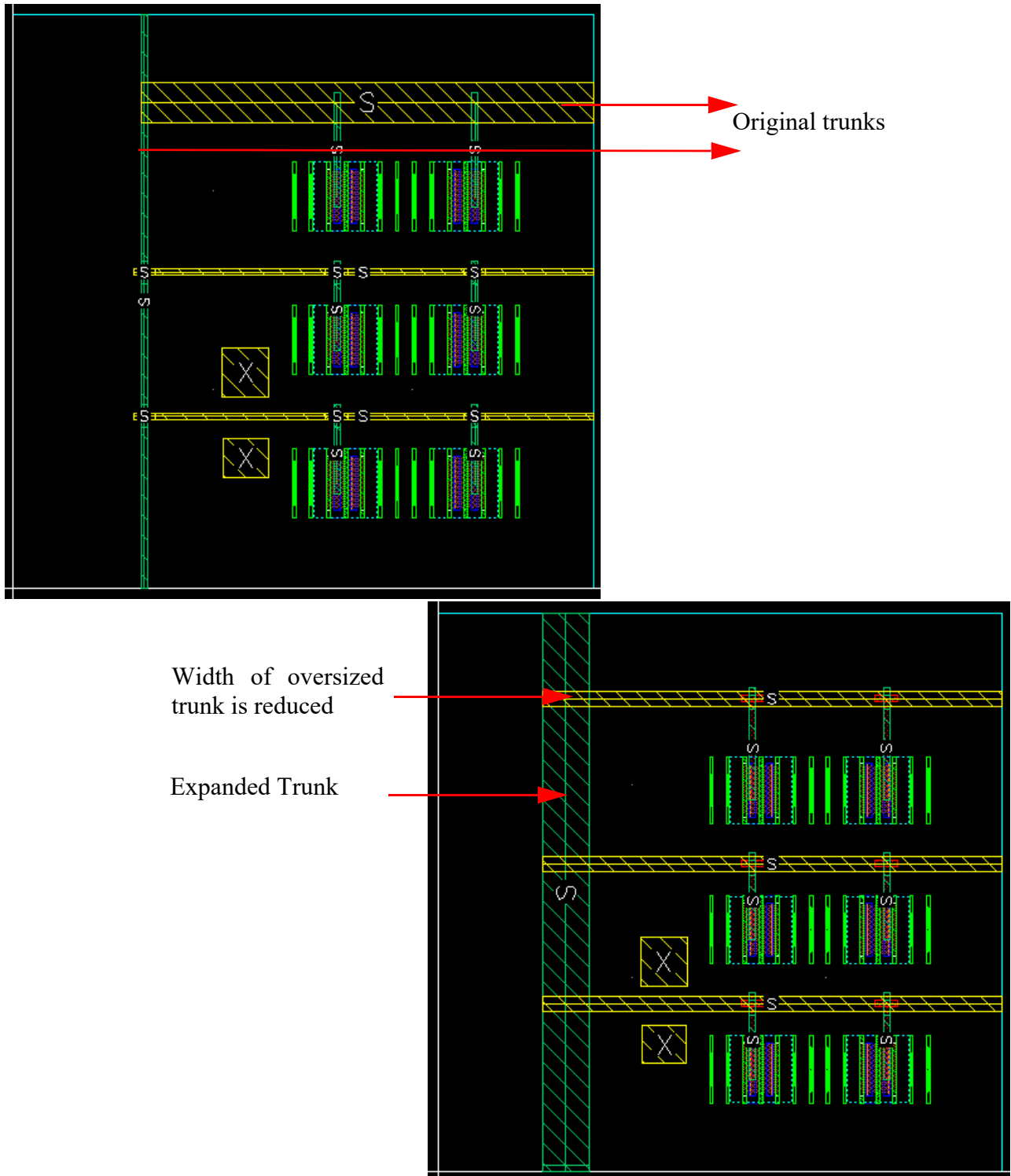
1. Select a net in the summary table to highlight the violations on that net in the layout canvas.
2. In the EAD toolbar, click the *Optimize Trunk Width for Complete Network* button.
3. From the drop-down list, select *Shrink and Expand* command. This is an ideal command to run if you want to both reduce the width of the oversized trunks and widen the trunks that violate the EM rules. When you select this command, two actions take place :
 - ☐ The width of the oversized trunks and vias is reduced by running the *Shrink Oversized Trunks* command.
 - ☐ The width of the trunks and vias is expanded by running the *Expand To Fix EM Violations* command.

For more information on these commands, see [Expanding Trunks to Fix EM Violations](#) and [Shrinking Oversized Trunks](#).

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

In the following figure, you can see that the width of the oversized trunk is reduced to the ideal width. At the same time, the other trunks were widened.





Shrinking only works when the Optimization Style is set to Widen – No Stranding and does not work when the Optimization Style is Parallel Stranding or Stacked Stranding. The Shrinking and Expanding Trunk command only shrinks the oversized trunks and does not reduce the number of strands.

Related Topics

[p2t](#)

[P2T EM Optimization](#)

[EAD Toolbar](#)

Running EM Analysis on Power Grids

EAD enables you to check the power grids by running EM analysis for the specified source current even when there are no devices connected to the grid.

EAD does so by inserting dummy instances as the current sink points at the via that is connected to the lowest metal layers. It then creates a dataset using the specified source current, which is distributed among the sink points in proportion to the number of via cuts at these sink points. This dataset is used to run EM analysis.

The results of EM analysis show the reported violations and the recommendations to fix these violations. You can then make the suggested modifications in the original grid so that it becomes EM-compliant.

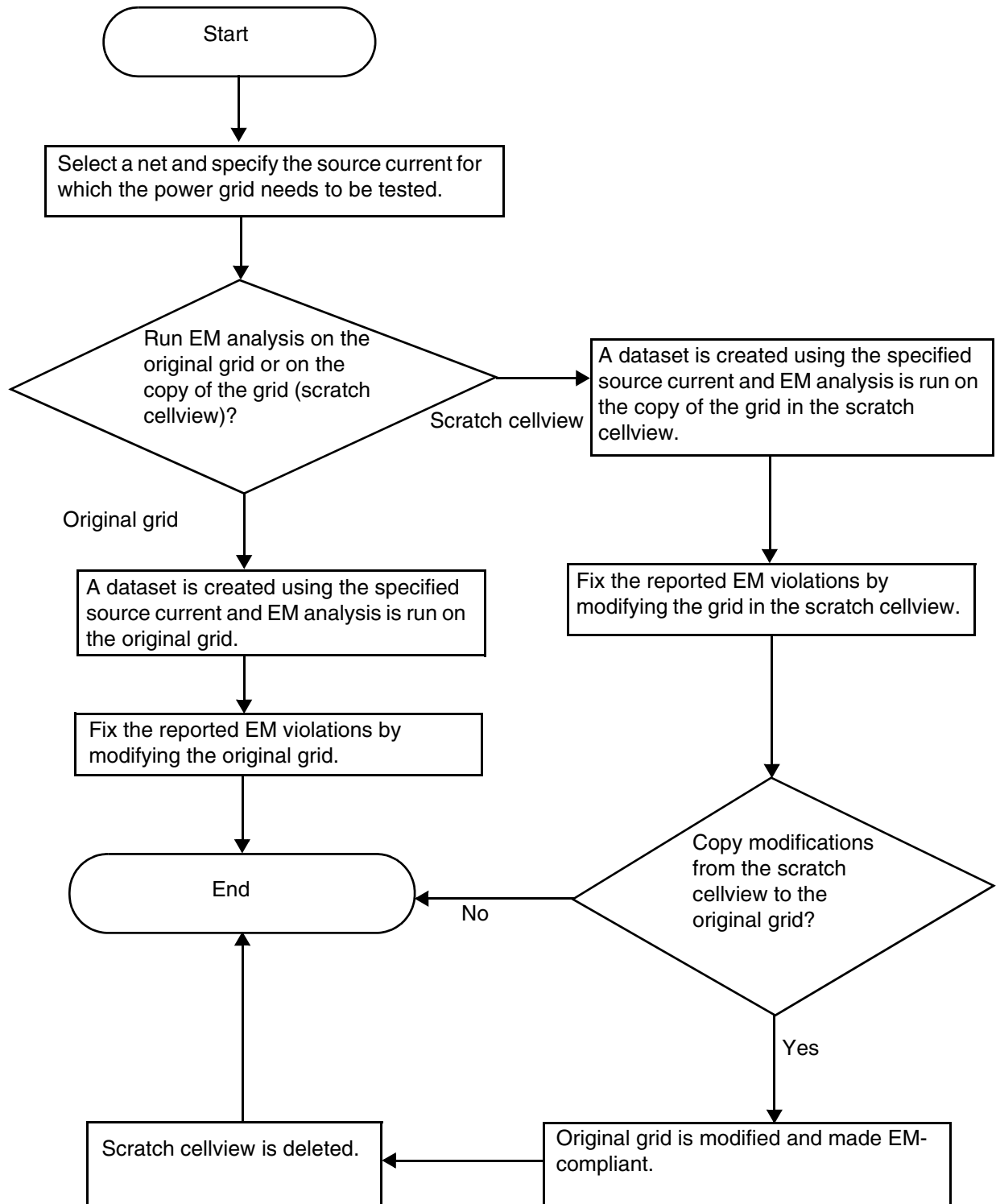
However, if you do not want to modify the original grid without first analyzing the effects of these modifications, you can choose to run the analysis on a scratch cellview, which is a copy of the original grid. You can then modify the grid in the scratch cellview to resolve the EM violations. After validating the fixes, you can copy these modifications from the scratch cellview to the original grid.

Note: Ensure that the EAD Browser is initialized before you run power grid analysis. See the [Starting EAD from a Layout Window](#) section for more information.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

The following workflow shows how to run the EM checks on power grids:



Prerequisites

Before running EM analysis, ensure that:

- The layout view of your design is open. See *Virtuoso Electrically Aware Design Flow Guide* for more information.
- The EAD Browser is initialized and the process setup is loaded. See *Virtuoso Electrically Aware Design Flow Guide* for more information.
- The Source current is specified. See the [Specifying the Source Current](#) section for more information.

Specifying the Source Current

To run EM analysis on a power grid, you can specify the source current by using one of the following methods:

- [Using EAD Menu](#)
- [Using the defaultCurrent Environment Variable](#)
- [Using the CUSTOM Dataset](#)

You can then run EM analysis either on the scratch cellview or on the original grid.

See the following sections for more information:

- [Running EM Analysis on the Scratch Cellview](#)
- [Running EM Analysis on the Original Grid](#)

Using EAD Menu

To specify the source current for a net using EAD menu, do the following:

1. Right-click the net in the layout canvas, and then choose *EAD – Specify source current for net*.
2. In the *Current (Ampere)* field on the form that appears next, specify the value of the source current.
3. Click *OK*.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

Using the defaultCurrent Environment Variable

Specify the source current by setting the defaultCurrent environment variable as follows:

```
envSetVal("layoutEAD.em" "defaultCurrent" 'float 0.001)
```

Note: The default value of this environment value is 1mA.

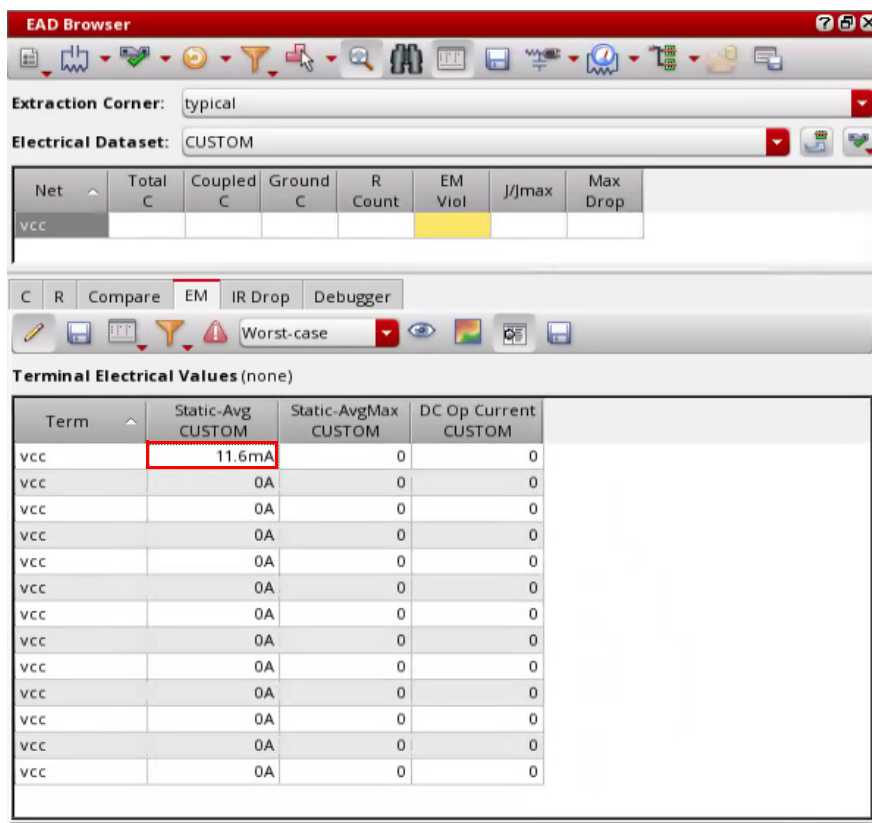
Using the CUSTOM Dataset

To specify the source current using the CUSTOM dataset, do the following:

1. In the EAD Browser, from the *Electrical Dataset* drop-down list, select CUSTOM.
2. In the summary section, select a net, and then click the *EM* tab.
3. Click *Edit Currents* on the *EM* tab toolbar.

The EM table lists all the terminals of the selected net.

4. In the *Static-Avg* column, specify the value of current for which the power net needs to be analyzed.



Running EM Analysis on the Scratch Cellview

1. Right-click a net on the layout canvas, and then choose *EAD – Run grid analysis on scratch cellview*.

EAD creates a scratch cellview, in which dummy instances are inserted as sink points at the lowest vias present in the layout. It then creates a dataset using the specified source current, which is distributed among the sink points in proportion to the number of via cuts at these sink points. This dataset is used to run EM analysis on the scratch cellview.

2. Double-click the *EM Viol* cell to view the results of EM analysis.

The reported EM violations are displayed in the EM table. Hover over any of the violations to view the recommended design changes in your grid so that it becomes EM-compliant.

After fixing the EM violations by modifying the grid in the scratch cellview, you can update the grid in the original layout with these modifications to make it EM-compliant for the specified source current. To upgrade the grid in the original layout, see the [Modifying the Original Layout](#) section.

Note: To discard the results, right-click the net in the layout view, and then choose *EAD – Discard edits from grid analysis* from the context menu.

Modifying the Original Layout

After resolving the EM violations by modifying the grid in the scratch cellview, you can replicate these modifications in the original grid.

To modify the grid in the original layout, do the following:

1. In the scratch cellview, right-click a net and then select *EAD – Upgrade grid in original layout*.

A message box appears informing you that the scratch layout will be deleted if you choose to continue.

2. Click *Yes* to confirm.

The changes are updated in the original grid and the scratch layout is deleted.

Important Points to Note

- By default, the scratch layout is created in the same library that contains the original layout. However, you can set the scratchLib environment variable to define a different scratch library.
- After running the EM Analysis on the scratch layout and making the modifications as needed in the original layout, you can delete data from the scratch library. Use the eadDeleteScratchLib SKILL function to delete all data from the scratch library.
- You can also set the default value of the current passing through the power grid (VDD) that you want to be considered during EM analysis. Set the defaultCurrent environment variable to define the default current.

Running EM Analysis on the Original Grid

To run EM analysis on the original grid, do the following:

1. Right-click a net on the layout canvas, and then choose *EAD – Run Supply Grid Analysis (modifies layout)*.

A dialog box appears informing you that the layout will be modified if you choose to continue.

2. Click *Yes* to confirm.

EAD creates a dataset using the specified source current, which is distributed among the sink points in proportion to the number of via cuts at these sink points. This dataset is used to run EM analysis on original grid.

3. Click *EM Viol* cell to view the results of EM analysis.

The reported EM violations are displayed in the EM table. Hover over any of the violations to view the recommended design changes in your grid so that it becomes EM-compliant.

4. Fix the EM violations by modifying the grid in the original layout.

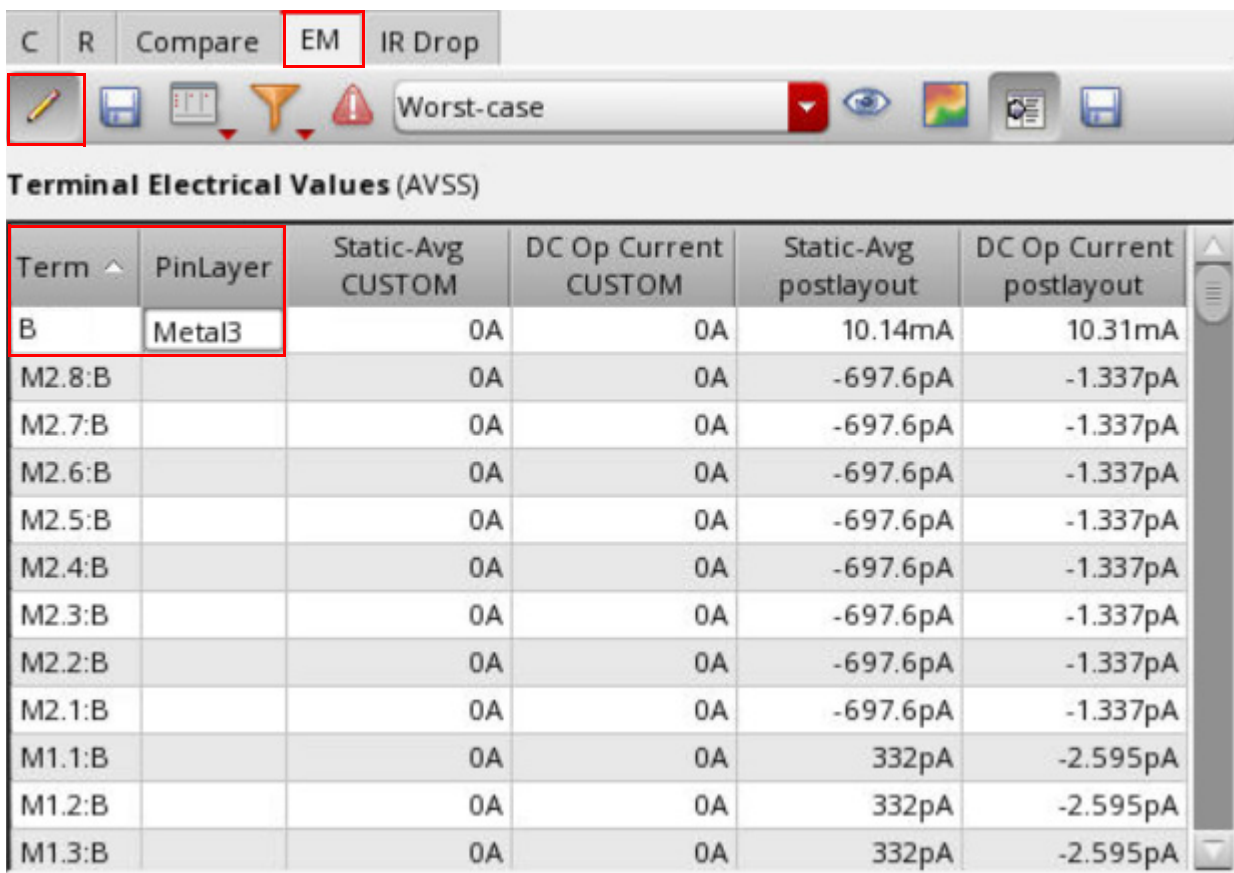
Specifying Mapping between Nets and Pin Layers for EM Check

You can specify the mapping between nets and pin layers for EM check. The nets or terminals serve as the voltage source only for the pin layers defined in the mapping.

To define the mapping between nest and pin layers for EM check:

1. Set the environment variable `layoutEAD.gui termPinLayersForEMCheck` to `t`.

```
envSetVal("layoutEAD.gui" "termPinLayersForEMCheck" 'boolean t)
```
2. In the EAD Browser, select a net.
3. In the *EM* tab of the EAD browser, click *Edit currents*.
4. Specify the name of the metal layer in the *PinLayer* column next to the net name, as shown in the following figure.



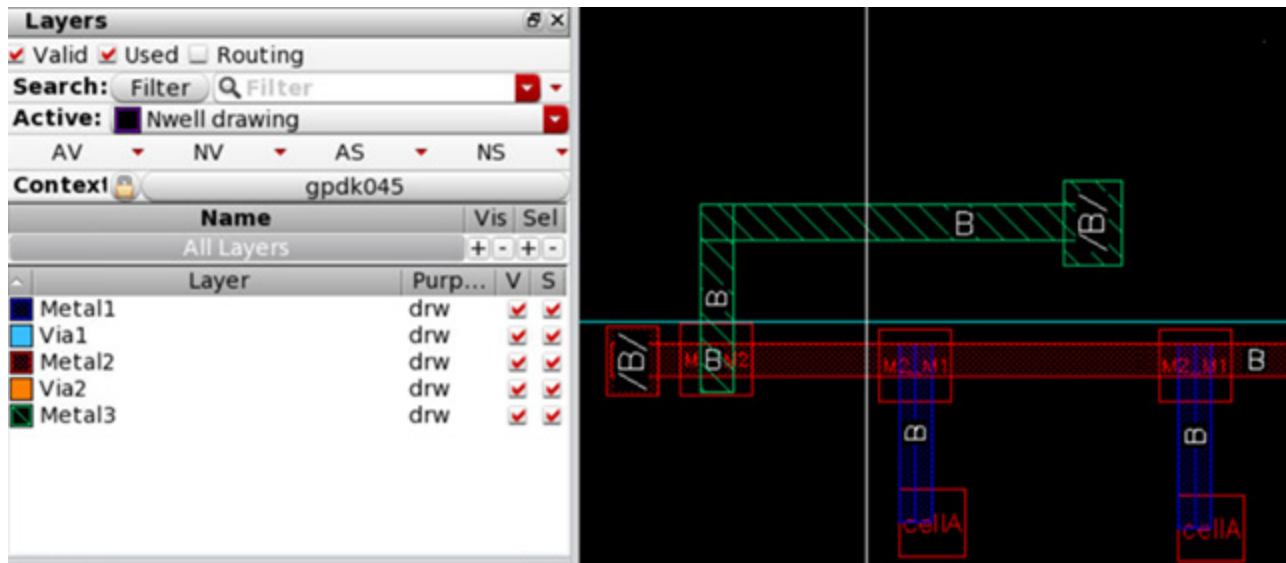
Term ^	PinLayer	Static-Avg CUSTOM	DC Op Current CUSTOM	Static-Avg postlayout	DC Op Current postlayout
B	Metal3	0A	0A	10.14mA	10.31mA
M2.8:B		0A	0A	-697.6pA	-1.337pA
M2.7:B		0A	0A	-697.6pA	-1.337pA
M2.6:B		0A	0A	-697.6pA	-1.337pA
M2.5:B		0A	0A	-697.6pA	-1.337pA
M2.4:B		0A	0A	-697.6pA	-1.337pA
M2.3:B		0A	0A	-697.6pA	-1.337pA
M2.2:B		0A	0A	-697.6pA	-1.337pA
M2.1:B		0A	0A	-697.6pA	-1.337pA
M1.1:B		0A	0A	332pA	-2.595pA
M1.2:B		0A	0A	332pA	-2.595pA
M1.3:B		0A	0A	332pA	-2.595pA

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

Example

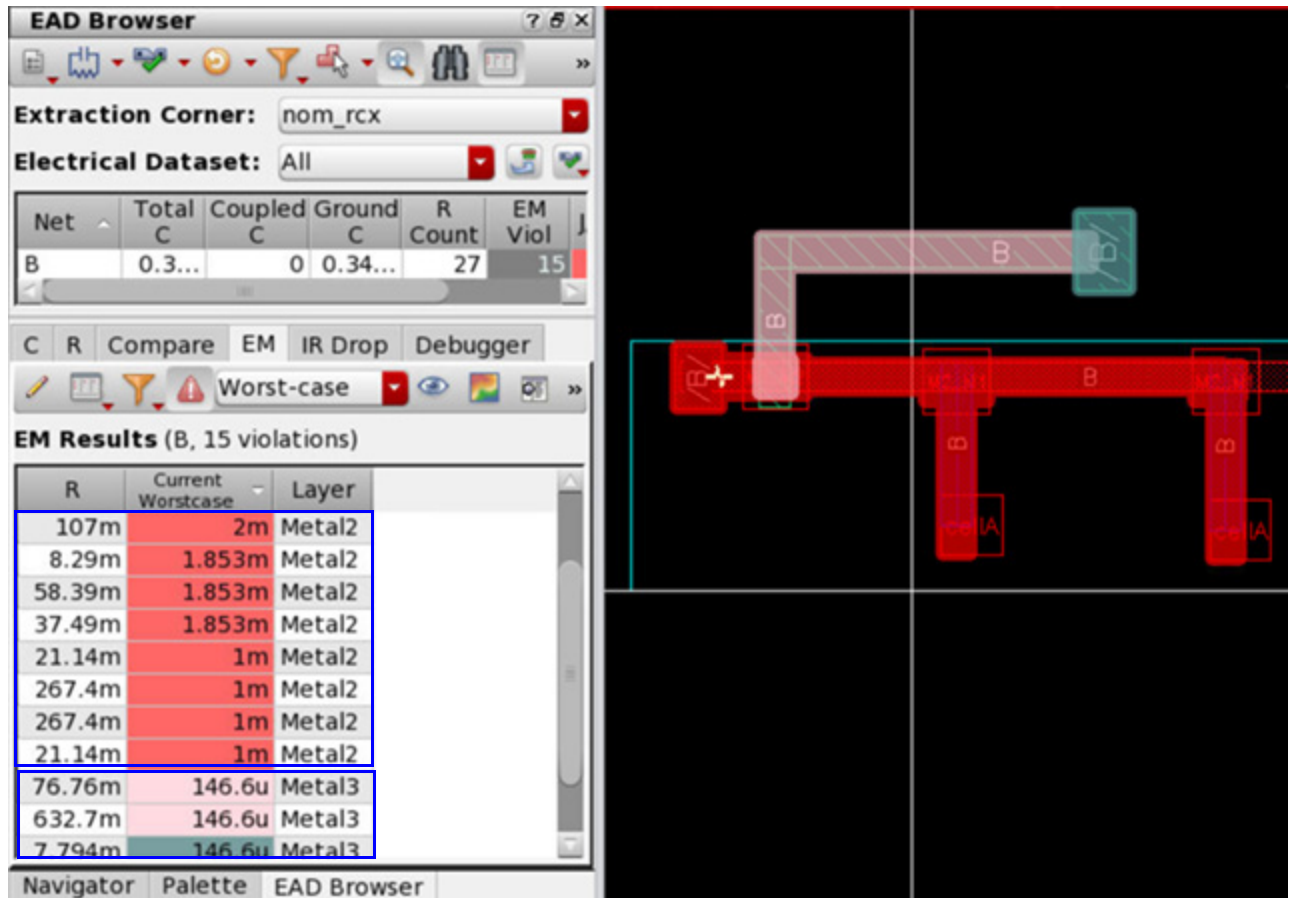
Consider the following design where term `B` has two pins; one is connected to the metal layer `Metal2` and another one is connected to the metal layer `Metal3`.



Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

If you observe the current distribution without considering the pin on `Metal3` as the voltage source for term `B`, you find that the current on the pin connected to `Metal3` is negligible as compared to `Metal2`.



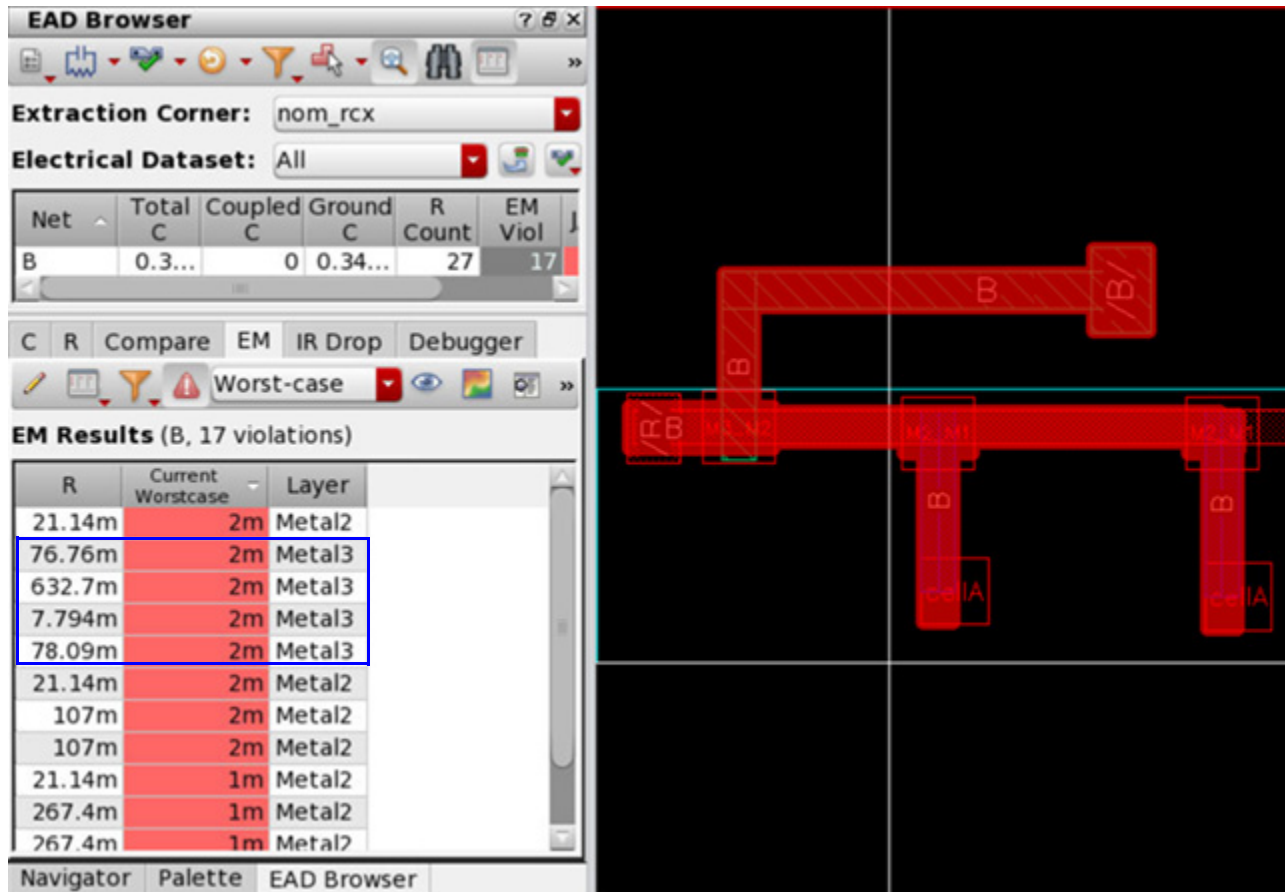
To consider the pin on the `Metal3` as the voltage source for term `B` for the EM check, specify the environment variable `termPinLayersForEMCheck` to `t` and then specify `Metal3` in the `PinLayer` column next to the net `B` in the *EM* tab of the EAD Browser.

```
envSetVal("layoutEAD.em" "termPinLayersForEMCheck" 'boolean t)
```

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

Now, if you observe the current distribution, you find that the current on the pins connected to Metal3 is now significant.



Generating EAD Dataset



Before generating EAD dataset, ensure that EAD setup is initialized.

To generate EAD Dataset without running a simulation or without a CSV file:

1. Right-click a net either in the layout canvas or in the Navigator assistant, and choose *EAD – Generate Dataset*.

Alternatively, right-click a net in the EAD Browser, and select *Generate Dataset*.

The Generate EAD Dataset form opens.

2. By default, the *Net* drop-down list displays the name of the net that you selected in the layout canvas, in the Navigator assistant, or in the *Net Summary* table of the EAD Browser. However, you can also choose another net from this drop-down list.
3. In the *Dataset Name* field, enter the name of the dataset that you want to create. If there are any existing datasets, you can select them from the *Dataset Name* drop-down list and edit them.
4. In the *Current* field, specify the value of current. The default value is 0.001.

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

5. From the *Current* type options, select one of the following:

- a. *Max Current*: Lets you specify the maximum current that can be passed through the net.
- b. *Pin Current*: Lets you specify the maximum current that can be passed through the pin.

6. From the *Distribution* drop-down list, select one of the following current distribution methods:

- ☐ *Aspect ratio*: Distributes the specified current among the devices in proportion to their aspect ratio, that is `width/length`.

For example, if the devices M10 and M5 have aspect ratios 150 and 75, respectively. The M10 gets twice as much current as M5 does.

Width and length information for a device is read from its CDF parameters. If EAD cannot find width and length properties from device CDF parameters, it checks whether the device is inside a macro Pcell, if yes, then it looks for a property on macro Pcell. The property must have values defined as `"...w=0.2 l=0.4..."`.

For more information, see `autoDatasetCustomProc`, `autoDatasetDeviceLengthPropList`, `autoDatasetMacroProp`, `autoDatasetDeviceWidthPropList`.

- ☐ *Equal*: Distributes the specified current equally among different device terminals connected to the net.
- ☐ *Custom*: Distributes the specified current among different device terminals according to the custom current distribution defined in the user-defined function specified by the environment variable `autoDatasetCustomProc`.

7. Select the *Detailed log* check box if you want to print a detailed log in the CIW.

8. Click *Apply* and then click *OK*.

The dataset with the name you specified is created and is available in the *Electrical Dataset* drop-down list in the EAD Browser.

9. Select this dataset from the *Electrical Dataset* drop-down list in the EAD Browser.

10. Select a net in the *Summary* pane of the EAD Browser.

11. On the *EM* tab, click the Edit currents icon to view the how the specified current is distributed across the different terminals of a net by the dataset.

Note the following points:

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

- Passive devices connected to the net are not assigned any current.
- Current is distributed among terminals of those devices that are registered with `ciRegisterDevice` either as `pfet` or as `nfet`.
- Current is distributed among terminals of those devices that are defined as `MOS` devices in the CPH component class.
- For *Max Current* distribution, drain terminal of `pmos` devices are considered to be source and `drain` terminal of `nmos` devices are considered to be sink points for current distribution.
- For *Pin Current* distribution, all device terminals are considered as sink points.
- Note that no current is distributed to bulk and gate terminals.

Related Topics

[autoDatasetCustomProc](#)

[autoDatasetDeviceLengthPropList](#)

[autoDatasetMacroProp](#)

[autoDatasetDeviceWidthPropList](#)

Virtuoso Electrically Aware Design Flow Guide

Performing Electromigration Checks and Fixing Violations

Introduction to EM Calculator

The EM Calculator is a tool that provides you recommendations on how you can make your design EM compliant based on certain inputs even before creating the actual circuit layout.

You can use the EM calculator to calculate the EM limits and to get the following recommendations for your design to be EM compliant:

- Width of the interconnect
- Length of the conductor layer
- Number of via cuts
- Number of tracks
- Maximum current

Related Topics

[Opening the EM Calculator](#)

[EM Calculator Graphical User Interface \(GUI\)](#)

[Working with the EM Calculator](#)

Opening the EM Calculator

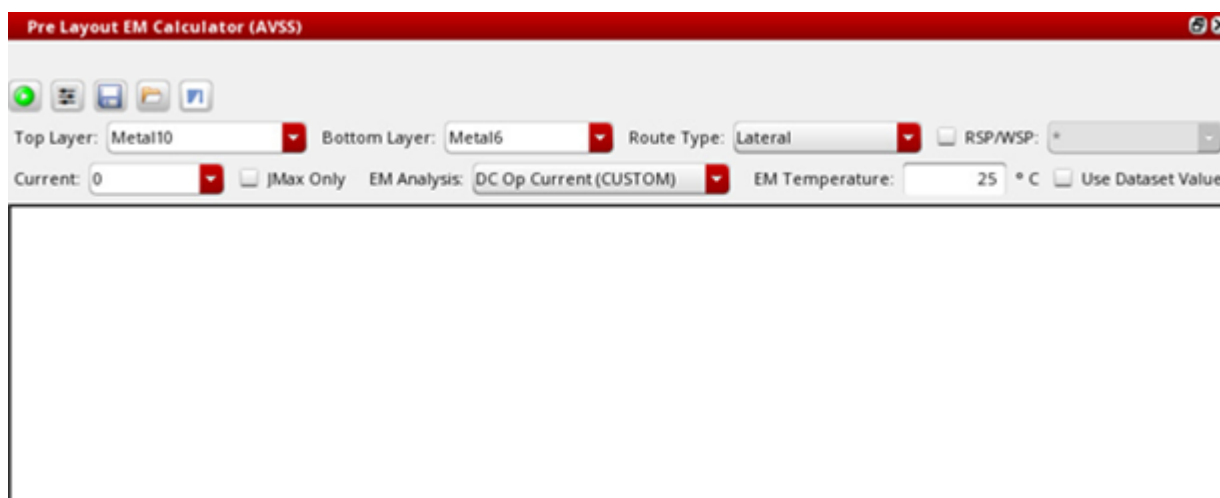
To open the EM Calculator:

Important

You need to set the `layoutEAD.gui enableEMCalculator` environment variable to `t` before using the EM Calculator.

1. In the summary pane of the EAD Browser, select a net for which you want to calculate the EM limits.
2. Right-click the selected net and then choose *EM Calculator* from the context menu.

The Pre Layout EM Calculator (*net name*) opens.



By default, the EM Calculator opens as a floating window. However, you can choose to open it as a docked widget inside the EAD Browser by setting the environment variable `showEMPlannerWindow` to `nil`.

Related Topics

[EM Calculator Graphical User Interface \(GUI\)](#)

[Working with the EM Calculator](#)

[enableEMCalculator](#)

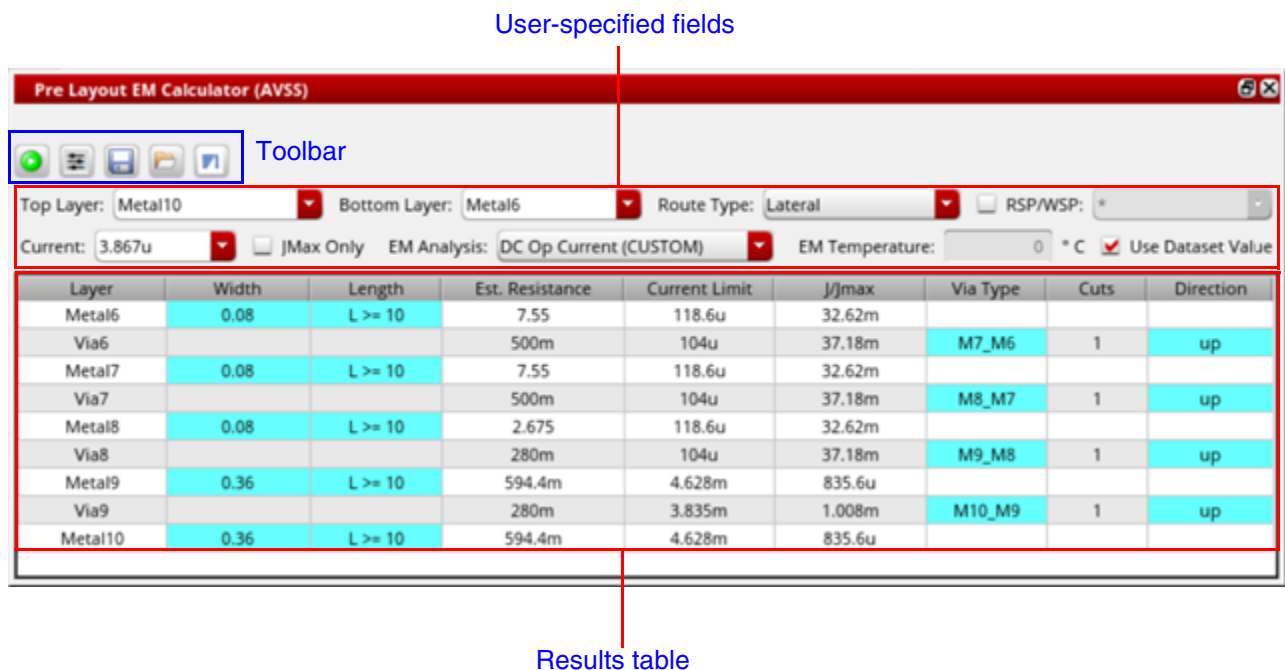
[showEMPlannerWindow](#)

EM Calculator Graphical User Interface (GUI)

The EM Calculator graphical user interface (GUI) consists of the following elements:

- EM Calculator Toolbar
- User-Specified Fields
- Results Table

The following figure shows the graphical user interface of EM Calculator.



Virtuoso Electrically Aware Design Flow Guide

Introduction to EM Calculator

EM Calculator Toolbar

The EM Calculator toolbar contains the commands that you can use to calculate EM limits, run what-if analysis, and export EM results.

The following table describes the various icons available on the EM Calculator toolbar.

Icon	Description
	Runs EM Calculator to calculate EM limits for the selected net based on the values specified in the User-Specified Fields. See the Calculating EM Limits section for more information.
	Runs the what-if analysis to see the effects of certain modifications on the EM limits. If you select the <i>JMax Only</i> check box, what-if analysis is disabled. See the Running What-If Analysis section for more information.
	Exports the EM results in a CSV file See the Exporting EM Results section for more information.
	Loads existing EM results that you earlier exported for a net. See the Loading EM Results section for more information.
	Resets the sorted EM results in the initial order.

Virtuoso Electrically Aware Design Flow Guide

Introduction to EM Calculator

User-Specified Fields

The following table describes the various user-specified fields based on which the EM Calculator calculates the EM limits.

Field	Description
<i>Top Layer</i>	Select the top-most routing layer.
<i>Bottom Layer</i>	Select the bottom-most routing layer.
<i>Route Type</i>	Select the routing type. Currently, only <code>Lateral</code> routing is supported.
<i>RSP/WSP</i>	Select this check box if you want to use widths from a selected WSP group or RSP group. If multiple WSP widths are found for the selected WSP or RSP group, a width for which the number of tracks will be minimum is recommended. By default, RSP group names are displayed in the drop-down list next to this check box. If you want to use widths from a WSP group, set the environment variable <code>useWSPGroupNames</code> to <code>t</code>: If you do not select this check box, width from the <code>allowedWidthRange</code> constraint defined in the technology file is used for calculating EM limits.
<i>Current</i>	Enter the value of current passing through the selected net. This field is disabled if you select the <i>JMax Only</i> check box.
<i>JMax Only</i>	Select this check box if you want to view the maximum allowed current.
<i>EM Analysis</i>	Select one of the following types of EM analysis: ■ <i>DC Op</i> ■ <i>Static-Avg</i> ■ <i>Dynamic-Peak</i> ■ <i>Dynamic-Rms</i> Note: The default EM analysis is <code>Static-Avg</code>.

Virtuoso Electrically Aware Design Flow Guide

Introduction to EM Calculator

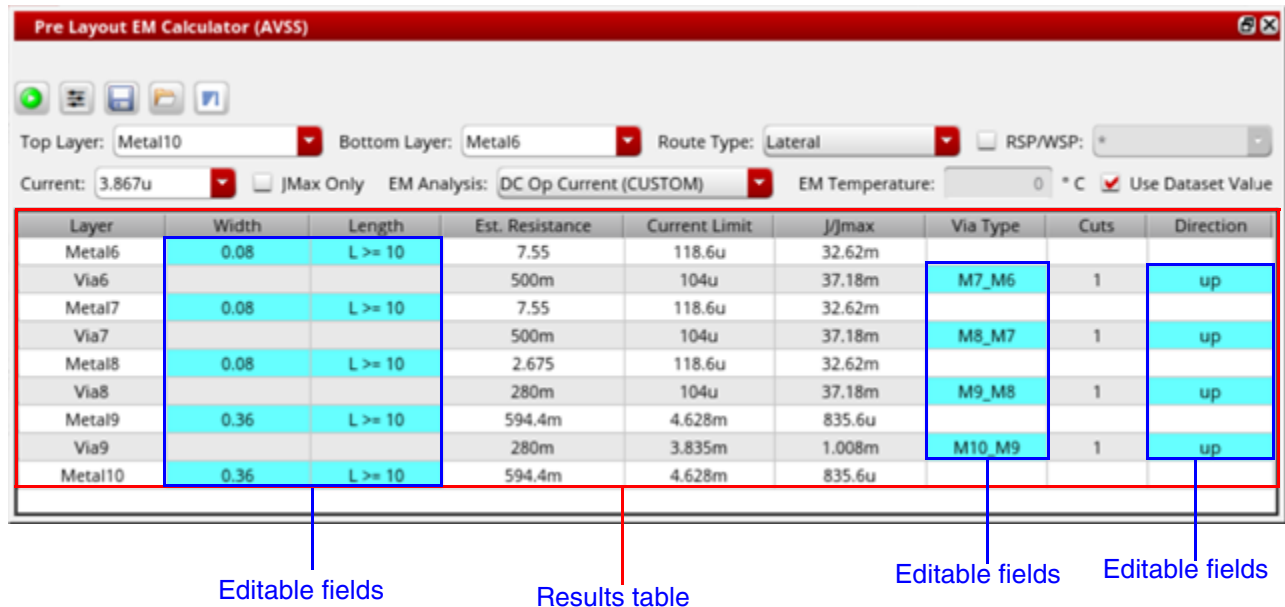
Field	Description
<i>EM Temperature</i>	The EM temperature in degrees Celsius. The value displayed in this field is same as that is set in the <i>EM Temperature</i> field on the <i>EM</i> tab of the EAD Options form. If you edit the value in this field, the same value is updated in the in the <i>EM Temperature</i> field on the <i>EM</i> tab of the EAD Options form
<i>Use Dataset Value</i>	Select this check box to use the temperature specified in the simulation dataset when computing EM effects. The <i>EM Temperature</i> field becomes unavailable when you select this check box. If the temperature value specified in the dataset is not valid, the default value specified in the <i>EM Temperature</i> field is used.

Virtuoso Electrically Aware Design Flow Guide

Introduction to EM Calculator

Results Table

The results table displays the EM calculations and recommendations of the design changes when you run the EM Calculator and the what-if analysis, respectively.



Pre Layout EM Calculator (AVSS)

Top Layer: Metal10 Bottom Layer: Metal6 Route Type: Lateral RSP/WSP: *

Current: 3.867u JMax Only EM Analysis: DC Op Current (CUSTOM) EM Temperature: 0 °C Use Dataset Value

Layer	Width	Length	Est. Resistance	Current Limit	J/Jmax	Via Type	Cuts	Direction
Metal6	0.08	L >= 10	7.55	118.6u	32.62m			
Via6			500m	104u	37.18m	M7_M6	1	up
Metal7	0.08	L >= 10	7.55	118.6u	32.62m			
Via7			500m	104u	37.18m	M8_M7	1	up
Metal8	0.08	L >= 10	2.675	118.6u	32.62m			
Via8			280m	104u	37.18m	M9_M8	1	up
Metal9	0.36	L >= 10	594.4m	4.628m	835.6u			
Via9			280m	3.835m	1.008m	M10_M9	1	up
Metal10	0.36	L >= 10	594.4m	4.628m	835.6u			

Editable fields Results table Editable fields Editable fields

Virtuoso Electrically Aware Design Flow Guide

Introduction to EM Calculator

The following table describes the various columns of the results table.

Column	Description										
Layer	Displays all the metal layers between the bottom-most and top-most layers you specified in the <i>Bottom Layer</i> and <i>Top Layer</i> fields, respectively. It also shows the via name connecting each subsequent metal layer. For example, if you select m10 and m14 as the bottom and top layers, the <i>Layer</i> column displays all the metal layers between m10 and m14 with the via names between each of these layers, as shown in the following figure: Via, v10 between the metal layers, m10 and m11 <table><tr><th>Layer</th></tr><tr><td>m10</td></tr><tr><td>v10</td></tr><tr><td>m11</td></tr><tr><td>v11</td></tr><tr><td>m12</td></tr><tr><td>v12</td></tr><tr><td>m13</td></tr><tr><td>v13</td></tr><tr><td>m14</td></tr></table> Metal layers in ascending order, from bottom-most to top-most layer	Layer	m10	v10	m11	v11	m12	v12	m13	v13	m14
Layer											
m10											
v10											
m11											
v11											
m12											
v12											
m13											
v13											
m14											
Tracks	Number of tracks This column is available only when the <u>showTracksWithoutWSP</u> environment variable is set to t.										
Width	Maximum Width This is an editable field.										
Length	Recommended length which satisfies the length-based EM rules defined in the technology file such that the design is EM-compliant. If N/A is displayed in the <i>Length</i> column, it means that no length-based rules are found for the layer. This is an editable field.										

Virtuoso Electrically Aware Design Flow Guide

Introduction to EM Calculator

Column	Description
<i>Est. Resistance</i>	Estimated resistance of the layer. Resistance of a metal layer depends on the <i>Length</i>, <i>Width</i>, and <i>EM Temperature</i> fields, whereas resistance of a via layer depends on the <i>Via Type</i>, <i>Cuts</i>, and <i>EM Temperature</i> fields. If the values of <i>Length</i> and <i>Width</i> are not valid, the value N/A is displayed in the <i>Est. Resistance</i> column. You can modify the length or width values after enabling the What-If analysis to view the updated estimated resistance value of the metal layer. This column is available only when the <u>showEstimatedResistance</u> environment variable is set to t. Calculation of the estimated resistance takes into consideration the color information for metal layers present in the ICT-EM file. The resistance of a metal layer is calculated for all colors, and the worst-case resistance for all color combinations is displayed in the <i>Est. Resistance</i> column.
<i>Current Limit</i> <i>J/J_{max}</i>	Maximum current that can flow through the layer or via EM limit expressed as the J/J_{max} ratio, which is calculated by the following expression: $J/J_{max} = \frac{\text{User-Specified Current}}{(\text{Current Limit} \times \text{Number of tracks or Number of via cuts})}$ This column is disabled when <i>JMax Only</i> check box is selected.
<i>Pulse Width</i>	The recommended values of pulse width. This information is available only for RMS or peak currents.
<i>Via Type</i>	cutClass names for a layer defined in the technology file This is an editable field.
<i>Cuts</i>	Number of via cuts required to flow the user-specified current through via such that the design is EM-compliant.
<i>Direction</i>	Direction of the current flow: up or down. Note: By default, the tool recommends a direction for which EM limit or J/J_{max} is worst. You can change the direction and see its effect on the EM limit by running the what-if analysis.

Working with the EM Calculator

Using EM Calculator, you can calculate the EM limits for the selected net, run what-if analysis, and export the EM results to a CSV file.

This section describes how to perform these tasks:

- [Calculating EM Limits](#)
- [Running What-If Analysis](#)
- [Exporting EM Results](#)
- [Loading EM Results](#)

Calculating EM Limits

Follow these steps to calculate the EM limits for a net using the EM Calculator:

1. From the summary pane of the EAD Browser, select a net for which you want to calculate the EM limits.
2. Right-click the selected net and then choose *EM Calculator* from the context menu.
The Pre Layout EM Calculator (*Net Name*) form opens.
3. From the *Route Type* drop-down list, select `Lateral`.
4. From the *Top Layer* and *Bottom Layer* drop-down lists, select the bottom and top metal layers.
5. From the *EM Analysis* drop-down list, select an electrical dataset that you want to use to calculate EM limits.

Note: The value of current from the selected dataset is automatically set in the *Current* field.

6. If you do not want to use the current from the dataset, specify the value of current manually in the *Current* field.
7. Select the *RSP/WSP* check box if you want to use minimum width from the selected RSP or WSP group. You can then select the corresponding RSP or WSP group from the drop-down list displayed next to the *RSP/WSP* check box.

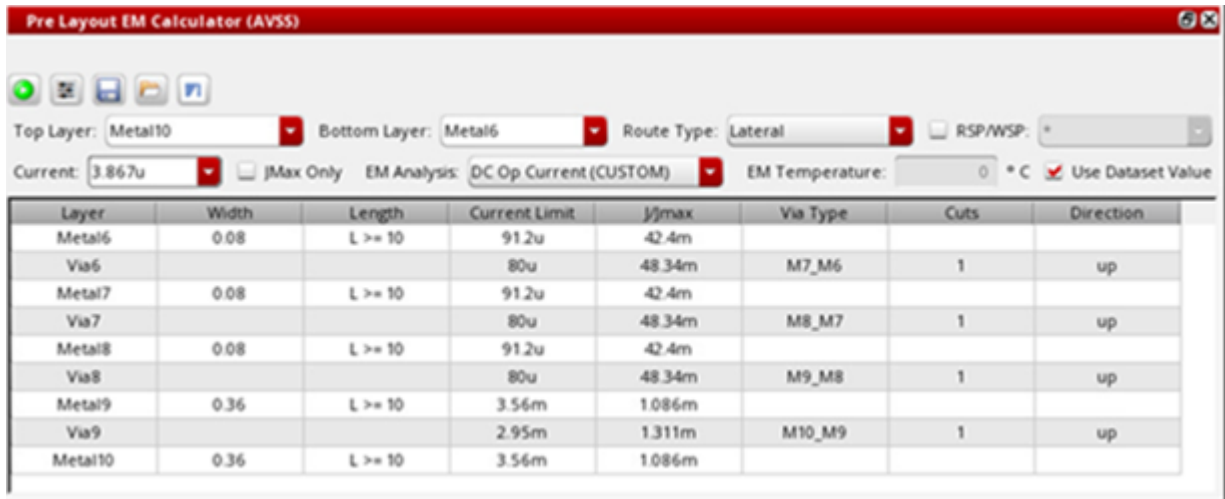
Note: If you want to use `minWidth` from all the RSP or WSP groups, select `*` from the drop-down list.

8. Edit the *EM Temperature* field if you want to specify a different temperature than what is set in the *EM* tab of the EAD Options form.
9. Select the Use Dataset Value check box if you want to use the temperature specified in the simulation dataset when computing EM effects.
10. From the EM Calculator toolbar, click *Run EM Calculator* to calculate the EM limits.

Virtuoso Electrically Aware Design Flow Guide

Introduction to EM Calculator

The EM results are displayed in the results table, as shown in the following figure.



The screenshot shows the 'Pre Layout EM Calculator (AVSS)' window. The input parameters are: Top Layer: Metal10, Bottom Layer: Metal6, Route Type: Lateral, RSP/WSP: *, Current: 3.867u, JMax Only: unchecked, EM Analysis: DC Op Current (CUSTOM), EM Temperature: 0 °C, and Use Dataset Value: checked. The results table below contains the following data:

Layer	Width	Length	Current Limit	J/Jmax	Via Type	Cuts	Direction
Metal6	0.08	L >= 10	91.2u	42.4m			
Via6			80u	48.34m	M7_M6	1	up
Metal7	0.08	L >= 10	91.2u	42.4m			
Via7			80u	48.34m	M8_M7	1	up
Metal8	0.08	L >= 10	91.2u	42.4m			
Via8			80u	48.34m	M9_M8	1	up
Metal9	0.36	L >= 10	3.56m	1.086m			
Via9			2.95m	1.311m	M10_M9	1	up
Metal10	0.36	L >= 10	3.56m	1.086m			

Running What-If Analysis

After analyzing the EM results, you can modify the following values in the results table:

- *Tracks*: Enter manually to modify.
- *Width*: Enter manually or select a value from the drop-down list.
- *Length*: Select a value from the drop-down list.
- *Via Type*: Choose the `cutClass` name for the selected via from the drop-down list.
- *Direction*: Choose either `up` or `down` from the drop-down list. Additionally, you can right-click the cell and select one of the following options to quickly change the direction of current flow for all or the selected vias:
 - ☐ *Change current direction for this via*: Changes the direction of the current flow from `up` to `down` or `down` to `up` for the selected via.
 - ☐ *Change current direction for all vias*: Changes the current direction for all the vias.
 - ☐ *Use up direction for all vias*: Changes the direction of the current flow to `up` for all the vias.
 - ☐ *Use down direction for all vias*: Changes the direction of the current flow to `down` for all the vias.

Note: See the [Results Table](#) section for more information about these columns.

You can then run the what-if analysis by clicking  from the EM Calculator toolbar to see their effects on the EM limits and get the recommended values of *Tracks*, *Current Limit*, J/J_{max} , and *Cuts* updated in the respective columns.

Important

If you select the *JMax Only* check box, what-if analysis is disabled.

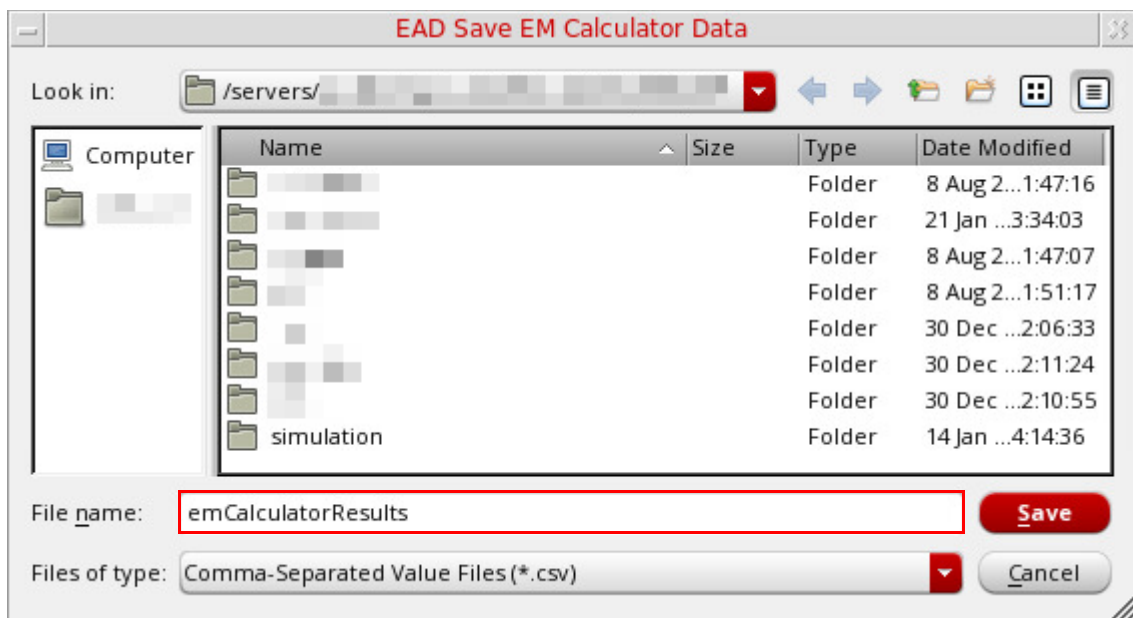
Exporting EM Results

After running the EM calculator or the what-if analysis for a net, you can export the EM results to a CSV file.

To export EM results for a net to a CSV file:

1. From the toolbar of the EM Calculator, click *Export the results* () .

The EAD Save EM Calculator Data form opens.



2. In the *File name* field, enter the name of the CSV file to which you want to export the EM results for the net.
3. Click *Save*.

You can later reuse this file to load these results for the net.

Loading EM Results

You can load existing results that you earlier exported for a net.

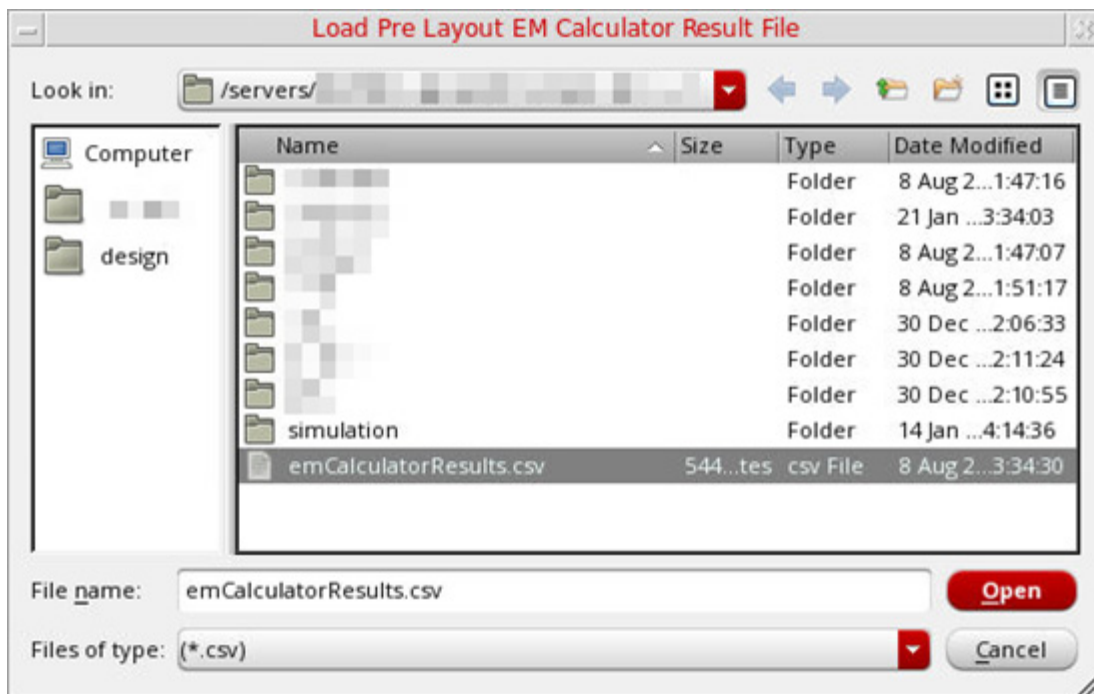
To load EM results:

1. From the summary pane of the EAD Browser, select a net for which you want to load the EM results.
2. Right-click the selected net, and then choose *EM Calculator*.

The Pre Layout EM Calculator (*Net Name*) form opens.

3. From the toolbar of the EM Calculator, click *Load existing results* (📁).

The Load Pre Layout EM Calculator Result File form opens.



4. Select the CSV file containing the EM results that you earlier exported for the same net.
5. Click *Open*.

Virtuoso Electrically Aware Design Flow Guide

Introduction to EM Calculator

The results are displayed in the results table of the EM Calculator.

Pre Layout EM Calculator (AVSS)

Route Type: Lateral Top Layer: Metal10 Bottom Layer: Metal6

EM Analysis: DC Op Current (CUSTOM) Current: 3.867u JMax Only RSP/WSP: *

Layer	Width	Length	Current Limit	J/Jmax	Via Type	Cuts	Direction
Metal6	0.08	L >= 10	91.2u	42.4m			
Via6			80u	48.34m	M7_M6	1	up
Metal7	0.08	L >= 10	91.2u	42.4m			
Via7			80u	48.34m	M8_M7	1	up
Metal8	0.08	L >= 10	91.2u	42.4m			
Via8			80u	48.34m	M9_M8	1	up
Metal9	0.36	L >= 10	3.56m	1.086m			
Via9			2.95m	1.311m	M10_M9	1	up
Metal10	0.36	L >= 10	3.56m	1.086m			

Resimulating Designs with Extracted Parasitics

The parasitic information of the design gets updated dynamically while you run electromigration analysis and make changes in the layout to resolve violations. You can use this information, which you can get from a partial or complete layout, to regenerate a netlist and then rerun simulations in ADE Assembler to check if the output specifications are still met. If required, you can further modify design variables and parameters or the layout to meet the desired results.

This chapter describes the enhancements in the parasitic aware design flow and how to use parasitics from the layout view to run simulations in ADE Assembler.

This chapter includes the following topics:

- [Preparing the Parasitic Setup](#)
- [Running Simulations and Verifying Specifications](#)
- [Using Extracted Parasitics to Run Spectre EM/IR Analysis](#)
- [Using Parasitic Values from Multiple Sources](#)
- [Running Advanced Simulation Runs to Improve Designs](#)
- [Viewing and Updating Layout Parasitics in VSE XL](#)

Running Simulations with Layout Parasitics

After the electromigration analysis is complete and violations are resolved, reopen the adexl view of your design. Next, prepare the parasitic setup by including layout parasitics and rerun simulations.

The following topics describe how to perform parasitic resimulations:

- [Preparing the Parasitic Setup](#)
- [Running Simulations and Verifying Specifications](#)

Preparing the Parasitic Setup

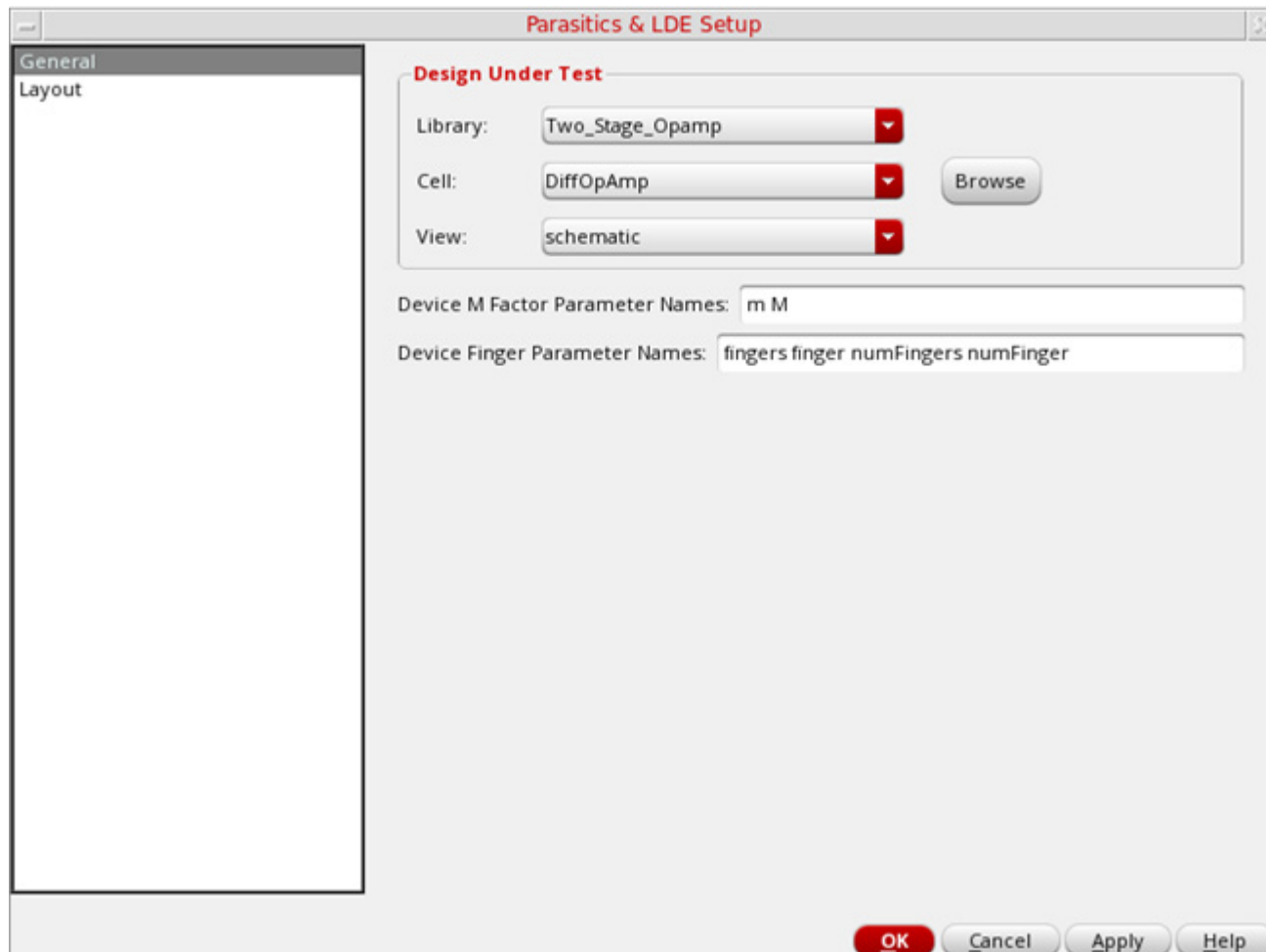
Preparation of parasitic setup involves creation of a netlist_layout view or a DSPF file that contains parasitics and other layout-dependent effects (LDE) to be used in simulation.

The following sections describe the steps to prepare a parasitic setup in different ways:

- [Creating a Netlist Cellview with Layout Parasitics in Virtuoso Layout EAD](#)
- [Creating a Netlist Cellview with Layout Parasitics in Virtuoso ADE Assembler](#)
- [Creating a DSPF File with Layout Parasitics in Virtuoso Layout EAD](#)

Creating a Netlist Cellview with Layout Parasitics in Virtuoso Layout EAD

To create a netlist cellview from Layout EAD, click  on the EAD toolbar or the EAD Browser toolbar and choose *Build Parasitic/LDE Netlist View*. The Parasitics & LDE Setup form is displayed as shown below.



The image shows the 'Parasitics & LDE Setup' dialog box. On the left is a sidebar with 'General' and 'Layout' tabs. The 'General' tab is selected. The main area is titled 'Design Under Test' and contains the following fields:

- Library:** A dropdown menu showing 'Two_Stage_Opamp'.
- Cell:** A dropdown menu showing 'DiffOpAmp' and a 'Browse' button to its right.
- View:** A dropdown menu showing 'schematic'.
- Device M Factor Parameter Names:** A text field containing 'm M'.
- Device Finger Parameter Names:** A text field containing 'fingers finger numFingers numFinger'.

At the bottom right are four buttons: 'OK' (highlighted in red), 'Cancel', 'Apply', and 'Help'.

On the *General* tab, the library and cell name of the layout view are displayed in the *Library Name* and *Cell Name* fields.

On the *Layout* tab, enter a name to be used for the netlist view in the *View Name* field. Specify other values on this form.

Related Topic:

- [Parasitics & LDE Setup form field description](#)

Click *OK* to close the form and to build the netlist view.

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

A message box displaying a summary of parasitics and other LDEs included in the netlist view is displayed. Click *OK* to close this message.

Note: This view is used in ADE Assembler when you select Layout (Parasitics/LDE) in the parasitic Mode toolbar.

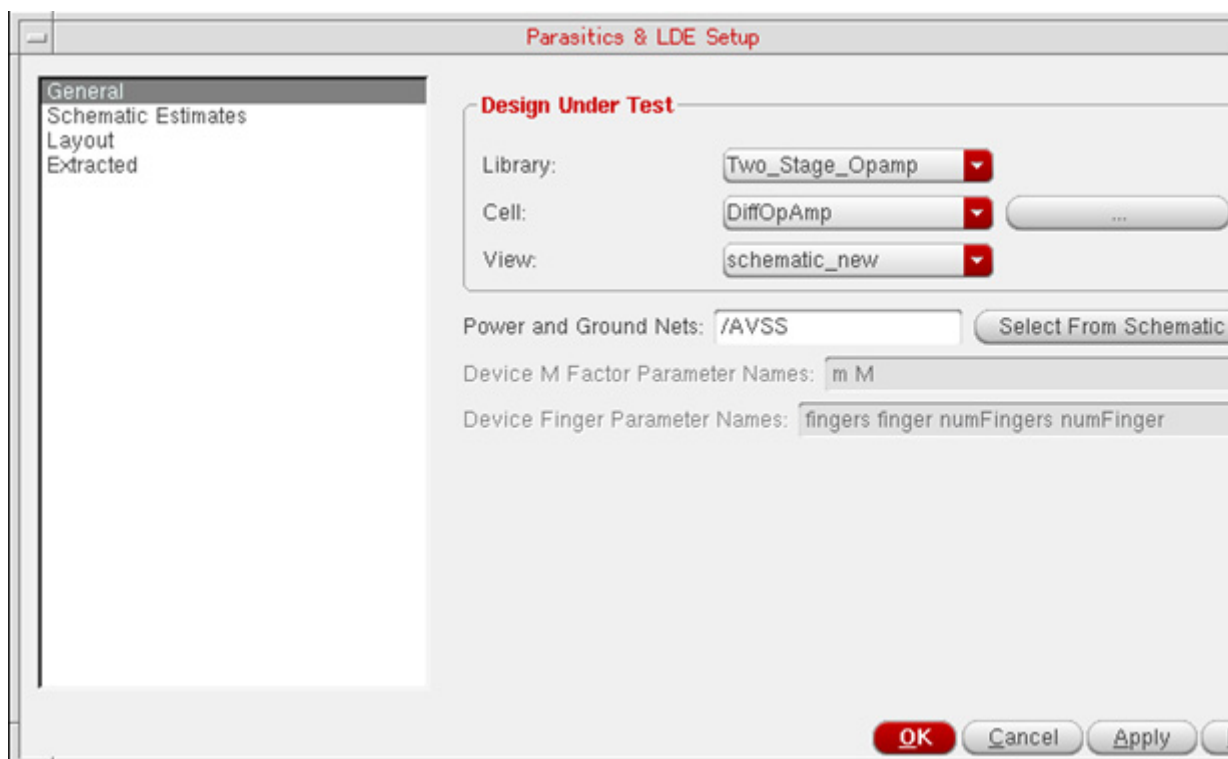


Creating a Netlist Cellview with Layout Parasitics in Virtuoso ADE Assembler

This method is helpful when the schematic and layout views are available in different libraries. Perform the following steps to prepare a netlist from ADE Assembler:

1. In the ADE Assembler window, choose *Parasitics/LDE – Setup*.

The **Parasitics & LDE Setup** form is displayed, as shown below.

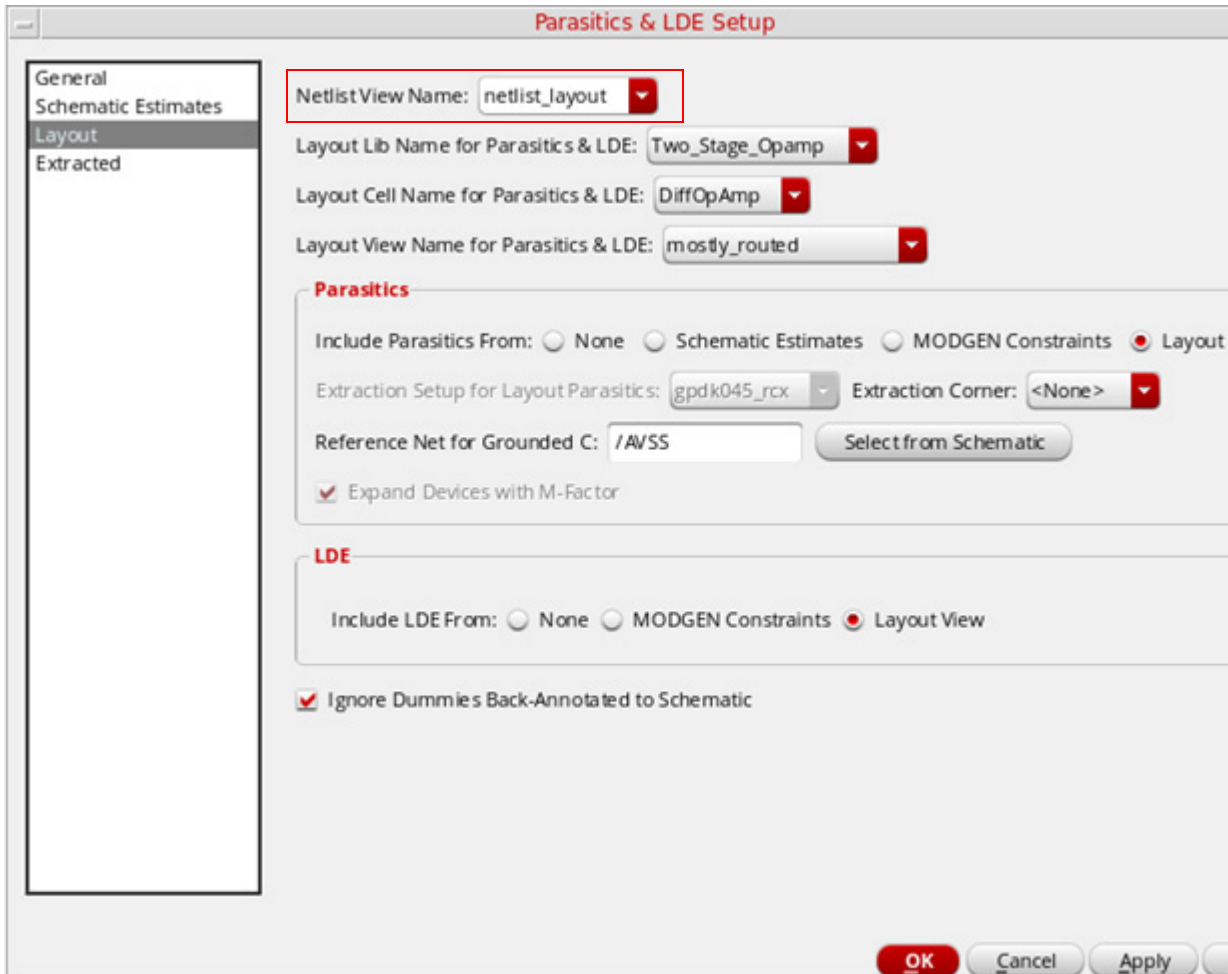


The General tab shows the schematic view of the DUT.

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

2. Open the Layout tab and specify a name for the netlist layout view to be created with the parasitics. By default, this is set to `netlist_layout`.



3. Using the following drop-down lists, select the library, cell, and view names of the layout view from where the parasitics will be read in:
 - ☐ *Layout Lib Name for Parasitics & LDE*
 - ☐ *Layout Cell Name for Parasitics & LDE*
 - ☐ *Layout View Name for Parasitics & LDE*
4. Specify the following values in the *Parasitics* section:
 - a. Select any one of the following options for the *Include parasitics from* field:
 - ☐ *Layout*: To include parasitics from the layout view for the design

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

- *MODGEN constraints*: To include parasitics from MODGEN constraints. When you select this option, the layout view is not checked for parasitics. A dummy layout is created by placing all MODGEN constraints and device instances. A netlist is then created by using that dummy layout.

If you extracted parasitics for multiple corners in the layout view, names of all the corners are listed in the *Extraction corner for layout parasitics* list.

- b. Select the name of corner for which you want to use parasitics in the netlist layout view.
- c. Specify the name of a ground net in the *Reference net for Grounded C* field. You can also select a net from the schematic by clicking on *Select from Schematic*.
- d. By default, the *Expand Devices with M-Factor* check box is selected, which implies that Virtuoso looks for cells in the layout that have parameters defining multiple factors. If the corresponding cell in the schematic view does not contain an equal number of factors, before generating the netlist, multiple copies of that schematic cell are created to establish the connection. If not required, clear this check box.
- e. When you select this check box, also specify the names of the CDF parameters that define m-factor parameters of devices in the *Device M Factor Parameter Names* field on the General tab.
- f. If there are any dummy cells backannotated from the layout to the schematic view, you can choose to include them while generating the netlist for simulation by clearing the *Ignore Dummies Back-Annotated to Schematic* check box. By default, such dummy cells are ignored.

For more details about the **Setup Parasitics & LDE** form, see [Parasitics/LDE –Setup](#) in the *Parasitic Aware Design User Guide*.

5. If required, configure other settings in this form.
6. Click *Apply* to save the settings.

The tool validates the entries in the form.

7. Click *OK* to close the form.

Virtuoso creates a netlist view with the name specified in the *Netlist View Name* field. This netlist view includes the parasitics from the layout view.

The simulation mode in ADE Assembler is automatically changed to *Layout (Parasitics/LDE)*. This is the mode in which ADE Assembler runs simulations with parasitics included in the netlist.

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

Important

A netlist created from a layout view merges the parasitics from both the layout view and the schematic view. While doing that, preference is given to the parasitics in the layout view. For example, if you have specified both resistance and capacitance for a net in the schematic view, whereas, only capacitance for that net in the layout view, the netlist will get the capacitance value from the layout and resistance value from the schematic. To use only the parasitics from the layout view and to ignore all the estimated parasitics from the schematic, set the `mixSchEstWithLayoutParasitics` environment variable to `nil`.

Note: If you change the layout view of your design, you need to rebuild the parasitic/LDE view before running simulations to include the parasitics from the layout correctly. For this, click *Build Parasitic/LDE View* on the *Parasitic Mode* toolbar in the ADE Assembler window.



The Build Parasitic/LDE View form is displayed. If required, specify the scaling factors for parasitics and click *OK*.

Creating a DSPF File with Layout Parasitics in Virtuoso Layout EAD

To export the parasitics from Layout EAD into a DSPF file, click  on the EAD toolbar or the EAD Browser toolbar and choose *Generate DSPF File*. The Parasitics & LDE Setup form is displayed.

Validate the information on the *General* tab and make changes, if required.

Open the *DSPF* tab, as shown below.

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

The screenshot shows the 'Parasitics & LDE Setup' dialog box with the 'DSPF' tab selected. The left pane shows a tree view with 'General' and 'DSPF' (selected). The right pane contains the following settings:

- DSPF File Name: DiffOpAmp (with a browse button)
- Subnode Delimiter: #
- Format for Capacitance Values**
 - Coupling Capacitance Prefix: (empty text box)
 - Ground Capacitance Prefix: (empty text box)
 - Unit: f (dropdown menu)
- Format for Resistance Values**
 - Resistance Prefix: (empty text box)
 - Unit: ohm (dropdown menu)
- ☒ Write Instances

At the bottom right are buttons for OK, Cancel, Apply, and Help.

This tab shows the settings used to write parasitic data in a DSPF file. If required, customize the settings. Click *OK* to close the form and generate the DSPF file.

The DSPF file generated in Virtuoso Layout EAD does not include LDE effects and cannot be used for parasitics resimulation.

Note: The subcircuit port order in the DSPF file is retrieved from terminal order (`termOrder`) defined in the *Simulation Information* (`simInfo`) of Spectre.

Related Topic:

- [Parasitics & LDE Setup](#) form field description

Running Simulations and Verifying Specifications

After the estimated view is built successfully, run a new simulation for your design and view the results to analyze how the desired output specifications are met. The color codes on the Results tab indicate the pass, fail, or near pass status, as shown in the figure below.

Outputs Setup Results Diagnostics										
Detail      Replace       										
	Parameter	Nominal						C0_0	C0_1	C0_2
	temperature	27						-40	0	125
Test	Output	Nominal	Spec	Weight	Pass/Fail	Min	Max	C0_0	C0_1	C0_2
AC	/V1/PLUS									
AC	/OUT									
AC	Current	2.736m	< 3m		pass	2.561m	2.919m	2.919m	2.802m	2.561m
AC	UGF	400.9M	> 300M		near	270.8M	582.8M	582.8M	465.7M	270.8M
AC	Gain	60.69	> 60		near	59.64	60.95	60.95	60.81	59.64
TRAN	SettlingTime	8.075n	< 10n		pass	7.843n	8.325n	7.843n	7.987n	8.325n
TRAN	/OUT									
TRAN	Swing	1.62	> 1		pass	1.57	1.653	1.653	1.633	1.57

Comparing the Simulation Results

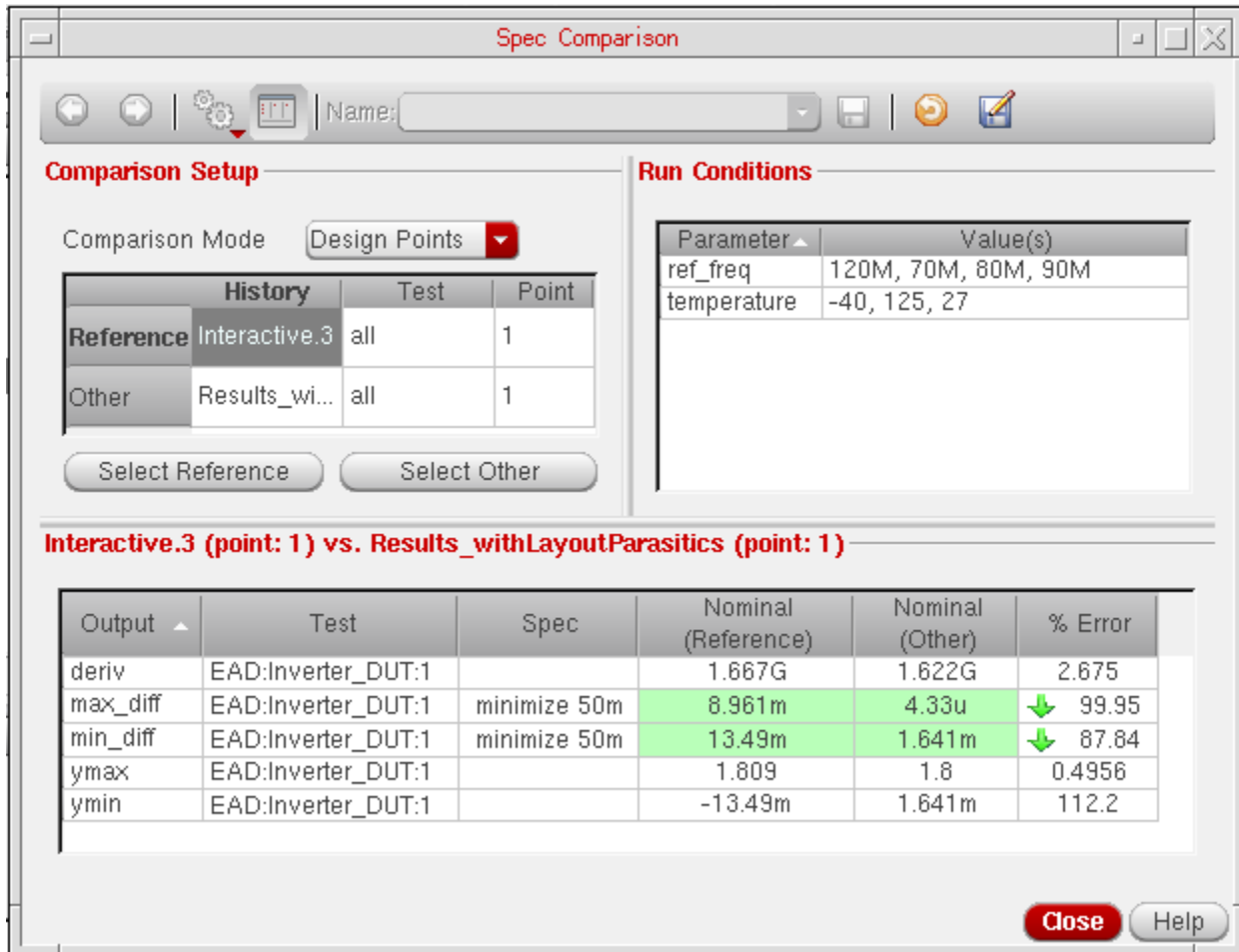
You can compare the results with that of the previous simulations, which were run without using the parasitic estimates. For this, you can use the Spec Comparison form in ADE Assembler.

On the Results toolbar, click  (*Spec Comparison*) to open the **Spec Comparison** window. Select two histories for which you need to compare the results, and specify the test and design points for which you need to compare the results. ADE Assembler creates a

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

comparison report and displays it in the reporting section of this window, as shown in the figure below.



For more details on Spec Comparison, refer to [Comparing Results](#) in the *Virtuoso ADE Assembler User Guide*.

The result comparison summary helps in analyzing the impact of parasitics on the design performance. Accordingly, you can decide the further actions to be taken to improve the design schematic or layout.

Using Extracted Parasitics to Run Spectre EM/IR Analysis

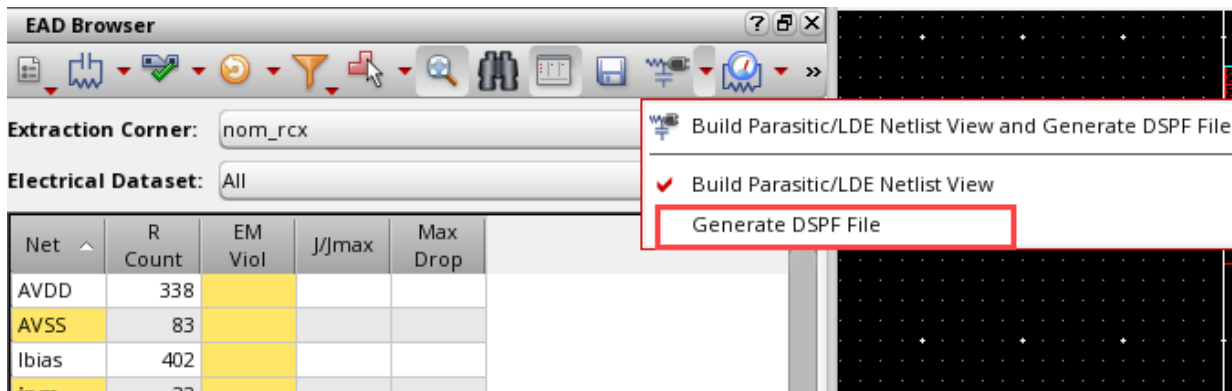
The extracted parasitics can be saved in DSPF files, which you can include in Spectre EM/IR analysis. This provides the capability to use the Spectre solver for the generation of current data. You can further use that current data to run EM checks in EAD.

To use the extracted parasitics to run Spectre EM/IR analysis, perform the following steps:

1. Open the layout view of the design.
2. Open the EAD Browser, and run parasitic extraction by using the *Extract Parasitics for All Nets* command.

Check the extracted parasitics data.

3. Use the *Export Parasitics – Generate DSPF File* command to save the parasitic details in a DSPF file.



4. Specify the settings for DSPF file generation as explained in [Creating a DSPF File with Layout Parasitics in Virtuoso Layout EAD](#).

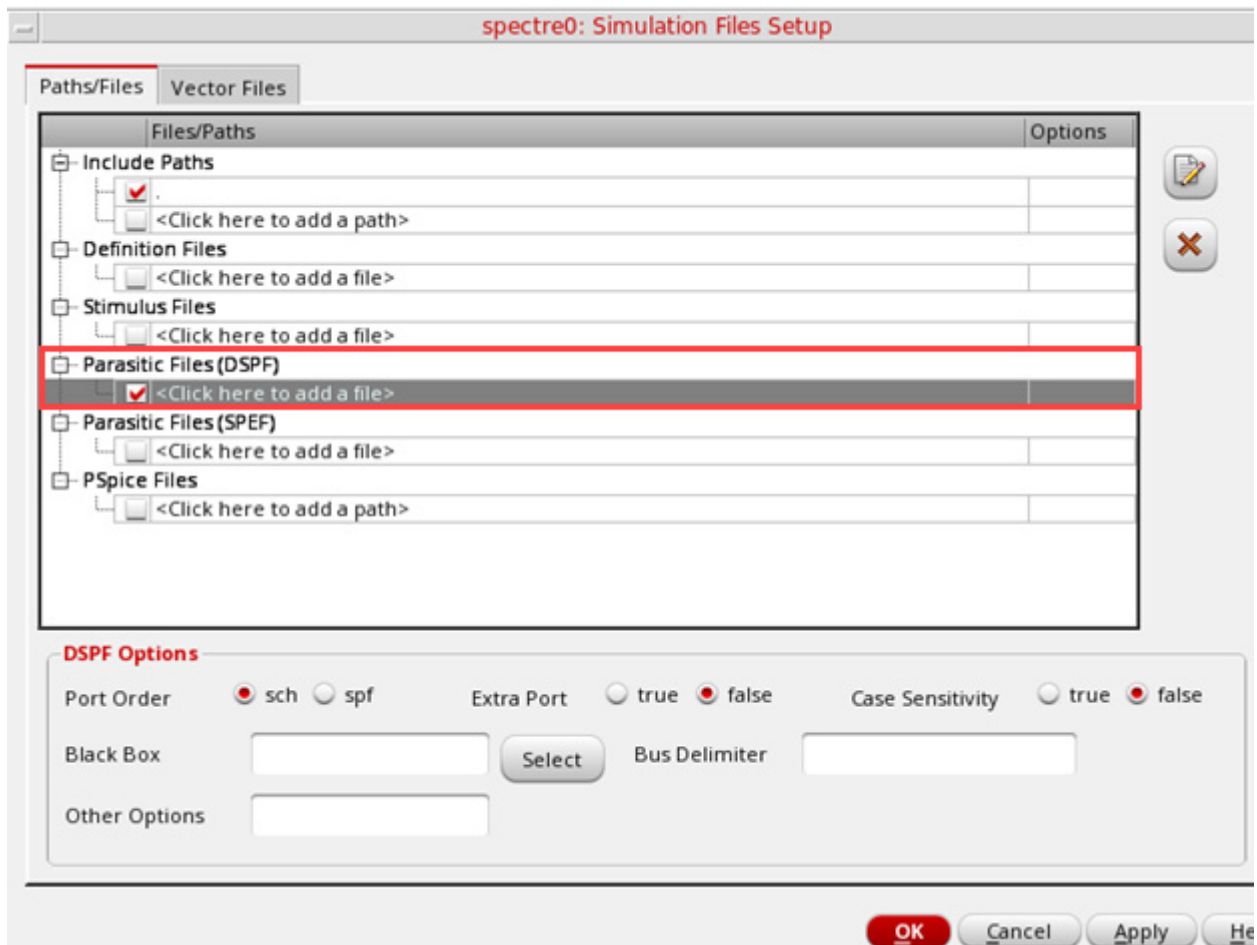
A .dspf file is saved with the extracted parasitics.

5. Open the maestro cellview for the design in Virtuoso ADE Explorer.
6. Choose *Setup – Simulation Files* to open the Simulation Files Setup form.

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

- Specify the name of the saved DSPF file in the Simulation Files Setup form.



- Choose *Setup – EM/IR Analysis* to open the *Spectre EMIR / Voltus-Fi XL Analysis Setup* form.
- Do the following:

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

- a. On the *Analysis* tab, select the *Enable EMIR Analysis in transient or DC Solution*.

The screenshot shows the 'spectre0: Spectre EMIR / Voltus-Fi XL Analysis Setup' dialog box with the 'Analysis' tab selected. The 'Type' section has 'Dynamic' selected. The 'Net Selection' section includes fields for 'Net Name' and 'Add/Modify' buttons, and checkboxes for 'IR Drop Analysis' (max, avg), 'EM Current Analysis' (max, avg, avgabs, rms, acpeak/pwc), 'Advanced IR Drop Analysis' (Signal Net IR Drop, Power Gate), and 'Analysis' (TMI Selfheating). The 'EM Rule Check Setup' section has fields for 'EM Tech File', 'Layer Map File', and 'EM Only ICT File', each with an 'Open' button. The 'Summary Information' section has a table with 'Options' and 'Value' columns, and buttons for 'Import', 'Export', 'Delete', and 'Clear'. At the bottom, there is a checkbox for 'Enable EMIR Analysis in Transient or DC Simulation' which is checked, and 'OK', 'Cancel', 'Apply', and 'Help' buttons.

spectre0: Spectre EMIR / Voltus-Fi XL Analysis Setup

Analysis | Solver | Options | Macro Model Generation

Type: ☒ Dynamic ☐ Static ☐ SPGS

Net Selection

Net Name:

IR Drop Analysis: ☐ max ☐ avg

EM Current Analysis: ☐ max ☐ avg ☐ avgabs ☐ rms ☐ acpeak/pwc

Advanced IR Drop Analysis: ☐ Signal Net IR Drop ☐ Power Gate

Analysis: ☐ TMI Selfheating

EM Rule Check Setup

EM Tech File

Layer Map File

EM Only ICT File

Summary Information

Options	Value

☒ Enable EMIR Analysis in Transient or DC Simulation

- b. On the *Solver* tab, ensure that the solver method is selected as *direct*.

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

10. Run simulation in ADE Explorer and save results.
11. Open the layout view again.
12. In the EAD Browser, create a user-defined dataset by using `Spectre EM/IR Results` as the current source.

For more details, refer to [Creating User-Defined Custom Datasets](#).

13. Run EM analysis in EAD Browser.

EAD runs EM analysis for the nets by using the saved current dataset.

Using Parasitic Values from Multiple Sources

If you have defined some important parasitics in the schematic view of your design, you can choose to use them along with the parasitics from layout.

To create parasitic estimates, in the ADE Assembler window, choose *Parasitics/LDE – Create Estimates*. The Parasitics & Electrical Setup assistant is displayed. Select a net or terminal on the schematic and choose the type of estimate to be created. A new estimate is created. For more details, refer to [Creating Parasitic Estimates](#) in the *Virtuoso Parasitic Aware Design User Guide*.

You can now add parasitic estimates from other sources, such as a layout view, in any one of the following ways:

- [By setting the source of a parasitic estimate to a layout](#)
- [By creating layout stitching estimates for a cell](#)

Setting the Source of a Parasitic Estimate to a Layout View

To specifically set the source for an estimate to a layout view, do the following:

1. Open the schematic view of your design.
2. Select the nets for which you want to create parasitic estimate constraints.
3. In the **Parasitics and Electrical Setup** assistant, click *Create Estimated Capacitance* to create a capacitance estimate, or *Create Estimated Resistance* to create a resistance estimate.

A new estimate is created for the net and displayed in the assistant.

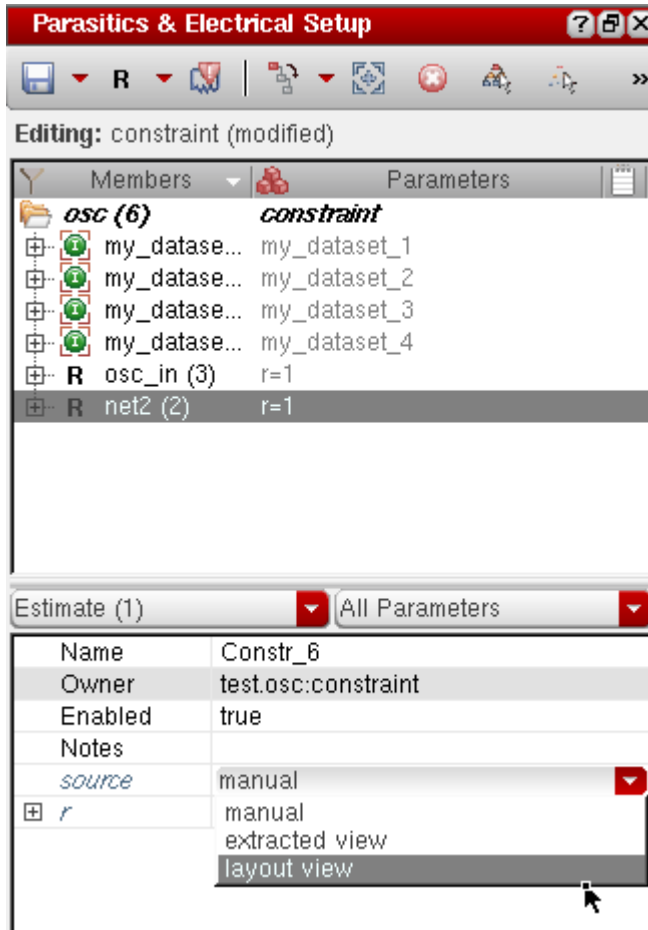
4. Select the newly created estimate and open the Estimate Editor section.

By default, the source property of a newly created estimate is set to `manual`.

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

5. Change the source property for the estimate to `layout view`, as shown in the figure below.



When you change the source of the estimate, the tool automatically adds three properties, *layLib*, *layCell*, and *layView*, and shows the names of the library, cell, and first found layout view for the current cell.

In this way, you can add a mixed set of estimates, with different sources, for a particular cellview. When you run a simulation with schematic estimates the next time, parasitic estimates will be included according to the source type.

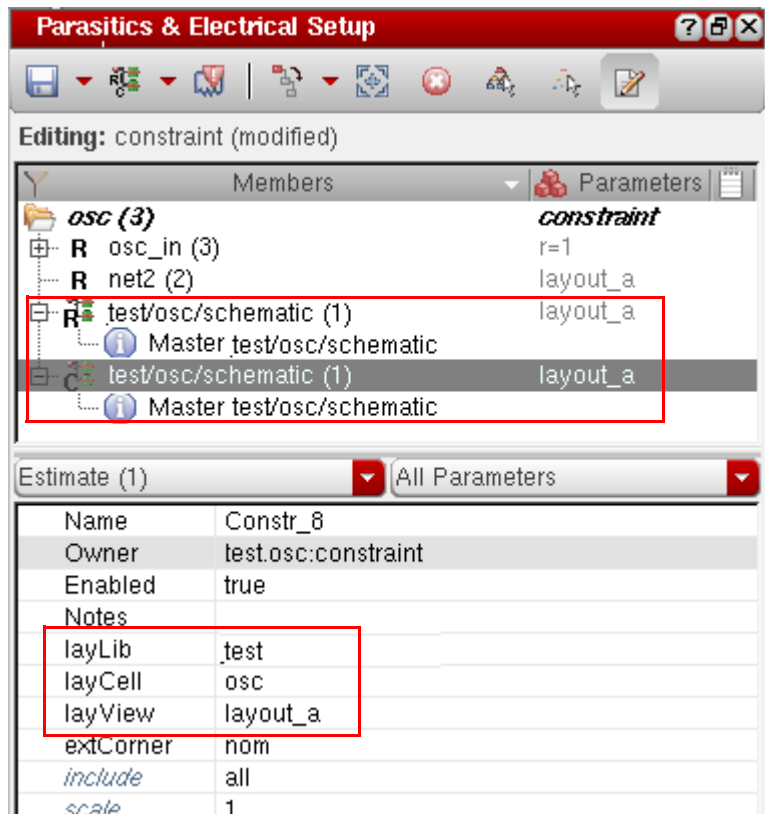
Important

You cannot set the source of L and K estimates to `layout view`.

Creating Layout Stitching Estimates for a Cell

If you want to get all the parasitics from a layout view, you can create a layout stitching estimate. For this, in the **Parasitics and Electrical Setup** assistant, click *Create Layout Stitching Estimates for Cell*.

Two new estimates, one of R and C type each, are created, as shown in the figure below.



The new estimate is attached to a layout view from where it is pulling all the R or C estimates.

Running Advanced Simulation Runs to Improve Designs

If the simulation results or the spec comparison reports indicate that the design performance is not as expected, you need to modify the design schematic or design parameters or the layout and rerun simulations until you get the desired results.

Some of the run modes in ADE Assembler help in identifying the design or parameter settings that can generate the desired results. Some of the possible ways to achieve the desired results are given below:

- Run a local or global optimization to find out the best design point. For more details, refer to *Virtuoso ADE Assembler User Guide*.
- Use the manual tuning run mode to run simulations with different parameterization setups and modify the design between different runs manually.
- Update the layout of devices, re-extract parasitics, run EM checks and resimulate the design with parameters.

Important Points to Note

- In the `Local Optimization` run mode, you can sweep the values of device parameters to identify the best design point that meets the desired specifications. For details on extracted view optimization, refer to Optimizing an Extracted or Layout View in the *Virtuoso Parasitic Aware Design Flow User Guide*.
- After you modify the devices in schematic or change the design parameters, you can rebuild the `netlist_layout` view with the same layout parasitics. You do not have to make any changes in the layout until you have found a good solution.

Viewing and Updating Layout Parasitics in VSE XL

During the resimulation phase of the EAD flow, you can identify the effect of layout parasitics on the simulation results of your design. As described in [Running Simulations with Layout Parasitics](#), you can create a netlist using the layout view that includes parasitics, run simulation, and compare the results with the simulation results obtained for the schematic view.

As a circuit designer, you can also visualize the layout parasitics on the schematic view to analyze how they have affected the simulation results. If the specifications for simulation results are not being met, you can tune the devices and apply constraints to provide guidance to the layout designer for the placement of those devices. In this way, you can collaborate with the layout designer to ensure that the layout is electrically correct.

The EAD Browser assistant in the schematic view provides the capability to overlay the layout parasitics on the schematic. This assistant provides access to parts of the EAD Browser that are useful to the circuit designer.

To open the EAD Browser assistant in VSE XL window,

➔ Choose *Launch – Plugins – EAD*.

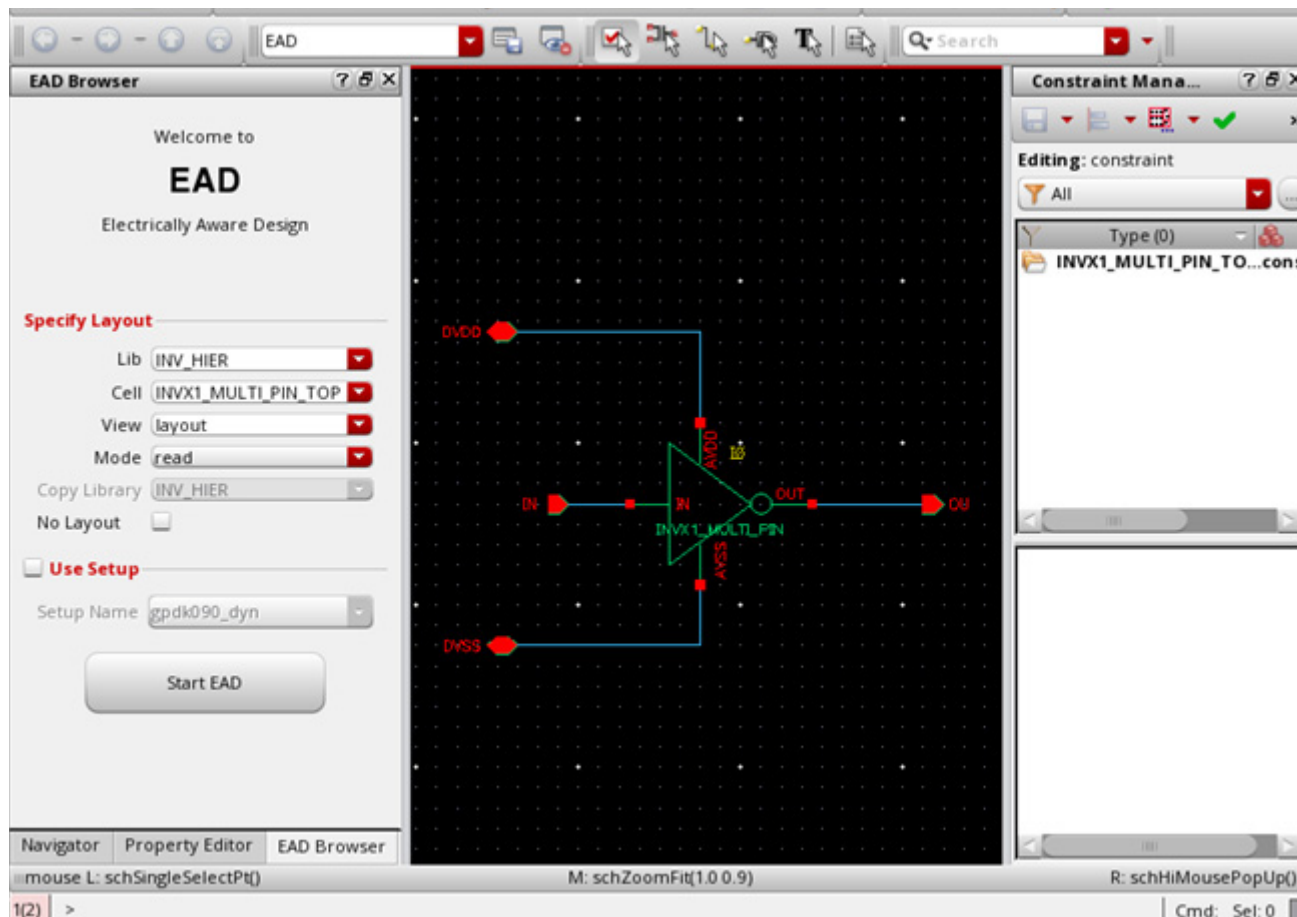
The EAD workspace for VSE XL is loaded. This workspace consists of:

- The EAD Browser assistant that is displayed to the left of the schematic diagram.
- The EAD toolbar that is displayed on top of the layout canvas. This toolbar is not visible by default. To view the toolbar, choose *Window – Toolbars – EAD Toolbar*.
- The EAD menu. This menu contains the command to open the EAD Options form.

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

Initially, the EAD Browser assistant shows the fields required to prepare the setup, as shown below.



For the EAD Browser to retrieve the parasitics from the layout view, the first step is to choose a layout view. Select the layout view corresponding to the current schematic, and then select a mode in which you want to access it.

You can choose from the following three modes:

- **Copy:** Copies the entire top-level cell to a new cell that you can manage. This allows you to make changes to the extraction engines, the filtering options, and the layout without involving the layout engineer. In this mode, when any changes are done in the source layout view, you are prompted to update the saved copy to reflect the changes. If you agree, the copy of the layout view is updated. If not, the saved copy is not updated at that time, but is automatically updated later when you use the *Update Parasitics & EM* command from the *EAD Browser Toolbar*.

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

By default, the new cell in which the top-level cell is copied is created in the same library. However, if you do not have write permission to that location, you can specify a different library name in the *Copy Library* drop-down list, which is enabled only in the copy mode.

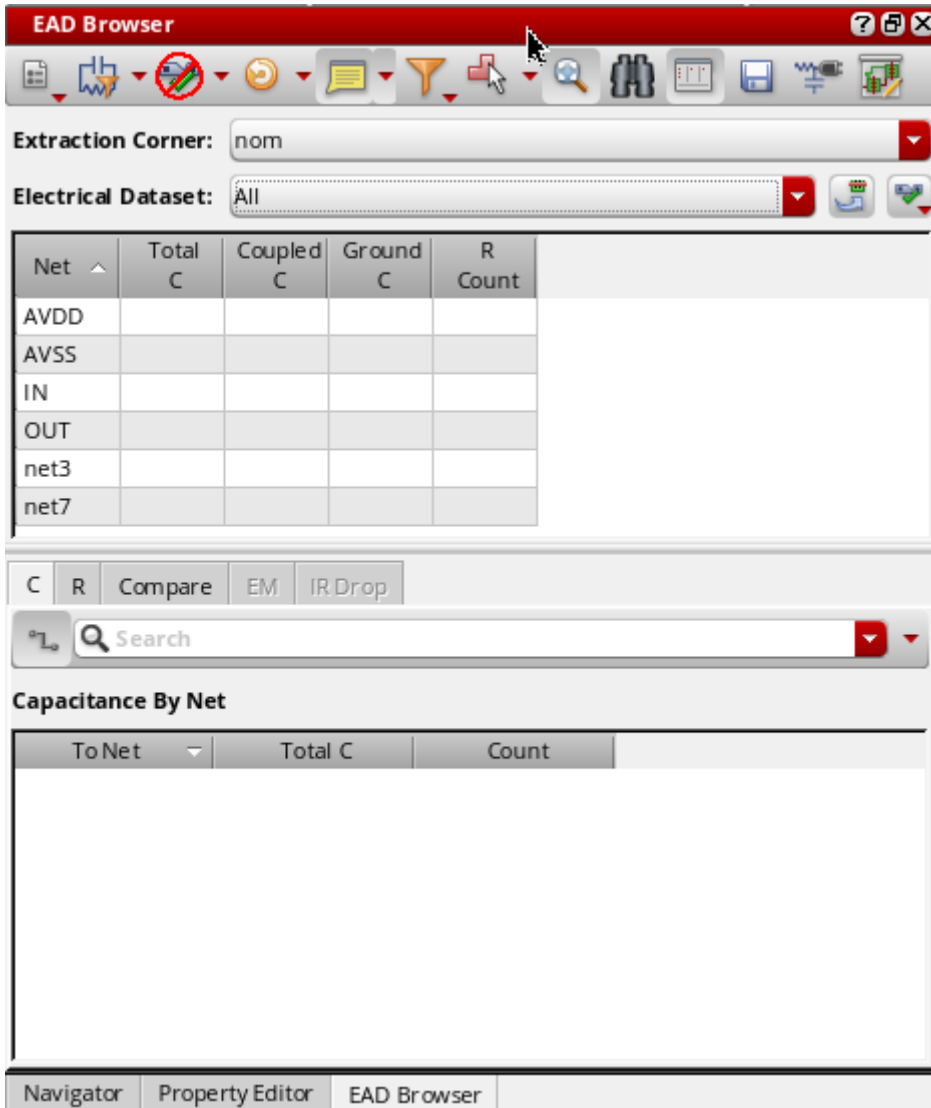
- **Read:** (Default) Opens the layout view in read mode, which means that you can view any saved parasitics, but cannot make changes to the extraction engines or the filtering options. Only the layout designer can make changes and those changes are always reflected in your view. This mode avoids duplication of data.
- **Edit:** Opens the layout view in edit mode, which means that you will be able to make changes to the extraction engines, the filtering options, and parasitics. The changes will be available to the layout engineer later, but the view will be locked while you are using it.

If the useSetup environment variable is set to the path of the file that contains the setup options to be used, the *Use Setup* check box is selected by default, and the name of the EAD setup is displayed in the *Setup Name* drop-down list. Otherwise, you can select the check box and choose a setup filename.

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

Click *Start EAD* in the EAD Browser assistant to bind the given layout cellview to the schematic cellview and to view the details of EAD Browser, as shown below.



Important Points to Note

- The setup details, including the process settings, extraction engine controls, high precision C and R extractor settings, and so on. are loaded from the EAD setup file selected on the Welcome to EAD form.
- The EAD Browser assistant in VSE XL is the same as that available in Virtuoso Layout Suite XL, except for the following differences:

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

- Display annotations on the schematic (): Annotates the net capacitance and current for instance terminals on the schematic.
- Show Layout (): Opens the layout view corresponding to the current schematic view. This layout is the same as that selected on the Welcome to EAD form.
- Run Pre-EM Checker & Generate Report (): Runs the pre-EM check in VSE XL and generates the pre-EM report. This option is available only if the *No Layout* check box is selected in EAD Browser assistant of the VSE XL.
- Currently, you cannot view the electromigration information for the nets or perform electromigration checks in the schematic view. Therefore, the commands in the EM Checker menu () are not available.

Depending on the mode (Copy, Read, or Edit) in which the layout view is being accessed, different sets of commands are enabled or disabled on the EAD Browser toolbar. For example, in Read mode, you can only read and display the available parasitics. Therefore, the commands to edit EAD setup, extract parasitics, and update electrical data are disabled on the EAD Browser toolbar, as shown below.



Commands not available when the layout view is accessed in the read mode. These commands are available when the layout view is used in the edit or copy mode.

For details of all the commands, refer to [EAD Browser Toolbar](#).

The following sections describe how you can use the EAD Browser in VSE XL to view and update parasitics in the schematic view:

- [Viewing Parasitics](#)
- [Extracting Parasitics](#)
- [Annotating Parasitics on the Schematic](#)
- [Modifying Parasitics on the Schematic](#)

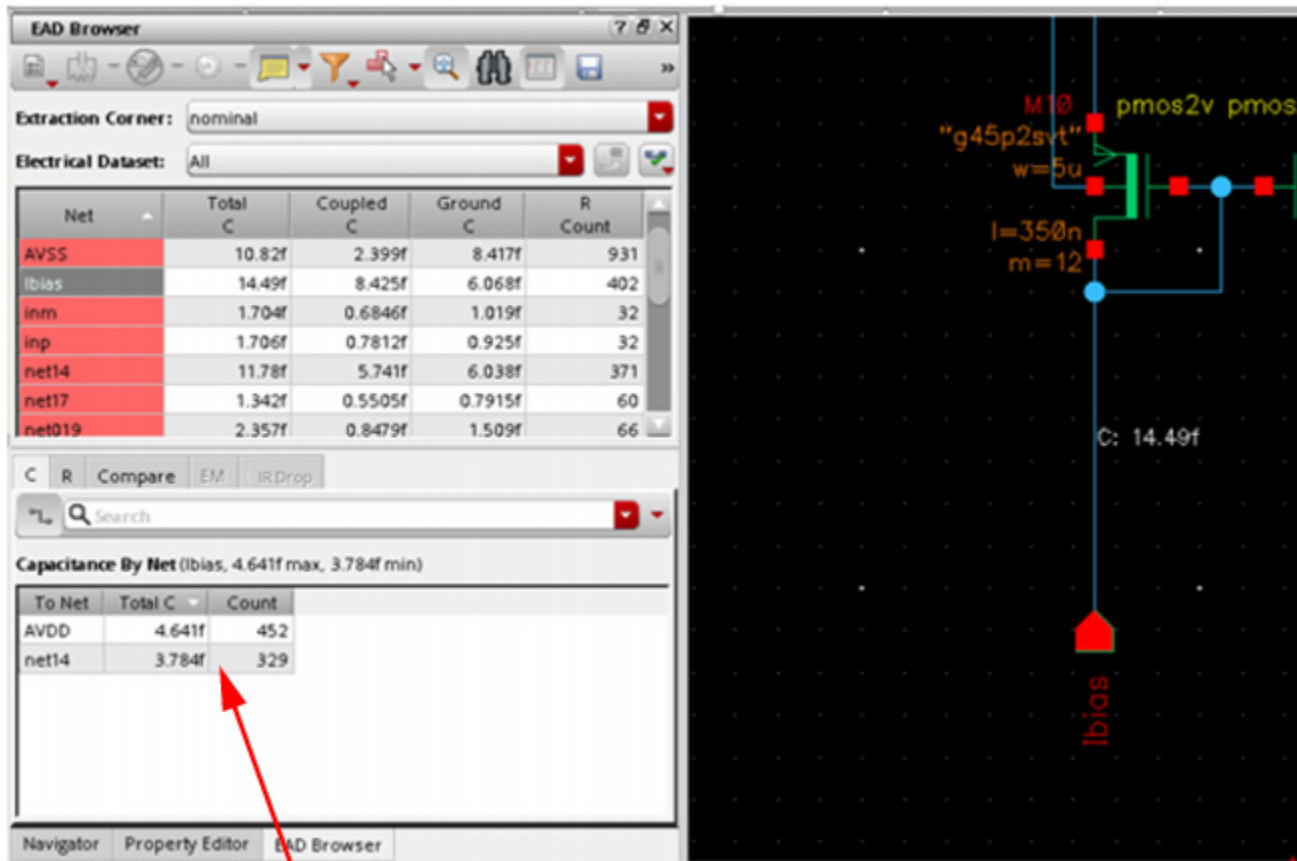
Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

■ Building Netlist to Run Simulations

Viewing Parasitics

Irrespective of the mode in which you access the layout view, details of the parasitics already available in the layout are displayed in the Summary pane. This includes the count of the capacitance and resistance for each net. When you click a net in the Summary pane, its details are displayed in the detail pane of the assistant, as shown below.



Capacitance from the selected net (Ibias)
to the other connected nets

Note: For a hierarchical design, the Summary pane lists all the nets available in the current hierarchy level or the levels below it.

By default, the *Capacitance By Layer* and *Capacitance by Node* tables are not visible on the C tab of the detail pane they require detailed knowledge of the layout topography which is not visible to the circuit view. You can display them by setting the `detailedParasiticTables` environment variable to `t` or by selecting the *Detailed Parasitic Tables* check box on the *Environment* tab of the EAD Options form.

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

By selecting a row in the detail pane, you can view how Rs and Cs are placed on the corresponding net in the circuit. Selecting a column for capacitance or resistance count opens the *C* tab or *R* tab, respectively, in the detail pane. Select rows in the detail pane to view the Rs or Cs. For example, the figure shown below shows the resistance path as a purple line for the selected net `net6`.

EAD Browser

Extraction Corner: nom

Electrical Dataset: All

Net	Total C	Coupled C	Ground C	R Count
inm	0	0	0	206
inp	0	0	0	206
net6	0	0	0	228
net9	0	0	0	126

Resistance Paths (net6, 6 paths, 74.05 max, 38.18 min)

Total R	Count	From	To
41.06	91	M1:D	M3:D
71.33	92	M1:D	M3:G
70.79	54	M1:D	M4:G
38.18	16	M3:D	M3:G
43.78	62	M3:D	
74.05	63	M3:G	

Resistance path

Shows the resistance path for this net selected in the detail pane

There can be multiple resistance paths in the layout between the two logical circuit points.

Probing Resistances for Nets, Terminals, or Instance Terminals Selected on Schematic

While you are working with the design schematic, you can interactively view resistance paths on a particular net or between selected terminals.

For selected probing, perform these steps:

1. To enable selected probing, choose *Filtering Options – Selected* on the EAD Browser toolbar.
2. To view the resistance paths on the *R* tab of the detail pane, select a net or terminal on the schematic as described below.

To view the resistance paths, do this:

- ☐ Select a net to view all the resistance paths from or to that net
- ☐ Select an instance terminal or a terminal to view all the resistance paths from or to that terminal on the connected net
- ☐ Select more than one terminal to view all the resistance paths between those two terminals

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

In the example shown below, the resistance paths on the selected net `net10` are displayed.

The screenshot shows the EAD Browser interface. The 'Extraction Corner' is set to 'nom' and the 'Electrical Dataset' is 'All'. The 'Show Only' dialog box is open, with the 'Selected' checkbox checked. A yellow callout '1' points to the 'Show Only' dialog box with the text 'Enable Selected probing'. A yellow callout '2' points to a net in the circuit diagram with the text 'Select a net'. A yellow callout '3' points to the 'Resistance Paths' table with the text 'Resistance paths on the selected net are displayed'.

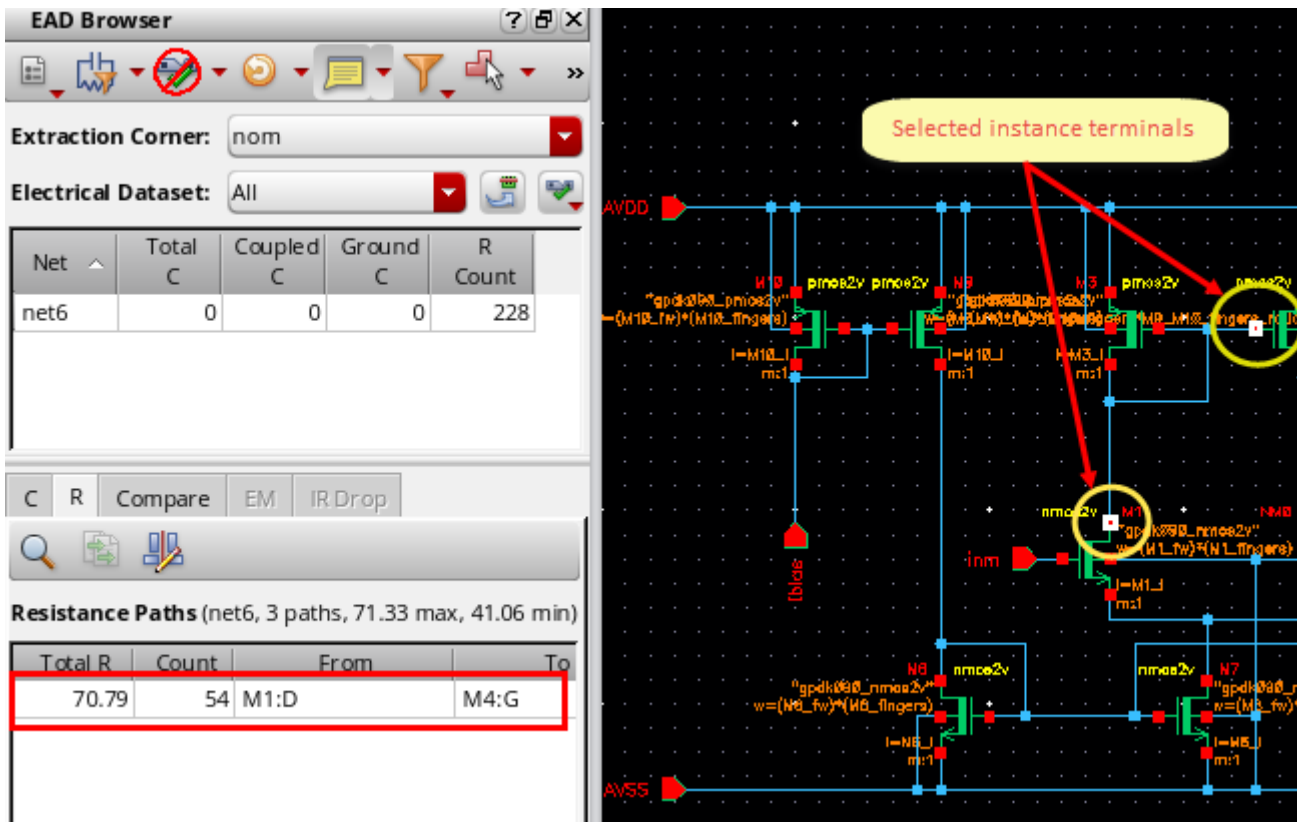
Resistance Paths (net10, 6 paths, 333.2 max, 16.6 min)

Total R	Count	From	To
16.6	16	NM0:D	M4:D
30.98	68	NM0:D	R0:PLUS
333.2	63	NM0:D	M5:G
30.75	76	M4:D	R0:PLUS
333	71	M4:D	M5:G
307.7	15	R0:PLUS	M5:G

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

Similarly, the following example shows the resistance path between two instance terminals.



Note: Resistance paths are displayed only when both the terminals of a net exist in the currently displayed schematic hierarchical level. If one terminal exists inside a subcell, EAD Browser does not display any resistance path.

For more details about the *Resistance Paths* table and the information contained in it, refer to [R tab](#).

Probing Capacitance for Nets Selected on Schematic

Similar to resistance, you can interactively view coupled capacitance values for one or more nets selected on the schematic.

For selected probing, perform these steps:

1. Choose *Filtering Options – Selected* on the EAD Browser toolbar.
2. On the schematic, select one or more nets, as described below:

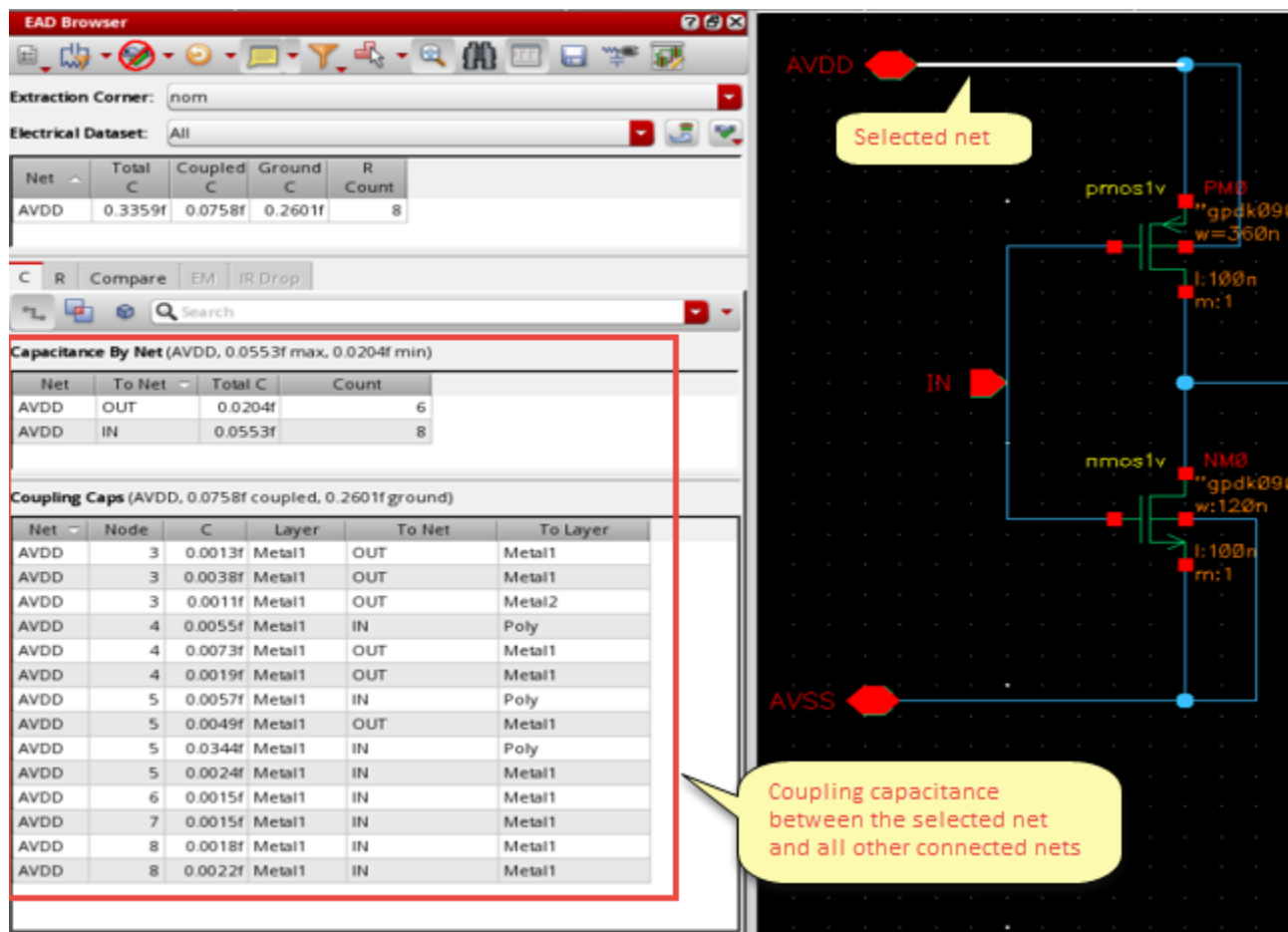
Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

- ❑ To view the coupling capacitance for a single net, select that net on the layout canvas
- ❑ To view the coupling capacitance between two selected nets, select those two nets
- ❑ To view the coupling capacitance between multiple nets, select all those nets

The coupling capacitance values corresponding to your selection are displayed on the C tab of the detail pane.

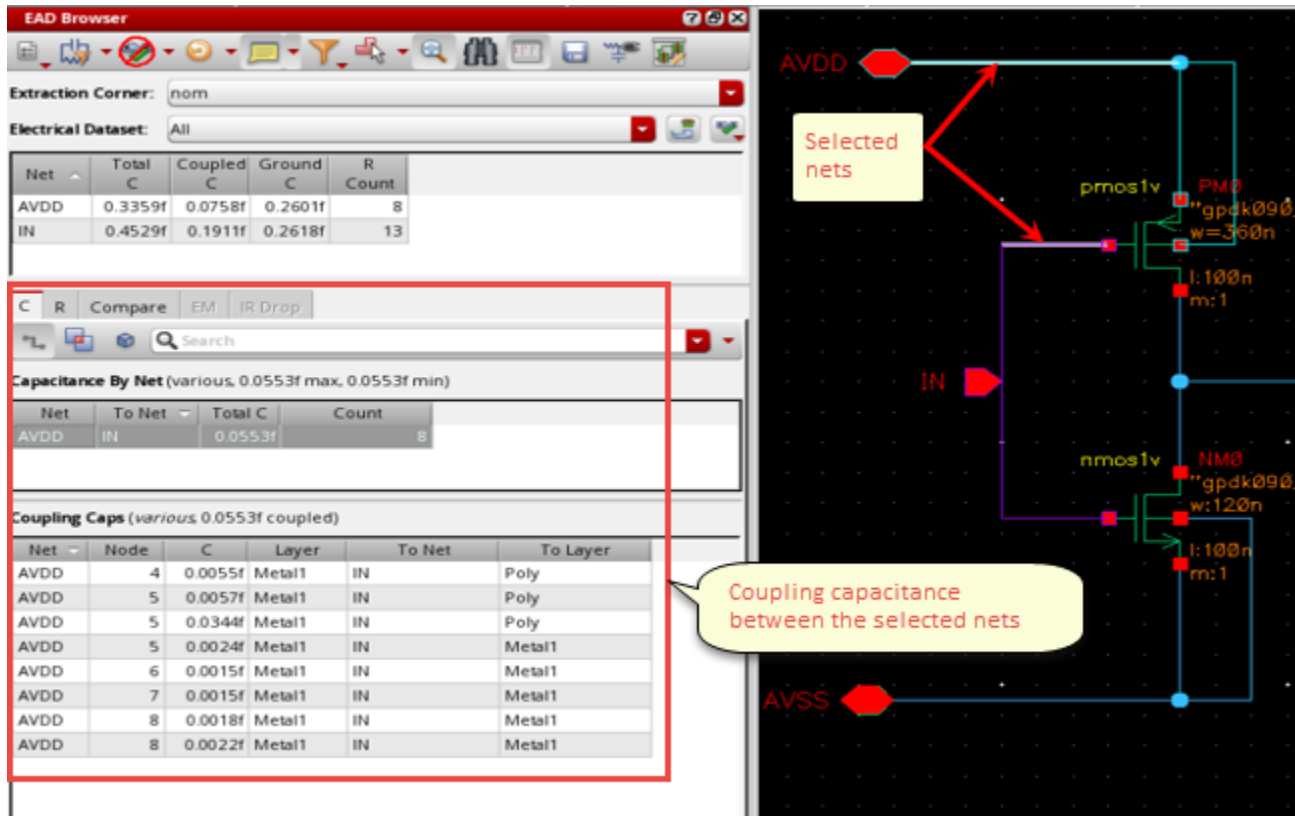
In the example shown below, the coupling capacitance between the selected net AVDD and all other nets connected to it, IN and OUT, is displayed.



Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

In the example shown below, the coupling capacitance between the selected nets AVDD and IN are displayed. The coupling capacitance between other nets is hidden.

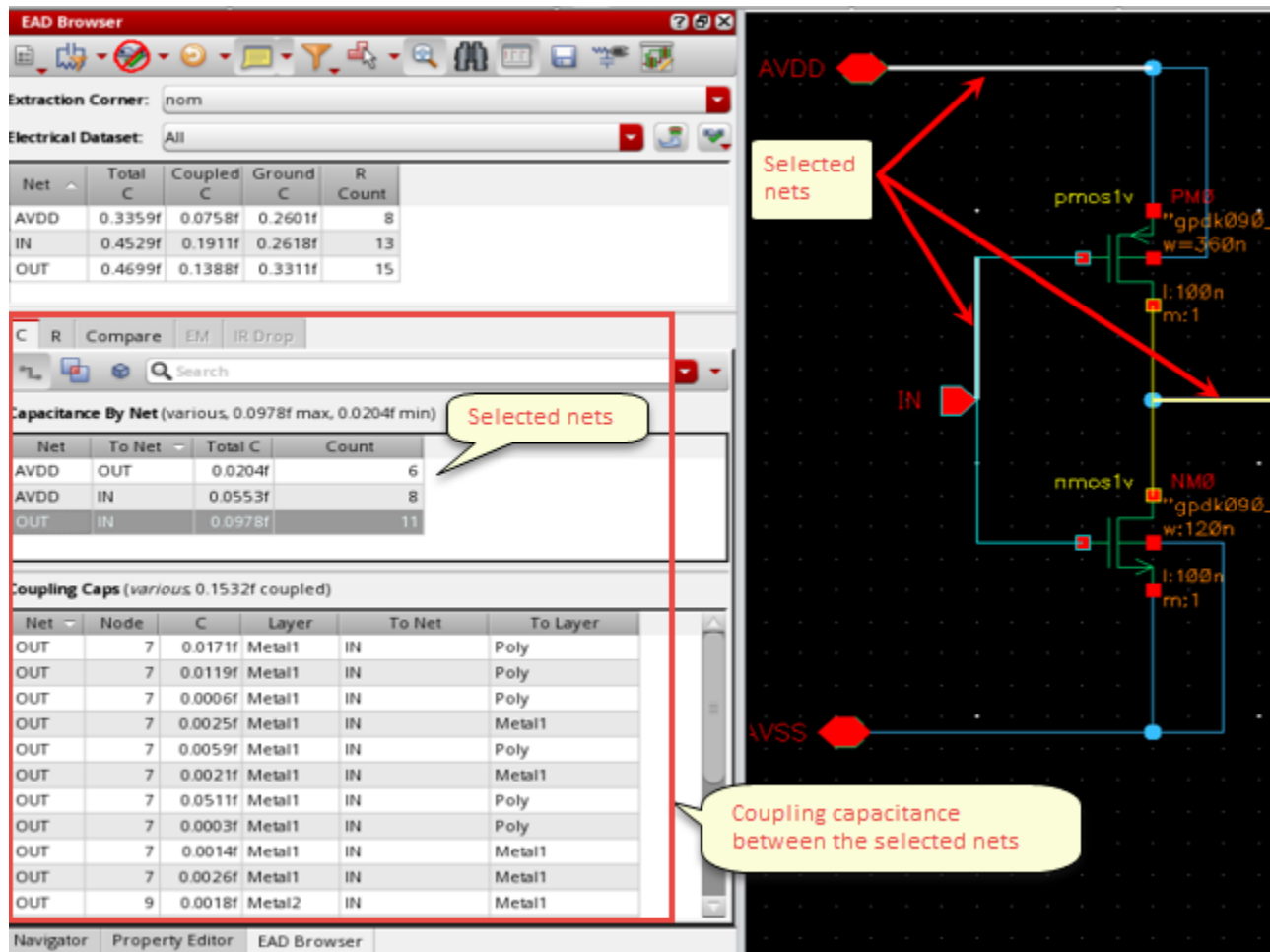


In the figure shown above, the *Net* and *To Net* columns in the *Coupling Caps* table show the from and to nets between which the coupling capacitance value is found.

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

Another example given below shows how the coupling capacitance is reported when multiple nets are selected.



Filtering the Coupling Caps Table by Layer

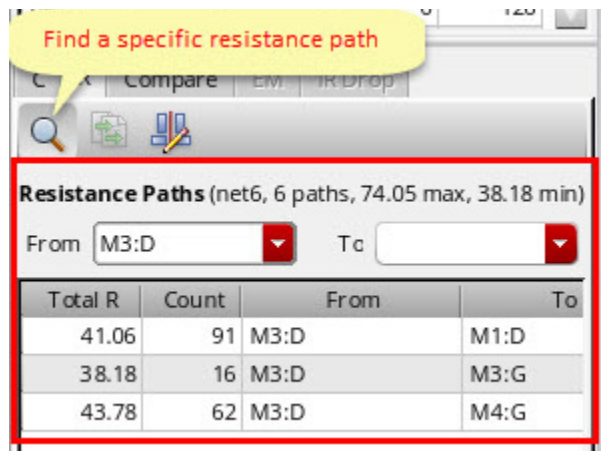
To filter entries in the *Coupling Caps* table:

- ➔ On the *C* tab of the *EAD Browser*, click  and select one of the following options to specify the layers you want to be displayed in the *Coupling Caps* table.
 - ☐ *Show All Layers*: Shows all layers.
 - ☐ *Hide All Layers*: Hides all layers.
 - ☐ *Show Only Via Layers*: Shows only via layers.
 - ☐ *Show Only Conductor Layers*: Shows only conductor layers.

- ❑ *Show Layers Above > Layer Name*: Shows only layers that are above the selected layer.

Finding a Specific Resistance Path

You can search for specific resistance paths between two instance terminals, or beginning from or ending at the selected instance terminal. For example, the figure given below shows the resistance paths from M3 : D.



Extracting Parasitics

Only when layout is accessed in edit or copy mode.

If the parasitics are not available in the layout view, you can extract them for all the nets by using the () command menu on the toolbar. Alternatively, you can extract parasitics for only a few selected nets by using the *Extract Parasitics* command from the context menu in the Summary pane. For more details, see [Extracting Parasitics in Layout EAD](#).

Viewing and Editing Constraints

In addition to the Constraint Manager assistant, you can directly view or edit the Max Resistance and Max C constraints for specific nets in the EAD Browser. For more details, refer to the following topics:

- [Viewing and Editing the Max Resistance Constraint in the EAD Browser](#)
- [Viewing and Editing the Max Capacitance Constraint in the EAD Browser](#)
- [Viewing and Editing the Max Coupling Capacitance Constraint in the EAD Browser](#)

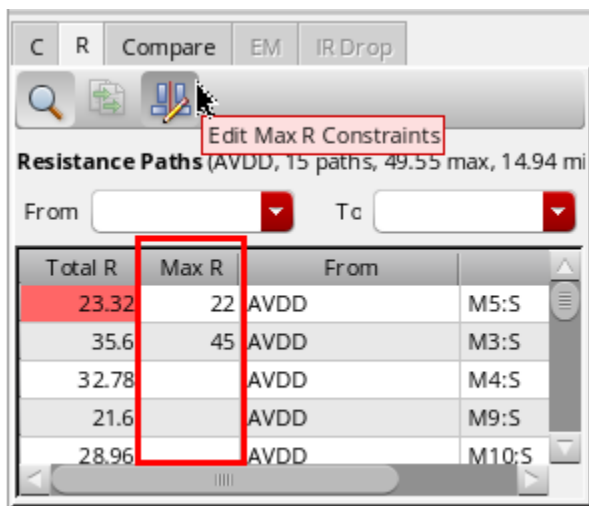
Viewing and Editing the Max Resistance Constraint in the EAD Browser

To edit the Max Resistance constraint for a net in the EAD Browser:

1. Click in the *R Count* column for a particular net on the Summary pane of the EAD Browser.

Alternatively, open the R tab in the detail pane, and then select the net on the schematic.

2. Click *Edit Max R Constraints* () on the toolbar of the R tab.
3. The *Count* column in the resistance paths table is changed to *Max R*. This column shows the existing constraint values, if defined in the constraint view. Otherwise, the cells are blank.



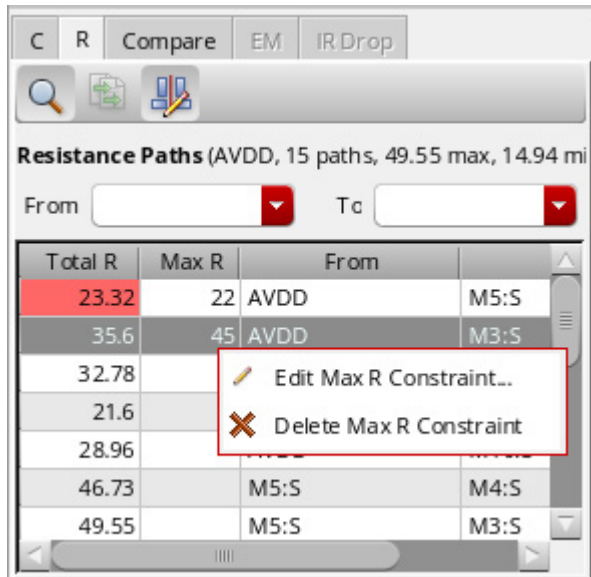
4. If required, double-click in the column and edit the constraint values.

Note: You can select and edit the Max R constraint for multiple nets together.

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

You can also use the *Edit Max R Constraint* and *Delete Max R Constraint* commands from the context-sensitive menu of the detail pane to edit or delete the constraints.



When you save the schematic view, the constraint values are also saved in the constraint view.

Viewing and Editing the Max Capacitance Constraint in the EAD Browser

To edit the Max Capacitance constraint for a net in the EAD Browser:

1. Right-click the row for a net on the Summary pane of the EAD Browser.
2. Choose *Constraints – Edit Max Capacitance Constraints* from the context menu.

The *EAD - Edit Max Capacitance Constraint* form is displayed, as shown below.

The screenshot shows the 'EAD - Edit Max Capacitance Constraint' dialog box. It has three input fields: 'Total Capacitance' with a value of '5f', 'Coupled Capacitance' with a value of '0', and 'Ground Capacitance' with a value of '0'. At the bottom, there are 'OK' and 'Cancel' buttons.

The default constraint values are displayed in the form.

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

3. Edit the required constraint values and click *OK*.

The constraint is added to the selected net and is visible in the Constraint Manager assistant, as shown in the example given below.

The screenshot displays the EAD Browser and Constraint Manager windows. The EAD Browser shows a table of parasitic extraction results for various nets. The 'net05' row is highlighted in red, indicating a violation. The Constraint Manager window shows the 'Max Capacitance (1)' constraint for 'net05' being edited.

Net	Total C	Coupled C	Ground C
AVDD	32.58f	3.208f	29.37f
AVSS	33.02f	4.638f	28.38f
IN	1.223f	0.2018f	1.021f
OUT	6.625f	3.407f	3.218f
net05	6.174f	2.34f	3.833f
net10	1.978f	0.389f	1.589f

The Constraint Manager window shows the following constraints:

- Max Capacitance (1) 5f 2f 0
- Max Capacitance (1) 5f 2.5f 0
- Max Capacitance (1) 5f 2.5f 0
- Max Resistance (2) 500m

The 'net05' constraint is highlighted in red. The 'Constraint (1)' table shows the following details:

Name	Constr_2
Owner	INV_HIER.INVX1_TOP:constraint
Enabled	true
Notes	
Total Capaci...	5f
Coupled Cap...	2.5f
Ground Cap...	0

A check for any capacitance values is also done and the violations are highlighted in red.

Note: If you need to apply the same constraint value to multiple nets, select the nets in the EAD Browser and use the *Edit Max Capacitance Constraints* command to create or edit the constraint.

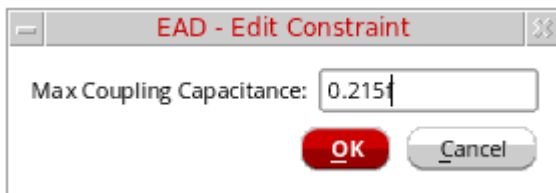
You can also use the *Delete Max Capacitance Constraint* command from the context-sensitive menu of the selected net on the Summary pane to delete the constraint.

Viewing and Editing the Max Coupling Capacitance Constraint in the EAD Browser

To edit the Max Coupling Capacitance constraint between two nets in the EAD Browser:

1. Select two nets in the Summary pane of the EAD Browser.
2. Right-click and choose *Constraints – Edit Max Coupling Capacitance Constraints* from the context menu.

The *EAD - Edit Constraint* form is displayed, as shown below.

The image shows a dialog box titled "EAD - Edit Constraint". Inside the dialog, there is a label "Max Coupling Capacitance:" followed by a text input field containing the value "0.215". Below the input field are two buttons: a red "OK" button and a grey "Cancel" button.

3. Specify a value in the *Max Coupling Capacitance* field.
4. Click *OK*.

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

The constraint is added to the selected nets and is visible in the Constraint Manager assistant, as shown in the example given below.

The screenshot displays the EAD Browser and Constraint Manager windows. The EAD Browser shows a table of capacitance values for various nets. The Constraint Manager shows a list of constraints, with a new 'Max Coupling Capacitance (2)' constraint added to nets 'inm' and 'net17'.

EAD Browser - Capacitance By Net

Net	Total C	Coupled C	Ground C	R Count
AVSS	10.43f	2.404f	8.029f	83
Ibias	13.87f	7.64f	6.232f	402
inm	1.756f	0.7864f	0.9696f	32
inp	2.223f	1.034f	1.19f	38
net14	15.34f	7.824f	7.52f	276
net17	1.391f	0.6527f	0.7383f	36
net019	2.31f	0.9078f	1.402f	39

Capacitance By Net (inm, 0.3544f max, 0.009f min)

To Net	Total C	Max C	Count
net17	0.2163f	215a	
outm	0.3544f	2f	

Coupling Caps (inm, 0.7864f Coupled, 0.9696f ground)

C	Layer	To Net	To Layer
0.162f	Metal1	outm	Metal3

Constraint Manager

Editing: constraint

Show: All

Type (38)

- Max Resistance (2)
- Max Capacitance (1)
- Max Resistance (1)
- Alignment (2)
- Max Coupling Capacitance (2)
- Max Capacitance (1)
- Max Coupling Capacitance (2)
- inm
- net17

Constraint (1) All Parameters

Name	Constr_94
Owner	Two_Stage_O
Enabled	true
Notes	
Coupled Capacitance	215a

A check for any coupling capacitance values is also done, and the violations are highlighted in red.

Other ways to edit the Max Coupling Capacitance constraint are as:

- ❑ Double-click a cell in the *Max C* column of the *Capacitance By Net* table in the detail pane and edit the value. If you remove the value, the constraint is deleted from Constraint Manager.

Note: By default, the *Max C* column is hidden if the Max Coupling Constraint is not defined for any net. To view this column, right-click in the column header of this table and choose *Max C*.

Virtuoso Electrically Aware Design Flow Guide

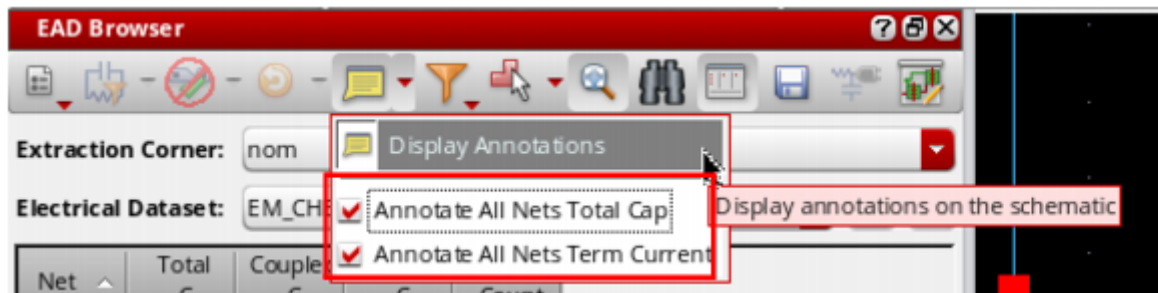
Resimulating Designs with Extracted Parasitics

- ❑ Right-click a net in the *Capacitance By Net* table in the detail pane of the EAD Browser, and to choose the *Edit Max Coupling Capacitance Constraints* command.

If a constraint is not required for any net, right-click the net in the summary pane of the EAD Browser and choose *Constraints – Delete Max Coupling Capacitance Constraints* from the context menu.

Annotating Parasitics on the Schematic

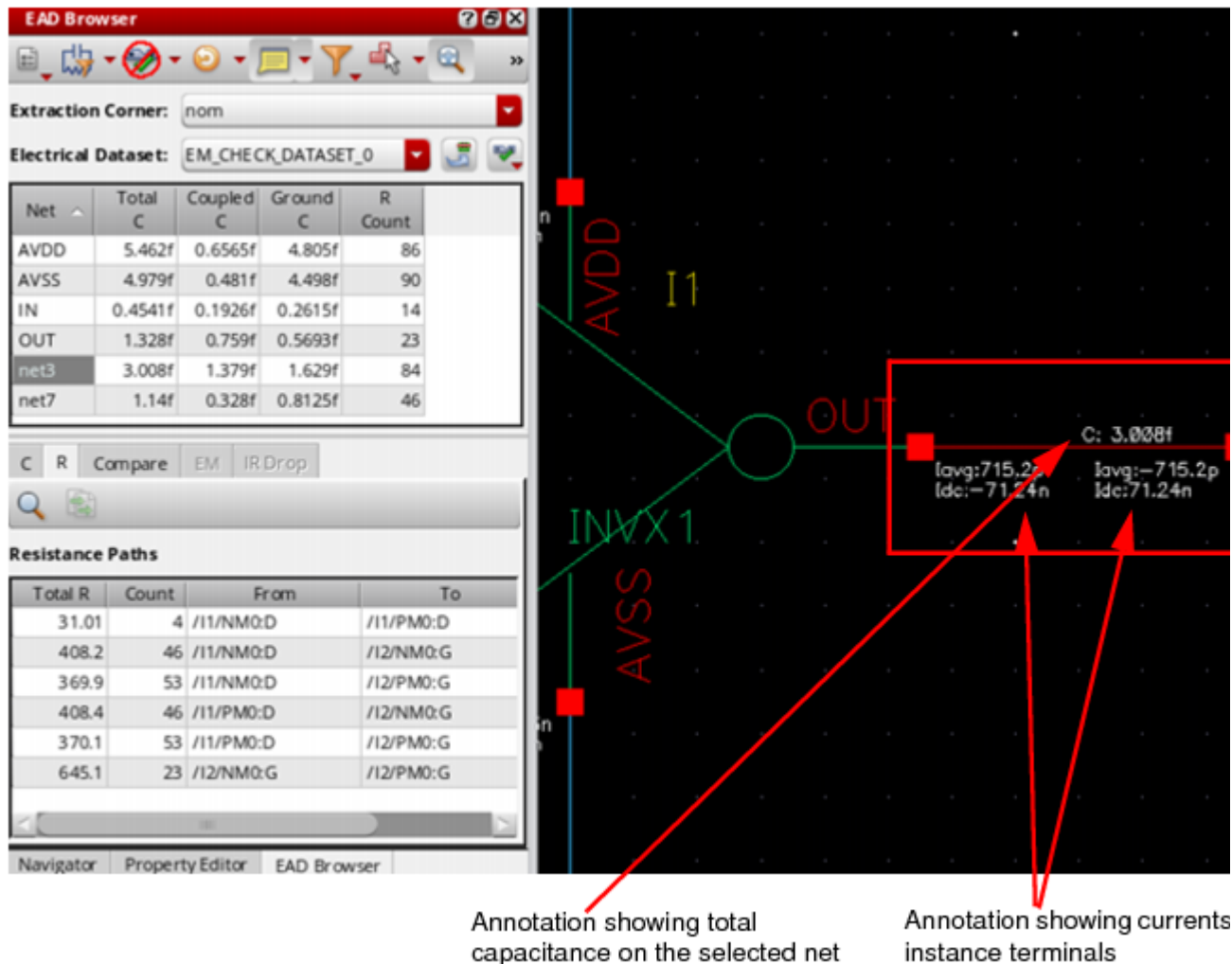
By default, the display of annotations on the schematic is enabled. Select a net in the EAD Browser and use the commands shown below to view annotations.



Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

When you select both the check boxes, the annotations for total capacitance and terminal current values are displayed on the selected net, as shown below.



Modifying Parasitics on the Schematic

Only when layout is accessed in the edit mode.

If any specifications were not met during the resimulation flow, you can check the parasitics of the nets whose results are being used in calculations. If required, you can modify parasitics

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

or add appropriate constraints. By doing this, you can provide suggestions to the layout engineer on a net-by-net basis.

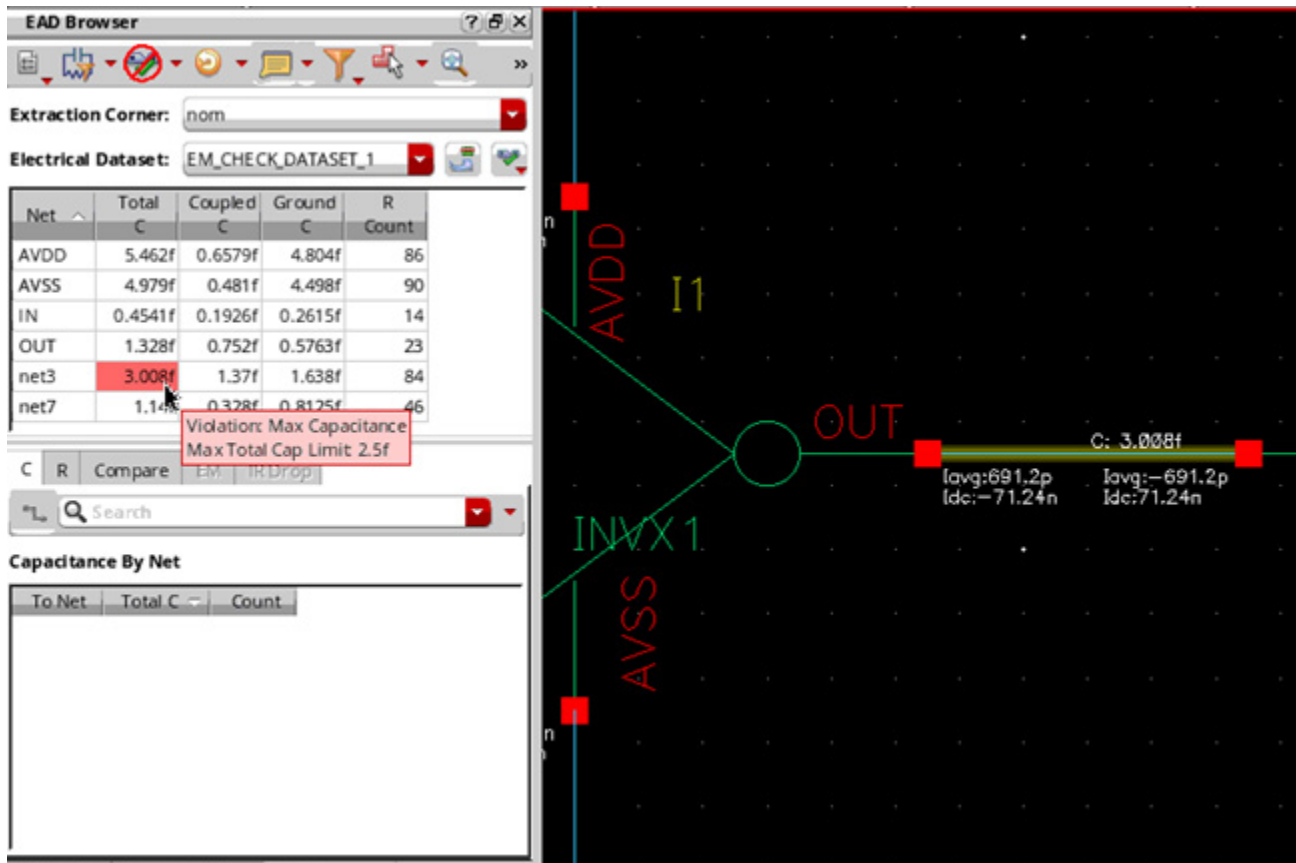


The effect of updated parasitics or constraints can be seen in the EAD Browser. For example, after you add or update the Max Capacitance constraint on a net, if the total capacitance

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

value for the net is more than the constraint value, it is highlighted as a violation, as shown below.



The layout engineers can update the layout view to view the constraints and the new violations reported due to those constraints. They can then address the same by making appropriate changes in the layout view.

Building Netlist to Run Simulations

Only when layout is accessed in the edit or copy mode.

If you updated the parasitics in the schematic view, you can verify that the modifications can help in meeting the specifications. For this, you can use the build a netlist view using the *Build Netlist View* the () command on the EAD Browser toolbar. The command opens the Build Parasitic/LDE Netlist View for <schematic> view form in which you can specify the name for the netlist view and other options for parasitic inclusion.

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

A netlist is created using the current schematic view as base. You can use this netlist view to run simulations in ADE Assembler and verify results.

Annotating Parasitics from Schematic EAD Browser

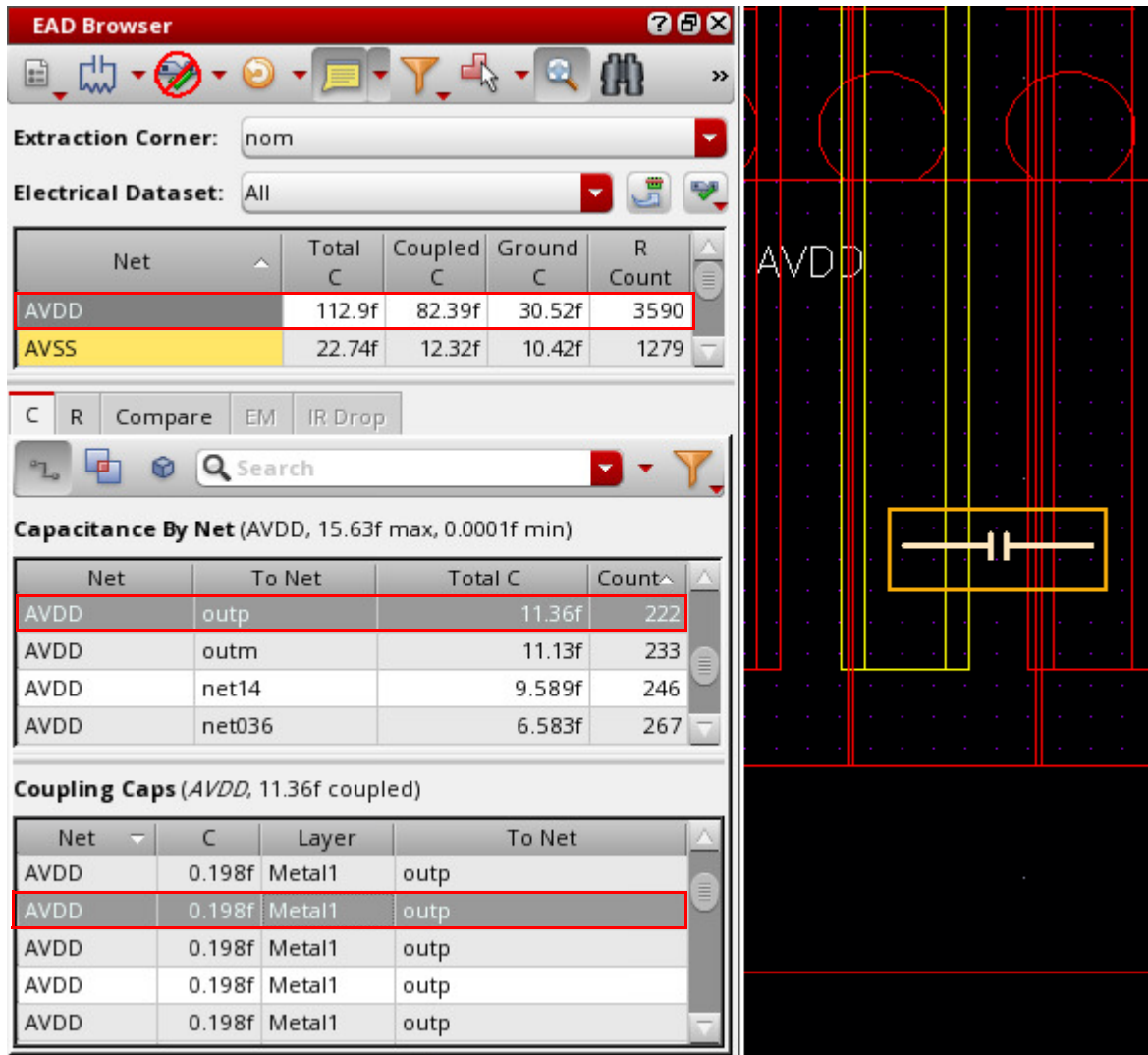
To annotate parasitics in Layout from the EAD Browser in the Schematic.

1. From the toolbar of the *EAD Browser* in Virtuoso Schematic Editor, click the *Open Layout*  icon.
Virtuoso Layout Suite opens.
2. Select a net in the EAD Browser of the Schematic.
3. Click either the *C* tab or the *R* tab and click an entry either in the *Coupling Caps* table or in the *Resistance Paths* table that you want to annotate in the layout, respectively.

Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

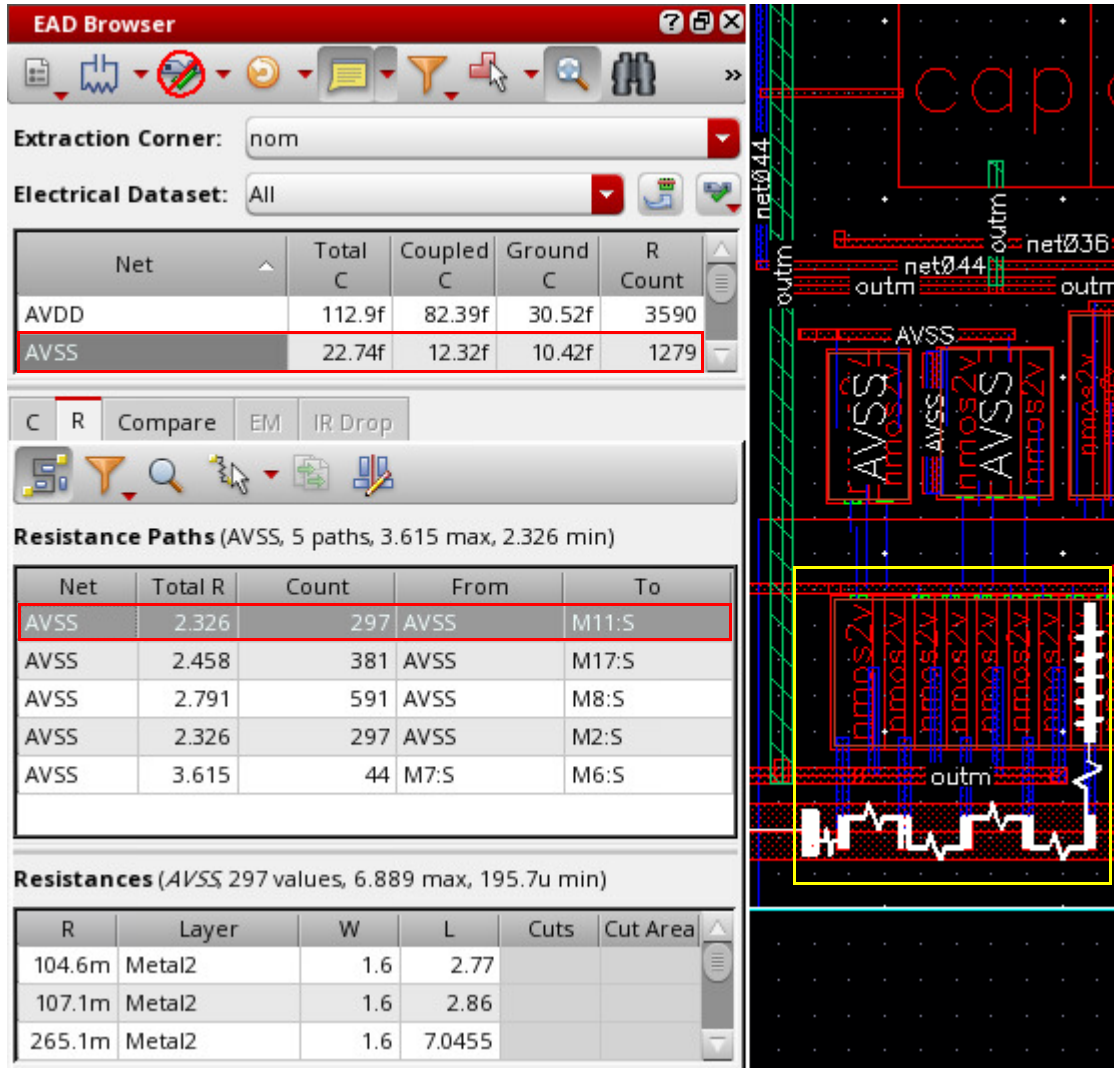
The following example shows the coupling capacitance between the nets `AVDD` and `outp` highlighted in the Layout.



Virtuoso Electrically Aware Design Flow Guide

Resimulating Designs with Extracted Parasitics

The following example shows the resistance path annotated in the layout for the selected net in the EAD Browser of the Schematic.



Quantus-Pegasus Integration for Partial Layouts

By combining parasitics extraction and physical verification sign-off capabilities of Cadence® Quantus™ Extraction Solution and Cadence® Pegasus™ Verification System, respectively, Virtuoso Layout Suite EXL provides sign-off level accuracy in EAD.

This flow requires all devices to be placed in the layout; the position of these devices may however not be the final. This flow enables the user to specify a reference net, known as the ground net, before extraction rather than before simulation.

Note: Similar to native EAD extraction, this flow also supports jobs for distributed processing.

Related Topics

[Creating Pegasus/Quantus Setup for Parasitics Extraction](#)

[EAD Pegasus/Quantus Create Setup File Dialog Form](#)

[Creating Process and Setup Files Manually](#)

[Extracting Parasitics from a Partial Layout](#)

[Viewing and Analyzing Parasitics](#)

[Extracting LDE Parameters](#)

[Resimulating Designs with Extracted Parasitics](#)

[Using Distributed Processing Jobs](#)

Creating Pegasus/Quantus Setup for Parasitics Extraction

Before creating the Pegasus/Quantus setup for parasitics extraction, you must have the following:

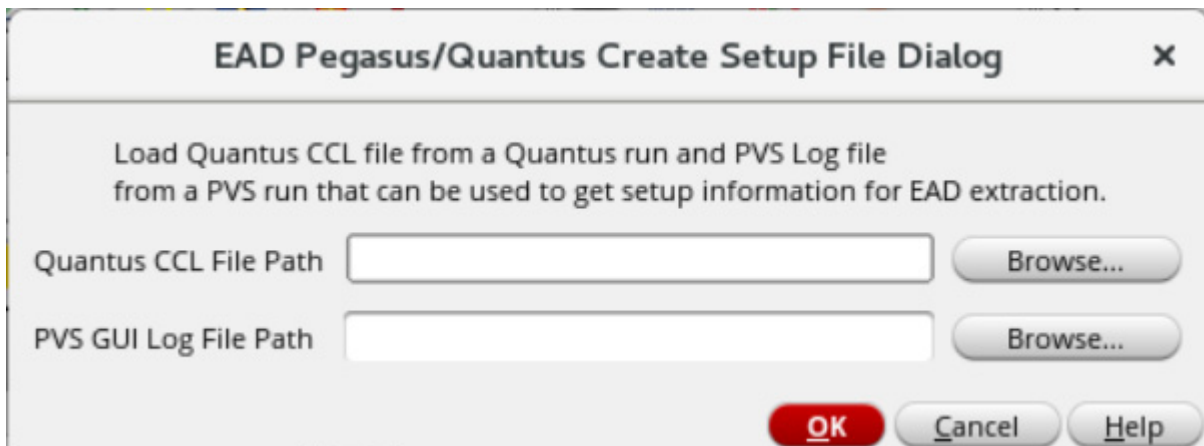
- LVS rules file with the LVS settings for the reference design
- Applicable Pegasus license
- Applicable Quantus license
- Quantus CCL file with the correct settings for the reference design

Note: Refer to README file for compatible Quantus and Pegasus versions.

To create Pegasus/Quantus setup for parasitics extraction:

1. Open EAD Browser from either Virtuoso Layout Suite EXL or MXL.
2. From the toolbar of EAD Browser, click  and select *Create Pegasus-Quantus Setup*.

The EAD Pegasus-Quantus Create Setup File Dialog form opens.



3. In the *Quantus CCL File Path* field, type the file path of the Quantus CCL file. You can also click *Browse* to select the Quantus CCL file.
4. In the *PVS GUI Log File Path* field, type the file path of the PVS GUI log file. You can also click *Browse* to select the PVS GUI log file.
5. Click *OK*.

Virtuoso Electrically Aware Design Flow Guide

Quantus-Pegasus Integration for Partial Layouts

The setup file and process files are saved to the following locations inside your current working directory, respectively:

- `.cadence/dfII/EAD/1/setup/setup.ini`
- `.cadence/dfII/EAD/1/process/process.ini`

After you load the EAD Pegasus/Quantus Setup for extraction, the following options become disabled:

- The *EM* tab on the EAD Option form.
- The *EM* and *IR Drop* tabs on the EAD Browser.

Related Topics

[EAD Pegasus/Quantus Create Setup File Dialog Form](#)

[Extracting Parasitics from a Partial Layout](#)

[Creating Process and Setup Files Manually](#)

[maxResistorLength](#)

Creating Process and Setup Files Manually

You can also create the `process.ini` and `setup.ini` files manually.

To create the `process.ini` file manually:

1. Open a new file using any text editor.
2. Type the following in the file:

```
[corners]
technologyLibraryFile= /<path_to_the_directory>/pvtech.lib
1\technologyName=<name_of_technology_file>
1\name=nom
size=1
```

3. Save the file at the following location inside the current working directory:

`.cadence/dfII/EAD/1/setup/setup.ini`

To create the `setup.ini` file manually:

1. Open a new file using any text editor.

2. Type the following in the file:

```
[connectivity]
pvsRulesFile=/<path_to_the_directory>/pvs.rul
[extraction]
processSettings=./.cadence/dfII/EAD/1/process/process.ini
```

3. Save the file at the following location inside the current working directory:

```
.cadence/dfII/EAD/1/setup/setup.ini
```

Related Topics

[EAD Pegasus/Quantus Create Setup File Dialog Form](#)

[Extracting Parasitics from a Partial Layout](#)

[Viewing and Analyzing Parasitics](#)

Extracting Parasitics from a Partial Layout

With the in-design integration of Quantus, Virtuoso Layout EXL provides ability to extract LDE parameters and RC parasitics from a partial layout. This enables quicker feedback loop to run layout-aware re-simulation, and tuning the circuit performance early without waiting for a complete LVS -lean layout.

To extract parasitics from a partial layout:

1. Create the Pegasus-Quantus Setup using the EAD Pegasus/Quantus Create Setup Dialog form.

2. From the toolbar of EAD Browser, click  and select the Pegasus-Quantus setup.

All available nets from the setup are loaded in the EAD Browser.

3. In the *Ground Net* field, specify a ground net. Alternatively, right-click a net in the EAD browser table and then *select Set As Ground Net*.

You can also keep the default selection 0, which indicates that extraction are run with respect to an imaginary ground net.

4. From the toolbar of EAD Browser, click  and select *Extract Parasitics For All Nets*.

The extraction process is completed in the following four phases.

- ☐ Preparing design for EAD

- ❑ Running job: Connectivity Extraction
- ❑ Running job: LVS
- ❑ Running job: Parasitic Extraction

Note: Additional nets may appear in the browser derived from the connectivity analysis engine.

5. After the extraction is complete, extraction results are displayed in the EAD Browser.

Related Topics

[EAD Pegasus/Quantus Create Setup File Dialog Form](#)

[Creating Pegasus/Quantus Setup for Parasitics Extraction](#)

[Extracting LDE Parameters](#)

[maxResistorLength](#)

[Viewing and Analyzing Parasitics](#)

Extracting LDE Parameters

In addition to parasitics extraction, you can also choose to extract only LDE (layout-dependent effects) parameters.

The extraction of LDE parameters is carried out in following phases:

- Pegasus is run with either a Pegasus Interactive or a sign-off license on the layout view and extracts all the devices and their device parameters.
- Quantus is run with either a Quantus Interactive or a sign-off license, keeping RC extraction turned off, to generate a SmartView. This SmartView consists only layout-extracted devices and their parameters.
- The SmartView is then used for simulation.

To extract LDE parameters:

1. Create the Pegasus-Quantus Setup using the EAD Pegasus/Quantus Create Setup Dialog form.
2. From the toolbar of EAD Browser, click  and select the Pegasus-Quantus setup.

All available nets from the setup are loaded in the EAD Browser.

3. In the *Ground Net* field, specify a ground net. Alternatively, right-click a net in the EAD browser table and then *select Set As Ground Net*.

You can also keep the default selection 0, which indicates that extraction are run with respect to an imaginary ground net.

4. From the toolbar of EAD Browser, click  and select *Extract LDE parameters only (no parasitics)*.

The extraction process is completed in the following three phases.

- ☐ Preparing design for EAD
- ☐ Running job: Connectivity Extraction
- ☐ Running job: Parasitic Extraction

Related Topics

[Creating Pegasus/Quantus Setup for Parasitics Extraction](#)

[EAD Pegasus/Quantus Create Setup File Dialog Form](#)

[Extracting Parasitics from a Partial Layout](#)

[Viewing and Analyzing Parasitics](#)

[Resimulating Designs with Extracted Parasitics](#)

Viewing Log Files

You can view the CDL netlist log file, LVS log file, and extraction log files.

To view these files:

- ➔ From the toolbar of EAD Browser, click  and select either one of the following options:
 - ☐ View CDL Netlist Log File
 - ☐ View LVS Log File
 - ☐ View Extraction Log File

For more information about these files, refer to *Quantus Extraction Users Manual*.

Virtuoso Electrically Aware Design Flow Guide

Quantus-Pegasus Integration for Partial Layouts

Related Topics

[Creating Pegasus/Quantus Setup for Parasitics Extraction](#)

[EAD Pegasus/Quantus Create Setup File Dialog Form](#)

[Extracting Parasitics from a Partial Layout](#)

[Extracting LDE Parameters](#)

Virtuoso Electrically Aware Design Flow Guide

Quantus-Pegasus Integration for Partial Layouts

Form Descriptions

This appendix describes the field descriptions of the following forms:

- [EAD Setup](#)
- [EAD Options](#)
- [EAD Pegasus/Quantus Create Setup File Dialog Form](#)
- [EAD Process Settings](#)
- [EAD Job Policy Editor](#)
- [EAD Net Options](#)
- [Parasitics & LDE Setup](#)
- [P2T EM Optimization](#)
- [The Pre-EM Setup Form](#)

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

EAD Setup

The following table describes the fields that are available on the EAD Setup form.

If you are viewing this form in smaller screens, a scroll bar appears on this form.

Form Field	Specifies...
<i>Enable Electrical Data Capture for EAD Flow</i>	If the EAD mode is to be enabled or disabled for the selected design under test.
Design Selection	
<i>Design Under Test</i>	The library name and cell name of the design for which electromigration analysis is required to be done.
Save Options	
<i>Signal Selection</i>	
<i>All Signals</i>	The current data is to be saved for all the signals in the specified DUT.
<i>Hierarchy Stop Level</i>	The design hierarchy level up to which the current data for all the signals is to be saved.
<i>Selected Signals</i>	The current data is to be saved for the selected signals only. You can select the signals using the Parasitic and Electrical Setup assistant.
Electromigration Checking	
<i>Type</i>	
<i>DC Current (Idc)</i>	The direct current data is to be saved.
<i>Average Current (Iavg)</i>	The average current data is to be saved.
<i>Scale</i>	A multiplier by which the direct and average current data is scaled.
<i>Waveforms for RMS and Peak Checks (Isignal)</i>	Directs to save the current waveforms with the simulation data to disk so that it can be processed later while performing RMS and Peak electromigration checks.

Voltage Dependent Rules

Type

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field	Specifies...
<i>Min and Max Voltage (Vmin/Vmax)</i>	That minimum and maximum voltages are to be captured during the simulation run for use in the voltage dependent rules flow.
<i>Scale</i>	A multiplier by which the voltage data is scaled before it is transferred to the layout view. The default is 1.00, which means that voltage data is not scaled.
<i>Create voltage constraints</i>	Whether to create post-simulation voltage constraints. Environment variable: <u>createVoltageConstraints</u>
Waveform Processing Options	
<i>Process Waveforms Inside Simulator</i>	Directs to process the current waveforms during simulation and save only the calculated current data to the disk.
<i>Process Waveforms Post Simulation</i>	Directs to process the current waveforms after simulation
<i>Clip Waveforms</i>	The saved waveform needs to be clipped for the given time interval.
<i>From</i>	The start time of the time interval for which the waveform of the current data is to be saved. You can also use the VAR expression in this field. For example, VAR("variable_name"). For more information, refer <u>Virtuoso ADE Assembler User Guide</u> .
<i>To</i>	The end time of the time interval for which the waveform of the current data is to be saved. You can also use the VAR expression in this field. For example, VAR("variable_name"). For more information, refer <u>Virtuoso ADE Assembler User Guide</u> .

EAD Options

The EAD Options form in Virtuoso Layout EAD is split into four tabs:

- General
- Extraction
- EM
- Environment
- Comments

The *Setups* group box is available on every tab:

- EAD Setup Options

If you are viewing this form in smaller screens, a scroll bar appears on this form.

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

General

The screenshot shows the 'EAD Options' dialog box with the 'General' tab selected. The dialog has five tabs: General, Extraction, EM, Environment, and Comments. The 'Process' section includes 'Process Settings' set to 'gpd090_dyn' and 'Temperature' set to '25 °C'. The 'Network' section has an 'Exclude Nets' group with 'Excluded Nets' and 'Except Nets' fields, each with a 'Select' button. Below this is a checkbox for 'Exclude Incomplete Nets'. The 'Primitives & Excluded Cells' section contains a table with columns 'Lib', 'Cell', and 'View'. The table is currently empty. Below the table are 'Add' and 'Delete' buttons. The 'Setups' section at the bottom has a 'Location' field set to './cadence/dfl/EAD/1/setup', a 'Name' field set to 'gpd090_dyn', and buttons for 'Save To', 'Load From', and 'Delete'. At the very bottom are buttons for 'OK', 'Cancel', 'Defaults', 'Apply', and 'Help'.

EAD Options

General Extraction EM Environment Comments

Process

Process Settings: gpd090_dyn

Temperature: 25 °C

Network

Exclude Nets

Excluded Nets: Select

Except Nets: Select

Exclude Incomplete Nets: ☐

Primitives & Excluded Cells

Lib	Cell	View

Add Delete

Setups

Location: ./cadence/dfl/EAD/1/setup

Name: gpd090_dyn

Save To Load From Delete

OK Cancel Defaults Apply Help

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field

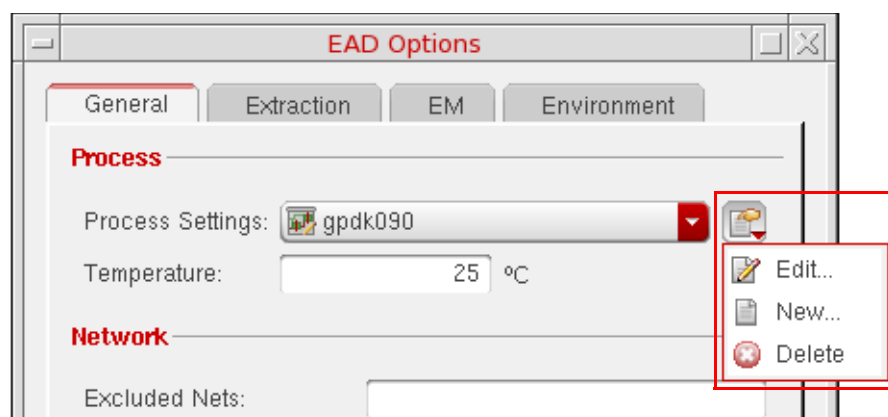
Specifies...

Process

Process Settings

The process settings file to be used in the current session.

The *Display process settings menu* button lets you add, change, or remove process settings files.



Environment variable: processSettings

See Editing Process Settings for more information.

Temperature

The extraction temperature in degrees Celsius.

Environment variable: temperature

Network

Excluded Nets

The nets to be excluded from parasitic extraction. Do one of the following:

- Type in a list of net names (complete, incomplete, power, etc) you do not want to extract. Separate the net names with a space or a comma. You can use wildcards to specify multiple net names.
- Click *Select* to select nets directly in the layout or schematic canvas.

Environment variable: excludedNets

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field	Specifies...
<i>Except Nets</i>	The nets that are exceptions to the list of nets specified in the <i>Excluded Nets</i> field. The nets specified in this field are extracted or displayed. Environment variable: excludedNetsExceptions
<i>Exclude Incomplete Nets</i>	That nets with incomplete routing or connectivity are to be excluded from extraction. Use this to skip extraction for <i>all</i> incomplete nets. Environment variable: includeSkippedNets

Primitives & Excluded Cells

Lib
Cell
View

The cellviews to be excluded from parasitic extraction. Parasitics are extracted up to and including the terminals of the excluded cells, but no extraction is performed inside the specified devices.

Click *Add* to add a row to the table, and type the required library, cell, and view names into the fields provided (or choose the names from the drop-down lists).

If you do not specify a cell or view name, then all cells or views are excluded. For example, the table below specifies the following exclusions:

- Cellview `Two_Stage_Opamp/OpAmp/layout_new`
- All views of cell `inv_2x_hv` in the `ether` library
- All the cellviews in the `em_test` library

Primitives & Excluded Cells

Lib	Cell	View
Two_Stage_Opamp	OpAmp	layout_new
ether	inv_2x_hv	
em_test		

Environment variable: [extractionPrimitives](#)

Setups

Location
Name

See [EAD Setup Options](#)

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Extraction

The screenshot shows the 'EAD Options' dialog box with the 'Extraction' tab selected. The dialog is divided into several sections: 'Engine Controls', 'Thresholds', 'High Precision C Solver', 'High Precision R Solver', and 'Setups'. Each section contains various configuration options with dropdown menus, checkboxes, and text input fields.

Engine Controls

- Extraction Mode: RC (dropdown), coupled (dropdown)
- Hierarchy Levels: 99 (spin box)
- Net Levels: 99 (spin box)
- Inst Term Hierarchy Levels: 99 (spin box)
- Erosion Effects: none (dropdown)
- Stop at Device Boundary: ☐
- Threads: 1 (spin box)

Thresholds

- Max Resistor Length: 100 (spin box) microns
- Minimum Coupling Cap: 0 (spin box) fF
- Resistance Scaling Factor: 1 (spin box)
- Gnd Cap Scaling Factor: 1 (spin box)
- Coupling Cap Scaling Factor: 1 (spin box)

High Precision C Solver

- Use for All Nets: ☐
- Default Convergence Target: 5.0 (spin box) %
- Metal Fill Modeling Method: none (dropdown)
- Via Effects: ☐
- Threads Per Net: 1 (spin box)

High Precision R Solver

- Use for All Nets: ☐
- Default Mesh Density: 1.000 (spin box)
- Threads Per Net: auto (dropdown)

Setups

- Location: J:\cadence\ /setup (text field with dropdown arrow)
- Name: (text field with dropdown arrow)
- Buttons: Save To, Load From, Delete
- Bottom Buttons: OK, Cancel, Defaults, Apply, Help

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field	Specifies...
Engine Controls	
<i>Extraction Mode</i>	The types of parasitic networks produced by the EAD extractor. Choose the mode (<i>RC</i>, <i>C</i>, or <i>R</i>) from the first pull-down and, where appropriate, the capacitance coupling type (<i>coupled</i>, <i>decoupled</i>, or <i>lumped coupled</i>) from the second pull-down. The default is <i>RC coupled</i>. Environment variable: <u>type</u>
<i>Hierarchy Levels</i>	Specifies the layout hierarchy depth for RC extraction. For example, if you type in 0, only top-level routing is extracted; if you type in 99 all hierarchy levels are extracted (nets that are wholly contained in lower levels are also extracted and reported). Environment variable: <u>hierarchyLevels</u>
<i>Net Levels</i>	Specifies the layout hierarchy depth for the lookup of nets for RC extraction. Value for this field should be less than or equal to the value of <i>Hierarchy Levels</i>. Environment variable: <u>netHierarchyLevels</u>
<i>Inst Term Hierarchy Levels</i>	Specifies the layout hierarchy depth from which the instance terminals are to be extracted during RC extraction. By default, the instance terminals at the bottom of the hierarchy are extracted. Value for this field should be less than or equal to the value of <i>Hierarchy Levels</i>. Environment variable: <u>instTermHierarchyLevels</u>
<i>Erosion Effects</i>	Specifies whether erosion effects are applied only to resistances (<i>R</i>), only to capacitances (<i>C</i>), to resistances and capacitances (<i>RC</i>), or to neither resistances nor capacitances (<i>none</i>). Environment variable: <u>enableErosionEffects</u>

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field	Specifies...
<i>Stop at Device Boundary</i>	Controls whether EAD extracts parasitics for shapes that are part of device terminals; for example, metal shapes in the source/drain regions of a transistor. This applies for instances at the level specified by the <i>Hierarchy Levels</i> setting. By default, EAD extracts parasitics for such shapes. But when this option is switched on, extraction stops at the device boundary, which means that terminals on Pcells and instances are treated as background shapes and are not extracted. Environment variable: <u>promotePorts</u>
<i>Threads</i>	The number of threads to use in multi-threaded R and C engines, both Fast and High Precision. It is applied only when multiple nets are being processed. Environment variable: <u>threads</u>
<i>Stop on LVS Mismatch</i>	Select this check box if you want to skip extraction when there is an LVS mismatch.
<i>Use Quantus timing mode</i>	Select this check box if you want to run Quantus extraction in timing mode. In this mode, simulation is faster but there are less details available for debugging.
Thresholds	
<i>Max Resistor Length</i>	The maximum length of resistor that will be extracted. Specify a value ≥ 0.5 in the text box. Resistors that are longer than the specified length are fractured into multiple resistors. The unit for this value is <code>microns</code>. Environment variable: <u>maxResistorLength</u>
<i>Minimum Coupling Cap</i>	A minimum coupling capacitance value in femtoFarad (fF). Capacitance values below the specified threshold are ignored during extraction. This lets you filter out small capacitance values that are likely to be inconsequential in any analysis. Environment variable: <u>couplingCThreshold</u>
<i>Resistance Scaling Factor</i>	Scaling factor to be applied on the extracted resistances. Environment variable: <u>resScaleFactor</u>

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field	Specifies...
<i>Gnd Cap Scaling Factor</i>	Scaling factor to be applied on the extracted ground capacitances. Environment variable: <u>gndCapScaleFactor</u>
<i>Coupling Cap Scaling Factor</i>	Scaling factor to be applied to the extracted coupling capacitances. Environment variable: <u>couplingCapScaleFactor</u>
High Precision C Solver	
<i>Use for All Nets</i>	That the high precision C solver is to be enabled for all nets. Use the <i>Threads</i> option to specify the number of processing threads to be used. Environment variable: <u>tileExtractor</u>
<i>Engine</i>	The engine to used for extracting parasitics. See the <u>iQuantus FS C Extraction (Advanced Nodes Layout EAD Only)</u> section for more information.
<i>Default Convergence Target</i>	The convergence goal (standard deviation) of the high precision C solver. It means that HPC will stop the random walk calculation process when it determines that standard deviation error has reached this convergence goal. In other words, if calculation continues with more walks, there is 99.7% probability that the result will fall into three sigma bounds comparing with the obtained values. Environment variable: <u>convergenceCThreshold</u>
<i>Metal Fill Modeling Method</i>	Controls how the EAD extraction treats fill shapes in the layout. ■ <i>floating</i> – Treat fill shapes as floating (unassigned to any net) ■ <i>ground</i> – Treat fill shapes as if they are connected to ground ■ <i>none</i> – Ignore fill shapes altogether Environment variable: <u>fillType</u>
<i>Via Effects</i>	Enables or disables via capacitance effects in the high precision C extraction. Environment variable: <u>modelViaCapacitanceEffect</u>

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field	Specifies...
<i>Threads Per Net</i>	The number of processing threads to be used by the high precision C solver for each net. This is another level of threading from the net-based processing. Setting this value to <code>auto</code> means that you want to use all the threads on the machine. For example, on a 16-core machine, 16 threads will be used. If you set this variable to a non-zero value, it will use the specified number of threads. The maximum number of threads that can be used will always be equal to the number of cores on the machine, regardless of the user-specified setting. For example, on a 16-core machine, if you set this variable to 8, only 8 threads will be used. But on the same machine, if you set this variable to 32, a maximum of 16 threads will be used regardless of the user-specified setting. Environment variable: <u>tileExtractorThreadCount</u>

High Precision R Solver

<i>Use for All Nets</i>	That the high precision R solver is to be used for all nets. Use the <i>Default Mesh Density</i> option to specify the mesh density to be used by the extractor.
<i>Default Mesh Density</i>	The mesh density to be used by the high precision R solver. The default value is 1.0. Environment variable: <u>resistanceMeshDensity</u> Note: You can use a customized mesh density for specific layers using the <u>resistanceMeshDensityFileName</u> environment variable.

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field	Specifies...
<i>Threads Per Net</i>	The number of processing threads to be used by the high precision R solver for each net. This is another level of threading from the net-based processing. Setting this value to <code>auto</code> means that you want to use as many threads as supported by the CPU. For example, on a 16-core machine, 16 threads will be used. If you set this variable to a non-zero value, it will use the specified number of threads. The maximum number of threads that can be used will always be equal to the number of cores on the machine, regardless of the user-specified setting. For example, on a 16-core machine, if you set this variable to 8, only 8 threads will be used. But on the same machine, if you set this variable to 32, a maximum of 16 threads will be used regardless of the user-specified setting. Note: Setting this value to <code>auto</code> means that you want to use as many threads as supported by the CPU. For distributed processing, the <code>auto</code> setting defaults to 1 (single thread) because it is difficult to predict on which host the job will be submitted to and how many extra licenses will be required for extraction. However, you can use the <code>autoThreadsPerNetOverride</code> environment variable to override the <code>auto</code> settings to a non-zero integer value when using distributed processing jobs for multi-threaded extraction.
<i>Location</i>	See EAD Setup Options
<i>Name</i>	

Note: Similar to the High Precision C solver, the high precision R engine also supports multiple threads per net, but there is no field in this form to specify the number of threads. The engine uses the setting of the `threads` environment variable.

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

EM

The screenshot shows the 'EAD Options' dialog box with the 'EM' tab selected. The dialog has five tabs: General, Extraction, EM, Environment, and Comments. The EM tab contains several sections: Temperature, Current, Resistor, Aging Effects, and Setups. The Temperature section has input fields for EM Temperature (25 °C) and Delta T (5 °C), with a checkbox for 'Use Dataset Value'. The Current section has input fields for Default (0 A), Minimum (0 A), and Scaling Factor (1). The Resistor section has an input field for Resistor Limit (50000) and a checkbox for 'Ignore Resistor Limit'. The Aging Effects section has an input field for Lifetime (10 years) and a dropdown menu. The Setups section has a checkbox for 'Dynamic Analysis', input fields for 'From' and 'To', a checkbox for 'Use Dataset Value', and a checkbox for 'Enable Self Heating Effects'. At the bottom, there are buttons for 'Save To', 'Load From', 'Delete', 'OK', 'Cancel', 'Defaults', 'Apply', and 'Help'.

EAD Options

General Extraction **EM** Environment Comments

Temperature

EM Temperature: °C ☐ Use Dataset Value

Delta T: °C

Current

Default: A

Minimum: A

Scaling Factor:

Resistor

Resistor Limit: ☐ Ignore Resistor Limit

Aging Effects

Lifetime: years ▼

☐ **Dynamic Analysis**

From To ☐ Use Dataset Value

☐ Enable Self Heating Effects

Setups

Location: ▼

Name: ▼

Save To Load From Delete

OK Cancel Defaults Apply Help

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field	Specifies...
Temperature	
<i>EM Temperature</i>	The EM temperature in degrees Celsius. This should be based on EM rules specified by the foundry. Environment variable: <u>temperature</u>
<i>Use Dataset Value</i>	That the temperature specified in the simulation dataset is to be used when computing EM effects. If the temperature value specified in the dataset is not valid, the default value specified in the <i>EM Temperature</i> field is used. ADE Assembler can correctly read a valid temperature if it is specified as a numerical value or as a variable using the VAR syntax. Environment variable: <u>useDatasetTemperature</u>
<i>Delta T</i>	The increase (in degrees Celsius) in junction temperature due to joule heating. Environment variable: <u>deltaT</u>
Current	
<i>Default</i>	The default terminal current to use when computing EM effects when there is no electrical data available from ADE Assembler. Environment variable: <u>defaultCurrent</u>
<i>Minimum</i>	The minimum threshold limit for I_{dc} and I_{avg} currents of the terminals and instance terminals of a net for using that net in EM calculation. If the current of all terminals and instance terminals of a net is below the threshold, EM calculation is skipped for that net. This minimum value is considered for the EM calculation of DC Op and Static-Avg current. It is not considered for the calculation of Dynamic-Rms and Dynamic-Peak current. Environment variable: <u>minCurrent</u>
<i>Scaling Factor</i>	The scaling factor to be used to scale currents when computing EM effects. All terminal currents are scaled by this value. Environment variable: <u>scalingFactor</u> and <u>scale</u>

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field	Specifies...
Resistor	
<i>Resistor Limit</i>	The maximum resistor limit against which the resistor count is compared while running EM checks.
<i>Ignore Resistor Limit</i>	Controls the comparison of the resistor count against the specified resistor limit while running EM checks. Environment variable: <u>ignoreResistorLimit</u>
Aging Effects	
<i>Lifetime</i>	The lifetime to be used for risk-based EM analysis. In risk-based EM analysis, MTTF (Mean Time To Failure) is first calculated using Black's equation. The calculated value is then used together with the specified lifetime to determine the probability of EM failure. Specify the value in the text field and choose either <i>years</i> or <i>hours</i> from the pull-down list. Environment variable: <u>lifetime</u> and <u>lifetimeUnits</u> Note: This field is disabled when the <i>EM Data Source</i> is set to <i>EM data files</i> in the <u>EAD Process Settings – Corners</u> form.
Dynamic Analysis	Enables dynamic EM analysis. Use the <i>From</i> and <i>To</i> fields to limit dynamic analysis to a specific time period in the simulation results. Environment variable: <u>dynamicAnalysis</u> When enabled, you can right-click in the <i>Dynamic Peak</i> and <i>Dynamic Rms</i> columns in EM tab and choose <i>Plot Terminal Currents</i> to view the currents associated with the terminals of the selected net or <i>Plot Resistor Currents</i> to view the intermediate currents through the selected resistor. Plots are displayed in a separate Virtuoso Visualization and Analysis XL window.
<i>From</i>	Start time for dynamic analysis (default is 0).
<i>To</i>	End time for dynamic analysis (default is the last time point in the simulation results).

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field	Specifies...
<i>Use Dataset Value</i>	Specifies if the information about the time interval for which the clipped waveform is to be used for EM analysis is to be taken from the dataset Environment variable: useDatasetClipInfo
<i>Enable Self Heating Effects</i>	Enables the use of self heating effects in EAD. Environment variable: enableSelfHeatingEffects When the self heating effects are used, Virtuoso Layout EAD calculates the delta-T values for devices by using the delta-T values saved in simulation result datasets and the RMS current. This calculation uses certain constant values that you can provide in a setup file specified by the sheSetupFile environment variable. To know about how to view the delta T results in EAD Browser, refer to Viewing EM Information in the EAD Browser. Note: When you have enabled the inclusion of self-heating effects in EM analysis, ensure that the datasets are generated from ADE Assembler simulation with the self-heating mode enabled on the Reliability Options form.
Setups	
<i>Location Name</i>	See EAD Setup Options

Environment

Form Field	Specifies...
Job Setup	
<i>Default Job Policy</i>	The job policy to be used for all the nets. The default job policy is <code>localhost</code>. Click <i>Edit job policies</i> () to open the <i>EAD Job Policy Editor</i> form, where you can set up, edit, and delete a job policy. See the Working with Job Policies section for more information.
<i>Wait For All Jobs To Complete</i>	

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field	Specifies...
	Waits for all the jobs to finish before showing the extraction or EM results.
EM Violation Highlighting	
<i>% of Limit</i>	The EM violation ranges used to highlight violations in the EAD Browser. Environment variable: <u>violationLevels</u>
<i>Color</i>	The colors used to highlight different violation levels in the EAD Browser. These are fully customizable, allowing you to color code the table display based on how much current capacity is being used by each wire.
EAD Browser Customization	
<i>Highlight Selected Nets</i>	That nets selected in the EAD Browser are automatically highlighted in the layout window as well. Environment variable: <u>highlightSelectedNets</u>
<i>Select Nets in Layout</i>	That nets selected in the EAD Browser are automatically selected in the layout window as well. Environment variable: <u>selectNetsInLayout</u>
<i>Detailed Parasitic Tables</i>	Displays detailed resistance and capacitance tables in the EAD Browser. When enabled, ■ If you select a capacitance value, the browser shows an extra table of individual coupling capacitance values ■ If you select a resistance value, the browser shows an additional table of individual resistance values along each path Environment variable: <u>detailedParasiticTables</u>
Setups	
<i>Location</i>	See <u>EAD Setup Options</u>
<i>Name</i>	

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Comments

The Comments tab contains a field text which displays the comments from the currently loaded EAD setup file. You can edit and save these comments. The updated comments are saved into the setup file.

EAD Setup Options

Form Field	Specifies...
<i>Location</i>	The directories where the setup files and process settings are stored. The list of locations includes all the <code>.cadence</code> directories found in the paths specified in the <code>Cadence setup.loc</code> file and in any additional directories specified by the <code>auxSearchPaths</code> environment variable; for example, Process settings: <pre><directory>/ .cadence/ [<user>] /dfII/EAD/ [<version>] /process/<process_settings_files></pre> Setups: <pre><directory>/ .cadence/ [<user>] /dfII/EAD/ [<version>] /setup/<setup_files></pre> Also listed are any technology-dependent setup files stored in a subdirectory with the same name as the technology library for the design. For more information, see “Understanding the .cadence Hierarchy” in the <i>Virtuoso Design Environment User Guide</i>.
<i>Name</i>	The name of the setup file. Use the buttons to perform the action required: ■ <i>Save To</i> saves the current settings to the specified setup file ■ <i>Load From</i> loads the settings from the specified setup file for use in the current EAD session ■ <i>Delete</i> deletes the specified setup file

EAD Pegasus/Quantus Create Setup File Dialog Form

The EAD Pegasus/Quantus Create Setup File Dialog form lets you specify the Quantus CCL and PVS GUI log files to create the setup required for electrically aware extraction.

Field	Description
<i>Quantus CCL File Path</i>	The file path to the Quantus CCL file that is generated after a Quantus run.
<i>PVS GUI Log File Path</i>	The file path to the PVS GUI log file that is generated after a PVS run.

Related Topics

[Creating Pegasus/Quantus Setup for Parasitics Extraction](#)

[Extracting Parasitics from a Partial Layout](#)

EAD Process Settings

The EAD process settings are split across four tabs:

- EAD Process Settings – Corners
- EAD Process Settings – Layer Mapping
- EAD Process Settings – Vias
- EAD Process Settings – Cell Shape Types

EAD Process Settings – Corners

Lets you define extraction corners by pointing to the corresponding ICT files that contain their

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

process information.

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field	Specifies...
<i>EM Data Source</i>	The source for EM rules. Choose any of the following: ■ <i>Virtuoso techfile</i>: to read the EM data from the Virtuoso technology file ■ <i>ICT files</i>: to read the EM data directly from the ICT file or QRC technology file specified in the <i>EAD Tech File</i> column Note: If you are performing pre-adoption testing of the EAD flow, you can directly use the EM data from a QRC technology file, but it cannot be used for fast C extraction. Therefore, in production use, it is recommended to generate an EAD technology file from the QRC technology file. ■ <i>ICT-EM</i>: to read the EM rules from partial ICT files that contain only the content required to define the EM rules for EM analysis. You can use this data source in the following scenarios: ☐ When you have an EAD technology file that contains process settings, but you need to load different sets of EM rules to test the design against different EM operating conditions. ☐ When the ICT file does not contain the final set of EM rules that are provided separately in partial ICT files. In this case, instead of modifying the ICT file to include the EM rules, you can load them from the EM rules file. When choosing this option, specify the name of the ICT file in the <i>EAD Tech File</i> column and the partial ICT file in the <i>EM Data File</i> column of the table given on this tab. Note: A partial ICT file uses the ICT syntax to define the EM parameters in the process section, and the EM models in the conductor and via layer sections. Also see: Using Custom Variables in ICT-EM Files ■ <i>EM data files</i> (same as VAVO/VAEO) to add the <i>EM Data File</i> column where you can specify an EM data file to use ■ <i>iRCX-EM</i>: to read the EM data directly from the iRCX-EM file without the need to update the existing EAD technology file or to first convert the iRCX-EM file to the ICT-EM file.

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field	Specifies...
<i>Name</i>	The name of a corner.
<i>EAD Tech File</i>	The location of the EAD technology file to be used for the specified corner. Type the path to the file into the field, or use the Browse button to locate it on the disk. For more details on the EAD tech file, refer to Generating an EAD Technology File.
<i>ICT-EM Data File</i>	The location of the ICT-EM data file to be used. When using an ICT-EM data file, ensure that the layer names in the EAD technology file and the given ICT-EM data file match. If the layer names in these files do not match, you must provide an EM Layer Map file along with the ICT-EM data file. The EM Layer Map file provides the missing layer mapping information. Note: This field is visible only when the <i>EM Data Source</i> is set to <i>ICT-EM</i>.
<i>EM Layer Map File</i>	The location the EM Layer Map file that contains the layer mapping information. Double click the cell in the column to specify the path either by typing or by clicking the ellipsis (...) button. You can also specify this file using the icitemLayerMapFile environment variable. Note: This field is visible only when the <i>EM Data Source</i> is set to <i>ICT-EM</i>.
<i>EM Data File</i>	The location of the EM data file for the corner in question. Type the path to the file into the field, or use the <i>Browse</i> button to locate it on disk. Note: This field is visible only when <i>EM Data Source</i> is set to <i>EM data files</i>.

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field	Specifies...
<i>iRCX-EM Data File</i>	The location of the iRCX-EM data file to be used for the specified corner. Double click the cell in the column to specify the path either by typing or by clicking the ellipsis (...) button. When using an iRCX-EM file, ensure that the layer names in the EAD technology file and the given iRCX-EM file match. If the layer names in these files do not match, you must provide an iRCX-RC file along with the iRCX-EM file. The iRCX-RC file provides the missing layer mapping information. Note: This field is visible only when the <i>EM Data Source</i> is set to <i>iRCX-EM</i>.
<i>iRCX-RC Layer Map File</i>	The location of the iRCX-RC file that contains the layer mapping information. Double click the cell in the column to specify the path either by typing or by clicking the ellipsis (...) button. Note: This field is visible only when the <i>EM Data Source</i> is set to <i>iRCX-EM</i>. You do not need to specify the iRCX-RC Layer Map File if the iRCX-EM file contains both the information, the EM rules and the layer mapping information.

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

EAD Process Settings – Layer Mapping

Specifies the mapping between the layers used in the ICT file and the layers defined in the Virtuoso technology file.

The system assumes that layers with the same names in the ICT file and Virtuoso technology file should be mapped. If this is incorrect, you can double-click the field you want to change and either type in an existing layer name directly or choose a layer name from the drop-down list.

Form Field	Specifies...
<i>ICT Layer</i>	The name of the layer in the ICT file.
<i>Virtuoso Layer</i>	The name of the corresponding layer in the Virtuoso technology file. The specified layers can be original layers or Virtuoso layers derived from the <code>techDerivedLayers</code> section of the technology file. An ICT file can have multiple poly layers for different devices, but the Virtuoso technology file has only a single poly layer. You need to derive and map additional poly layers. If there is no obvious mapping, this entry is left blank. Type in or choose the layer you need by double-clicking on the entry in the table. The drop-down list shows only those layers that are relevant to extraction.

EAD Process Settings – Vias

EAD Process Settings - gpdk045_rcx

Tabs: Corners | Layer Mapping | **Vias** | Cell Shape Types | Comments | Advanced Mode

Via Instance Mode: **flatten**

Layer	Max Spacing	Max Cuts
Via9	0.4	100
Via8	0.4	100
Via7	0.4	100
Via6	0.4	100
Via5	0.4	100
Via4	0.4	100
Via3	0.4	100
Via2	0.4	100
Via10	0.4	100
Via1	0.4	100
Cont	0.4	100

Buttons: Add, Delete, **Save**, Save As..., Cancel, Help

Specifies how vias are flattened and clustered.

Form Field

Specifies...

Via Instance Mode

The clustering style to be used. Choose one of the following:

- *cluster*: Clustering parameters are applied inside each via instance. No clustering occurs across hierarchical boundaries.
- *flatten*: Flattens all via instances (across hierarchy) and then applies clustering parameters
- *one_resistor*: Extracts one resistor for each via instance. Clustering parameters not followed, there is no clustering across hierarchical instance or Pcell boundaries. Flat vias follow cluster parameters.

Environment variable: [viaInstanceMode](#)

Also see: [viaShapesScanMode](#)

Layer

The via layer name.

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field	Specifies...
<i>Max Spacing</i>	The maximum spacing allowed between vias. If the space between vias for a given layer is below the specified value, they are clustered. Note: If this value is not specified, the <code>min_spacing</code> rule defined in the Virtuoso technology file for the via layer is used to calculate the max spacing value as shown below. $\text{Max Spacing} = 2 * \text{minSpacing} + \text{minWidth} - (1/\text{DBUperUU})$ If <code>minSpacing</code> value is not defined in the Virtuoso technology file for the via layer, no clustering is performed, which means that one resistor is extracted for each via shape. DBUperUU is <u>database units per user unit</u> specified in the technology file.
<i>Max Cuts</i>	Maximum number of vias that are clustered. Default value: 100

The details of via layers are imported from the .ini files. If required, you can manually add or delete via layers by using the *Add* and *Delete* commands given below the Vias table.

EAD Process Settings – Cell Shape Types

Layout EAD can treat cells, views, terminals, and layers as different shape types for extraction purposes. This lets you perform a differentiated extraction of terminals on Pcells by defining how non-terminal shapes in sub-cells contribute to the extraction.

You can use wildcards to define the Pcell names, specify which terminals are to be extracted for RC, and specify whether to extract capacitance to the terminal layer even if RC is not extracted on the terminal itself.

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

In the table, click *Add* to add a row and complete the entry as required. Click *Delete* to remove a row from the table.

Form Field	Specifies...
<i>Lib</i> <i>Cell</i> <i>View</i>	The cellviews containing the shapes to be specified. Type the required library, cell, and view names into the fields provided (or choose the names from the drop-down lists). If you do not specify a cell or view name, then all cells or views are assumed. you can also use an asterisk as a wildcard as shown in the examples below.
<i>Term</i>	A terminal name. If you do not specify a terminal name, then all terminals are assumed.
<i>Layer</i>	A layer name. You can either type in a layer name or choose a layer name from the drop-down list. The list shows only those layers that are relevant to extraction. If you do not specify a layer name, then all layers are assumed.
<i>Shape Type</i>	How a shape is treated during extraction: ■ <i>regular</i> refers to shapes that have a net assignment. EAD extracts coupling capacitance to these shapes ■ <i>background</i> refers to shapes that do not have a net assignment. EAD extracts grounded capacitance to these shapes (exception: for “background” shapes on device or instance terminals, EAD extracts coupling capacitance) ■ <i>invisible</i>: Shapes are ignored by the extraction engine Environment variable: <u>hierarchyShapeType</u>

EAD Job Policy Editor

The following figure shows the *EAD Job Policy Editor* form.

The following table describes the fields that are available in the EAD Job Policy Editor form.

Form Field	Specifies...
<i>Job Policy Name</i>	Displays the currently selected job policy.
<i>Distribution Method</i>	One of the following distribution methods: ■ Command ■ LBS ■ Local ■ Remote-Host
<i>Command</i>	The <code>bsub</code> command to submit a job for batched execution by a distributed load-sharing batch system Note: This field is displayed only when <code>Command</code> is selected as the <i>Distribution Method</i> .
<i>Queue</i>	The name of queue to which the distributed job is submitted. Note: This field is displayed only when <code>LBS</code> is selected as the <i>Distribution Method</i> .
<i>Hosts</i>	The list of remote hosts to be used for evaluating the submitted jobs. Note: This field is displayed only when <code>Remote-Host</code> is selected as the <i>Distribution Method</i> .

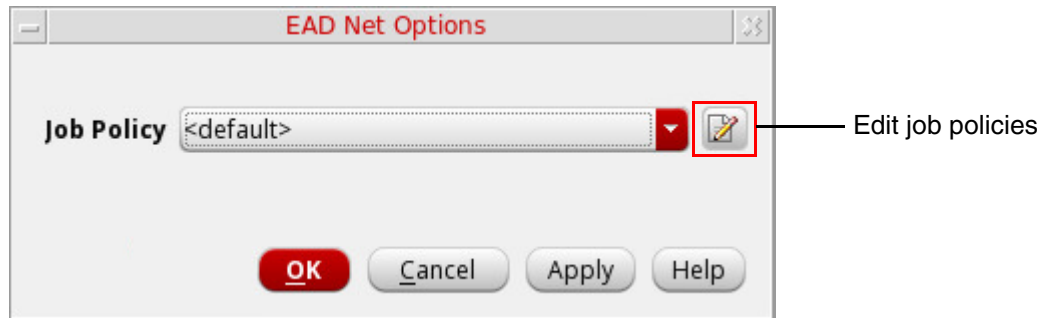
Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field	Specifies...
<i>Max Jobs</i>	Maximum number of simultaneous jobs Note: This field is displayed when the selected <i>Distribution Method</i> is either <i>Command</i> or <i>LBS</i> .

EAD Net Options

The following figure shows the *EAD Net Options* form.



The following table describes the fields that are available in the EAD Net Options form.

Form Field	Specifies...
<i>Job Policy</i>	Displays the currently selected job policy for a net. The <default> policy indicates that the policy that is selected as the <i>Default Job Policy</i> in the <i>Job Setup</i> section on the <i>Environment</i> tab of the EAD Options form is also selected for the net. You can also select different policies for different nets. Click the <i>Edit job policies</i> icon next to the <i>Job Policy</i> field to open the <u>EAD Job Policy Editor</u> form.

Parasitics & LDE Setup

Lets you specify the options for netlist_layout view or DSPF file to be used to run simulations with layout parasitics. Depending on the command used to open this form, this form shows the following tabs:

- General tab and Layout tab when you use the *Build Parasitic/LDE Netlist View* command.
- General tab and DSPF tab when you use the *Build Parasitic/LDE Netlist View* command.

General Tab

Lets you specify the options for netlist view or DSPF file generation.

Form Field	Description
<i>Library</i>	Name of the library
<i>Cell</i>	Name of the cell
<i>View</i>	Name of the view
<i>Device M Factor Parameter Names</i>	Names of the parameters that define m factor for devices
<i>Device Finger Parameter Names</i>	Names of the parameters that define the number of fingers for devices

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Layout Tab

Lets you specify the options for netlist_layout view creation.

Form Field	Description
<i>Netlist View Name</i>	Specifies the name of the view to be used to create a netlist that includes parasitics or LDE parameters from the layout view. Both LDE parameters and layout parasitics can be added to the same netlist view.
<i>Layout Lib Name for Parasitics & LDE</i>	Name of the library of the design that contains a layout cellview with layout parasitics.
<i>Layout Cell Name for Parasitics & LDE</i>	Name of the design cell that contains a layout view with layout parasitics.
<i>Layout View Name for Parasitics & LDE</i>	Name of the layout view that contains layout parasitics to be used in resimulation of a design. This can be a partial or a complete layout view. When you specify a layout name in this field, select Include Parasitics From Layout option in the Parasitics section on the Layout tab or the Layout View option in the LDE section on the Layout tab. This is helpful in the EAD or LDE flow. You can choose to include both the LDE parameters and the layout parasitics from the same layout view.
<i>Parasitics</i>	This section contains fields to be used while including parasitics from a layout view. This is required while simulating a design with layout dependent effects or in the Electrically Aware Design flow.

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field	Description
<i>Include parasitics from</i>	Specifies the source of parasitics. You can select any one of the following three sources: ■ <i>None</i> - No parasitics are used. This is the default option. ■ <i>Schematic Estimates</i> - Includes estimates from the Parasitics & Electrical assistant. These are the same estimates used in the Schematic Estimates mode. This option allows the use of LDE with estimated parasitics ■ <i>Layout</i> - Includes parasitic estimates from the layout view specified in the Layout View Name for Parasitics & LDE field. When you choose to include layout parasitics in simulation, you also need to change the parasitic mode to Layout (Parasitics/LDE).
<i>Extraction Setup for Layout Parasitics</i>	When you choose to include parasitics from the specified layout view, names of the all the extracted corners for that layout are listed in this field. Select name of the desired corner.
<i>Extraction Corner</i>	When you choose to include parasitics from the specified layout view, names of the all the extracted corners for that layout are listed in this field. Select name of the desired corner. The extraction corner that is selected in the EAD Browser is displayed as the default corer in this drop-down list. The default extraction corner is defined in the process settings file (<i>.ini</i> file) that is saved to the specified location, for example: <pre>/.cadence/dfII/EAD/1/process/gpdk045.ini</pre>

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field	Description
<i>Reference Net for Grounded C</i>	Specifies name of the ground net to be used for grounded capacitance. You can either type a name in this field or click Select from Schematic to select a net from the design schematic. Important Points ■ A reference net for grounded C is not required if you are extracting only R estimates or if you are stitching only R estimates from RC extraction. ■ You can specify name of a global net that does not exist in the design under test. The net will be created in the netlist view as an inherited connection so that you can use a netSet property on the cell instance to specify the net to identify it with. The netSet property name is the same as the net name. ■ You can use the <code>msps.layout.referenceNet</code> environment variable to set the initial default value of the reference net field.
<i>Expand Devices with M-Factor</i>	Specifies if it is required to expand the devices with multiple factors before generating a netlist. When this option is selected, the m-factor parameter on the schematic device is then used to create multiple parallel devices in the netlist view. This allows the parasitics for the nets between each m-factor instance in the layout to be brought into the netlist view for simulation. If this check box is disabled, the device is represented as a single instance in the netlist view, and the parasitics between m-factor instances will not be included.

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field	Description
<i>LDE</i>	This section contains fields to be used while including layout dependent effects from a layout view. This is required while re-simulating a design in the Layout Dependent Effects flow. When you choose to include LDE parameters in simulation, you also need to change the parasitic mode to Layout (Parasitics/LDE).
Include LDE From	Specifies the source of LDEs. You can choose any one of the following options: ■ <i>None</i> - This is the default option, which specifies that no parameters from layout or MODGEN constraints are to be used. ■ <i>MODGEN Constraints</i> - Includes LDEs extracted from the Modgen constraints defined for the given constraint cellview. ■ <i>Layout View</i> - Includes estimates from the layout view specified in the Layout View Name for Parasitics & LDE field. If you include parasitics from layout in the Parasitics section, you cannot choose to use LDEs defined in MODGEN constraints. This is because parasitics and LDE parameters must come from the same source.
<i>Include LDE from Layout View</i>	Includes estimates from the layout view into the netlist view.
<i>Device M-Factor parameter Names</i>	While generating a netlist with parasitics, if the tool finds any cell with parameter names listed in field, it expands the cell if the Expand Devices with M-Factor check box is selected. However, when using LDE parameters, cells will be expanded according to the multiple factors used in the layout or MODGEN constraint. In both cases, the m-factor value on the individual expanded devices in the netlist is reset to 1.

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field	Description
<i>Device Finger Parameter Names</i>	While generating a netlist with parasitics, if the tool finds any cell with the given finger parameter names, it expands the cell if the Expand Devices with M-Factor check box is selected. However, when using LDE parameters, cells will be expanded according to the multiple fingers used in the layout or MODGEN constraint.
<i>Ignore Dummies Back-Annotated to Schematic</i>	Specifies that if there are any dummy cells backannotated from layout to the schematic view, they need to be ignored while generating a netlist for simulation. For more details on backannotated dummy instances, refer to Dummy Instances Backannotation.

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

DSPF Tab

Lets you specify the options for DSPF file generation.

Form Field	Description
<i>DSPF File Name</i>	Name of the DSPF file. The default file name is <code><cellName>.dspf</code> . The file is saved in the cellview directory. Related environment variable: <u>dspfFileName</u>
<i>Subnode Delimiter</i>	The delimiter to be used while writing subnodes in the DSPF file. Default value: # Related environment variables: <u>dspfSubNodeDelimiter</u> <u>dspfWriteSubNodes</u>
<i>Coupling Capacitance Prefix</i>	The default prefix to be used for the coupling capacitance values in the DSPF file. Related environment variable: <u>dspfCouplingCapPrefix</u>
<i>Ground Capacitance Prefix</i>	The default prefix to be used for the ground capacitance values in the DSPF file Related environment variable: <u>dspfGroundCapPrefix</u>
<i>Unit</i>	The unit for capacitance values to be written in the DSPF file. Default value: f Related environment variable: <u>dspfCapUnits</u>
<i>Resistance Prefix</i>	The default prefix to be used for the resistance values in the DSPF file. Related environment variable: <u>dspfResistorPrefix</u>

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Form Field	Description
<i>Unit</i>	The unit for resistance values to be written in the DSPF file. Default value: <code>ohm</code> Related environment variable: <code>dspfResUnits</code>
<i>Write Instances</i>	Specifies whether to write instances in the DSPF file. Related environment variable: <code>dspfWriteInstances</code>

P2T EM Optimization

Lets you specify the options for EM trunk optimization.

Form Field	Description
<i>Stack Twigs</i>	Adjusts the width of a twig that has an EM violation by performing stacked stranding. When this option is deselected, twigs with EM violations are ignored. By default this option is deselected Twig stacking is only supported when the optimization mode is <i>Expand to Fix EM violations</i> or <i>Shrink and Expand</i>. It is not supported when the optimization mode is <i>Shrink Oversized Trunks</i>.
<i>Fill Stacked Trunks with Vias</i>	Applies to widening of trunks using <i>Stacked Stranding</i>. This command inserts the maximum possible additional vias after a trunk is widened using stacked stranding. By default this option is deselected.
<i>Prefer DRC Violations to EM Violations</i>	Allows DRC violations (shorts and spacing errors) when widening the trunks, twigs, and vias. By default this option is deselected. This means that no DRC violations are allowed.

Related Topics

[Fixing Violations Using EM Driven Trunk Optimization](#)

[p2t](#)

[EAD Toolbar](#)

The Pre-EM Setup Form

Lets you specify the options for EM trunk optimization.

Form Field	Description
<i>Extraction Setup</i>	Specifies the path to the file containing the setup options. For example: " ./cadence/dfII/EAD/1/setup/gp-dk45.ini " Environment Variables: <u>useSetup</u>
<i>Extraction Corner</i>	Specifies the extraction corner that is mapped to an ICT file or eadTechFile to be used for extraction and the source of the EM models to be used for EM Checking. For example, nominal. Environment Variables: <u>useCorner</u>
<i>Pre-EM Check Layers</i>	Specifies the comma-separated list of layers on which pre-EM check is performed. For example, Via1, Via2, Via3 Environment Variables: <u>preEMCheckLayers</u>

Related Topics

Setting Up Environment Variables for the Pre-EM check

Virtuoso Electrically Aware Design Flow Guide

Form Descriptions

Environment Variables

The following sections describe the environment variables used in the electromigration flow:

- [elec.gui](#)
- [elec.results](#)
- [ei.dataSet](#)
- [ei.setup](#)
- [msps.layout](#)
- [layoutEAD](#)
- [layoutEAD.calc](#)
- [layoutEAD.em](#)
- [layoutEAD.gui](#)
- [layoutEAD.pex](#)
- [layoutEAD.solver](#)
- [p2t](#)

elec.gui

The following environment variables provide default values to the **EAD Setup** form in Design Environment XL:

- createVoltageConstraints
- customVoltageConstraintFunc
- customVminCalc
- customVmaxCalc
- enableCustomVminVmaxCalc
- enableEADMode
- saveAll
- useHierarchyLevel
- hierarchyLevel
- enableIdc
- idcScaleFactor
- enableIavg
- iavgScaleFactor
- enableIsignal
- saveWaveforms
- clipWaveforms
- clipFrom
- clipTo
- showOnlyTopLevelNets
- skipLibListForDataset
- vdrcPerf
- vdrcPerfSigCount
- vdrcCsvDatasetDir
- vdrcCsvDatasetFileSuffix

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

- vdrCsvDatasetNameSuffix
- warnOnCacheEdit
- warnOnCacheSave
- warnOnDatasetOverwrite

createVoltageConstraints

```
elec.gui createVoltageConstraints boolean { t | nil }
```

Description

Specifies whether to create voltage constraints. The default value is `nil`, which indicates that voltage constraints are not created. Set this value to `t`, if you want to create voltage constraints.

GUI Equivalent

EAD Setup – Voltage Dependent Rules – Create voltage constraints

Examples

```
envGetVal("elec.gui" "createVoltageConstraints")
envSetVal("elec.gui" "createVoltageConstraints" 'boolean t)
```

Related Topics

[EAD Setup](#)

[customVoltageConstraintFunc](#)

[elecVdrCreateConstraints](#)

[elecVdrCreateDeltaV](#)

customVoltageConstraintFunc

`elec.gui customVoltageConstraintFunc string nameOfTheUserDefinedFunction`

Description

Specifies the name of the user-defined function to be called to create `deltaVoltage` and `voltageSyncedNet` constraints.

GUI Equivalent

None

Examples

```
envGetVal("elec.gui" "customVoltageConstraintFunc")
envSetVal("elec.gui" "customVoltageConstraintFunc" 'string "userDefinedElecFunc")
```

Related Topics

[elecVdrCreateConstraints](#)

[elecVdrCreateDeltaV](#)

[createVoltageConstraints](#)

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

customVmaxCalc

Specifies the name of a user-defined SKILL procedure to be used in the calculation of the maximum voltage value during simulation.

In `.cdsenv`:

```
elec.gui customVmaxCalc string "demoCalcVmax"
```

In `.cdsinit` or the CIW:

```
envSetVal("elec.gui" "customVmaxCalc" 'string "demoCalcVmax")
```

Valid Values:

Name of a user-defined SKILL procedure to be used to calculate the maximum voltage value

Default Value: ""

GUI Equivalent: None

Note: If the job control mode is LSCS, define this setting in the `.vdsinit` file for custom voltage-dependent rules.

For more information, see [Customizing the Calculation of Minimum and Maximum Voltages](#) in the *Virtuoso Voltage Dependent Rules Flow Guide*.

customVminCalc

Specifies the name of a user-defined SKILL procedure to be used in the calculation of the minimum voltage value during simulation.

In `.cdsenv`:

```
elec.gui customVminCalc string "My_Vmin"
```

In `.cdsinit` or the CIW:

```
envSetVal("elec.gui" "customVminCalc" 'string "My_Vmin")
```

Valid Values:

Name of a user-defined SKILL procedure to be used
to calculate the minimum voltage value

Default Value: ""

GUI Equivalent: None

Note: If the job control mode is LSCS, define this setting in the `.vdsinit` file for custom voltage-dependent rules.

For more information, see [Customizing the Calculation of Minimum and Maximum Voltages](#) in the *Virtuoso Voltage Dependent Rules Flow Guide*.

enableCustomVminVmaxCalc

Enables or disables the feature to use custom SKILL procedures for the calculation of the minimum and maximum voltage values to be used in the voltage dependent rules flow.

In `.cdsenv`:

```
elec.gui enableCustomVminVmaxCalc boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("elec.gui" "enableCustomVminVmaxCalc" 'boolean t)
```

Valid Values:

<code>t</code>	Enables the feature to use custom SKILL procedures for the calculation of voltage values.
<code>nil</code>	Disables the feature to use custom SKILL procedures for the calculation of voltage values.

Default Value: `nil`

GUI Equivalent: `None`

Related Variables:

- [customVminCalc](#)
- [customVmaxCalc](#)

Note: If the job control mode is LSCS, define this setting in the `.vdsinit` file for custom voltage-dependent rules.

For more information, see [Customizing the Calculation of Minimum and Maximum Voltages](#) in the *Virtuoso Voltage Dependent Rules Flow Guide*.

enableEADMode

Enables or disables the EAD mode for the current design. When the EAD mode is enabled, the current data is saved with the simulation results. This data is further used in electromigration analysis.

EAD mode can be enabled in SKILL scripts using the following codes:

```
sess=axlGetWindowSession()  
sdb=axlGetMainSetupDB(sess)  
elecSetup=axlSDBGetExtension(sdb "ElectricalSetup")
```

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

```
when (elecSetup>0
  design=axlSDBGetChild(elecSetup "Design")
  elecMode=axlSDBGetChild(design "ElectricalMode")
  axlSDBSetVal(elecMode "True")
)
```

You can use the following command to disable EAD mode:

```
axlSDBSetVal(elecMode "False")
```

In `.cdsenv`:

```
elec.gui enableEADMode boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("elec.gui" "enableEADMode" 'boolean t)
```

Valid Values:

<code>t</code>	Enables the EAD mode for the current design.
<code>nil</code>	Disables the EAD mode for the current design.

Default Value: `nil`

GUI Equivalent: Command: *EAD – EAD Setup*

Field: *Enable Electrical Data Capture for EAD Flow*

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

saveAll

Specifies if the current data is to be saved for all the signals in the selected design.

In `.cdsenv`:

```
elec.gui saveAll boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("elec.gui" "saveAll" 'boolean t)
```

Valid Values:

<code>t</code>	Saves the current data for all the signals.
<code>nil</code>	Specifies that the current data is to be saved only for the selected signals. To know more about how to select signals, refer to Selecting Signals to Save Currents .

Default Value: `t`

GUI Equivalent: Command: *EAD – EAD Setup*

Field: *Signal Selection – Save EAD Data – All Signals*

useHierarchyLevel

Specifies if the current data is to be saved only upto a given hierarchy level in the selected design.

In `.cdsenv`:

```
elec.gui useHierarchyLevel boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("elec.gui" "useHierarchyLevel" 'boolean t)
```

Valid Values:

<code>t</code>	Saves the current data for all the signals upto a level specified by the <u>hierarchyLevel</u> environment variable.
<code>nil</code>	Saves the current data for all the signals in the complete hierarchy of the selected design.

Default Value: `t`

GUI Equivalent: Command: *EAD – EAD Setup*

Field: *Save Signal Selection – All Signals – Hierarchy Stop Level*

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

hierarchyLevel

Specifies the level of hierarchy upto which the current data is to be saved for all the signals.

In `.cdsenv`:

```
elec.gui hierarchyLevel int 5
```

In `.cdsinit` or the CIW:

```
envSetVal("elec.gui" "hierarchyLevel" 'int 4)
```

Valid Values:

any integer value	Specifies the level of hierarchy.

Default Value: 99

GUI Equivalent: Command: *EAD – EAD Setup*

Field: *Signal Selection – All Signals – Hierarchy Stop Level*

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

enableIdc

Saves the DC current data for the specified signals with the results database. In this case, ensure that DC analysis has been enabled for the test.

In `.cdsenv`:

```
elec.gui enableIdc boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("elec.gui" "enableIdc" 'boolean t)
```

Valid Values:

<code>t</code>	Saves the direct current data for the specified signals with the simulation results.
<code>nil</code>	Does not save the direct current data with the simulation results.

Default Value: `t`

GUI Equivalent: Command: *EAD – EAD Setup*
Field: *Type – DC Current (Idc)*

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

idcScaleFactor

Specifies the scale factor for the DC current data. This scale factor is used to scale the current data only while transferring the currents from schematic to the layout view.

In `.cdsenv`:

```
elec.gui idcScaleFactor float 1.0
```

In `.cdsinit` or the CIW:

```
envSetVal("elec.gui" "idcScaleFactor" 'float 2.0)
```

Valid Values:

A positive float value.

Default Value: 1.0

GUI Equivalent: Command: *EAD – EAD Setup*

Field: *Type – DC Current (Idc) – Scale* scroll list

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

enableIavg

Saves the average current data for the specified signals with the simulation results. In this case, ensure that transient analysis has been enabled for the test.

In `.cdsenv`:

```
elec.gui enableIavg boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("elec.gui" "enableIavg" 'boolean t)
```

Valid Values:

<code>t</code>	Saves the average current data with the simulation results.
<code>nil</code>	Does not save the average current data with the simulation results.

Default Value: `t`

GUI Equivalent: Command: *EAD – EAD Setup*

Field: *Type – Average Current (Iavg)*

iavgScaleFactor

Specifies the scale factor for the average current data.

In `.cdsenv`:

```
elec.gui iavgScaleFactor float 1.0
```

In `.cdsinit` or the CIW:

```
envSetVal("elec.gui" "iavgScaleFactor" 'float 2.0)
```

Valid Values:

A positive float value.

Default Value: 1.0

GUI Equivalent: Command: *EAD – EAD Setup*

Field: *Type – Average Current (Iavg) – Scale* scroll list

enableIsignal

Saves the transient waveform data for the selected signals so that it can be used for dynamic EM analysis.

In `.cdsenv`:

```
elec.gui enableIsignal boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("elec.gui" "enableIsignal" 'boolean t)
```

Valid Values:

<code>t</code>	Saves the transient waveform data for the selected signals.
<code>nil</code>	Does not save the transient waveform data for the selected signals.

Default Value: `nil`

GUI Equivalent: Command: *EAD – EAD Setup*

Field: *Type – Waveforms for RMS and Peak Checks (Isignal)*

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

saveWaveforms

Saves the current waveforms with the simulation data to disk so that it can be processed later while performing electromigration checks.

In `.cdsenv`:

```
elec.gui saveWaveforms boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("elec.gui" "saveWaveforms" 'boolean t)
```

Valid Values:

<code>t</code>	Saves the current waveforms with the simulation data.
<code>nil</code>	Does not save the current waveforms with the simulation data.

Note: The GUI equivalent for this value is *Process Waveforms Inside Simulator* and is currently available only for the Spectre and APS simulators

Default Value: `t`

GUI Equivalent: Command: *EAD – EAD Setup*

Field: *Process Waveforms Post Simulation*

clipWaveforms

If you choose to save the current waveforms with the simulation data, this option specifies whether the currents are to be compiled for a specific time interval or for the complete simulation run.

Note: This does not impact the waveforms saved in the results database.

The time interval can be specified using the clipFrom and clipTo environment variables.

In `.cdsenv`:

```
elec.gui clipWaveforms boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("elec.gui" "clipWaveforms" 'boolean t)
```

Valid Values:

<code>t</code>	Saves the current waveforms for a specific time period.
<code>nil</code>	Saves the current waveforms for complete simulation run.

Default Value: `nil`

GUI Equivalent: Command: *EAD – EAD Setup*

Field: *Clip Waveforms*

clipFrom

Specifies a start time of the time interval for which the waveform clip of the current data is to be saved.

In `.cdsenv`:

```
elec.gui clipFrom string ""
```

In `.cdsinit` or the CIW:

```
envSetVal("elec.gui" "clipFrom" 'string "10u")
```

Valid Values:

A string value specifying a start time for the waveform clip.

Default Value: `""`

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

GUI Equivalent: Command: *EAD – EAD Setup*
Field: *From*

clipTo

Specifies an end time of the time interval for which the waveform clip of the current data is to be saved.

In `.cdsenv`:

```
elec.gui clipTo string ""
```

In `.cdsinit` or the CIW:

```
envSetVal("elec.gui" "clipTo" 'string "40u")
```

Valid Values:

A string value specifying an end time for the waveform clip.

Default Value: ""

GUI Equivalent: Command: *EAD – EAD Setup*
Field: *To*

skipLibListForDataset

```
elec.gui skipLibListForDataset string space-separatedLibraryNames
```

Description

Specifies the space-separated library names for which you do not want to create datasets. The default value is "", which indicates that datasets are created for all libraries.

You can also use regular expressions. The following example uses regular expressions to skip dataset creation for all libraries whose name start with `testLib` and `myLib`:

```
envSetVal("elec.gui" "skipLibListForDataset" 'string "testLib* myLib*")
```

GUI Equivalent

None

Examples

```
envGetVal("elec.gui" "skipLibListForDataset")  
envSetVal("elec.gui" "skipLibListForDataset" 'string "myLib1 myLib2")
```

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

vdrcPerf

```
elec.gui vdrcPerf boolean { t | nil }
```

Description

Specifies whether to select *All* in the *DUT Masters* drop-down list.

The default is `nil`, which indicates that the current DUT is selected in the *DUT Masters* drop-down list. If a DUT master has constraint cache, viewing results also opens the constraints cache associated with the DUT master. Therefore, opening results may take a lot of time for large datasets.

When you set this environment variable to `t`, *All* is selected in the *DUT Masters* drop-down list and opening results does not open constraint cache. This helps in improving performance. Note that all signals are still saved.

GUI Equivalent

None

Examples

```
envGetVal("elec.gui" "vdrcPerf")  
envSetVal("elec.gui" "vdrcPerf" 'boolean t)
```


vdrcPerfSigCount

```
elec.gui vdrcPerfSigCount int numberOfSignals
```

Description

Specifies the threshold count of signals, exceeding which all instances for a DUT master are listed in *DUT Instance* drop-down list, even if an instance has no signals saved to them.

When number of signals is very large, it takes time to determine which instances have signals saved to them.

The default value is 10000, which indicates that if there are more than 10000 signals saved, all instances will be listed in the *DUT Instance* drop-down list, including the ones that do not have any signals.

If the number of signals saved are less than the specified value, only the instance that have at least one signal saved are displayed in the *DUT Instance* drop-down list.

GUI Equivalent

None

Examples

```
envGetVal("elec.gui" "vdrcPerfSigCount")  
envSetVal("elec.gui" "vdrcPerfSigCount" 'int 30000)
```

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

vdrCsvDatasetDir

`elec.gui vdrCsvDatasetDir string pathToTheDirectory`

Description

Specifies the path to the directory where the edited simulation dataset is saved as the CSV file. The default is ".", which indicates that the file is saved in the current working directory.

GUI Equivalent

None

Examples

```
envGetVal("elec.gui" "vdrCsvDatasetDir")
envSetVal("elec.gui" "vdrCsvDatasetDir" 'string "/home/user/simulation")
```

vdrCsvDatasetFileSuffix

```
elec.gui vdrCsvDatasetFileSuffix string suffixOfTheCSVFile
```

Description

Specifies the suffix of the CSV file containing information about the edited simulation dataset that is being saved as the CSV dataset. The default is "csv", which indicates that the CSV file is saved with the suffix `csv`.

GUI Equivalent

None

Examples

```
envGetVal("elec.gui" "vdrCsvDatasetFileSuffix")  
envSetVal("elec.gui" "vdrCsvDatasetFileSuffix" 'string "testCSV")
```

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

vdrCsvDatasetNameSuffix

`elec.gui vdrCsvDatasetNameSuffix string suffixOfTheDatasetName`

Description

Specifies the suffix to be added to the name of the CSV dataset. The default is "csv", which indicates that the name of the CSV Dataset is saved with the suffix `csv`.

GUI Equivalent

None

Examples

```
envGetVal("elec.gui" "vdrCsvDatasetNameSuffix")
envSetVal("elec.gui" "vdrCsvDatasetNameSuffix" 'string "csvDataset")
```

showOnlyTopLevelNets

```
elec.gui showOnlyTopLevelNets boolean { t | nil }
```

Description

Specifies whether to display only top-level nets for *Vmin* and *Vmax* calculations in the simulation datasets on the VDR Dataset form. The default is `nil`, which indicates that all nets are displayed.

GUI Equivalent

None

Examples

```
envGetVal("elec.gui" "showOnlyTopLevelNets")  
envSetVal("elec.gui" "showOnlyTopLevelNets" 'boolean t)
```

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

warnOnCacheEdit

Specifies if a warning message is to be displayed before making the constraint view editable while creating a dataset in ADE Assembler.

In `.cdsenv`:

```
elec.gui warnOnCacheEdit boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("elec.gui" "warnOnCacheEdit" 'boolean t)
```

Valid Values:

<code>t</code>	A warning message is displayed in the GUI to confirm that ADE Assembler can make the constraint view editable before creating the dataset.
<code>nil</code>	No warning message is displayed in the GUI. ADE Assembler makes the constraint view editable and creates the dataset. In addition, an info message is displayed in the CIW.

For more details, see [Creating Datasets](#).

Default Value: `t`

GUI Equivalent: `None`

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

warnOnCacheSave

Specifies if a warning message is to be displayed while saving a dataset in the constraint view.

In `.cdsenv`:

```
elec.gui warnOnCacheSave boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("elec.gui" "warnOnCacheSave" 'boolean t)
```

Valid Values:

<code>t</code>	A warning message is displayed to confirm if ADE Assembler can save the dataset in the constraint view.
<code>nil</code>	No warning message is displayed in the GUI. ADE Assembler saves the dataset in the constraint view and displays an info message in the CIW.

For more details, see [Creating Datasets](#).

Default Value: `t`

GUI Equivalent: `None`

warnOnDatasetOverwrite

Specifies if a warning message needs to be displayed when an existing dataset is being overwritten by the dataset being saved.

In `.cdsenv`:

```
elec.gui warnOnDatasetOverwrite boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("elec.gui" "warnOnDatasetOverwrite" 'boolean t)
```

Valid Values:

<code>t</code>	A warning popup is displayed
<code>nil</code>	No warning is displayed and the existing dataset is overwritten.

For more details, see [Creating Datasets](#).

Default Value: `t`

GUI Equivalent: None

elec.results

- dcResultName
- tranResultName

dcResultName

Specifies the result name for DC data that can be passed to the selectResult OCEAN command to access terminal currents. If not set, the EAD first looks for "dcOp". If that too is not found, uses "dc" as the result name.

In .cdsenv:

```
elec.results dcResultName string ""
```

In .cdsinit or the CIW:

```
envSetVal("elec.results" "dcResultName" 'string "")
```

Valid Values:

A string value specifying the result name for DC data.

Default Value: ""

GUI Equivalent: None

tranResultName

Specifies the result name for transient data that can be passed to the selectResult OCEAN command to access terminal currents. If not set, the tool uses "tran".

In .cdsenv:

```
elec.results tranResultName string ""
```

In .cdsinit or the CIW:

```
envSetVal("elec.results" "tranResultName" 'string "")
```

Valid Values:

A string value specifying the result name for transient data.

Default Value: ""

GUI Equivalent: None

ei.dataSet

- excludeCurrentsInCI
- storeRelativePaths
- transferPerf

excludeCurrentsInCI

Specifies if the current data is to be saved as part of the constraint view where a dataset is being saved. By default, when a dataset is saved in the constraint view, the current data is not saved in it. Instead, only the paths to the simulation results directory are saved.

In `.cdsenv`:

```
ei.dataSet excludeCurrentsInCI boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("ei.dataSet" "excludeCurrentsInCI" 'boolean t)
```

Valid Values:

<code>t</code>	Saves the current data in the ADE Assembler results database. Path to that directory is saved as a dataset property in the constraint view.
<code>nil</code>	Saves the current data in the constraint view.

Default Value: `t`

GUI Equivalent: `None`

storeRelativePaths

Saves the paths of the results database and the simulation directory as relative paths. By default, absolute paths are saved for these directories.

In `.cdsenv`:

```
ei.dataSet storeRelativePaths boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("ei.dataSet" "storeRelativePaths" 'boolean t)
```

Valid Values:

<code>t</code>	Saves relative paths for the results database and the simulation directory
<code>nil</code>	Saves absolute paths for the results database and the simulation directory

Default Value: `nil`

GUI Equivalent: `None`

transferPerf

Enables you to improve the performance when updating electrical data. By default, this environment variable is set to `t` so that you do not face any performance-related issues when updating electrical data. However, if this causes any issues in creating datasets, you can set it to `nil`.

In `.cdsenv`:

```
ei.dataSet transferPerf boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("ei.dataSet" "transferPerf" 'boolean t)
```

Valid Values:

<code>t</code>	Turns on the performance improvement.
<code>nil</code>	Turns off the performance improvement.

Default Value: `t`

GUI Equivalent: `None`

ei.setup

- enableWorstCaseDatasetForVoltages
- expandIteratedInstancesForCurrents
- expandIteratedInstancesForVoltages

enableWorstCaseDatasetForVoltages

```
ei.setup enableWorstCaseDatasetForVoltages boolean {t | nil}
```

Description

Specifies whether to create a single worst-case simulation voltage dataset having worst *Vmin* and *Vmax* values for all nets. The dataset is saved in the constraint view.

Worst *Vmin* value is the minimum voltage value across all corners, sweep points, and DUT instances. Worst *Vmax* value is the maximum voltage value across all corners, sweep points, and DUT instances.

The dataset with the net names and worst voltage values (*Vmin* and *Vmax*) are displayed in the *Simulation Dataset* tab of the VDR Dataset form.

The default value is `nil`, which indicates that simulation datasets are created for unique combinations of values in the *Point*, *Corner*, *DUT Masters*, and *DUT Instance* drop-down lists.

This environment variable works only if the environment variable `ei.dataSet excludeCurrentsInCI` is set to `nil`.

GUI Equivalent

None

Examples

```
envGetVal("ei.setup" "enableWorstCaseDatasetForVoltages")
envSetVal("ei.setup" "enableWorstCaseDatasetForVoltages" 'boolean t)
```

Related Topics

[excludeCurrentsInCI](#)

[Creating Datasets](#)

[Viewing Simulation Datasets](#)

expandIteratedInstancesForCurrents

Specifies whether iterated instances are to be expanded when saving the current data. For example, if there is a hierarchical instance terminal `/I1<0:2>/I2<0:1>/OUT`, the current data will be saved as described below.

When the `expandIteratedInstancesForCurrents` variable is set to `t`, the data would be saved for the terminals of all the iterated instances:

```
/I1<0>/I2<0>/OUT
/I1<1>/I2<0>/OUT
/I1<2>/I2<0>/OUT
/I1<0>/I2<1>/OUT
/I1<1>/I2<1>/OUT
/I1<2>/I2<1>/OUT
```

When the `expandIteratedInstancesForCurrents` variable is set to `nil`, only the first instance of the iterated instance is taken into account, ignoring the rest. In this example, the current would be saved only for `/I1<0>/I2<0>/OUT`.

Note: This variable is used only when you choose the *All Signals* option on the EAD Setup form for signal selection.

In `.cdsenv`:

```
ei.setup expandIteratedInstancesForCurrents boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("ei.setup" "expandIteratedInstancesForCurrents"
'boolean t)
```

Valid Values:

<code>t</code>	Expands the iterated instances when the instance terminals are selected to save currents.
<code>nil</code>	Does not iterate through the instances. Instead, saves the current data for the terminals of the first instance.

Default Value: `nil`

GUI Equivalent: `None`

expandIteratedInstancesForVoltages

Specifies whether iterated instances are to be expanded when saving the voltage data. For example, if there is a hierarchical instance voltage `/I1<0:2>/I2<0:1>/net4`, the voltage data will be saved as described below.

When the `expandIteratedInstancesForVoltages` variable is set to `t`, the data would be saved for the nets connected to all the iterated instances:

```
/I1<0>/I2<0>/net4  
/I1<1>/I2<0>/net4  
/I1<2>/I2<0>/net4  
/I1<0>/I2<1>/net4  
/I1<1>/I2<1>/net4  
/I1<2>/I2<1>/net4
```

When the `expandIteratedInstancesForVoltages` variable is set to `nil`, only the first instance of the iterated instance is taken into account, ignoring the rest. In this example, the voltage would be saved only for `/I1<0>/I2<0>/net4`.

Note: This variable is used only when you choose the *All Signals* option on the EAD Setup form for signal selection.

In `.cdsenv`:

```
ei.setup expandIteratedInstancesForVoltages boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("ei.setup" "expandIteratedInstancesForVoltages"  
          'boolean nil)
```

Valid Values:

<code>t</code>	Expands the iterated instances when the nets are selected to save voltages.
<code>nil</code>	Does not iterate through the instances. Instead, saves the voltage data for the net connected to the first instance.

Default Value: `nil`

GUI Equivalent: `None`

Also see: [expandIteratedInstancesForCurrents](#)

msps.layout

- expandSchematicDevices
- fingeringNames
- ignoreBackAnnotatedDummyDevices
- includeLayoutParasitics
- includeSchematicEstimates
- individualInstCdfCallbacks
- mfactorNames
- mixSchEstWithLayoutParasitics
- netlistView
- referenceNet

Note: These variables are described in the *Parasitic Aware Design User Guide*.

individualInstCdfCallbacks

When expanding an m-factor or fingered device in the parasitic/LDE view, the tool executes a CDF callback associated with that device. This variable specifies if the CDF callback is to be executed for each instance of a device or only once for a device.

By default, this variable is set to `nil` and while building a parasitic/LDE view, Virtuoso does not execute the CDF callback for each instance of a device. Instead, it executes the callback only once for a device and the result is applicable to all the instances. This helps in optimizing the build process.

To execute the callback for each cell instance separately, set this variable to `t`.

Note: If your design has PODE devices, all devices must be instantiated one by one. For such cases, you must set this environment variable to `t`.

In `.cdsenv`:

```
mmps.layout individualInstCdfCallbacks boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("mmps.layout" "individualInstCdfCallbacks" 'boolean  
nil)
```

Valid Values:

<code>t</code>	Executes the CDF callback for each instance of a device separately.
<code>nil</code>	Executes the CDF callback for a device only once.

Default Value: `nil`

layoutEAD

- auxSearchPaths
- checkEM
- defaultJobPolicy
- doAutoLayerMapping
- excludedNets
- excludedNetsExceptions
- extractConnectivity
- extractNetsWithConstraints
- flattenOtherInstances
- highlightNetShapes
- includeEmptyNets
- loadSubconductorLayers
- processSettings
- processStrappedMosfets
- resistancePathLimit
- stampNullNetTiles
- updateMode
- updateStyle
- useCorner
- useSetup
- xlNetsOnly

auxSearchPaths

Specifies additional directories that are to be searched for EAD setup and process settings files.

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

Note: Setups and process settings files must be stored under the standard `.cadence` directory structure in the specified directories. For example, if you set `auxSearchPaths` to `"/home/login/my_ead"`, EAD looks for setup and process setting files in `"/home/login/my_ead/.cadence/dfII/EAD/1"`.

In `.cdsenv`:

```
layoutEAD auxSearchPaths string "path"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD" "auxSearchPaths" 'string "path")
```

Valid Values:

<i>string</i>	Paths to search EAD setup and process settings files. Separate multiple paths using a colon.

Default Value: (empty string)

GUI Equivalent: None

checkEM

Disables EM checking and display for subsequent changes to the layout. EM checking and all EM-related options are disabled.

In `.cdsenv`:

```
layoutEAD checkEM boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD" "checkEM" 'boolean t)
```

Valid Values:

<i>t</i>	EM checking and options are enabled.
<i>nil</i>	EM checking and options are disabled.

Default Value: *t*

GUI Equivalent: Command: *Run EM Check – Disable EM Checker* (EAD Browser toolbar)



defaultJobPolicy

Specifies the default job policy to be used to run extraction.

In `.cdsenv`:

```
layoutEAD defaultJobPolicy string "localhost"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD" "defaultJobPolicy" 'string "localhost")
```

Valid Values:

<code>localhost</code>	Name of the default job policy

Default Value: `localhost`

GUI Equivalent: Command: *Layout EAD Options – General* tab

Field: The *Job Policy Name* field in the *EAD Job Policy Editor* form

doAutoLayerMapping

Controls whether to automatically map a design layer to a layer in the EAD technology file if both the layers have the same name.

For example, if you set this environment variable to `t`, and both the design and EAD technology file have a layer named, 'Metal1', these layers will be automatically mapped without any explicit mapping defined in the EAD setup.

In `.cdsenv`:

```
layoutEAD doAutoLayerMapping boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD" "doAutoLayerMapping" 'boolean t)
```

Valid Values:

<code>t</code>	Layers having same name are automatically mapped.
<code>nil</code>	Layers are not automatically mapped. You need to manually specify the mapping by using either the EAD Process Settings form or the setup file (<code>.ini</code>).

Default Value: `t`

GUI Equivalent: None

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

excludedNets

Lists the nets that are to be excluded from parasitic extraction. Use this to specify any net (complete, incomplete, power, etc) that you do not want to extract.

In `.cdsenv`:

```
layoutEAD excludedNets string "string"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD" "excludedNets" 'string "string"')
```

Valid Values:

<i>string</i>	String listing the nets that are to be excluded from extraction. Separate the names with a space or a comma. You can use a wildcard to specify net names.

Default Value: Empty string.

GUI Equivalent: Command: *Layout EAD Options – General* tab

Field: *Excluded Nets*

excludedNetsExceptions

Lists the nets that are exceptions to the list of excluded nets specified using `excludedNets`. Basically, the extraction setting is overridden for the nets given in this list and they are displayed or extracted.

In `.cdsenv`:

```
layoutEAD excludedNetsExceptions string "string"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD" "excludedNetsExceptions" 'string "string"')
```

Valid Values:

<i>string</i>	String listing the nets that are exceptions to the list of excluded nets. Separate the names with a space or a comma. You can use a wildcard to specify net names.

Default Value: Empty string.

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

GUI Equivalent: Command: *Layout EAD Options – General* tab
Field: *Except Nets*

extractConnectivity

Controls whether layout shapes that are not explicitly assigned to a net, but need to be extracted with a net for accurate extraction results, are assigned the same net as any shape they touch. This assignment is used for parasitic extraction purposes only.

When C extraction is done, any shape with no net is treated as connected to ground.

Important Points to Note

- This variable is helpful in assigning nets to the null net shapes only in Pcells. It does not stamp null net shapes with sources other than Pcells. For non-Pcell shapes, you can use [stampNullNetTiles](#).
- By setting this variable to `none`, you can prevent the extraction of incorrect shapes when the device instances, which are not to be extracted, are not distinguishable from the interconnect. Therefore, it is recommended to be careful when changing the value to `"pcells"`.
- If your design have guard rings, set this variable to `none`.
- This variable does not assign nets to the null net shapes on the derived layers or the local interconnect layers.

In `.cdsenv`:

```
layoutEAD extractConnectivity cyclic "pcells"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD" "extractConnectivity" 'cyclic "pcells")
```

Valid Values:

"pcells"	Net assignment is performed only for null net shapes in Pcell masters
"none"	No net assignment is performed

Default Value: "pcells"

GUI Equivalent: None

Also see: [stampNullNetTiles](#)

extractNetsWithConstraints

Controls extraction of nets displayed in the [EAD Browser](#). If this environment variable is set to `t`, the EAD Browser shows only those nets that have an EAD constraint defined on them. If the EAD Browser is already open and an EAD constraint is added to or deleted from a net, the [Summary Pane](#) is dynamically updated to show or hide that net.

In `.cdsenv`:

```
layoutEAD extractNetsWithConstraints boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD" "extractNetsWithConstraints" 'boolean t)
```

Note: Currently, if you set this environment variable after initializing the EAD session, the UI is not reloaded with the new setting. Therefore, it is recommended to apply the settings before opening Virtuoso Layout EAD.

Valid Values:

<code>t</code>	Displays only those nets in the EAD Browser that have EAD constraints applied on them.
<code>nil</code>	Displays the nets irrespective of any EAD constraints applied on them.

Default Value: `nil`

GUI Equivalent: None

flattenOtherInstances

Specifies whether the interconnect instances in the layout need to be flattened.

In `.cdsenv`:

```
layoutEAD flattenOtherInstances boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD" "flattenOtherInstances" 'boolean t)
```

Valid Values:

<code>t</code>	The interconnected shapes are flattened and not extracted as part of a source net.
<code>nil</code>	The interconnected shapes are not flattened. They are extracted as part of a source net.

Default Value: `nil`

GUI Equivalent: `None`

highlightNetShapes

Controls which types of net shapes (see [hierarchyShapeType](#)) are highlighted in the layout canvas when you click a net in the EAD Browser.

In `.cdsenv`:

```
layoutEAD highlightNetShapes string "rb"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD" "highlightNetShapes" 'string "rb")
```

Valid Values:

<code>string</code>	Any combination of the following letters: <code>r</code> – regular net shapes <code>b</code> – background net shapes <code>i</code> – invisible net shapes

Default Value: `rb` Regular and background net shapes are selected in the canvas when you click a net in the EAD Browser

GUI Equivalent: `None`

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

includeEmptyNets

Enables or disables display of empty nets, that is, the nets without any terminal, instance terminal, shape, or via on them, in the Summary pane of the EAD Browser.

In `.cdsenv`:

```
layoutEAD includeEmptyNets boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD" "includeEmptyNets" 'boolean t)
```

Valid Values:

<code>t</code>	Displays empty nets in the Summary pane of the EAD Browser.
<code>nil</code>	Does not display empty nets in the Summary pane of the EAD Browser.

Default Value: `t`

GUI Equivalent: None

loadSubconductorLayers

Controls whether to include the `sub_conductor` layers in the ICT file. When included, the `sub_conductor` layers are available to create the mapping from Virtuoso layers to ICT layers in the EAD Process Settings form. Any geometry mapped to a `sub_conductor` layer is extracted according to the corresponding parameters in the ICT file. When not included, only the parent conductor layer is available for mapping and extraction.

In `.cdsenv`:

```
layoutEAD loadSubconductorLayers boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD" "loadSubconductorLayers" 'boolean t)
```

Valid Values:

<code>t</code>	Includes the <code>sub_conductor</code> layers in the ICT file
<code>nil</code>	Does not include the <code>sub_conductor</code> layers in the ICT file

Default Value: `t`

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

GUI Equivalent: None

processSettings

Specifies the process settings file to be used in the current session.

In `.cdsenv`:

```
layoutEAD processSettings string "file_name"
```

In `.cdsinit` or the CIW:

```
envSetVal( "layoutEAD" "processSettings" 'string "file_name")
```

Valid Values:

string Name of the process settings file to use.

Default Value: Empty string

GUI Equivalent: Command: *Layout EAD Options – General* tab

Field: *Process Settings*

processStrappedMosfets

EAD splits the current across multiple fingers of a transistor except when the transistor Pcell source or drain terminals are strapped together inside the Pcell. In that case, there is only one terminal defined for source and drain, respectively. When transferring the currents for EM analysis, EAD assigns all the current to the device terminal, which may not be correct. Set this variable to `t` to force EAD to split the current evenly between the fingers.

In `.cdsenv`:

```
layoutEAD processStrappedMosfets boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "layoutEAD" "processStrappedMosfets" 'boolean t)
```

Valid Values:

<code>t</code>	Current is divided evenly between the fingers of the device.
<code>nil</code>	Current is assigned to the device terminals.

Default Value: `nil`

GUI Equivalent: `None`

resistancePathLimit

Specifies the maximum number of resistance paths to evaluate and show in the EAD Browser.

In `.cdsenv`:

```
layoutEAD resistancePathLimit int 100
```

In `.cdsinit` or the CIW:

```
envSetVal( "layoutEAD" "resistancePathLimit" 'int 100)
```

Valid Values:

<code>int</code>	Positive integer value specifying the maximum number of resistance paths to evaluate.

Default Value: `100`

GUI Equivalent: `None`

stampNullNetTiles

Specifies whether shapes that are not assigned to a net in the layout will be extracted because they physically connect to some shape other that does have a net.

Setting this variable to `t` is particularly helpful on layouts that contain subcells with only routing shapes (no devices).

During C extraction, any shape with no net is treated as connected to ground.

In `.cdsenv`:

```
layoutEAD stampNullNetTiles boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD" "stampNullNetTiles" 'boolean t)
```

Valid Values:

<code>t</code>	The unassigned shapes are extracted as part of a source net
<code>nil</code>	The unassigned shapes are not extracted as part of a source net

Default Value: `t`

GUI Equivalent: None

Also see: [extractConnectivity](#)

updateMode

Specifies whether the extraction and EM checking results displayed in the EAD Browser are updated **automatically** as the layout is altered, or **manually** by clicking the *Update Parasitics & EM* button in the EAD Browser toolbar.

In `.cdsenv`:

```
layoutEAD updateMode cyclic "manual"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD" "updateMode" 'cyclic "manual")
```

Valid Values:

<code>manual</code>	Extraction and EM checking are run manually
<code>auto</code>	Extraction and EM checking are run automatically

Default Value: `manual`

GUI Equivalent: *Update Parasitics & EM – Manual Update Mode* and *Automatic Update Mode* (EAD Browser toolbar)



updateStyle

Specifies whether parasitic extraction is performed continuously, in discrete steps, or only after the current action completes.

In `.cdsenv`:

```
layoutEAD updateStyle cyclic "discrete"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD" "updateStyle" 'cyclic "discrete")
```

Valid Values:

<code>discrete</code>	Extraction is performed in discrete steps.
<code>continuous</code>	Extraction is performed continuously.
<code>actionComplete</code>	Extraction is performed when the current action completes.

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

Default Value: `discrete`

GUI Equivalent: `None`

useCorner

Specifies the extraction corner which is mapped to an ICT file or eadTechFile to be used for extraction and the source of the EM models to be used for EM Checking. This variable is used to initialize EAD in non-GUI flows, such as the pre-EM flow.

In `.cdsenv`:

```
layoutEAD useCorner string "C0"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD" "useCorner" 'string "C0")
```

Valid Values:

<code>string</code>	Corner name

Default Value: `""`

GUI Equivalent: The *Extraction Corner* field in the Pre-EM Setup form.

useSetup

Specifies the name of the file containing the setup options.

In `.cdsenv`:

```
layoutEAD useSetup string "path"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD" "useSetup" 'string "path")
```

Valid Values:

<code>string</code>	Path to the setup file to be used when EAD is invoked; for example: <code>"./cadence/dfII/EAD/1/setup/gpdk45.ini"</code>

Default Value: `""`

GUI Equivalent: The *Extraction Setup* field in the Pre-EM Setup form.

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

xlNetsOnly

Specifies that EAD examines only those nets that are part of an OpenAccess design.

In `.cdsenv`:

```
layoutEAD xlNetsOnly boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD" "xlNetsOnly" 'boolean t)
```

Valid Values:

<code>t</code>	EAD examines only nets that are part of an OpenAccess design.
<code>nil</code>	EAD examines all nets.

Default Value: `t`

GUI Equivalent: `None`

layoutEAD.calc

showEstimatedResistance

showEstimatedResistance

```
layoutEAD.calc showEstimatedResistance boolean {t | nil}
```

Description

Specifies whether to display the *Est. Resistance* column in the results table of the EM Calculator.

The default value is `nil`, which indicates that the *Est. Resistance* column is hidden.

The *Est. Resistance* column displays the estimated resistance of the layer.

Resistance of a metal layer depends on the *Length*, *Width*, and *EM Temperature* fields, whereas resistance of a via layer depends on the *Via Type*, *Cuts*, and *EM Temperature* fields.

If the *Length* field is specified as a condition, for example:

- $L < 5$ or $L \leq 5$, the length is considered as 5 when calculating estimated resistance.
- $L > 5$ or $L \geq 5$, the length is considered as 5 when calculating the estimated resistance.
- $5 < L < 7$ or $5 \leq L \leq 7$, the length is considered as $(5+7)/2$ when calculating the estimated resistance.

GUI Equivalent

The *Est. Resistance* column in the results table of EM Calculator.

Examples

```
envGetVal("layoutEAD.calc" "showEstimatedResistance")  
envSetVal("layoutEAD.calc" "showEstimatedResistance" 'boolean t)
```

Related Topics

Results Table

layoutEAD.em

- autoDatasetCustomProc
- autoDatasetDeviceLengthPropList
- autoDatasetMacroProp
- autoDatasetDeviceWidthPropList
- autoDatasetDistributionMethod
- adjustedCurrentMsgRelTol
- currentDensityMPVDefault
- customSHESetupFile
- cdfPercentage
- computeIRDropWhenLoadingSetup
- checkResWidthAndEMLength
- defaultCurrent
- deltaT
- distributeTermCurrent
- dynamicAnalysis
- eadVerbosityLevel
- emViolThresholdPercent
- enableDynamicRead
- enableSelfHeatingEffects
- enableViaElseRule
- forceSegmentTotal
- icemLayerMapFile
- includeDeltaT_Margin
- lifetime
- lifetimeUnits

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

- metalRegionSearchExtent
- minCurrent
- odRegionSearchExtent
- preloadWaveforms
- preserveResistorCurrents
- printAdjustedCurrents
- resistorLimit
- resPathsResistorLimit
- rmsMetalLineNumAC
- rmsMetalLineNumDC
- scale
- scalingFactor
- scratchLib
- segmentLength
- sheSetupFile
- setTermNodeAsReference
- skipViaStackMetalResistor
- sumThreshold
- termPinLayersForEMCheck
- useDatasetClipInfo
- useDatasetTemperature
- useMaxRule
- viaOverlapPercentage
- warnOnEmLayer
- warnOnEmLimit
- writeEMResultsToOA
- preEMCheckLayers

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

- preEMDescendIntoPCell
- preEMDetailReport
- preEMLogFile
- preEMViolThresholdPercent
- minResistorCurrent

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

autoDatasetCustomProc

layoutEAD.em autoDatasetCustomProc string *nameOfTheUserDefinedFunction*

Description

Specifies the name of the user-defined function to be called for custom current distribution strategy.

Consider that you have defined a custom procedure named, `autoDatasetUserFunc` as follows:

```
(define (autoDatasetUserFunc t_dataSetName d_masterCellViewId d_instId
  t_deviceTermName t_termName)  aspectRatio = userFuncToComputeAspectRatio(
  d_InstId)  aspectRatio ; returnValue)
```

This user-defined function is run for each device terminal connected to the net and returns a floating-point value. The current among the device terminals is distributed in the proportion of the values returned by the user-defined function.

Consider that there are three device terminals, `M0:D`, `M1:D`, `M2:S`, connected to a net `N0`. Now, if the function returns 1, 2, 3 values for these device terminals respectively, then current is distributed in the proportion of $1/6$, $2/6$, $3/6$, respectively, as described in the following table.

Total current	100mA		
Device terminals	M0:D	M1:D	M2:S
Return values for each terminal	1.0	2.0	3.0
Ratio of current distribution	$1/6$	$2/6 = 1/3$	$3/6 = 1/2$
Current distribution	16.66mA	33.33mA	50mA

GUI Equivalent

None

Examples

```
envGetVal("layoutEAD.em" "autoDatasetCustomProc")
envSetVal("layoutEAD.em" "autoDatasetCustomProc" 'string "autoDatasetUserFunc")
```

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

Related Topics

[Generating EAD Dataset](#)

[autoDatasetDeviceLengthPropList](#)

[autoDatasetMacroProp](#)

[autoDatasetDeviceWidthPropList](#)

autoDatasetDeviceLengthPropList

```
layoutEAD.em autoDatasetDeviceLengthPropList string "listOfLengthProperties"
```

Description

Specifies a list of length properties for a device. These properties are used to find out the device length, which is then used to calculate the aspect ratio ($\text{width}/\text{length}$) for the device. The aspect ratio can be used as a current distribution strategy in case of automatic dataset creation.

The default value is "1 L", which indicates property 1 is searched first to find the device length. If the property 1 is not found, then property L is used.

GUI Equivalent

None

Examples

```
envGetVal("layoutEAD.em" "autoDatasetDeviceLengthPropList")
envSetVal("layoutEAD.em" "autoDatasetDeviceLengthPropList" 'string "length
Length")
```

Related Topics

[Generating EAD Dataset](#)

[autoDatasetCustomProc](#)

[autoDatasetMacroProp](#)

[autoDatasetDeviceWidthPropList](#)

autoDatasetMacroProp

```
layoutEAD.em autoDatasetMacroProp string "listOfLengthProperties"
```

Description

Specifies the property name that is read on a macro Pcell to find out aspect ratio of devices inside the Pcell.

GUI Equivalent

None

Examples

```
envGetVal("layoutEAD.em" "autoDatasetMacroProp")  
envSetVal("layoutEAD.em" "autoDatasetMacroProp" 'string "macroCellProp")
```

Related Topics

[Generating EAD Dataset](#)

[autoDatasetCustomProc](#)

[autoDatasetDeviceLengthPropList](#)

[autoDatasetDeviceWidthPropList](#)

autoDatasetDeviceWidthPropList

```
layoutEAD.em autoDatasetDeviceWidthPropList string "listOfWidthProperties"
```

Description

Specifies a list of width properties for a device. These properties are used to find out the device width, which is then used to calculate the aspect ratio ($\text{width}/\text{length}$) for the device. The aspect ratio can be used as a current distribution strategy in case of automatic dataset creation.

The default value is " w W ", which indicates property w is searched first to find the device width. If the property w is not found, then property W is used.

GUI Equivalent

None

Examples

```
envGetVal("layoutEAD.em" "autoDatasetDeviceWidthPropList")  
envSetVal("layoutEAD.em" "autoDatasetDeviceWidthPropList" 'string "width Width")
```

Related Topics

[Generating EAD Dataset](#)

[autoDatasetCustomProc](#)

[autoDatasetDeviceLengthPropList](#)

[autoDatasetMacroProp](#)

autoDatasetDistributionMethod

```
layoutEAD.em autoDatasetDistributionMethod cyclic {Aspect ratio| Equal | Custom}
```

Description

Specifies the current distribution method to use when generating EAD datasets.

Valid values are:

- **Aspect ratio:** Distributes the specified current among the devices in proportion to their aspect ratio, that is width/length. For example, if the devices M10 and M5 have aspect ratios 150 and 75, respectively, the M10 gets twice as much current as M5 does.
- **Equal:** Distributes the specified current equally among different device terminals connected to the net.
- **Custom:** Distributes the specified current among different device terminals according to the custom current distribution defined in the user-defined function specified by the environment variable autoDatasetCustomProc.

The default value is `Aspect ratio`.

GUI Equivalent

None

Examples

```
envGetVal("layoutEAD.em" "autoDatasetDistributionMethod")
envSetVal("layoutEAD.em" "autoDatasetDistributionMethod" 'cyclic "Equal")
envSetVal("layoutEAD.em" "autoDatasetDistributionMethod" 'cyclic "Custom")
```

Related Topics

Generating EAD Dataset

autoDatasetDeviceLengthPropList

autoDatasetMacroProp

autoDatasetDeviceWidthPropList

adjustedCurrentMsgRelTol

Specifies the relative tolerance limit to determine whether to print the adjusted current message if adjustment is not within the specified relative tolerance.

In `.cdsenv`:

```
layoutEAD.em adjustedCurrentMsgRelTol float 0
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "adjustedCurrentMsgRelTol" 'float 0)
```

Valid Values:

float	Floating point value specifying the relative tolerance limit to be applied to filter the terminals.

Default Value: 0

GUI Equivalent: None

When source and sink currents for a net do not sum up to zero, EAD adjusts these current values to make them KCL compliant and displays a warning message in CIW. The warning message also prints the original and adjusted values for each terminal, if the `printAdjustedCurrents` environment variable is set to `t`.

When the number of terminals printed in the message are too high, you may want to selectively view only those terminals for which the difference between the original and adjusted currents is significantly high. Such terminals can be filtered based on a user-specified relative tolerance value.

Consider the following example, where a non-KCL compliant terminal, `Ibias`, is printed in the warning message:

Adjusted currents for 'Iavg (AVG)' are as follows:

Term Name	Specified Value	Adjusted Value
Ibias	1A	0.9A

Here, the difference between the specified and adjusted current values is 0.1A ($=1\text{A}-0.9\text{A}$), which is 10% of the specified value 1A. If you do not want to print such terminals for which the difference between the specified and adjusted value is within 10% of the specified value, set the relative tolerance value to 0.1, as follows:

```
envSetVal("layoutEAD.em" "adjustedCurrentMsgRelTol" 'float 0.1)
```

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

customSHESetupFile

```
layoutEAD.em customSHESetupFile string "pathToCustomSHESetupFile"
```

Description

Specifies the path to the custom setup file that contains constant values and custom equations to be used while calculating delta-T in the custom self-heating effects (SHE) flow.

The default value is an empty string.

When using the custom SHE flow, custom equations are applied on the `simDeltaT` temperature that is obtained from the simulator.

To use the custom SHE flow, ensure that both the environment variables, `sheSetupFile` and `customSheSetupFile`, are defined.

GUI Equivalent

None

Examples

```
envGetVal("layoutEAD.em" "customSHESetupFile")
envSetVal("layoutEAD.em" "customSHESetupFile" 'string "/servers/user/SHE/
custom.sh")
```

Related Topics

[Viewing Delta T Information](#)

[sheSetupFile](#)

currentDensityMPVDefault

By default, the unit for current density limit for via layers depends on the value of `currentDensityMPV` in the EM rules. If the EM rules contain `currentDensityMPV=true`, the current density unit is mA/via. If the EM rules contain `currentDensityMPV=false`, the current density unit is mA/um.

If `currentDensityMPV` is not set in the EM rules, unit to be used for current density limit depends on the value of this environment variable.

Note: This variable is considered only when the EM rules are read from an `emDataFile`. It does not apply to EM rules read from ICT or ICT-EM files.

In `.cdsenv`:

```
layoutEAD.em currentDensityMPVDefault boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "currentDensityMPVDefault" 'boolean t)
```

Valid Values:

<code>t</code>	If <code>currentDensityMPV</code> is not set in <code>currentDensityMPVDefault</code> EM rules, the current density unit is taken as mA/um.
<code>nil</code>	If <code>currentDensityMPV</code> is not set in the EM rules, the current density unit is taken as mA/via.

Default Value: `nil`

GUI Equivalent: None

cdfPercentage

Defines the value to be used by the `cdf_percentage` variable for the calculation of CDF (Cumulative Distribution Factor) required for determining the maximum current limit.

Note: If EM rules for a process node depend on the `cdf_percentage` variable, ensure that the value of the environment variable, `cdfPercentage`, is set to a value greater than zero.

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

For more details, refer to *Rules for Defining EM Parameters* in the *Voltus-Fi Custom Power Integrity Solution XL User Guide*.

In `.cdsenv`:

```
layoutEAD.em cdfPercentage float 0.0
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "cdfPercentage" 'float 0.0)
```

Valid Values:

A numerical value between 0.0 and 100.0

Default Value: 0.0

GUI Equivalent: None

computeIRDropWhenLoadingSetup

When the environment variable `writeEMResultsToOA` is set to `t`, electromigration results are saved to the OA database while saving the layout view. When you reopen the design in Virtuoso Layout EAD, the saved results are loaded and IR drop is also calculated for all resistance paths.

For designs that have nets with large number of resistors, this calculation may take time. The environment variable `computeIRDropWhenLoadingSetup` enables you to choose whether to calculate IR drop along resistance paths in the layout when the EAD Browser is initialized.

In `.cdsenv`:

```
layoutEAD.em computeIRDropWhenLoadingSetup boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "computeIRDropWhenLoadingSetup"  
          'boolean t)
```

Valid Values:

`t` IR drop is calculated when a design is reopened and the EAD browser is initialized.

`nil` IR drop is not calculated when a design is reopened and the EAD browser is initialized.

Default Value: `t`

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

GUI Equivalent: None

checkResWidthAndEMLength

```
layoutEAD.em checkResWidthAndEMLength boolean {t | nil}
```

Description

Specifies whether to compute the width for a single-tile resistor in the orthogonal direction of the EM length or in the same direction of the EM length.

The default value is `nil`, which indicates that width of a single-tile resistor is computed in the orthogonal direction of the EM length.

GUI Equivalent

None

Examples

```
envGetVal("layoutEAD.em" "checkResWidthAndEMLength")
envSetVal("layoutEAD.em" "checkResWidthAndEMLength" 'boolean t)
```

defaultCurrent

Specifies the default terminal current to use when computing EM effects.

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

Important

If you specify 1mA as the default current value, all the terminals will become non-KCL compliant and the currents will be adjusted to value 0. In this case, EM checks will not be run.

In `.cdsenv`:

```
layoutEAD.em defaultCurrent float 0.0
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "defaultCurrent" 'float 0.0)
```

Valid Values:

float Floating point value specifying the default current.

Default Value: 0.0

GUI Equivalent: Command: *Layout EAD Options – EM* tab

Field: *Default*

deltaT

Specifies in degrees Celsius the increase in junction temperature due to joule heating.

In `.cdsenv`:

```
layoutEAD.em deltaT float 5
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "deltaT" 'float 5)
```

Valid Values:

float Floating point value specifying the increase in junction temperature due to joule heating.

Default Value: 5

GUI Equivalent: Command: *Layout EAD Options – EM* tab

Field: *Delta T*

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

distributeTermCurrent

Produces a more accurate modeling of the terminal region, especially with wide source/drain terminals that are strapped by a row of vias. Set this variable to `t` if there is a suspected false EM violation in or near a terminal geometry.

In `.cdsenv`:

```
layoutEAD.em distributeTermCurrent boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "distributeTermCurrent" 'boolean t)
```

Valid Values:

<code>t</code>	The terminal current is distributed equally among the nodes within a terminal.
<code>nil</code>	The terminal current is assigned to a single node within the terminal geometry. This node is usually located in the center of the terminal shape and on the lower layer, if there are multiple layers in the terminal geometry.

Default Value: `nil`

GUI Equivalent: `None`

dynamicAnalysis

Enables dynamic EM analysis.

In `.cdsenv`:

```
layoutEAD.em dynamicAnalysis boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "dynamicAnalysis" 'boolean nil)
```

Valid Values:

<code>t</code>	Dynamic EM analysis is enabled.
<code>nil</code>	Dynamic EM analysis is disabled.

Default Value: `nil`

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

GUI Equivalent: Command: *Layout EAD Options – EM* tab
Field: *Dynamic Analysis*

eadVerbosityLevel

```
layoutEAD.em eadVerbosityLevel cyclic {low | high}
```

Description

Specifies whether to display multiple warning messages for each layer or a single warning message per unsupported rule for all layers.

The default value is `low`, which indicates that only a single warning message per unsupported layer is displayed for all the layers.

This environment variable works only if the `warnOnEmLimit` is set to `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutEAD.em" "eadVerbosityLevel")  
envSetVal("layoutEAD.em" "eadVerbosityLevel" 'cyclic "high")
```

Related Topic

[warnOnEmLimit](#)

emViolThresholdPercent

Specifies the threshold value of $J/J_{\max}$ ratio in percentage beyond which the EM analysis reports them as EM violations.

In `.cdsenv`:

```
layoutEAD.em emViolThresholdPercent float 100.0
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "emViolThresholdPercent" 'float 100.0)
```

Valid Values:

float	Any floating point value.

Default Value: 100.0

GUI Equivalent: None

enableDynamicRead

Specifies that currents for terminals that do not have datasets are dynamically read and transferred from the schematic source before an EM check is run.

Switch on this option when working with partial layouts that have incomplete nets to ensure that currents are loaded for any new terminals added to the resistor network since the last EM check.

In `.cdsenv`:

```
layoutEAD.em enableDynamicRead boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "enableDynamicRead" 'boolean nil)
```

Valid Values:

t	Dynamic read and transfer are enabled.
nil	Dynamic read and transfer are disabled.

Default Value: nil

GUI Equivalent: None

enableSelfHeatingEffects

Enables the self-heating flow in EAD.

In `.cdsenv`:

```
layoutEAD.em enableSelfHeatingEffects boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "enableSelfHeatingEffects" 'boolean  
nil)
```

Valid Values:

<code>t</code>	Considers self-heating effects.
<code>nil</code>	Does not consider self-heating effects.

Default Value: `nil`

GUI Equivalent: *Enable Self Heating Effects* check box on the EM tab of the Layout EAD Options form

enableViaElseRule

Controls how EM limits are calculated when applying a PWL-based rule if no exact match of via area is found in the EM rule file.

In `.cdsenv`:

```
layoutEAD.em enableViaElseRule boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "enableViaElseRule" 'boolean nil)
```

Valid Values:

<code>t</code>	■ If an exact match of the via area is not found, the default rule is applied. ■ If there is no default rule, interpolation is carried out to calculate the EM limit.
<code>nil</code>	If an exact match of the via area is not found, interpolation is carried out to calculate the EM limit.

Default Value: `nil`

GUI Equivalent: `None`

forceSegmentTotal

Overrides the blech length calculation methods defined in the ICT file or specified by the segmentLength variable and sets the calculation method to `total`. In this method, the blech length is calculated as the accumulated length of all the resistors associated with a contiguous metal geometry.

In `.cdsenv`:

```
layoutEAD.em forceSegmentTotal boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "forceSegmentTotal" 'boolean t)
```

Valid Values:

<code>t</code>	Sets the blech length calculation method to <code>total</code> . In this method, the blech length is calculated as the accumulated length of all the resistors associated with a contiguous metal geometry.
<code>nil</code>	Uses the blech length calculation method as defined in the ICT file or as specified by the <u>segmentLength</u> variable.

Default Value: `nil`

GUI Equivalent: `None`

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

icitemLayerMapFile

Specifies the path to EM Layer Map file that contains the layer mapping information.

In `.cdsenv`:

```
layoutEAD.em icitemLayerMapFile string
"<path_to_the_EM_Layer_Map_file>"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "icitemLayerMapFile" 'string
"<path_to_the_EM_Layer_Map_file>")
```

Valid Values:

<code>string</code>	Path to the EM Layer Map file.

Default Value:	<code>""</code>	An empty string.

GUI Equivalent:	<i>EM Layer Map File</i> field on the <i>Corners</i> tab of the EAD Process Settings form.

Note: This field is available only when the *EM Data Source* is set to *ICT-EM*.

includeDeltaT_Margin

Enables or disables addition of the delta-T value specified on the EAD Options form to the EM temperature while finding the temperature derating factor.

In `.cdsenv`:

```
layoutEAD.em includeDeltaT_Margin boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "includeDeltaT_Margin" 'boolean t)
```

Valid Values:

<code>t</code>	Delta-T value is added to the EM temperature to calculate the temperature derating factor

<code>nil</code>	The temperature derating factor is equal to the EM temperature

Default Value:	<code>nil</code>

GUI Equivalent:	None

lifetime

The lifetime to be used for risk-based EM analysis. In risk-based EM analysis, MTTF (Mean Time To Failure) is first calculated using Black's equation. The calculated value is then used together with the specified lifetime to determine the probability of EM failure. Specify the units using the lifetimeUnits environment variable.

In `.cdsenv`:

```
layoutEAD.em lifetime float 10
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "lifetime" 'float 10)
```

Valid Values:

float	Floating point value specifying the number of <u>lifetimeUnits</u> .

Default Value: 10

GUI Equivalent: Command: *Layout EAD Options – EM* tab

Field: *Lifetime*

ignoreResistorLimit

Controls the comparison of the resistor count against the specified resistor limit while running EM checks.

In `.cdsenv`:

```
layoutEAD.em ignoreResistorLimit boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "ignoreResistorLimit" 'boolean t)
```

Valid Values:

t	The resistor count is not compared with the resistor limit specified in the <i>Resistor Limit</i> field on the EM tab of the Layout EAD Options form.
nil	The resistor count is compared with the resistor limit specified in the <i>Resistor Limit</i> field on the EM tab of the Layout EAD Options form.

Default Value: nil

GUI Equivalent: Command: *Layout EAD Options – EM* tab
Field: *Ignore Resistor Limit*

lifetimeUnits

The units for the lifetime value for EM analysis.

In `.cdsenv`:

```
layoutEAD.em lifetimeUnits cyclic "years"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "lifetimeUnits" 'cyclic "years")
```

Valid Values:

years	The <u>lifetime</u> value is measured in years.
hours	The <u>lifetime</u> value is measured in hours.

Default Value: years

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

GUI Equivalent: Command: *Layout EAD Options – EM* tab

Field: *Lifetime* units pull-down

metalRegionSearchExtent

Used to identify the extent of the search region around a resistor to find any other resistor that reports the maximum change in temperature (delta T) due to RMS current. To get the dimensions of the search region, EAD multiplies the value specified for this variable with the minimum pitch (minimum width + minimum spacing) for the conductor layer on which the resistor is extracted. If EAD finds any other resistor with a higher delta T value, that value is used in the computation of the final delta T for the conductor.

Note: This variable is used when the *Enable Self Heating Effects* check box on the *EM* tab of the EAD Options form is selected. For more details, refer to Viewing Delta T Information.

In `.cdsenv`:

```
layoutEAD.em metalRegionSearchExtent float 15
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "metalRegionSearchExtent" 'float 15)
```

Valid Values:

float

Floating point value specifying the search extent for the metal region.

Default Value: 15

GUI Equivalent: None

minCurrent

The minimum threshold limit for *Idc* and *Iavg* currents of the terminals and instance terminals of a net for using the net in EM calculation. If the current of all terminals and instance terminals of a net is below the threshold, then EM calculation is skipped for that net.

In `.cdsenv`:

```
layoutEAD.em minCurrent float 0.0
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "minCurrent" 'float 0.0)
```

Valid Values:

<i>float</i>	Floating point value specifying the minimum terminal current for computing EM effects.

Default Value: 0

GUI Equivalent: Command: *Layout EAD Options – EM* tab

Field: *Minimum*

odRegionSearchExtent

Specifies the dimensions (in microns) by which the bounds of a diffusion shape needs to be extended to check the overlap with the metal layer on which resistor is extracted. This variable is considered while calculating the delta T values for the conductor.

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

Note: This variable is used when the *Enable Self Heating Effects* check box on the *EM* tab of the EAD Options form is selected. For more details, refer to Viewing Delta T Information.

In `.cdsenv`:

```
layoutEAD.em odRegionSearchExtent float 1.8
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "odRegionSearchExtent" 'float 1.8)
```

Valid Values:

<code>float</code>	Floating point value specifying the search extent to check for overlap of a diffusion shape with the metal layer.

Default Value: 1.8

GUI Equivalent: None

preloadWaveforms

Specifies whether all waveforms for dynamic EM analysis are loaded at the start, or are loaded only during the EM analysis and then unloaded again afterwards.

In `.cdsenv`:

```
layoutEAD.em preloadWaveforms boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "preloadWaveforms" 'boolean t)
```

Valid Values:

<code>t</code>	All waveforms for dynamic EM analysis are loaded at the start.
<code>nil</code>	Waveforms for dynamic EM analysis are loaded only during the dynamic EM analysis and are unloaded again when the analysis is complete.

Default Value: `nil`

GUI Equivalent: None

preserveResistorCurrents

Preserves intermediate currents for resistors during dynamic EM analysis so that you can plot these in a Virtuoso Visualization and Analysis XL window. The default is `nil`, which means that currents are not preserved. When set to `t`, you can view the currents through any resistor in the parasitic network by clicking the right mouse button on one of the values in either the *Dynamic-Rms* or *Dynamic-Peak* column and selecting *Plot Resistor Currents*. For example, you might first run an EM check without preserving currents (the default setting) and then enable this option to rerun the check on selected nets of interest.

In `.cdsenv`:

```
layoutEAD.em preserveResistorCurrents boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "preserveResistorCurrents" 'boolean t)
```

Valid Values:

<code>t</code>	Preserves intermediate resistor currents during dynamic EM analysis.
<code>nil</code>	Intermediate resistor currents are not preserved during dynamic EM analysis.

Default Value: `nil`

GUI Equivalent: `None`

printAdjustedCurrents

Enables printing of messages in CIW that inform about the current adjustments done by the solver to correct KCL non-compliance during EM checking. In case of incomplete routing, or when EM currents are entered manually, EAD adjusts the current values to make them KCL compliant and displays a warning message in the CIW, as shown below:

```
*WARNING* Current sources or sinks for net outp do not sum to zero (possibly due
to incomplete routing). Some currents have been adjusted to present the most
accurate information possible for dataset dataset_0.
```

In addition to this message, the original and adjusted value for each terminal are also displayed when this environment variable is set to `t`.

```
Adjusted currents for 'Iavg (AVG)' are as follows:
Term Name      Specified Value      Adjusted Value
outp           -3.24805e-11A          0A
|M18.7:G       -3.24805e-11A          0A
|M18.6:G       -3.24805e-11A          0A
|M18.5:G       -3.24805e-11A          0A
|M12.6:G       8.87067e-10A           0A
```

In `.cdsenv`:

```
layoutEAD.em printAdjustedCurrents boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "printAdjustedCurrents" 'boolean t)
```

Valid Values:

<code>t</code>	Prints the messages for current adjustments done during EM checking in CIW.
<code>nil</code>	Does not print the messages for current adjustments done during EM checking in CIW.

Default Value: `t`

GUI Equivalent: `None`

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

resistorLimit

Specifies the maximum number of resistors that EM checking can handle on a particular net. If the number of resistors exceeds this value, EM checking is not run for the net.

In `.cdsenv`:

```
layoutEAD.em resistorLimit int 50000
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "resistorLimit" 'int 50000)
```

Valid Values:

<i>int</i>	Positive integer value specifying the number of resistors.

Default Value: 50000

GUI Equivalent: Command: *Layout EAD Options – EM* tab

Field: *Resistor Limit*

Related environment variable: [ignoreResistorLimit](#)

resPathsResistorLimit

Specifies the resistor limit up to which the resistance paths are resolved for a net while populating details in the *Resistance Paths* table on the R tab in EAD Browser. For more details, refer to [R tab](#).

In `.cdsenv`:

```
layoutEAD.em resPathsResistorLimit int 50000
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "resPathsResistorLimit" 'int 50000)
```

Valid Values:

<i>int</i>	Positive integer value specifying the maximum resistor limit.

Default Value: 50000

GUI Equivalent: None

rmsMetalLineNumAC

Specifies the RMS metal line number for signal nets. You can set this value to override the default value of the metal line number taken from the header section of an ICT file.

This number is used to look up the metal line rating tables in the EM models.

In `.cdsenv`:

```
layoutEAD.em rmsMetalLineNumAC int 2
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "rmsMetalLineNumAC" 'int 2)
```

Valid Values:

int Zero or a positive integer value

Default Value: 0

GUI Equivalent: None

rmsMetalLineNumDC

Specifies the RMS metal line number for the power or ground nets. You can set this value to override the default value of the metal line number taken from the header section of an ICT file.

This number is used to look up the metal line rating tables in the EM models.

In `.cdsenv`:

```
layoutEAD.em rmsMetalLineNumDC int 2
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "rmsMetalLineNumDC" 'int 2)
```

Valid Values:

int Zero or a positive integer value

Default Value: 0

GUI Equivalent: None

scale

Specifies that terminal currents are to be scaled when computing EM effects. You specify the factor using the scalingFactor environment variable.

In `.cdsenv`:

```
layoutEAD.em scale boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "scale" 'boolean t)
```

Valid Values:

<code>t</code>	Terminal currents are to be scaled using the <u>scalingFactor</u> .
<code>nil</code>	Terminal currents are not to be scaled.

Default Value: `t`

GUI Equivalent: `None`

scalingFactor

Specifies the scaling factor to be used when computing EM effects. All terminal currents are scaled by this value. This factor is considered only when the scale environment variable is set to `t`.

In `.cdsenv`:

```
layoutEAD.em scalingFactor float 1
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "scalingFactor" 'float 1)
```

Valid Values:

<code>float</code>	Floating point value specifying the scaling factor to be used.

Default Value: `1`

GUI Equivalent: Command: *Layout EAD Options – EM* tab
Field: *Scaling Factor*

scratchLib

Specifies the library in which the scratch layout is created when performing EM analysis on power grids. See the [Running EM Analysis on Power Grids](#) section for more information.

In `.cdsenv`:

```
layoutEAD.em scratchLib string
"<name_of_the_scratch_library>"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "scratchLib" 'string
"<name_of_the_scratch_library>")
```

Valid Values:

<i>string</i>	Name of the library where the scratch layout is created.

Default Value:	""	An empty string.

Note: By default, the scratch layout is created in same library that contains the original layout.

GUI Equivalent:	No

segmentLength

Specifies the default method for EM length calculation to be used when no method is specified in the ICT file. If a method is specified in the ICT file, that is used instead of the default method.

Note: If an EM length calculation method is given in the ICT file, you can override that to use `total` by setting the `forceSegmentTotal` variable to `t`.

In `.cdsenv`:

```
layoutEAD.em segmentLength cyclic "centerline"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "segmentLength" 'cyclic "centerline")
```

Valid Values:

<code>total</code>	EM length is calculated as the accumulated length of all the resistors associated with a contiguous metal geometry.
<code>centerline</code>	EM length is calculated as the longest path measured across a contiguous geometry from end-to-end along the centerline of the path.
<code>perimeter</code>	EM length is calculated as the longest path measured across a contiguous geometry from end-to-end along the outside edges of the path.

Default Value: `centerline`

GUI Equivalent: `None`

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

sheSetupFile

```
layoutEAD.em sheSetupFile string "pathToSHESetupFile"
```

Specifies the path to the setup file that contains the constant values to be used while calculating delta-T in the self-heating effects flow.

Description

Specifies the path to the setup file that contains constant values to be used while calculating delta-T in the self-heating effects flow.

The default value is an empty string.

GUI Equivalent

None

Examples

```
envGetVal("layoutEAD.em" "sheSetupFile")  
envSetVal("layoutEAD.em" "sheSetupFile" 'string "/servers/user/SHE/sheSetup.sh")
```

Related Topics

[Viewing Delta T Information](#)

[customSHESetupFile](#)

setTermNodeAsReference

Specifies whether to set the top-level term as the voltage reference node when running EM analysis.

In `.cdsenv`:

```
layoutEAD.em setTermNodeAsReference boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "setTermNodeAsReference" 'boolean nil)
```

Valid Values:

<code>t</code>	Top-level terms are used as voltage reference nodes.
<code>nil</code>	The default behavior of selecting voltage reference node will continue.

Default Value: `nil`

GUI Equivalent: `None`

skipViaStackMetalResistor

Specifies whether to skip the minor metal resistor part introduced between the two vias of a stacked via for EM analysis.

In `.cdsenv`:

```
layoutEAD.em skipViaStackMetalResistor boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "skipViaStackMetalResistor" 'boolean  
t)
```

Valid Values:

<code>t</code>	The metal resistor part between the vias of a stacked via is skipped for EM analysis.
<code>nil</code>	The metal resistor part between the vias of a stacked via is considered for EM analysis.

Default Value: `t`

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

GUI Equivalent: None

sumThreshold

Specifies the tolerance for the Kirchoff's Current Law (KCL) compliance check.

This environment variable is used only when there is no voltage source defined in the input electrical dataset, in which case, if the network is either incomplete or not KCL-compliant, the EM checker chooses the best voltage source to use for EM analysis.

In `.cdsenv`:

```
layoutEAD.em sumThreshold float 1e-8
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "sumThreshold" 'float 1e-8)
```

Valid Values:

float

Floating point value specifying the tolerance to be applied when checking KCL compliance.

Default Value: `1e-8`

GUI Equivalent: None

termPinLayersForEMCheck

```
layoutEAD.em termPinLayersForEMCheck boolean {t | nil}
```

Description

Specifies whether to enable mapping between nets and pin layers for EM check. The nets or terminals serve as the voltage source only for the pin layers defined in the mapping.

The default value is `nil`, which indicates that you cannot specify mapping between nets and pin layer for EM check.

If you set this environment variable to `t`, the `PinLayer` column is displayed in the *EM* tab of the EAD Browser where you can specify the name of the metal layer against a net.

GUI Equivalent

The *PinLayer* column in the EM tab.

Examples

```
envGetVal("layoutEAD.em" "termPinLayersForEMCheck")  
envSetVal("layoutEAD.em" "termPinLayersForEMCheck" 'boolean t)
```

Related Topics

[Specifying Mapping between Nets and Pin Layers for EM Check](#)

useDatasetClipInfo

Specifies if the information about the time interval for which the clipped waveform is to be used for EM analysis is to be taken from the dataset.

In `.cdsenv`:

```
layoutEAD.em useDatasetClipInfo boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "useDatasetClipInfo" 'boolean nil)
```

Valid Values:

<code>t</code>	The information about the time interval for waveform clipping is used from the dataset.
<code>nil</code>	The information about the time interval for waveform clipping is not used from the dataset. Instead, the values specified in the <i>From</i> and <i>To</i> fields in the <i>Dynamic Analysis</i> section of the <i>Layout EAD Options</i> form are used.

Default Value: `nil`

GUI Equivalent: Command: *Layout EAD Options – EM tab – Dynamic Analysis*
Field: *Use Dataset Value*

useDatasetTemperature

Specifies that the temperature specified in the simulation dataset is to be used when computing EM effects.

In `.cdsenv`:

```
layoutEAD.em useDatasetTemperature boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "useDatasetTemperature" 'boolean nil)
```

Valid Values:

<code>t</code>	The simulation dataset temperature is used.
<code>nil</code>	The simulation dataset temperature is not used.

Default Value: `nil`

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

GUI Equivalent: Command: *Layout EAD Options – EM tab – Temperature*
Field: *Use Dataset Value*

useMaxRule

Specifies whether the rule for which the J_{max} value is maximum is used to calculate EM limits, if more than one EM rules are satisfied for that resistor.

In `.cdsenv`:

```
layoutEAD.em useMaxRule boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "useMaxRule" 'boolean nil)
```

Valid Values:

<code>t</code>	The rule for which the J_{max} value is maximum is used to calculate EM limits.
<code>nil</code>	The rule for which the J_{max} value is minimum is used to calculate EM limits

Default Value: `nil`

GUI Equivalent: `None`

viaOverlapPercentage

Specifies the threshold value of overlapping area (in percentage) between vias based on which it is decided whether to consider them as stacked vias. If this environment variable is specified, `stack_up_via` and `stack_down_via` rules will be applied only to such vias.

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

For example, if the value of `viaOverlapPercentage` is set to `30.0`, only those vias which have at least 30% of overlapping areas are considered as stacked vias.

In `.cdsenv`:

```
layoutEAD.em viaOverlapPercentage float 20.0
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "viaOverlapPercentage" 'float 20.0)
```

Valid Values:

<code>float</code>	Floating point value specifying the threshold value of overlapping area (in percentage) between the vias.

Default Value: `20.0`

GUI Equivalent: `None`

warnOnEmLayer

Displays a warning message when Layout EAD encounters EM rules that are defined on a layer for which there is no matching layer in the ICT-EM file. Messages are displayed by default; set the environment variable to `nil` to suppress them.

Note: You must set this environment variable before starting EAD.

In `.cdsenv`:

```
layoutEAD.em warnOnEmLayer boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "warnOnEmLayer" 'boolean t)
```

Valid Values:

<code>t</code>	The messages are displayed.
<code>nil</code>	The messages are not displayed.

Default Value: `t`

GUI Equivalent: `None`

warnOnEmLimit

Displays a warning message when Layout EAD encounters EM limits that are not supported. You can also use this environment variable to control whether to display warning messages related to unsupported power rules in the ICT-EM file. Messages are displayed by default; set the environment variable to `nil` to suppress them.

Note: You must set this environment variable before starting EAD.

In `.cdsenv`:

```
layoutEAD.em warnOnEmLimit boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "warnOnEmLimit" 'boolean t)
```

Valid Values:

<code>t</code>	The warning messages are displayed.
<code>nil</code>	The warning messages are not displayed.

Default Value: `t`

GUI Equivalent: `None`

writeEMResultsToOA

Enables writing of the electromigration results to the OA database while saving the layout view. When you reopen the design in Virtuoso Layout EAD, the saved results are also loaded and displayed in the Summary pane of the EAD Browser.

Virtuoso Electrically Aware Design Flow Guide

Environment Variables



Set this variable before running EM checks.

In `.cdsenv`:

```
layoutEAD.em writeEMResultsToOA boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "writeEMResultsToOA" 'boolean t)
```

Valid Values:

<code>t</code>	Enables writing of the electromigration results to the OA database.
<code>nil</code>	Disables writing of the electromigration results to the OA database.

Default Value: `nil`

GUI Equivalent: `None`

preEMCheckLayers

Specifies the layer on which pre-EM check is performed.

In `.cdsenv`:

```
layoutEAD.em preEMCheckLayers string <name-of-the-layer>
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "preEMCheckLayers" 'string "<name-of-the-layer">")
```

Valid Values:

<code>string</code>	Name of the layer. For example: <code>"Metal1"</code>

Default Value: `""`

GUI Equivalent: The *Pre-EM Check Layers* field in the Pre-EM Setup form.

preEMDescendIntoPCell

Specifies whether to descend into a Pcell during the pre-EM check.

In `.cdsenv`:

```
layoutEAD.em preEMDescendIntoPCell boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "preEMDescendIntoPCell" 'boolean nil)
```

Valid Values:

<code>t</code>	Pcell instances are further descended during the pre-EM checks.
<code>nil</code>	Pcell instances are not descended during the pre-EM checks.

Default Value: `nil`

GUI Equivalent: `None`

preEMDetailReport

Specifies whether to generate the detailed report of pre-EM analysis.

In `.cdsenv`:

```
layoutEAD.em preEMDetailReport boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "preEMDetailReport" 'boolean t)
```

Valid Values:

<code>t</code>	Detailed report is generated.
<code>nil</code>	Detailed report is not generated.

Default Value: `t`

GUI Equivalent: `None`

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

preEMLogFile

Name of the log file containing the report of pre-EM analysis.

Note: This file is saved in the simulation results directory path. For example, if you used the maestro view, the log file will be saved in the following location:

```
/simulation/<LibraryName>/<CellName>/<ViewName>/results/maestro/  
<SimulationHistoryName>/<PreEMLogFileName>
```

In .cdsenv:

```
layoutEAD.em preEMLogFile string preEmCheck.log
```

In .cdsinit or the CIW:

```
envSetVal("layoutEAD.em" "preEMLogFile" 'string  
"preEmCheck.log")
```

Valid Values:

string	Name of the log file containing the report of pre-EM analysis. For example:
	"preEmCheck.log"

Default Value: preEmCheck.log

GUI Equivalent: None

preEMViolThresholdPercent

Specifies the threshold value of J/Jmax ratio in percentage reaching which the pre-EM check starts reporting EM violations.

In .cdsenv:

```
layoutEAD.em preEMViolThresholdPercent float 100.0
```

In .cdsinit or the CIW:

```
envSetVal("layoutEAD.em" "preEMViolThresholdPercent" 'float  
100.0)
```

Valid Values:

float	Any floating point value.

Default Value: 100.0

GUI Equivalent: None

minResistorCurrent

Specifies the threshold value for the minimum resistor current. If the actual resistor current is less than the specified threshold value, it is considered as 'zero' during EM analysis.

In `.cdsenv`:

```
layoutEAD.em minResistorCurrent float 1.0e-9
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.em" "minResistorCurrent" 'float 1.0e-9)
```

Valid Values:

`float`

Floating point value specifying the threshold value for the minimum resistor current.

Default Value: `1.0e-9`

GUI Equivalent: `None`

layoutEAD.pex

- autoInjectionPointFile
- autoInjectionPointDebug
- capacitanceMethod
- checkInterrupt
- couplingCThreshold
- coupleToBulkNets
- doSlotting
- dspfCapUnits
- dspfCouplingCapPrefix
- dspfFileName
- dspfGroundCapPrefix
- dspfResistorPrefix
- dspfResUnits
- dspfSubNodeDelimiter
- dspfWriteInstances
- dspfWriteSubNodes
- enableErosionEffects
- enableLoadingEffects
- enableWbeEffects
- enableWeeEffects
- excludePolyDiffusionCap
- extractDegenerateNets
- extractLocalInterconnectVias
- extractNetsWithOpens
- extractNonOrthoNets

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

- extractResOnDiffusion
- extractToPins
- extractionPrimitives
- enableCapOnlyModels
- fillPurposeNames
- fillType
- forceExtraction
- groundNetName
- hideDummyResistors
- hierarchyLevels
- hierarchyShapeType
- insertResNodeLabelPurpose
- insertResNodes
- instTermHierarchyLevels
- instTermSuperconductive
- maxResistorLength
- netHierarchyLevels
- nonTermDiffusionShapeType
- promotePorts
- recordCoupledCap
- relativeCThreshold
- resScaleFactor
- gndCapScaleFactor
- couplingCapScaleFactor
- slottingPurposeNames
- stackViaArrayHandling
- skipPowerGround

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

- supportSDRDynamicDataset
- swapLithoBiasDirection
- temperature
- threads
- type
- unusedInstTermShapeType
- viaInstanceMode
- viaShapesScanMode
- warnOnLayerMapping

autoInjectionPointFile

```
layoutEAD.pex autoInjectionPointFile string fileName
```

Description

Specifies the name of the text file that contains the details of the nodes to be inserted with respect to the defined pitch and offset on net shapes for calculation of current. The file is used when extraction is run on nets.

The default value is "" (an empty string).

- The text file must contain data in the following format:

```
grid_est=-net netName -layer metalName -pitch pitchValue -offset  
offsetValue
```

For example:

```
grid_est=-net vcch_io -layer m7 -pitch 1.632 -offset 0
```

- The value of pitch must be greater than 0 and that of offset must be greater than or equal to 0.
- There can be multiple entries for a net with distinct metal name.
- Based on the values of pitch and offset, injection points are created only in preferred layer routing directory defined in the technology file.
- It is assumed that prBoundary is defined, and the net has an associated terminal.

GUI Equivalent

None

Examples

```
envGetVal("layoutEAD.pex" "autoInjectionPointFile")  
envSetVal("layoutEAD.pex" "autoInjectionPointFile" 'string ./injectionPointFile)
```

Related Topics

[autoInjectionPointDebug](#)

[insertResNodes](#)

[Modifying EM Analysis for Wide Pins](#)

autoInjectionPointDebug

```
layoutEAD.pex autoInjectionPointDebug boolean { t | nil }
```

Description

Specifies whether to enable debugging for the auto-injection points.

The default value is nil, which indicates that debugging is not performed.

GUI Equivalent

None

Examples

```
envGetVal("layoutEAD.pex" "autoInjectionPointDebug")  
envSetVal("layoutEAD.pex" "autoInjectionPointDebug" 'boolean t)
```

Related Topics

[autoInjectionPointFile](#)

[insertResNodes](#)

[Modifying EM Analysis for Wide Pins](#)

capacitanceMethod

Specifies whether to use the built-in capacitance models (`profile`) or external tile-based parasitic extraction (`tile`) when performing capacitance extraction.

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

Note: To use the high-precision solver to extract capacitance for all nets, set this environment variable to "tile" and the [tileExtractor](#) environment variable to "sparse".

In `.cdsenv`:

```
layoutEAD.pex capacitanceMethod cyclic "profile"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "capacitanceMethod" 'cyclic  
"profile")
```

Valid Values:

profile	Use the built-in capacitance models
tile	Use external tile-based extraction. To use the high-precision solver, you must also set the tileExtractor environment variable to "sparse".

Default Value: profile

GUI Equivalent: None

checkInterrupt

Specifies whether Layout EAD needs to check for the `Ctrl + C` command that you can use to interrupt the extraction process. This check can increase the extraction run time. Therefore, if you are sure that you do not need to interrupt the extraction, disable the check by setting this variable to `nil`.

In `.cdsenv`:

```
layoutEAD.pex checkInterrupt boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "checkInterrupt" 'boolean t)
```

Valid Values:

t	Enables the check for <code>Ctrl + C</code> command to interrupt extraction
nil	Disables the check for <code>Ctrl + C</code> command to interrupt extraction

Default Value: t

GUI Equivalent: None

couplingCThreshold

Specifies a minimum coupling capacitance value in femtoFarad (fF). Capacitance values that fall below the threshold are excluded from parasitic extraction. This lets you filter out small capacitance values that are likely to be inconsequential in any analysis.

In `.cdsenv`:

```
layoutEAD.pex couplingCThreshold float 0.1
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "couplingCThreshold" 'float 0)
```

Valid Values:

<i>float</i>	Floating point value specifying the minimum coupling capacitance threshold.

Default Value: 0

GUI Equivalent: Command: *Layout EAD Options – Extraction* tab
Field: *Minimum Coupling Cap*

coupleToBulkNets

Specifies whether High Precision C and Fast C extraction need to extract coupling to any substrate or well shapes that have net assignments. Typically, these shapes are tied to the power or ground nets, and affect the capacitances of these nets the most. Therefore, the ground capacitance (and total capacitance) of the power nets gets reduced because of the presence of these shapes. Set this variable to `t` for an accurate calculation of capacitance of these nets.

In `.cdsenv`:

```
layoutEAD.pex coupleToBulkNets boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "coupleToBulkNets" 'boolean t)
```

Valid Values:

<code>t</code>	Enables extraction of coupling to any substrate or well shapes
<code>nil</code>	Disables extraction of coupling to any substrate or well shapes

Default Value: `t`

GUI Equivalent: `None`

doSlotting

Specifies whether Layout EAD needs for slotting shapes and use them to cut holes in the routing shapes to be extracted. By default, EAD does not recognize any slot shape. Set this variable to `t` to enable search for slotting shapes in specific layers that use the purpose names recognized by EAD for slotting. Related env var: [slottingPurposeNames](#)

In `.cdsenv`:

```
layoutEAD.pex doSlotting boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "doSlotting" 'boolean t)
```

Valid Values:

<code>t</code>	Enables the search for slotting shapes
<code>nil</code>	Disables the search for slotting shapes

Default Value: `nil`

GUI Equivalent: `None`

dspfCapUnits

Specifies the default unit to be used for the capacitance values in the DSPF file.

In `.cdsenv`:

```
layoutEAD.pex dspfCapUnits cyclic "ff"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "dspfCapUnits" 'cyclic "ff")
```

Valid Values:

```
"f" "ff" "pf"
```

Default Value: `"f"`

GUI Equivalent: *Parasitics & LDE Setup – DSPF tab – Format For Capacitance Values – Unit*

dspfCouplingCapPrefix

Specifies the default prefix to be used for the coupling capacitance values in the DSPF file.

In `.cdsenv`:

```
layoutEAD.pex dspfCouplingCapPrefix string "CC"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "dspfCouplingCapPrefix" 'string "CC")
```

Valid Values:

A string value specifying a prefix

Default Value: `""`

GUI Equivalent: *Parasitics & LDE Setup – DSPF tab – Format For Capacitance Values – Coupling Capacitance Prefix*

dspfFileName

Specifies the name of the DSPF file saved with parasitic details.

In `.cdsenv`:

```
layoutEAD.pex dspfFileName string "dspfProj1"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "dspfFileName" 'string "dspfProj1")
```

Valid Values:

A string value specifying a file name

Default Value: `""`

GUI Equivalent: *Parasitics & LDE Setup – DSPF tab – File Name*

dspfGroundCapPrefix

Specifies the default prefix to be used for the ground capacitance values in the DSPF file.

In `.cdsenv`:

```
layoutEAD.pex dspfGroundCapPrefix string "CC"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "dspfGroundCapPrefix" 'string "CC")
```

Valid Values:

A string value specifying a prefix

Default Value: `""`

GUI Equivalent: *Parasitics & LDE Setup – DSPF tab – Format For Capacitance Values – Ground Capacitance Prefix*

dspfResistorPrefix

Specifies the default prefix to be used for the resistance values in the DSPF file.

In `.cdsenv`:

```
layoutEAD.pex dspfResistorPrefix string "R"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "dspfResistorPrefix" 'string "R")
```

Valid Values:

A string value specifying a prefix

Default Value: `""`

GUI Equivalent: *Parasitics & LDE Setup – DSPF tab – Format For Capacitance Values – Resistance Prefix*

dspfResUnits

Specifies the default unit to be used for the resistance values in the DSPF file.

In `.cdsenv`:

```
layoutEAD.pex dspfResUnits cyclic "ohm"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "dspfResUnits" 'cyclic "ohm")
```

Valid Values:

```
"ohm" "kohm"
```

Default Value: `"ohm"`

GUI Equivalent: *Parasitics & LDE Setup – DSPF tab – Format For Resistance Values – Resistance Prefix*

dspfSubNodeDelimiter

Specifies the delimiter to be used while writing sub nodes in the DSPF file.

In `.cdsenv`:

```
layoutEAD.pex dspfSubNodeDelimiter string "#"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "dspfSubNodeDelimiter" 'string "#")
```

Valid Values:

A string value specifying a delimiter for sub nodes

Default Value: `"#"`

GUI Equivalent: *Parasitics & LDE Setup – DSPF tab – Subnode Delimiter*

dspfWriteInstances

Specifies whether to write instances in the DSPF file.

In `.cdsenv`:

```
layoutEAD.pex dspfWriteInstances boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "dspfWriteInstances" 'boolean t)
```

Valid Values:

<code>t</code>	Writes the instances in the DSPF file
<code>nil</code>	Does not write the instances in the DSPF file

Default Value: `t`

GUI Equivalent: *Parasitics & LDE Setup – DSPF tab – Write Instances*

dspfWriteSubNodes

Specifies whether to write subnodes in the DSPF file.

In `.cdsenv`:

```
layoutEAD.pex dspfWriteSubNodes boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "dspfWriteSubNodes" 'boolean t)
```

Valid Values:

<code>t</code>	Writes the subnodes in the DSPF file
<code>nil</code>	Does not write the subnodes in the DSPF file

Default Value: `t`

GUI Equivalent: `None`

enableErosionEffects

Specifies whether erosion effects are applied only to resistances, only to capacitances, to resistances and capacitances, or to neither resistances nor capacitances.

Note: For more information on erosion effects, see the *Quantas QRC Techgen Reference Manual*.

In `.cdsenv`:

```
layoutEAD.pex enableErosionEffects cyclic "none"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "enableErosionEffects" 'cyclic  
"none")
```

Valid Values:

<code>none</code>	Erosion effects are not applied
<code>c_only</code>	Erosion effects are applied only for capacitance extraction
<code>r_only</code>	Erosion effects are applied only for resistance extraction
<code>rc</code>	Erosion effects are applied to resistance and capacitance extraction

Default Value: `none`

GUI Equivalent: Command: *Layout EAD Options – Extraction* tab
Field: *Erosion Effects*

enableLoadingEffects

Controls whether dielectric layer loading effects are applied to capacitance values.

Note: For more information on dielectric layer loading effects, see the *Quantas QRC Techgen Reference Manual*.

In `.cdsenv`:

```
layoutEAD.pex enableLoadingEffects boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "enableLoadingEffects" 'boolean t)
```

Valid Values:

<code>t</code>	Loading effects are applied to capacitances
<code>nil</code>	Loading effects are not applied to capacitances

Default Value: `t`

GUI Equivalent: `None`

enableWbeEffects

Controls whether wire bottom etching effects are applied during parasitic extraction at advanced process nodes. You can specify that wire bottom etching effects are applied only to resistances, only to capacitances, to both resistances and capacitances (the default), or to neither resistances nor capacitances.

Note: For more information on wire bottom etching effects, see the *Quantas QRC Techgen Reference Manual*.

In `.cdsenv`:

```
layoutEAD.pex enableWbeEffects cyclic "rc"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "enableWbeEffects" 'cyclic "rc")
```

Valid Values:

<code>none</code>	Wire bottom etching effects are not applied
<code>c_only</code>	Wire bottom etching effects are applied to capacitances only

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

<code>r_only</code>	Wire bottom etching effects are applied to resistances only

<code>rc</code>	Wire bottom etching effects are applied to both resistances and capacitances

Default Value: `rc`

GUI Equivalent: `None`

enableWeeEffects

Controls whether wire edge enlargement (WEE) effects are applied during parasitic extraction. You can specify that WEE effects are applied only to resistances, only to capacitances, to resistances and capacitances (the default), or to neither resistances nor capacitances.

Turning on WEE effects automatically applies any lithoBias rules that are present in the EAD technology file. In the lithoBias flow, base WEE effects are specified in lithoBias rules and a higher order WEE table is specified in the ICT file. The lithoBias rules are based on the drawn wire width and spacing and the higher order WEE effects are based on silicon width and spacing parameters after the lithoBias rules are applied.

For more information on wire edge enlargement effects and lithoBias, see Chapters 3 and 11 of the *Quantas QRC Techgen Reference Manual*.

In `.cdsenv`:

```
layoutEAD.pex enableWeeEffects cyclic "rc"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "enableWeeEffects" 'cyclic "rc")
```

Valid Values:

<code>none</code>	Wire edge enlargement effects are not applied

<code>c_only</code>	Wire edge enlargement effects are applied only to capacitances

<code>r_only</code>	Wire edge enlargement effects are applied only to resistances

<code>rc</code>	Wire edge enlargement effects are applied to both resistances and capacitances

Default Value: `rc`

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

GUI Equivalent: None

excludePolyDiffusionCap

This variable is considered by both the High Precision C solver and the fast C models. For the fast C models, it controls the inclusion of both gate-to-diffusion overlap and fringing capacitance. For the High Precision C solver, it controls the inclusion of only the gate-to-diffusion overlap capacitance (also see [includePolyFringingCap](#) for High Precision C solver).

In `.cdsenv`:

```
layoutEAD.pex excludePolyDiffusionCap boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "excludePolyDiffusionCap" 'boolean t)
```

Valid Values:

t	Considers the estimated capacitance values given in the device simulation models in the PDK. For fast C models, the gate-to-diffusion overlap and fringing capacitance are ignored. For High Precision C solver, the gate-to-diffusion overlap capacitance are ignored.
nil	Includes the gate-to-diffusion overlap and fringing capacitance for fast C models. Includes the gate-to-diffusion overlap capacitance for High Precision C solver.

Default Value: t

GUI Equivalent: None

extractDegenerateNets

Specifies that degenerate nets (nets with zero or one instance terminal) are to be extracted.

In `.cdsenv`:

```
layoutEAD.pex extractDegenerateNets boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "extractDegenerateNets" 'boolean t)
```

Valid Values:

<code>t</code>	Degenerate nets are extracted.
<code>nil</code>	Degenerate nets are skipped by the extractor.

Default Value: `t`

GUI Equivalent: `None`

extractLocalInterconnectVias

Specifies that the extractor derives via connections between local interconnect layers and other layers by overlap instead of relying on layout data. The EM checker applies the vertical EM limit to these connections.

In `.cdsenv`:

```
layoutEAD.pex extractLocalInterconnectVias boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "extractLocalInterconnectVias"  
          'boolean t)
```

Valid Values:

<code>t</code>	The extractor derives via connections between LI layers.
<code>nil</code>	The extractor does not derive via connections and relies instead on the layout data.

Default Value: `t`

GUI Equivalent: `None`

extractNetsWithOpens

Controls whether the extractor extracts nets that are not fully routed.

In `.cdsenv`:

```
layoutEAD.pex extractNetsWithOpens boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "extractNetsWithOpens" 'boolean t)
```

Valid Values:

<code>t</code>	The extractor extracts nets that are not fully routed.

<code>nil</code>	The extractor skips nets that are not fully routed.

Default Value: `t`

GUI Equivalent: `None`

extractNonOrthoNets

Controls whether the extractor extracts non-orthogonal nets.

In `.cdsenv`:

```
layoutEAD.pex extractNonOrthoNets boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "extractNonOrthoNets" 'boolean t)
```

Valid Values:

<code>t</code>	The extractor extracts non-orthogonal nets.

<code>nil</code>	The extractor skips non-orthogonal nets.

Default Value: `t`

GUI Equivalent: `None`

extractResOnDiffusion

Controls whether the extractor extracts resistance on diffusion layers.

Note: Any diffusion shape that is not a terminal is treated as a background shape, or a part of substrate, and no resistance is extracted for it. To extract resistance on regular diffusion shapes, set the nonTermDiffusionShapeType environment variable to `regular`.

In `.cdsenv`:

```
layoutEAD.pex extractResOnDiffusion boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "extractResOnDiffusion" 'boolean nil)
```

Valid Values:

<code>t</code>	The extractor extracts resistance on diffusion layers.
<code>nil</code>	The extractor does not extract resistance on diffusion layers.

Default Value: `nil`

GUI Equivalent: No

extractToPins

Controls how multiple pins on a resistive network of a net are handled during RC extraction.

In `.cdsenv`:

```
layoutEAD.pex extractToPins boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "extractToPins" 'boolean nil)
```

Valid Values:

<code>t</code>	In addition to terminals and instance terminals, pins are also considered as network terminal objects and a resistance path is computed to them.
<code>nil</code>	Only terminals and instance terminals are considered as network terminal objects and a resistance path is computed to them.

Default Value: `nil`

GUI Equivalent: No

extractionPrimitives

Specifies the cells to be excluded from parasitic extraction. Parasitics are extracted up to and including the terminals of the excluded cells, but no extraction is performed inside the specified devices.

Note: Use this environment variable to exclude devices such as capacitors and resistors, where you do not want to extract resistance and capacitance twice, or for polygonal standard cells or other instances that are not XL compliant. Device terminals are extracted by default, which means that the resistance on the terminals is computed and all terminal shapes are passed to the extraction engine for capacitance extraction as well (allowing calculation of the coupling capacitance from the terminals to nets outside the Pcell).

In `.cdsenv`:

```
layoutEAD.pex extractionPrimitives string "testLib/buffer/
layout testLib/hier_buffer/layout"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "extractionPrimitives" 'string
"testLib/buffer/layout testLib/hier_buffer/layout")
```

Valid Values:

<i>string</i>	A list of cellviews (<i>lib/cell/view</i>) each separated by a space

Default Value: Empty string

GUI Equivalent: Command: *Layout EAD Options – General* tab

Field: *Primitives & Excluded Cells* table

enableCapOnlyModels

Controls whether to extract capacitor-only layers.

Some processes have specially designated capacitor-only layers that are used to create capacitor devices. These capacitor devices have some predetermined estimated capacitance values assigned by the Pcell.

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

Using this environment variable, you can extract a more accurate capacitance value for the capacitor-only layers in these devices from the actual layout for a given design.

In `.cdsenv`:

```
layoutEAD.pex enableCapOnlyModels boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "enableCapOnlyModels" 'boolean t)
```

Valid Values:

<code>t</code>	Capacitor-only layers are extracted.
<code>nil</code>	Capacitor-only layers are not extracted.

Default Value: `t`

fillPurposeNames

Registers the layer purpose names in which Layout EAD needs to search for fill shapes. By default, EAD recognizes the fill shapes on the layer purpose "fill" or "opcFill".

In `.cdsenv`:

```
layoutEAD.pex fillPurposeNames string "fill opcFill"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "fillPurposeNames" 'string "fill  
opcFill")
```

Valid Values:

A string value containing a space-separated list of layer purpose names

Default Value: "fill opcFill"

GUI Equivalent: None

fillType

Controls how the EAD extraction treats fill shapes in the layout.

In `.cdsenv`:

```
layoutEAD.pex fillType cyclic "none"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "fillType" 'cyclic "none")
```

Valid Values:

<code>none</code>	Ignore fill shapes altogether
<code>floating</code>	Treat fill shapes as floating (unassigned to any net)
<code>grounded</code>	Treat fill shapes as if they are connected to ground

Default Value: `none`

GUI Equivalent: Command: *Layout EAD Options – Extraction* tab
Field: *Metal Fill Modeling Method*

forceExtraction

Forces parasitic extraction for all nets when updating results, even if the nets in question have not been affected since the last extraction was run.

In `.cdsenv`:

```
layoutEAD.pex forceExtraction boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "forceExtraction" 'boolean nil)
```

Valid Values:

<code>t</code>	All nets are extracted.
<code>nil</code>	Only affected nets are extracted.

Default Value: `nil`

GUI Equivalent: `None`

groundNetName

Specifies the net name to use for the ground net when writing directly from the extractor to a netlist.

In `.cdsenv`:

```
layoutEAD.pex groundNetName string "0"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "groundNetName" 'string "0")
```

Valid Values:

<i>string</i>	String specifying the ground net name

Default Value: `0`

GUI Equivalent: `None`

hideDummyResistors

```
layoutEAD.pex hideDummyResistors boolean { t | nil }
```

Description

Specifies whether to hide dummy resistor having 1m ohm resistance from the detailed parasitics table.

The default value is `nil`, which indicates that these dummy resistors are displayed in the detailed parasitics table.

GUI Equivalent

None

Examples

```
envGetVal("layoutEAD.pex" "hideDummyResistors")  
envSetVal("layoutEAD.pex" "hideDummyResistors" 'boolean t)
```


hierarchyLevels

Specifies the layout hierarchy depth from which layout shapes are to be extracted during RC extraction. For example, if you specify 0, only top-level routing is extracted; if you specify 99, all hierarchy levels are extracted (nets that are wholly contained in lower levels are also extracted and reported). Decoupled capacitance is extracted to shapes for levels greater than the *Hierarchy Levels* setting.

Note: This option also works in conjunction with the [promotePorts](#) option.

Also see: [instTermHierarchyLevels](#)

In `.cdsenv`:

```
layoutEAD.pex hierarchyLevels int 99
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "hierarchyLevels" 'int 99)
```

Valid Values:

int

Positive integer value specifying the number of hierarchy levels from which layout shapes are to be extracted.

The default value, 99, specifies that all hierarchy levels are extracted. Nets that are wholly contained in lower levels are also extracted and reported.

If you set the value to 0, only top-level routing is extracted.

Default Value: 99

GUI Equivalent: Command: *Layout EAD Options – Extraction* tab

Field: *Hierarchy Levels*

hierarchyShapeType

Controls how non-terminal shapes in sub-cells are handled during parasitic extraction. This applies only to hierarchy levels greater than the [hierarchyLevels](#) setting.

In `.cdsenv`:

```
layoutEAD.pex hierarchyShapeType cyclic "regular"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "hierarchyShapeType" 'cyclic  
"regular")
```

Valid Values:

<code>regular</code>	Refers to shapes that have a net assignment. EAD extracts coupled capacitance to these shapes
<code>background</code>	Refers to shapes that do not have a net assignment. EAD extracts grounded capacitance to these shapes (exception: for “background” shapes on device or instance terminals, EAD extracts coupled capacitance)
<code>invisible</code>	Shapes are ignored by the extraction engine

Default Value: `regular`

GUI Equivalent: `None`

insertResNodeLabelPurpose

Specifies the name of the purpose to be used while creating labels to to insert nodes for R extraction in wide pins.

In `.cdsenv`:

```
layoutEAD.pex insertResNodeLabelPurpose string "forLabels"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "insertResNodeLabelPurpose" 'string  
"forLabels")
```

Valid Values:

A string value specifying the purpose name.

Default Value: `""`

GUI Equivalent: None

Related Topics

[Modifying EM Analysis for Wide Pins](#)

[autoInjectionPointFile](#)

[autoInjectionPointDebug](#)

insertResNodes

Specifies the name of the `text` file that provides the details of the nodes to be inserted on wide pins for the purpose of calculation of current.

In `.cdsenv`:

```
layoutEAD.pex insertResNodes string "labelDetails.txt"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "insertResNodes" 'string  
"labelDetails.txt")
```

Valid Values:

A string value specifying the file name.

Default Value: `""`

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

GUI Equivalent: None

Related topic: [Modifying EM Analysis for Wide Pins](#)

instTermHierarchyLevels

Specifies the layout hierarchy depth from which the instance terminals are to be extracted during RC extraction. By default, the instance terminals at the bottom of the hierarchy are extracted. You can set this variable to extract instance terminals from upto a certain level only.

Also see: [hierarchyLevels](#)

In `.cdsenv`:

```
layoutEAD.pex instTermHierarchyLevels int 99
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "instTermHierarchyLevels" 'int 99)
```

Valid Values:

int

Positive integer value specifying the hierarchy level depth from which instance terminals are to be extracted.

The default value, 99, specifies that instance terminals from all hierarchy levels are to be extracted. Instance terminals from nets that are wholly contained in lower levels are also extracted and reported.

If you set the value to 0, instance terminals are extracted only from the top-level.

Default Value: 99

GUI Equivalent: *EAD Options – Extraction – Inst Term Hierarchy Levels*

instTermSuperconductive

Specifies whether to consider a top pin or terminal shape as a superconductor that does not have any resistance. If a terminal shape is considered as a superconductor, Layout EAD uses a small resistance value of less than 0.001 ohms to connect it to a non-terminal shape.

In `.cdsenv`:

```
layoutEAD.pex instTermSuperconductive boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "instTermSuperconductive" 'boolean t)
```

Valid Values:

<code>t</code>	Layout EAD uses a shorting resistor as a superconductor resistor to connect a non-terminal shape with a terminal shape.
<code>nil</code>	The resistance of the terminal shape will be extracted same as normal shape.

Default Value: `nil`

GUI Equivalent: *None*

maxResistorLength

Specifies the maximum length of resistor (in microns) that can be extracted. Resistors that are longer than the specified maximum length are fractured into multiple resistors.

In `.cdsenv`:

```
layoutEAD.pex maxResistorLength float 100.0
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "maxResistorLength" 'float 100.0)
```

Valid Values:

<code>float</code>	Floating point value ≥ 0.5 specifying the maximum length of resistor that will be extracted. Resistors that are longer than the specified maximum are fractured into multiple resistors.

Default Value: `100.0`

GUI Equivalent: Command: *Layout EAD Options – Extraction* tab
Field: *Max Resistor Length*

Important

Quantus allows a string value of "infinite" for the CCL parameter `extraction_setup -max_fracture_length`. This value is triggered by the environment variable `layoutEAD.pex maxResistorLength` set to 0.0 or by the keyword `maxResLen=0` in the `setup.ini` file. In addition, Quantus allows a numeric value for the CCL parameter `max_fracture_length`, but a value greater than 800 is considered an error and the extraction stops. However, when running Quantus/Pegasus setup for parasitic extraction in EAD, if the value of the environment variable `maxResistorLength` is set to greater than 800, it writes `-max_fracture_length 800` to the CCL file without printing any error message in the CIW. Therefore, the extraction will run, but the fracturing behavior may not be as expected.

Related Topics

[Creating Pegasus/Quantus Setup for Parasitics Extraction](#)

[Extracting Parasitics from a Partial Layout](#)

netHierarchyLevels

Controls the layout hierarchy depth for net lookup for parasitic extraction.

In `.cdsenv`:

```
layoutEAD.pex netHierarchyLevels int 99
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "netHierarchyLevels" 'int 99)
```

Valid Values:

int

Positive integer value specifying the number of layout hierarchy levels. This value should be less than or equal to the value specified by the `hierarchyLevels` environment variable.

Default Value: 99

GUI Equivalent: None

nonTermDiffusionShapeType

Controls how non-terminal diffusion layer shapes contribute to parasitic extraction.

In `.cdsenv`:

```
layoutEAD.pex nonTermDiffusionShapeType cyclic "background"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "nonTermDiffusionShapeType" 'cyclic  
"background")
```

Valid Values:

<code>regular</code>	Both the coupling to the shape and the shape itself are extracted
<code>background</code>	Coupling capacitance is calculated to the shapes but extraction of the shape is not performed
<code>invisible</code>	Shapes are ignored by the extraction engine
<code>default</code>	The value of hierarchyShapeType is used

Default Value: `background`

GUI Equivalent: None

promotePorts

Controls whether EAD extracts parasitics for shapes that are part of device terminals; for example, metal shapes in the source/drain regions of a transistor. This applies for instances at the level specified by the [hierarchyLevels](#) environment variable.

By default, EAD extracts parasitics for such shapes. But when this option is switched on, extraction stops at the device boundary, which means that terminals on Pcells and instances

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

are treated as background shapes and are **not** extracted.

In `.cdsenv`:

```
layoutEAD.pex promotePorts boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "promotePorts" 'boolean t)
```

Valid Values:

<code>t</code>	Shapes that are part of terminals are extracted
<code>nil</code>	Shapes that are part of terminals are not extracted.

Default Value: `t`

GUI Equivalent: Command: *Layout EAD Options – Extraction* tab

Field: *Stop at Device Boundary*

recordCoupledCap

Records how much of the total capacitance was due to coupling (for decoupled capacitance extraction types).

In `.cdsenv`:

```
layoutEAD.pex recordCoupledCap boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "recordCoupledCap" 'boolean nil)
```

Valid Values:

<code>t</code>	Records how much of the total capacitance was due to coupling
<code>nil</code>	Amount of total capacitance due to coupling is not recorded.

Default Value: `nil` None

GUI Equivalent: None

relativeCThreshold

Specifies a threshold for the ratio of the coupling between the extracted net and a neighbor net divided by the total capacitance of the extracted net. If the ratio is smaller than `relativeCThreshold`, then coupling capacitances to the neighbor net are not written. This lets you filter out small capacitance values that are likely to be inconsequential in any analysis.

In `.cdsenv`:

```
layoutEAD.pex relativeCThreshold float 0.00
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "relativeCThreshold" 'float 0.00)
```

Valid Values:

<i>float</i>	Floating point value specifying the relative capacitance threshold.

Default Value: 0.00

GUI Equivalent: None

resScaleFactor

```
layoutEAD.pex resScaleFactor float scalingFactor
```

Description

Specifies the scaling factor to be applied on the extracted resistances.

The default value is 1, which indicates that no scaling is applied to the extracted value of the resistance.

GUI Equivalent

The *Resistance Scaling Factor* field on the *Extraction* tab of the EAD Options form.

Examples

```
envGetVal("layoutEAD.pex" "resScaleFactor")  
envSetVal("layoutEAD.pex" "resScaleFactor" 'float "2")
```

gndCapScaleFactor

```
layoutEAD.pex gndCapScaleFactor float scalingFactor
```

Description

Specifies the scaling factor to be applied on the extracted ground capacitance.

The default value is 1, which indicates that no scaling is applied to the extracted value of the ground capacitance.

GUI Equivalent

The *Gnd Cap Scaling Factor* field on the *Extraction* tab of the EAD Options form.

Examples

```
envGetVal("layoutEAD.pex" "gndCapScaleFactor")  
envSetVal("layoutEAD.pex" "gndCapScaleFactor" 'float "3")
```

couplingCapScaleFactor

```
layoutEAD.pex couplingCapScaleFactor float scalingFactor
```

Description

Specifies the scaling factor to be applied on the extracted coupling capacitance.

The default value is 1, which indicates that no scaling is applied to the extracted value of the coupling capacitance.

GUI Equivalent

The *Coupling Cap Scaling Factor* field on the *Extraction* tab of the EAD Options form.

Examples

```
envGetVal("layoutEAD.pex" "couplingCapScaleFactor")  
envSetVal("layoutEAD.pex" "couplingCapScaleFactor" 'float "4")
```

slottingPurposeNames

Registers the layer purpose names in which Layout EAD needs to search for slotting shapes if the doSlotting environment variable is enabled. By default, EAD searches the purpose "slot" for slotting shapes. Use this environment variable to specify a list of purpose names in which EAD can search for slotting shapes.

In `.cdsenv`:

```
layoutEAD.pex slottingPurposeNames string "slotA"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "slottingPurposeNames" 'string "slotA  
slotB")
```

Valid Values:

A string value containing a space-separated list of layer purpose names

Default Value: "slot"

GUI Equivalent: None

stackViaArrayHandling

Specifies how via arrays are to be handled while calculating EM violations.

In `.cdsenv`:

```
layoutEAD.pex stackViaArrayHandling cyclic "none"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "stackViaArrayHandling" 'cyclic  
"none")
```

Valid Values:

"mesh"	Builds a mesh network for all vias in a via array. All resistors inside via arrays are considered for EM calculations.
"cluster"	The clustering options specified on the EAD Process Settings form are used to extract resistances.
"none"	Resistors are extracted using minimum spanning tree.

Default Value: "cluster"

GUI Equivalent: None

skipPowerGround

Controls whether power and ground nets are extracted or skipped.

In `.cdsenv`:

```
layoutEAD.pex skipPowerGround boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "skipPowerGround" 'boolean nil)
```

Valid Values:

t	Power and ground nets are skipped.
nil	Power and ground nets are extracted.

Default Value: nil

GUI Equivalent: None

supportSDRDynamicDataset

```
layoutEAD.pex supportSDRDynamicDataset boolean { t | nil }
```

Description

Controls whether or not the peak and RMS current scalar values are visible in EAD browser. These peak and RMS current scalar values are used by SDR for current estimation and for the conversion of design intent information into a dataset. When set to `t`, design intent scalar peak and RMS values are visible in the EAD browser and can be used by SDR to size the wires according to the current estimation. The default value is `nil`.

GUI Equivalent

Command	<i>SDR toolbar – Options</i>
Field	<i>Consider scalar dynamic currents</i>

Examples

```
envGetVal("layoutEAD.pex" "supportSDRDynamicDataset")  
envSetVal("layoutEAD.pex" "supportSDRDynamicDataset" 'boolean t)
```

Related Topics

[SDR Options Form](#)

swapLithoBiasDirection

Swaps the edge directions of the shapes when applying lithoBias effects during parasitic extraction. Use this when the layout being extracted will be rotated 90 (or 270) degrees in the final layout, meaning that the horizontal edges of the shapes will be the vertical edges in the final implementation.

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

For more details on lithoBias, see Chapter 11, “LithoBias Flow” in the *Quantas QRC Techgen Reference Manual*.

In `.cdsenv`:

```
layoutEAD.pex swapLithoBiasDirection boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "swapLithoBiasDirection" 'boolean t)
```

Valid Values:

<code>t</code>	Swap edge directions of shapes when applying lithoBias effects during parasitic extraction
<code>nil</code>	Do not swap edge directions when applying lithoBias effects during parasitic extraction

Default Value: `nil`

GUI Equivalent: `None`

temperature

Specifies the extraction temperature in degrees Celsius.

In `.cdsenv`:

```
layoutEAD.pex temperature float 25
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "temperature" 'float 25)
```

Valid Values:

<code>float</code>	Floating point value specifying a temperature in degrees Celsius.

Default Value: `25`

GUI Equivalent: Command: *Layout EAD Options – General* tab
Field: *Temperature*

threads

Specifies the number of threads to use in multi-threaded R and C engines, both Fast and High Precision. It applies only when multiple nets are being processed or for High Precision R extraction.

In `.cdsenv`:

```
layoutEAD.pex threads int 1
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "threads" 'int 1)
```

Valid Values:

<code>int</code>	Positive integer value specifying a number of execution threads.

Default Value: 1

GUI Equivalent: Command: *Layout EAD Options – Extraction* tab

Field: *Threads*

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

type

Controls the type of parasitic networks produced by the EAD extractor.

In `.cdsenv`:

```
layoutEAD.pex type cyclic "coupled_rc"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "type" 'cyclic "coupled_rc")
```

Valid Values:

decoupled_rc	A network with distributed resistances and grounded capacitances, not including coupling capacitors to neighboring nets GUI option: <i>RC decoupled</i> Note: The coupling effects to neighboring nets are still taken into account, but that coupling capacitance is grounded.
coupled_rc	A network with distributed resistances and capacitances, including coupling capacitors to neighboring nets GUI option: <i>RC coupled</i>
lumped_c	A network with resistance values and a single total coupling capacitance value between nets GUI option: <i>C decoupled</i>
lumped_cc	A network with a single total capacitance value and one coupling capacitor to each neighboring net GUI option: <i>C lumped coupled</i>
r_only	A network with only resistance values and no capacitances. For EM checking purposes, only resistance is required. GUI option: <i>R</i>
coupled_c_only	Perform coupled capacitance extraction only GUI option: <i>C coupled</i>
none	Do not perform any extraction

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

Default Value: `coupled_rc`

GUI Equivalent: Command: *Layout EAD Options – Extraction* tab

Field: *Extraction Mode*

unusedInstTermShapeType

The connectivity extractor assigns connectivity from top to bottom through the design hierarchy. While doing this, if it finds a break in the hierarchical net assignment, there can be instance terminals without any net assignment. This variable determines how to handle the shapes of the instance terminals with no net assignment.

In `.cdsenv`:

```
layoutEAD.pex unusedInstTermShapeType cyclic "background"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "unusedInstTermShapeType" 'cyclic  
"invisible")
```

Valid Values:

"invisible"	The shapes are ignored and not passed to the extraction engine
"background"	The shapes can be stamped by another net touching them
"regular"	The shapes are treated as regular shapes and can be stamped by another net touching them

Default Value: `"background"`

GUI Equivalent: No

viaInstanceMode

Specifies the default via instance mode.

In `.cdsenv`:

```
layoutEAD.pex viaInstanceMode cyclic "cluster"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "viaInstanceMode" 'cyclic "cluster")
```

Valid Values:

<code>cluster</code>	Clustering parameters are applied inside each via instance. No clustering occurs across hierarchical boundaries.
<code>flatten</code>	All via instances (across hierarchy) are flattened and then clustering parameters are applied.
<code>one_resistor</code>	Extracts one resistor for each via instance. Clustering parameters not followed, there is no clustering across hierarchical instance or Pcell boundaries. Flat vias follow cluster parameters.

Default Value: `cluster`

GUI Equivalent: Command: *Edit Process Settings – Vias* tab

Field: *Via Instance Mode*

viaShapesScanMode

Controls the way in which via shapes are scanned while creating clusters.

In `.cdsenv`:

```
layoutEAD.pex viaShapesScanMode cyclic "edgeToEdge"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "viaShapesScanMode" 'cyclic  
"edgeToEdge")
```

Valid Values:

<code>edgeToEdge</code>	Scanning starts along the left and bottom edges (lower-left corner of the via array) and clusters are formed till the right and top edges (upper-right corner of the via array) are reached.
<code>edgesToCenter</code>	Scanning starts along the left and bottom edges (lower-left corner of via array) and the right and top edges (upper-right corner of via array) and clusters are formed until the center of via array is reached. This approach makes the clusters more symmetric around the center of the via array.

Default Value: `edgeToEdge`

GUI Equivalent: `None`

warnOnLayerMapping

Displays a warning message when Layout EAD encounters a missing layer and via mappings between OpenAccess technology files and ICT files. Set the environment variable to nil to suppress these warning messages since they are displayed by default.

In `.cdsenv`:

```
layoutEAD.pex warnOnLayerMapping boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.pex" "warnOnLayerMapping" 'boolean t)
```

Valid Values:

<code>t</code>	The messages are displayed.
<code>nil</code>	The messages are suppressed.

Default Value: `t`

GUI Equivalent: None

layoutEAD.solver

- convergenceCThreshold
- finWidth
- finSpacing
- hpcMetalFillDefFileName
- hprDisableViaClustering
- hprStampDataFileName
- includePolyFringingCap
- modelNonGateFormingPolyCap
- modelViaCapacitanceEffect
- resistanceMeshDensity
- resistanceMeshDensityFileName
- resistanceMeshFileName
- resistanceMesherThreadCount
- resistanceMesherTimeOut
- resistancePathThreshold
- resistanceFieldFileName
- showVoltageDropViolation
- tileExtractor
- tileExtractorThreadCount
- useNumberOfGPU
- qrcTechFileName
- qrcLayerSetupFileName
- qrcCapgenFileName
- hpcEngine

autoThreadsPerNetOverride

Overrides the *Threads Per Net* setting from `Auto` to an integer value. Override this setting to a non-zero integer value is important if you want to use distributed processing jobs for multi-threaded extraction.

Note: The *Threads Per Net* setting is defined in the *High Precision R Solver* section on the *Extraction* tab of the EAD Options form.

In `.cdsenv`:

```
layoutEAD.solver autoThreadsPerNetOverride int 0
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "autoThreadsPerNetOverride" 'int 0)
```

Valid Values:

`int` Any integer value greater than zero.

Default Value: 0

GUI Equivalent Command: *Layout EAD Options – Extraction* tab – *High Precision R Solver* section

Field: *Threads Per Net*

convergenceCThreshold

Specifies the convergence goal (standard deviation) of the high precision C solver.

In `.cdsenv`:

```
layoutEAD.solver convergenceCThreshold float 2.0
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "convergenceCThreshold" 'float  
2.0)
```

Valid Values:

`float` Floating point value specifying the convergence goal.

Default Value: 2.0

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

GUI Equivalent: Command: *Layout EAD Options – Extraction* tab
Field: *Default Convergence Target*

finWidth

(Advanced Nodes Only) Specifies the width of fins used to create fin geometries in high-precision capacitance models. The unit of this value is μm . To enable finFET modeling, set this variable to a value greater than 0.0.

In `.cdsenv`:

```
layoutEAD.solver finWidth float 0.012
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "finWidth" 'float 0.012)
```

Valid Values:

float Floating point value specifying the fin width value.

Default Value: 0.0

GUI Equivalent: None

finSpacing

(Advanced Nodes Only) Specifies the spacing between the fins used to create fin geometries in high-precision capacitance models. The unit of this value is μm . To enable finFET modeling, set this variable to a value greater than 0.0.

In `.cdsenv`:

```
layoutEAD.solver finSpacing float 0.040
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "finSpacing" 'float 0.040)
```

Valid Values:

float Floating point value specifying the fin spacing value.

Default Value: 0.0

GUI Equivalent: None

hpcMetalFillDefFileName

Enables the virtual metal fill modeling capability in the high precision C extraction flow. By default, the value of this variable is an empty string "", and the virtual metal fill modeling capability is disabled.

Set this variable to the name and path of the file that contains the virtual metal fill definition to be used by the flow. For details about the syntax of the virtual metal fill, refer to the *Virtual Metal Fill* section in the *Quantas QRC Techgen Reference manual*.

If an ICT file contains virtual metal fill definitions and you want to read the details from that file, specify a single line with "0" in your virtual metal definition file. This implies that no custom virtual fill definition is to be used. Instead, the virtual metal fill modeling parameters are to be read from the ICT file.

Note: If the virtual metal fill modeling parameters of a particular layer are defined in both the ICT file and the metal fill definition file, the latter has higher precedence.

In `.cdsenv`:

```
layoutEAD.solver hpcMetalFillDefFileName string ""
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "hpcMetalFillDefFileName" 'string  
      "")
```

Valid Values:

String	Path to a text file enclosed in double quotes. If the text file contains a single line, "0", the virtual metal fill information given in the ICT file is used. Otherwise, the text file should contain the details of the virtual metal fill in the standard syntax.
""	Disables the virtual metal fill modeling capacity in the high precision C extraction flow.

Default Value: ""

GUI Equivalent: None

Also see: [Customizing High precision R and C Extraction](#)

hprDisableViaClustering

Specifies whether the High Precision Resistance extraction flow needs to disable via clustering. By default, the value of this variable is `nil`, and High Precision Resistance extraction flow will observe the via clustering rules defined on the Vias tab of the EAD Process Settings form and perform corresponding via clustering.

In `.cdsenv`:

```
layoutEAD.solver hprDisableViaClustering boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "hprDisableViaClustering" 'boolean  
nil)
```

Valid Values:

<code>t</code>	High Precision Resistance extraction will not perform via clustering and will ignore the rules defined on the EAD Process Settings form.
<code>nil</code>	High Precision Resistance extraction will perform via clustering and will consider the rules defined on the <u>Vias</u> tab of the EAD Process Settings form.

Default Value: `nil`

GUI Equivalent: `None`

hprStampDataFileName

Specifies whether cell identification information should be used to construct data or mesh file names along with the resistanceFieldFileName and resistanceMeshFileName environment variables. When this variable is set to `t`, the cell information is included in the file names in the following format:

`<prefix>_<libName>_<cellName>_<viewname>_<sequenceID>` where, `<prefix>` is set by resistanceFieldFileName or resistanceMeshFileName.

For example, if the mesh file name set by variable resistanceMeshFileName is `./HPR/mesh.bin`, the mesh file names for the `layout` view of cell `INVX1` in library `INV_HIER` are named as `mesh.bin_INV_HIER_INVX1_layout_000001`, `mesh.bin_INV_HIER_INVX1_layout_000002`, and so on.

Adding cell information to the file names helps in avoiding files from being overwritten when resistanceFieldFileName and resistanceMeshFileName are not changed while you extract data from multiple cellviews.

In `.cdsenv`:

```
layoutEAD.solver hprStampDataFileName boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "hprStampDataFileName" 'boolean
nil)
```

Valid Values:

<code>t</code>	Cell information is used in the mesh or data file names
<code>nil</code>	Cell information is not used in the mesh or data file names

Default Value: `nil`

GUI Equivalent: `None`

includePolyFringingCap

Specifies that the fringing capacitance between the side walls of polysilicon and diffusion shapes is to be considered by the High Precision C solver. Also see [excludePolyDiffusionCap](#).

In `.cdsenv`:

```
layoutEAD.solver includePolyFringingCap boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "includePolyFringingCap" 'boolean  
nil)
```

Valid Values:

<code>t</code>	Fringing capacitance is included.
<code>nil</code>	Fringing capacitance is ignored.

Default Value: `nil`

GUI Equivalent: `None`

modelNonGateFormingPolyCap

Specifies if the high precision C solver needs to include the non-gate forming poly shapes in the modeling for poly to diffusion capacitance extraction.

In `.cdsenv`:

```
layoutEAD.solver modelNonGateFormingPolyCap boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "modelNonGateFormingPolyCap"  
'boolean nil)
```

Valid Values:

<code>t</code>	Non-gate forming capacitances are included in the modeling for poly to diffusion capacitance extraction.
<code>nil</code>	Non-gate forming capacitances are not included in the modeling for poly to diffusion capacitance extraction.

Default Value: `nil`

GUI Equivalent: `None`

modelViaCapacitanceEffect

Specifies if the high precision C solver needs to include vias in the modeling for capacitance extraction. When this variable is set to `t`, vias are included in capacitance extraction models and vias in the standard cells are clustered using the rules defined on the *Vias* tab of the EAD Process Settings form. When this variable is set to `nil`, vias are not included in the capacitance extraction model, therefore, the via clustering rules do not have any effect on high precision C extraction.

In `.cdsenv`:

```
layoutEAD.solver modelViaCapacitanceEffect boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "modelViaCapacitanceEffect"  
          'boolean nil)
```

Valid Values:

<code>t</code>	Vias are included in the modeling for capacitance extraction.
<code>nil</code>	Vias are not included in the modeling for capacitance extraction.

Default Value: `nil`

GUI Equivalent: `None`

resistanceMeshDensity

Specifies the default mesh density to be used by the high precision R solver.

Note: To specify separate mesh density values to be used for different layers, use the [resistanceMeshDensityFileName](#) variable.

In `.cdsenv`:

```
layoutEAD.solver resistanceMeshDensity float 1.0
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "resistanceMeshDensity" 'float  
1.0)
```

Valid Values:

<i>float</i>	Floating point value specifying the default mesh density for high precision R extraction.

Default Value: 1.0

GUI Equivalent: Command: *Layout EAD Options – Extraction*

Field: *Default Mesh Density*

resistanceMeshDensityFileName

Specifies the name of the file that provides layer-wise mesh density values to be used by the high precision R solver. By default, the mesh density specified by the [resistanceMeshDensity](#) variable is used for all the layers. However, you can use different mesh density values for

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

different layers by adding those details in a space-separated text file and providing the name of the file using this environment variable.

In `.cdsenv`:

```
layoutEAD.solver resistanceMeshDensityFileName string "./  
mesh.defs"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "resistanceMeshDensityFileName"  
'string "./mesh.defs"')
```

Valid Values:

string

Name and path of the text file that contains the layer-wise mesh density values. The format of the text file is explained below.

Each line in the text file should contain two space-separated values, `layer_name` and `mesh_density_multiplier`.

The mesh density for the specified layers is calculated as:

```
default_mesh_density * mesh_density_multiplier.
```

For example, if the default mesh density is 0.4 and the text file `mesh.defs` contains the following lines:

```
M3 1.0  
M4 0.01
```

the mesh density used for the two layers is calculated as shown below.

mesh density for M3 = $0.4 * 1.0 = 0.4$

mesh density for M4 = $0.4 * 0.01 = 0.004$

All other layers will use the default mesh density, 0.4, because no specification for those layers is given in the `mesh.defs` file.

Default Value: ""

GUI Equivalent: None

resistanceMeshFileName

Specifies the name of the file to be used by the high precision R solver to save the binary mesh data for selected nets. The data from this file is used to overlay mesh on the selected net on the layout canvas.

Note: After this variable is set by using the `.cdsinit` file or in the CIW, for extraction of each net, a new file is created with a unique file name that consists of the string specified by this environmental variable suffixed by an index number. The file will be overwritten if users re-extract the same net.

Also see:

- [hprStampDataFileName](#)
- [Viewing Mesh for Selected Nets](#)

In `.cdsenv`:

```
layoutEAD.solver resistanceMeshFileName string "mesh.bin"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "resistanceMeshFileName" 'string  
"mesh.bin")
```

Valid Values:

File name (including the path).

Default Value: `"./HPR/mesh.bin"`

GUI Equivalent: None

resistanceMesherThreadCount

Specifies the number of processing threads to be used by the high precision R solver for each net.

The default value of `resistanceMesherThreadCount` is 0, which means that you want to use as many threads as supported by the current machine. For example, on a 16-core machine, 16 threads will be used.

If you set this variable to a non-zero value, it will use the specified number of threads. The maximum number of threads that can be used will always be equal to the number of cores on the machine, regardless of the user setting.

For example, on a 16-core machine, if you set this variable to 8, only 8 threads will be used. But on the same machine, if you set this variable to 32, a maximum of 16 threads will be used regardless of the user setting.

In `.cdsenv`:

```
layoutEAD.solver resistanceMesherThreadCount int 0
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "resistanceMesherThreadCount" 'int  
0)
```

Valid Values:

`int` Any integer value greater than or equal to zero.

Default Value: 0

resistanceMesherTimeOut

Specifies the threshold time (in minutes) allocated to high precision resistance extraction for one single net.

During field solving for a net, Layout EAD keeps tracking and estimates the run time to finish the high precision R extraction for the current net. If the run time consumed or estimated time is longer than the threshold time specified by this variable, high precision R extraction is stopped. A warning message is displayed and the extraction for the remaining parts (if any) of that net are skipped.

In `.cdsenv`:

```
layoutEAD.solver resistanceMesherTimeOut int 5
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "resistanceMesherTimeOut" 'int 10)
```

Valid Values:

A positive integer value

Default Value: 5

GUI Equivalent: None

resistancePathThreshold

Specifies the maximum resistance value in `ohms` that can be reported by the high precision R solver and displayed in the EAD Browser. If this limit exceeds, the high precision R solver does not report any resistance.

Note: If the value of this variable is set to `0.0` (`zero`), the high precision R solver reports all computed resistance values and R counts without any limitation of the threshold value.

In `.cdsenv`:

```
layoutEAD.solver resistancePathThreshold float 1000.0
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "resistancePathThreshold" 'float  
1000.0)
```

Valid Values:

float

Floating point value specifying the maximum resistance value that can be reported by the high precision R solver.

Default Value: `1000.0`

GUI Equivalent: `None`

resistanceFieldFileName

Specifies the name of the file to be used by the high precision R solver to save the current density and voltage drop field data for selected nets. The data from this file is used to overlay current density and voltage drop field data on the selected net on the layout canvas.

Once this variable is set in a Virtuoso session, for every subsequent extraction, a new file is created. Unique file names are created by suffixing an incremental number.

Also see:

- [hprStampDataFileName](#)
- [Running EM Checks with High Precision R and C Extraction](#)

In `.cdsenv`:

```
layoutEAD.solver resistanceFieldFileName string "data.bin"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "resistanceFieldFileName" 'string  
"data.bin")
```

Valid Values:

File name (including the path)

Default Value: `"./HPR/data.bin"`

GUI Equivalent: None

showVoltageDropViolation

Specifies if voltage drop highlight should show the absolute voltage drop values or the violation levels calculated using the voltage drop tolerance data provided in a reference ASCII text file. The name of the reference file should be `<field-data-file-name>.ref_voltage`, where `<field-data-file-name>` is the field data file name specified by the `resistanceFieldFileName` environment variable.

In `.cdsenv`:

```
layoutEAD.solver showVoltageDropViolation boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "showVoltageDropViolation"  
          'boolean t)
```

Valid Values:

<code>t</code>	Voltage drop highlight shows the voltage drop violation levels
<code>nil</code>	Voltage drop highlight shows the real voltage drop values

Default Value: `nil`

GUI Equivalent: `None`

Also see: [Viewing Voltage Drop](#)

tileExtractor

Enables the high precision C solver for all nets.

Note: The `capacitanceMethod` environment variable must be set to "tile". Use `tileExtractorThreadCount` to specify the number of processing threads to be used.

In `.cdsenv`:

```
layoutEAD.solver tileExtractor string "sparce"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "tileExtractor" 'string "sparce")
```

Valid Values:

<code>sparce</code>	Enables the high precision solver for all nets (assuming the <code>capacitanceMethod</code> environment variable is set to "tile")

Default Value: "" Empty string

GUI Equivalent: Command: *Layout EAD Options – Extraction*
 Field: *Use Solver for All Nets*

tileExtractorThreadCount

Specifies the number of processing threads per net to be used by the high precision C solver.

The default value of this environment variable is 0, which corresponds to the `auto` setting of the field *Threads Per Net* in the *High Precision C Solver* section of the *Extraction* tab.

Total number of extraction threads to be used by the high precision C solver depends upon the values of the environment variables `layoutEAD.pex threads` and `layoutEAD.solver tileExtractorThreadCount`, as follows:

Total number of extraction threads = Value of `layoutEAD.pex threads` x Value of `layoutEAD.solver tileExtractorThreadCount`

Therefore, if the values of the environment variables `layoutEAD.pex threads` and `layoutEAD.solver tileExtractorThreadCount` are set to 2 and 4 respectively, the high precision C solver will use a total of 8 (2 x 4) processing threads.

The default number of extraction threads to be used by the high precision C solver = Default value of `layoutEAD.pex threads` x Default value of `layoutEAD.solver`

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

`tileExtractorThreadCount => 1 x auto`, which means that you want to use all the threads on the machine.

In `.cdsenv`:

```
layoutEAD.solver tileExtractorThreadCount int 0
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "tileExtractorThreadCount" 'int 0)
```

Valid Values:

<code>int</code>	Positive integer specifying the number of processing threads to be used.

Default Value:	0	Corresponds to the <code>auto</code> setting of the field <i>Threads Per Net</i> , which means that you want to use all the threads on the machine.

GUI Equivalent:	Command: <i>Layout EAD Options – Extraction</i> tab – <i>High Precision C Solver</i> section
	Field: <i>Threads Per Net</i>

useNumberOfGPU

Enables Graphical Processing Unit (GPU) computing for high precision C extraction and specifies the number of GPUs to be used. By default, this variable is set to 0, and GPU

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

computing is disabled. When this variable is set to a value greater than 0, GPU computing is enabled for the nets selected for high precision C extraction.

In `.cdsenv`:

```
layoutEAD.solver useNumberOfGPU int 0
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "useNumberOfGPU" 'int 0)
```

Valid Values:

int

Positive integer specifying the number of graphical processing units to be used.

This value should be equal to or smaller than the number of GPUs that exist in the hardware system.

Default Value: 0

GUI Equivalent: None

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

qrcTechFileName

Specifies the path to the QRC technology file to be used by the iQuantus FS engine to perform high precision C extraction. This file must be aligned with the EAD technology file.

In `.cdsenv`:

```
layoutEAD.solver qrcTechFileName string  
    "<path_to_the_QRC_tech_file>"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "qrcTechFileName" 'string  
    "<path_to_the_QRC_tech_file>")
```

Valid Values:

Path to the QRC technology file.

Default Value: " " (An empty string)

GUI Equivalent: None

qrcLayerSetupFileName

Specifies the path to the file in which the QRC layer setup data is saved. This file is used by the iQuantus FS engine to perform high precision C extraction.

In `.cdsenv`:

```
layoutEAD.solver qrcLayerSetupFileName string  
    "<path_to_the_QRC_layer_setup_file>"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "qrcLayerSetupFileName" 'string  
    "<path_to_the_QRC_layer_setup_file>")
```

Valid Values:

Path to the QRC layer setup file.

Default Value: " " (An empty string)

GUI Equivalent: None

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

qrcCapgenFileName

Specifies the path to the Quantus capgen (also called Techgen) command file. Techgen is a Quantus program that creates the QRC technology file.

In `.cdsenv`:

```
layoutEAD.solver qrcCapgenFileName string  
    "<path_to_the_QRC_Cap_Gen_File>"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "qrcCapgenFileName" 'string  
    "<path_to_the_QRC_Cap_Gen_File>>")
```

Valid Values:

Path to the Quantus capgen command file.

Default Value: " " (An empty string)

GUI Equivalent: None

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

hpcEngine

Specifies the high precision C extraction engine to be used.

In `.cdsenv`:

```
layoutEAD.solver hpcEngine cyclic "native"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.solver" "hpcEngine" 'cyclic "native")
```

Valid Values:

<code>native</code>	Native engine is used for high precision C extraction.
<code>quantusFS</code>	iQuantus Fast Solver (iQuantus FS) engine is used for high precision C extraction.

Default Value: `native`

GUI Equivalent: The *Engine* field in the *High Precision C Solver* section on the *Extraction* tab of the *EAD Options* form.

Note: The *Engine* field is not displayed by default. It is displayed only when you set this environment variable.

layoutEAD.gui

- autoHideParasiticHighlights
- applyLayerFilterOnCanvasHighlight
- capacitanceFixedSuffix
- clickClickUpdateDelay
- clickClickUseNetCache
- detailedParasiticTables
- dualPaneMode
- emResultsShowAllResistors
- enableEMCalculator
- enableFindNet
- emTableResultType
- fitToSelected
- fitToSelectedZoomScale
- hideCheckLayerMapping
- hideHPRCExtraction
- highlightSelectedNets
- layerFilterOnEMCanvas
- loadStateOnOpen
- includeSkippedNets
- maxTooltipLength
- netGroupsPath
- parasiticHighlightColor
- P2PSnapping
- resistanceDisplay
- displayP2PResistancePath

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

- resistancePathsTableTimeout
- saveStateOnClose
- saveUserSpecificSetups
- selectNetsInLayout
- selectedParasiticHighlightColor
- significantDigits
- showEMPlannerWindow
- showRecommendedTracks
- showTracksWithoutWSP
- statePath
- strikeOutDeletedDatasets
- updateP2POnDrag
- useWSPGroupNames
- violationLevels
- violationPacketStipple
- violationTransparency
- warnOnDatasetOverwrite
- warnOnTemperatureMismatch
- warnOnTemperatureOutOfRange
- warnWhenEMDataSourcesInvalid
- warnWhenNoParasiticsForDSPF

autoHideParasiticHighlights

Automatically hides any parasitic highlighting in the layout window when the mouse cursor moves away from the parasitic and EM tables in the EAD Browser. By default, parasitic highlighting in the layout canvas is persistent, i.e., it remains visible even if the mouse cursor is moved away from the browser tables.

In `.cdsenv`:

```
layoutEAD.gui autoHideParasiticHighlights boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "autoHideParasiticHighlights"  
          'boolean nil)
```

Valid Values:

<code>t</code>	Parasitic highlights are automatically hidden in the layout canvas
<code>nil</code>	Parasitic highlights are persistent

Default Value: `nil`

GUI Equivalent: `None`

applyLayerFilterOnCanvasHighlight

Enables layer-based filtering for IR Drop heat map or EM violation. When this feature is enabled, you can view heat maps for selected layers by selecting layers from the *Select which layers are displayed in the table* drop-down command list.

In `.cdsenv`:

```
layoutEAD.gui applyLayerFilterOnCanvasHighlight boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "applyLayerFilterOnCanvasHighlight"  
          'boolean nil)
```

Valid Values:

<code>t</code>	Enables filtering of layers to show the IR drop or EM violation heat map.
<code>nil</code>	Disables filtering of layers to show the IR drop or EM violation heat map.

Default Value: `nil`

GUI Equivalent: `None`

capacitanceFixedSuffix

Specifies that all capacitance values are displayed in fF. When set to `nil`, the units vary depending on the magnitude of the capacitance value.

In `.cdsenv`:

```
layoutEAD.gui capacitanceFixedSuffix boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "capacitanceFixedSuffix" 'boolean t)
```

Valid Values:

<code>t</code>	All capacitance values are shown in fF
<code>nil</code>	Units for capacitance values depend on the magnitude of the value

Default Value: `t`

GUI Equivalent: `None`

clickClickUpdateDelay

Specifies the delay in seconds after a mouse cursor pause before info balloon data is updated. Set this variable to a higher value to prevent any updates while you are moving the mouse to select the second point for info balloon.

In `.cdsenv`:

```
layoutEAD.gui clickClickUpdateDelay float 0.5
```

In `.cdsinit` or the CIW:

```
envSetVal( "layoutEAD.gui" "clickClickUpdateDelay" 'float 0.5)
```

Valid Values:

<i>float</i>	Floating point value specifying a number of seconds

Default Value: 0.5

GUI Equivalent: None

Also see: [Using Point-To-Point Info Balloons](#)

clickClickUseNetCache

Specifies whether to process all the shapes found on a net while selecting points for the creation of point-to-point info balloons. When this variable is set to `t`, all shapes present on a net are properly stamped and extracted.

Note: When there are a large number of shapes on a net, this can impact the performance of selection of points for info balloon creation. In that case, you can set the variable to `nil`.

In `.cdsenv`:

```
layoutEAD.gui clickClickUseNetCache boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "layoutEAD.gui" "clickClickUseNetCache" 'boolean t)
```

Valid Values:

<code>t</code>	Processes all the shapes found on a net while selecting points for the creation of point-to-point info balloons.
<code>nil</code>	Skips the processing of shapes while selecting points for the creation of point-to-point info balloons.

Default Value: `t`

GUI Equivalent: None

Also see: [Using Point-To-Point Info Balloons](#)

detailedParasiticTables

Displays detailed resistance and capacitance tables in the EAD Browser. When enabled, if you select a capacitance value, the browser shows an extra table of individual coupling

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

capacitance values; if you select a resistance value, the browser shows an additional table of individual resistance values along each path. The default is `nil`.

In `.cdsenv`:

```
layoutEAD.gui detailedParasiticTables boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "detailedParasiticTables" 'boolean t)
```

Valid Values:

<code>t</code>	Displays detailed parasitic tables in the EAD Browser.

<code>nil</code>	Detailed parasitic tables are not displayed.

Default Value: `nil`

GUI Equivalent: Command: *Layout EAD Options – Environment* tab
Field: *Detailed Parasitic Tables*

dualPaneMode

Toggles the dual pane view, with the summary view on top and detailed view underneath it.

In `.cdsenv`:

```
layoutEAD.gui dualPaneMode boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "dualPaneMode" 'boolean t)
```

Valid Values:

<code>t</code>	Dual pane view is enabled.

<code>nil</code>	Dual pane view is disabled.

Default Value: `t`

GUI Equivalent: *Toggle dual pane view* button on the EAD Browser toolbar



emResultsShowAllResistors

Shows **all** resistors in the EM results table (by default, the table shows only those resistors with violations; that is, a ratio of greater than 1 or the minimum violation level whichever is lower).

In `.cdsenv`:

```
layoutEAD.gui emResultsShowAllResistors boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "emResultsShowAllResistors" 'boolean  
nil)
```

Valid Values:

<code>t</code>	Shows all resistors in the EM violation table
<code>nil</code>	Shows only resistors with J/Jmax greater than 1 or the minimum violation level whichever is lower

Default Value: `nil`

GUI Equivalent: `None`

emTableResultType

Specifies the default EM result type to show in the EM violation table.

In `.cdsenv`:

```
layoutEAD.gui emTableResultType cyclic "J/Jmax"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "emTableResultType" 'cyclic "J/Jmax")
```

Valid Values:

<code>J/Jmax</code>	Shows the ratio of current to maximum current. A value greater than or equal to 1.0 is a violation.
<code>(J/Jmax) * W</code>	Shows the ratio of current to maximum current multiplied by the resistor width This gives an estimated width for this segment required to fix a violation.

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

$(J/J_{max}) * N$	Shows the ratio of current to maximum current multiplied by the existing number of via cuts. This gives the number of via cuts that could possibly fix the EM violation reported for the via resistor.
Current	Shows the current from simulation results
Max Current	Shows the maximum current that can flow through this resistor
Percent of Max Current	Shows the value as a percentage of the maximum current
Delta T	Shows the delta T values for RMS currents in the metal resistors

Default Value: `J/Jmax`

GUI Equivalent: *Result type* button on the EM violation table toolbar



enableFindNet

Specifies that results can be found for a specific net.

In `.cdsenv`:

```
layoutEAD.gui enableFindNet boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "enableFindNet" 'boolean nil)
```

Valid Values:

<code>t</code>	Find results by net is enabled.
<code>nil</code>	Find results by net is disabled.

Default Value: `nil`

GUI Equivalent: *Find results for a specific net* button on the EAD Browser toolbar



fitToSelected

Automatically pans and zooms to the correct area of the layout view when you select a net or parasitic in the EAD Browser tables.

Note: You can also specify the zoom factor using the fitToSelectedZoomScale variable.

In `.cdsenv`:

```
layoutEAD.gui fitToSelected boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "fitToSelected" 'boolean nil)
```

Valid Values:

<code>t</code>	Automatically pans and zooms to the correct area of the layout view
<code>nil</code>	Does not automatically pan and zoom

Default Value: `t`

GUI Equivalent: *Automatically zoom display to selected network or parasitic button on the EAD Browser toolbar*



hideCheckLayerMapping

```
layoutEAD.gui hideCheckLayerMapping boolean {t | nil}
```

Description

Specifies whether to show the *Check Layer Mapping* check box in the *Options* menu of the EAD Browser.

The default value is `t`, which indicates the *Check Layer Mapping* options is hidden by default.

GUI Equivalent

The *Check Layer Mapping* check box in the *Options* menu of the EAD Browser.

Examples

```
envGetVal("layoutEAD.gui" "hideCheckLayerMapping")  
envSetVal("layoutEAD.gui" "hideCheckLayerMapping" 'boolean nil)
```

hideHPRCExtraction

```
layoutEAD.gui hideHPRCExtraction boolean {t | nil}
```

Description

Specifies whether to show the *High Precision C Extraction* and *High Precision R Extraction* options in the context menu when you right-click any net in the *Net Summary Table* of the EAD Browser.

The default value is `t`, which indicates the *High Precision C Extraction* and *High Precision R Extraction* options are hidden by default.

GUI Equivalent

The *High Precision C Extraction* and *High Precision R Extraction* options in the context menu

Examples

```
envGetVal("layoutEAD.gui" "hideHPRCExtraction")
envSetVal("layoutEAD.gui" "hideHPRCExtraction" 'boolean nil)
```

fitToSelectedZoomScale

Specifies the zoom factor to be used when you have chosen to zoom the layout display to the selected network or parasitic, that is, when the *Automatically zoom display to selected network or parasitic* button on the EAD Browser toolbar is selected.

In `.cdsenv`:

```
layoutEAD.gui fitToSelectedZoomScale float 0.25
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "fitToSelectedZoomScale" 'float 0.25)
```

Valid Values:

<i>float</i>	A valid float value between the range of 0.05 to 0.9

Default Value: 0.25

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

GUI Equivalent: None

highlightSelectedNets

Automatically highlights nets in the layout when they are selected in the EAD Browser.

In `.cdsenv`:

```
layoutEAD.gui highlightSelectedNets boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "highlightSelectedNets" 'boolean nil)
```

Valid Values:

<code>t</code>	Nets are automatically highlighted
<code>nil</code>	Nets are not automatically highlighted

Default Value: `nil`

GUI Equivalent: Command: *Layout EAD Options – Environment* tab

Field: *Highlight Selected Nets*

layerFilterOnEMCanvas

Specifies whether to use layer filter to selectively display heat map only for the selected layers on the layout canvas.

In `.cdsenv`:

```
layoutEAD.gui layerFilterOnEMCanvas boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "layerFilterOnEMCanvas" 'boolean nil)
```

Valid Values:

<code>t</code>	Displays heat map only for the selected layers on the layout canvas
<code>nil</code>	Displays heat map for all the layers on the layout canvas

Default Value: `nil`

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

GUI Equivalent: None

loadTablesOnDemand

```
layoutEAD.gui loadTablesOnDemand boolean {t | nil}
```

Description

Specifies whether to enable on-demand loading of the capacitance and resistance tables when you select a net in the EAD Browser.

The default value is `nil`, which indicates that the on-demand loading of the tables in the EAD Browser is disabled by default.

You can use set this environment variable to improve performance when loading resistors and capacitance for the net selected in the EAD Browser.

GUI Equivalent

None

Examples

```
envGetVal("layoutEAD.gui" "loadTablesOnDemand")
envSetVal("layoutEAD.gui" "loadTablesOnDemand" 'boolean t)
```

includeSkippedNets

Specifies that skipped nets are displayed in the summary table.

In `.cdsenv`:

```
layoutEAD.gui includeSkippedNets boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "includeSkippedNets" 'boolean nil)
```

Valid Values:

<code>t</code>	Nets with incomplete routing are extracted
<code>nil</code>	Nets with incomplete routing are not extracted

Default Value: `nil`

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

GUI Equivalent: Command: *Layout EAD Options – General* tab
Field: *Exclude Incomplete Nets*

maxTooltipLength

Specifies the maximum number of lines to be displayed in the tooltip. This variable is especially helpful in adjusting the size of tooltips that display the constraint results (violations and warnings) for nets. For example, the tooltips for cells in the *Total R* column on the *R* tab of the EAD Browser. If the tooltip text gets truncated, you can either increase the number of lines to be displayed in the tooltip or use the *View Constraint Data* command in the context-sensitive menu of the net to view complete results.

In `.cdsenv`:

```
layoutEAD.gui maxTooltipLength int 25
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "maxTooltipLength" 'int 25)
```

Valid Values:

integer A positive integer value

Default Value: 20

GUI Equivalent: None

netGroupsPath

Specifies the path to the net group state .ini files for saving and reading the net group data. The net groups are saved in the `<library-name>/<cell-name>` directory in the specified path. By default, the net groups are saved in the `.cadence` directory.

In `.cdsenv`:

```
layoutEAD.gui netGroupsPath string "/usr/groupPath"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "netGroupsPath" 'string "/usr/  
groupPath")
```

Valid Values:

A string value defining the path to save the net group paths.

Default Value: `""`

GUI Equivalent: `None`

parasiticHighlightColor

Specifies the color used to dynamically highlight resistance paths in the layout canvas. Dynamic highlights are drawn when the pointer is moved over a resistance value in the parasitic and EM tables in the EAD Browser. By default, these are drawn in a different color than the resistance paths that are selected in the EAD Browser (see [selectedParasiticHighlightColor](#) for more information).

In `.cdsenv`:

```
layoutEAD.gui parasiticHighlightColor string "white"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "parasiticHighlightColor" 'string  
"white")
```

Valid Values:

Any color defined in the display resources file loaded in the current session. For more information, see the [Virtuoso Technology Data User Guide](#).

Default Value: `white`

GUI Equivalent: `None`

P2PSnapping

Specifies where to snap the points created for point-to-point info balloons.

In `.cdsenv`:

```
layoutEAD.gui P2PSnapping cyclic "term"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "P2PSnapping" 'cyclic "node")
```

Valid Values:

"node"	Snaps the point to the nearest node
"term"	Snaps the point to the nearest terminal
"off"	Does not snap the points. They are created at the selected point.

Default Value: "node"

GUI Equivalent: None

resistanceDisplay

Specifies how to highlight resistance paths and resistors on the layout canvas.

In `.cdsenv`:

```
layoutEAD.gui resistanceDisplay cyclic "resistor"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "resistanceDisplay" 'cyclic  
"resistor")
```

Valid Values:

"resistor"	Uses a resistor symbol to highlight resistance paths and resistors.
"halo"	Uses a halo to highlight resistance paths and resistors.

Default Value: "resistor"

GUI Equivalent: None

displayP2PResistancePath

Specifies whether to show resistance paths while displaying a point-to-point info balloon. If the resistance paths are shown, you can choose to show all the paths or only the shortest path containing the limiting resistor.

In `.cdsenv`:

```
layoutEAD.gui displayP2PResistancePath cyclic "singlePath"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "displayP2PResistancePath" 'cyclic  
"singlePath")
```

Valid Values:

"singlePath"	Only one resistance path between the start and end point is shown. This is the shortest path containing the limiting resistor.
"allPaths"	All resistance paths between the start and end point are shown.
"none"	Does not show any resistance path

Default Value: "none"

GUI Equivalent: None

Also see:

- [Displaying Resistance Paths Between the Start and End Points of an Info Balloon](#)
- [clickClickUpdateDelay](#)
- [updateP2POnDrag](#)

resistancePathsTableTimeout

Specifies the time (in second) to be allocated for searching the resistance paths for the net selected in the summary pane. If all the resistance paths are not resolved, you can use the [Search](#) feature to explicitly select and resolve the required paths.

In `.cdsenv`:

```
layoutEAD.gui resistancePathsTableTimeout float 5.0
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "resistancePathsTableTimeout" 'float  
2.0)
```

Valid Values:

A positive float value specifying the time (in seconds) to be allocated for searching the resistance paths for the selected nets.

Value 0 indicates that there is no time limit.

Default Value: 2.0

GUI Equivalent: None

saveStateOnClose

Automatically restores EAD to a state in which the design was last exited.

In `.cdsenv`:

```
layoutEAD.gui saveStateOnClose boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "saveStateOnClose" 'boolean nil)
```

Valid Values:

<code>t</code>	Automatically restores the EAD state.
<code>nil</code>	Does not restore the EAD state automatically

Default Value: `nil`

GUI Equivalent: `None`

saveUserSpecificSetups

Stores setups in user-specified locations instead of the default location in the `.cadence` directory.

In `.cdsenv`:

```
layoutEAD.gui saveUserSpecificSetups boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "saveUserSpecificSetups" 'boolean  
nil)
```

Valid Values:

<code>t</code>	Setups are saved in a user-specific <code>.cadence</code> directory
<code>nil</code>	Setups are saved in the default <code>.cadence</code> directory

Default Value: `nil`

GUI Equivalent: `None`

selectNetsInLayout

Automatically selects nets in the layout when they are selected in the EAD Browser.

In `.cdsenv`:

```
layoutEAD.gui selectNetsInLayout boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "selectNetsInLayout" 'boolean nil)
```

Valid Values:

<code>t</code>	Nets are automatically selected in the layout view
<code>nil</code>	Nets are not automatically selected in the layout view

Default Value: `nil`

GUI Equivalent: Command: *Layout EAD Options – Environment* tab

Field: *Select Nets in Layout*

selectedParasiticHighlightColor

Specifies the color used in the layout canvas to highlight resistance paths selected in the EAD Browser. By default, paths that are selected are drawn in a different color than the dynamic highlights that appear when the pointer moves over a value in the EAD Browser tables (see [parasiticHighlightColor](#) for more information).

In `.cdsenv`:

```
layoutEAD.gui selectedParasiticHighlightColor string "tan"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "selectedParasiticHighlightColor"  
  'string "tan")
```

Valid Values:

Any color defined in the display resources file loaded in the current session. For more information, see the [Virtuoso Technology Data User Guide](#).

Default Value: `tan`

GUI Equivalent: None

significantDigits

Specifies the number of significant digits to use throughout the EAD Browser tables.

In `.cdsenv`:

```
layoutEAD.gui significantDigits int 4
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "significantDigits" 'int 4)
```

Valid Values:

<code>int</code>	Positive integer value specifying the number of significant digits to use

Default Value: 4

GUI Equivalent: None

statePath

Specifies the path to the directory to be used to save or load EAD state data. See the [Working with EAD States](#) section for more information.

Note: Ensure that the directory already exists in the specified path and you have the read/write permission to the directory.

In `.cdsenv`:

```
layoutEAD.gui statePath string ""
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "statePath" 'string  
"<path_to_the_directory">')
```

Valid Values:

A string value specifying the directory path to save or load EAD state data.

Default Value: "" An empty string.

GUI Equivalent: None

strikeOutDeletedDatasets

Specifies whether the names of datasets deleted from the constraint view should appear in the ~~striketrough~~ font style in the context-menu of column headers on the EM tab. You can remove such datasets from this list by using the *Delete Electrical Data command* on the EAD toolbar.

In `.cdsenv`:

```
layoutEAD.gui strikeOutDeletedDatasets boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "strikeOutDeletedDatasets" 'boolean
t)
```

Valid Values:

t	The names of deleted datasets appear in the ~~striketrough~~ font style
nil	The names of deleted datasets do not appear in the ~~striketrough~~ font style

Default Value: t

GUI Equivalent: None

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

updateP2POnDrag

Enables or disables update of point-to-point info balloons while you move the pointer over a net after selecting the first point.

In `.cdsenv`:

```
layoutEAD.gui updateP2POnDrag boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "updateP2POnDrag" 'boolean t)
```

Valid Values:

<code>t</code>	After the first point is selected, EAD updates the data in info balloons while you move the pointer as per the delay (in seconds) specified using the clickClickUpdateDelay environment variable.
<code>nil</code>	Info balloons are updated only after the second point is selected. They are not updated while you move the pointer towards the second point.

Default Value: `t`

GUI Equivalent: None

Also see: [Using Point-To-Point Info Balloons](#)

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

violationLevels

Specifies the ranges and colors used for EM violation highlighting.

In `.cdsenv`:

```
layoutEAD.gui violationLevels string "100:red 75:orange"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "violationLevels" 'string "100:red  
75:orange")
```

Valid Values:

<i>string</i>	List of violation level values and the colors used to represent them in the EAD Browser.

Default Value:	100:red 75:orange	Violations between 76% and 100% of the limit are highlighted in red; violations up to 75% of the limit are highlighted in orange

GUI Equivalent: Command: *Layout EAD Options – Environment* tab

Field: *EM Violation Highlighting*

violationPacketStipple

Specifies the stipple pattern to use when highlighting EM violations.

In `.cdsenv`:

```
layoutEAD.gui violationPacketStipple string "solid"
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "violationPacketStipple" 'string  
"solid")
```

Valid Values:

<i>string</i>	String specifying the stipple pattern to use

Default Value:	solid

GUI Equivalent: None

violationTransparency

Specifies the default value for the *Highlight Transparency* slider in the *EM* tab toolbar. Choose a percentage value between 1 and 60.

See [Highlighting and Fixing Violations in the Layout Window](#) for more information.

In `.cdsenv`:

```
layoutEAD.gui violationTransparency int 30
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "violationTransparency" 'int 30)
```

Valid Values:

<code>int</code>	Integer value between 1 and 60 inclusive specifying the percentage transparency of the violation overlay highlights.

Default Value: 30

GUI Equivalent: Command: *Highlight Transparency* (*EM* tab toolbar)

warnOnTemperatureMismatch

Specifies that a warning popup is displayed if EM, extraction, or dataset temperatures do not match.

In `.cdsenv`:

```
layoutEAD.gui warnOnTemperatureMismatch boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "warnOnTemperatureMismatch" 'boolean  
t)
```

Valid Values:

<code>t</code>	A warning popup is displayed
<code>nil</code>	No warning popup is displayed

Default Value: `t`

GUI Equivalent: None

warnOnTemperatureOutOfRange

During EM checks, Virtuoso compares the user-specified EM temperature at which the current limit is to be calculated with the reference temperature range specified by the EM rules in `eadTechFile` or `emDataFile`. If the user-specified EM temperature is out of the range boundaries specified by the EM rules, a warning pop-up message is displayed with the names of layers for which the temperature values will be readjusted to the nearest reference value. This environment variable enables or disables the display of such warning messages.

Note: When the *Use Dataset Value* check box on the *EM* tab of the **Layout EAD Options** form is selected, the user-specified EM temperature is the temperature value in the dataset saved from ADE Assembler. If the check box is cleared, it is the temperature value specified in the *EM Temperature* field.

In `.cdsenv`:

```
layoutEAD.gui warnOnTemperatureOutOfRange boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "warnOnTemperatureOutOfRange"  
          'boolean t)
```

Valid Values:

<code>t</code>	A warning popup is displayed
<code>nil</code>	No warning popup is displayed

Default Value: `t`

GUI Equivalent: None

warnWhenEMDataSourcesInvalid

When initializing the EAD environment using the given setup information, EAD checks for the existence of the given EM data source file (EM data file, ICT file, or ICT-EM file). If the specified file is not found at the given location or is not readable, a warning message is displayed. This variable enables or disables the display of that warning message.

In `.cdsenv`:

```
layoutEAD.gui warnWhenEMDataSourceIsInvalid boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "warnWhenEMDataSourceIsInvalid"  
          'boolean t)
```

Valid Values:

<code>t</code>	A warning popup is displayed if the EM data source file is not found or is not readable
<code>nil</code>	No warning popup is displayed if the EM data source file is not found or is not readable

Default Value: `t`

GUI Equivalent: `None`

warnWhenNoParasiticsForDSPF

EAD displays the following warning message if you try to generate a DSPF file when there are no parasitic results saved in the OA database:

```
"Cannot generate a DSPF file because parasitics have not been  
extracted from the layout. Run the 'Extract Parasitics' command to  
extract the parasitics and then generate a DSPF file."
```

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

This variable enables or disables the display of this warning message.

In `.cdsenv`:

```
layoutEAD.gui warnWhenNoParasiticsForDSPF
boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "warnWhenNoParasiticsForDSPF"
" 'boolean t)
```

Valid Values:

<code>t</code>	A warning message is displayed when generating the DSPF file if there are no parasitic results saved in the OA database.
<code>nil</code>	No warning message is displayed.

Default Value: `t`

GUI Equivalent: `None`

loadStateOnOpen

Automatically loads an EAD state on startup.

In `.cdsenv`:

```
layoutEAD.gui loadStateOnOpen boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("layoutEAD.gui" "loadStateOnOpen" 'boolean nil)
```

Valid Values:

<code>t</code>	Automatically loads the EAD state on startup.
<code>nil</code>	Does not load the EAD state automatically

Default Value: `nil`

GUI Equivalent: `None`

enableEMCalculator

```
layoutEAD.gui enableEMCalculator boolean {t | nil}
```

Description

Specifies whether to show the *EM Calculator* command in the context-sensitive menu when you right-click a net in the summary pane of the EAD Browser.

The default value is `nil`, which indicates that the EM Calculator command is hidden by default.

GUI Equivalent

Command	Right-click a net in the summary pane of the EAD Browser
Field	<i>EM Calculator</i>

Examples

```
envGetVal("layoutEAD.gui" "enableEMCalculator")  
envSetVal("layoutEAD.gui" "enableEMCalculator" 'boolean t)
```

showEMPlannerWindow

```
layoutEAD.gui showEMPlannerWindow boolean {t | nil}
```

Description

Specifies whether to open the EM Calculator as a floating window or as a docked widget inside the EAD Browser.

The default value is `t`, which indicates that the EM Calculator opens as a floating window.

GUI Equivalent

None

Examples

```
envGetVal("layoutEAD.gui" "showEMPlannerWindow")  
envSetVal("layoutEAD.gui" "showEMPlannerWindow" 'boolean nil)
```

Related Topics

[enableEMCalculator](#)

showRecommendedTracks

```
layoutEAD.gui showRecommendedTracks boolean {t | nil}
```

Description

Specifies whether to display the *Tracks* column in the EM Results table.

The default value is `nil`, which indicates that the *Tracks* column is not displayed in the EM Results table.

The *Track* column in the EM Results table shows the addition number of tracks recommended to fix the reported EM violations.

The total number of recommended tracks is calculated using the following formula.

$$\text{Recommended number of tracks} = \left(\frac{\frac{J}{J_{max}} \times W}{W} \right)$$

Where:

- $(J/J_{max}) \times W$ is the ratio of actual current to maximum current multiplied by the resistor width.
- W is drawn width.

If the expression of the recommended tracks evaluates to a floating point number, the value is rounded to the next integer. For example, if the expression evaluates to `10.45`, it is rounded to the next integer, `11`. The value displayed in the *Tracks* column is `one` less than the calculated value of the recommended number of tracks.

GUI Equivalent

The *Tracks* column in the EM Results table

Examples

```
envGetVal("layoutEAD.gui" "showRecommendedTracks")
envSetVal("layoutEAD.gui" "showRecommendedTracks" 'boolean t)
```

showTracksWithoutWSP

```
layoutEAD.gui showTracksWithoutWSP boolean {t | nil}
```

Description

Specifies whether to display the *Tracks* column in the results table of the EM Calculator.

The default value is `nil`, which indicates that the *Tracks* column is not displayed in the results table of the EM Calculator.

GUI Equivalent

Command	None
Field	The <i>Tracks</i> column in the results table of the EM Calculator

Examples

```
envGetVal("layoutEAD.gui" "showTracksWithoutWSP")  
envSetVal("layoutEAD.gui" "showTracksWithoutWSP" 'boolean t)
```

Related Topics

[enableEMCalculator](#)

[Results Table](#)

useWSPGroupNames

```
layoutEAD.gui useWSPGroupNames boolean {t | nil}
```

Description

Specifies whether to use widths from a WSP group or from an RSP group when calculating EM limits for a net.

The default value is `nil`, which indicates that the widths are used from the RSP group.

GUI Equivalent

Command	None
Field	The drop-down list displayed next to the <i>RSP/WSP</i> check box in the EM Calculator

Examples

```
envGetVal("layoutEAD.gui" "useWSPGroupNames")
envSetVal("layoutEAD.gui" "useWSPGroupNames" 'boolean t)
```

Related Topics

[enableEMCalculator](#)

[User-Specified Fields](#)

p2t

- EMStackTwigs
- EMFillStackedTrunksWithVias
- EMAllowDRCViolations
- EMOptimizationMode
- EMTrunkStrandMode
- EMTrunkTapering

EMStackTwigs

Adjusts the width of a twig that has an EM violation by performing stacked stranding.

In `.cdsenv`:

```
p2t EMStackTwigs boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("p2t" "EMStackTwigs" 'boolean t)
```

Valid Values:

<code>t</code>	Stacked stranding is performed for twigs with EM violations.

<code>nil</code>	Twigs with EM violations are ignored.

Default Value: `nil`

GUI Equivalent: **Stack Twigs**

EMFillStackedTrunksWithVias

Applies to trunks that are optimized using the Stacked Stranding trunk optimization style.

In `.cdsenv`:

```
p2t EMFillStackedTrunksWithVias boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("p2t" "EMFillStackedTrunksWithVias" 'boolean t)
```

Valid Values:

<code>t</code>	Additional vias are inserted along the entire stack of trunks after stacked stranding is done.
<code>nil</code>	Via addition after stacked stranding is ignored.

Default Value: `nil`

GUI Equivalent: **Fill Stack trunks with Vias**

EMAllowDRCViolations

Allows DRC violations (shorts and spacing errors) when trunks, twigs, and vias are expanded to remove EM violations.

In `.cdsenv`:

```
p2t EMAllowDRCViolations boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("p2t" "EMAllowDRCViolations" 'boolean t)
```

Valid Values:

<code>t</code>	Allows DRC violations (shorts, spacing errors) when trunks, twigs, and vias are expanded.
<code>nil</code>	DRC violations are not allowed.

Default Value: `nil`

GUI Equivalent: **Prefer DRC Violations to EM Violations**

EMOptimizationMode

Specifies the EM trunk optimization command.

In `.cdsenv`:

```
p2t EMOptimizationMode cyclic "Expand"
```

In `.cdsinit` or the CIW:

```
envSetVal("p2t" "EMOptimizationMode" 'cyclic "Expand")
```

Valid Values:

Expand	Expands the width of trunks and increases the number of cuts in via arrays according to the hints provided by EAD.
Shrink	Reduces the width of the oversized trunks and the number of cuts of oversized via arrays.
Shrink And Expand	This option consists of two steps: ■ Shrink trunks and via arrays by running the <i>Shrink Oversized Trunks</i> command. ■ Expand trunks and vias by running the <i>Expand To Fix EM Violations</i> command.

Default Value: Expand

GUI Equivalent: *Optimize Trunk Width for Complete Network*

EMTrunkStrandMode

Selects the style for optimizing a trunk to resolve EM violations.

In `.cdsenv`:

```
p2t EMTrunkStrandMode cyclic "Parallel"
```

In `.cdsinit` or the CIW:

```
envSetVal("p2t" "EMTrunkStrandMode" 'cyclic "Parallel")
```

Valid Values:

None	Widens the trunks to fix EM violations.
Parallel	Inserts additional parallel trunks to fix EM violations.

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

`Stack` Stacks the trunks to fix EM violations.

Default Value: `None`

GUI Equivalent: *Select Trunk Optimization Style*

EMTrunkTapering

Enables trunk tapering during trunk widening. Trunk tapering works for only the widen style and not for parallel or stacked stranding.

In `.cdsenv`:

```
p2t EMTrunkTapering boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("p2t" "EMTrunkTapering" 'boolean t)
```

Valid Values:

`t` Enables trunk tapering

`nil` Disables trunk tapering

Default Value: `nil`

GUI Equivalent: *Toggle Trunk Tapering for Complete Network*

Related Topics

[Fixing Violations Using EM Driven Trunk Optimization](#)

[P2T EM Optimization](#)

[EAD Toolbar](#)

Virtuoso Electrically Aware Design Flow Guide

Environment Variables

Electrically Aware Design Functions

This section provides syntax, descriptions, and examples for the Cadence SKILL functions associated with the Virtuoso Electrically Aware Design (EAD) flow.

Only the functions documented in this chapter are supported for public use. Any other functions, and undocumented aspects of the functions described below, are private and subject to change or removal at any time.

- [elecCreateDataSetCSVTemplate](#)
- [elecDeleteDataSets](#)
- [eadDeleteScratchLib](#)
- [elecGetEnableEADMode](#)
- [elecSetEnableEADMode](#)
- [elecGetCurrentData](#)
- [elecGetDataSetNames](#)
- [elecGetDataSetParamsPropValue](#)
- [elecMakeDefaultDatasetName](#)
- [elecUpdateDataSetParamsPropValue](#)
- [elecGetVoltageData](#)
- [elecGetVsyncEnabled](#)
- [elecSetVsyncEnabled](#)
- [elecVdrCreateConstraints](#)
- [elecVdrCreateDeltaV](#)
- [eadAddNetsToNetGroup](#)
- [eadCreateDataSetFromLayCSV](#)

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

- eadCreateDataSetFromSchCSV
- eadCreateDataSetLayCSVTemplate
- eadCreateNetGroup
- eadDeleteNetGroup
- eadDeleteScratchLib
- eadExtractParasiticsForDesign
- eadExtractParasiticsForDesign
- eadGetAllEMDatasets
- eadGetNetsInNetGroup
- eadGetTriggerDesc
- eadGetLocCurrentLimit
- eadJobClose
- eadJobStatus
- eadJobSubmit
- eadLoadSetup
- eadPreEMSetUserStopCVList
- eadRegTrigger
- eadRegisterCustomEMVariable
- eadRemoveNetsFromNetGroup
- eadRemoveParasitics
- eadRemoveParasitics
- elecTransferDataSets
- eadUnregTrigger
- eadInitializeDesign
- eadExtractParasitics
- eadRemoveDesignParasitics
- eadRunEM

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

- [eadSaveSummaryTableToCSV](#)
- [eadSetCurrentDataset](#)

List of EAD-related OCEAN Commands

- [ocnxIEADAddMeasurement](#)
- [ocnxIEADCreateDataSet](#)
- [ocnxIEADEnableLiveProcessing](#)
- [ocnxIEADEnableVoltageConstraints](#)
- [ocnxIEADSelectAllSignals](#)
- [ocnxIEADSetDutMaster](#)
- [ocnxIEADSetHierarchyLevel](#)
- [ocnxIEADSetWaveFormClipping](#)

elecCreateDataSetCSVTemplate

```
elecCreateDataSetCSVTemplate(  
    t_libName  
    t_cellName  
    t_viewName  
    t_fileName  
    [ ?resultTypes l_resultTypes ]  
    [ ?maxHierLevel n_maxHierLevel ]  
    [ ?scaleFactors l_scaleFactors ]  
)  
=> t / nil
```

Description

Creates a CSV template file for a given schematic cellview in a format that can be later used to fill in the current or voltage values, and to create an EAD dataset from it.

Arguments

<i>t_libName</i>	Name of the library that contains the schematic view.
<i>t_cellName</i>	Name of the cell that contains the schematic view.
<i>t_viewName</i>	Name of the schematic view for which you need to save a template.
<i>t_fileName</i>	Name of the file in which you need to save the CSV template for the given cellview. A new file is created with the given name if it does not exist already.

[?resultTypes l_resultTypes]

A list of the result types representing the current types and voltage types to be saved. Using this argument, you can specify whether a list of instance-terminals (terminals) or nets, or both are to be saved in the template.

Default value: ("Idc" "Iavg")

[?maxHierLevel n_maxHierLevel]

An integer value specifying the hierarchy level upto which you want to save the instance or terminal names in the template.

By default, the hierarchy level saved on the EAD Setup form is used.

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

[?scaleFactors *l_scaleFactors*]

A list of values of scale factors for each result type.

The number of items specified in this list should be same as the number of items in the list of result types.

Value Returned

t When a CSV template file is successfully created.

nil The operation was unsuccessful.

Examples

The following examples show the different ways in which you can save CSV templates:

```
■ elecCreateDataSetCSVTemplate("Two_Stage_Opamp" "OpAmp" "schematic" "./
templateForDataset.csv")
==> t

elecCreateDataSetCSVTemplate("Two_Stage_Opamp" "OpAmp" "schematic" "./
templateForDataset.csv" ?resultTypes list("Idc" "Iavg" "Vmin" "Vmax"))
==> t

elecCreateDataSetCSVTemplate("Two_Stage_Opamp" "OpAmp" "schematic" "./
templateForDataset.csv" ?resultTypes list("Idc" "Iavg" "Vmin" "Vmax")
?scaleFactors list(2.0 3.0 1.0 1.0))
==> t

elecCreateDataSetCSVTemplate("Two_Stage_Opamp" "OpAmp" "schematic" "./
templateForDataset.csv" ?resultTypes list("Idc" "Iavg" "Vmin" "Vmax")
?scaleFactors list(2.0 3.0 1.0 1.0) ?maxHierLevel 0)
==> t

elecCreateDataSetCSVTemplate("Two_Stage_Opamp" "OpAmp" "schematic" "./
templateForDataset.csv" ?resultTypes list("Idc" "Iavg" "Vmin" "Vmax")
?scaleFactors list(2.0 3.0 1.0 1.0) ?maxHierLevel 99)
==> t
```

elecDeleteDataSets

```
elecDeleteDataSets(  
    t_libName  
    t_cellName  
    t_viewName  
    [ ?dataSetList l_dataSetList ]  
    [ ?dataSetType l_dataSetType ]  
)  
=> t / nil
```

Description

Deletes electrical data from the specified design. This function can be used for both layout and schematic view of a design.

Arguments

<i>t_libName</i>	Name of the library.
<i>t_cellName</i>	Name of the cell.
<i>t_viewName</i>	Name of the view. You can specify the name of a constraint or layout view.
[?dataSetList <i>l_dataSetList</i>]	List of the names of the datasets to be deleted. If you do not specify any list, all the datasets for the given cellview are deleted.
[?dataSetType <i>l_dataSetType</i>]	Type of the dataset to be deleted. Valid values: <code>currentData</code> , <code>voltageData</code>

Value Returned

<i>t</i>	The specified datasets are deleted successfully.
<i>nil</i>	The operation was unsuccessful.

Example

The following example shows how to use this function to delete datasets:

```
;The following example deletes all the datasets saved in the layout view of the  
given cellview
```

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

```
elecDeleteDataSets("Two_Stage_Opamp" "OpAmp" "layout")  
=> t
```

;The following example deletes all the datasets saved in the schematic view of the given cellview

```
elecDeleteDataSets("Two_Stage_Opamp" "OpAmp" "schematic")  
=> t
```

;The following example deletes the dataset named 'worstcase' from the layout view of the given cellview

```
elecDeleteDataSets("Two_Stage_Opamp" "OpAmp" "layout" ?dataSetList  
list("worstcase"))  
=> t
```

eadDeleteScratchLib

```
eadDeleteScratchLib(  
    )  
=> t / nil
```

Description

Deletes all data from the scratch library that contains data related to the scratch layout.

Note: By default, the scratch layout is created in the same library that contains the original layout. However, you can define a separate scratch library using the [scratchLib](#) environment variable.

See the [Running EM Analysis on Power Grids](#) section for more information

Arguments

None

Value Returned

t	All data from the scratch library is deleted successfully.
nil	The operation was unsuccessful.

Example

The following example deletes all data from the scratch library:

```
eadDeleteScratchLib()  
=> t
```

elecGetEnableEADMode

```
elecGetEnableEADMode(  
    t_sessionName  
    [ ?historyName t_historyName ]  
)  
=> t / nil
```

Description

Returns the current status of the *Enable Electrical Data Capture for EAD Flow* check box in the EAD Setup form for the specified session and history name.

Arguments

<code>t_sessionName</code>	Name of the maestro session.
<code>[?historyName t_historyName]</code>	Name of the history.

Value Returned

<code>t</code>	When the <i>Enable Electrical Data Capture for EAD Flow</i> check box is selected.
<code>nil</code>	When the <i>Enable Electrical Data Capture for EAD Flow</i> check box is not selected.

Example

The following example shows how to use this function to get the current status of the *Enable Electrical Data Capture for EAD Flow* check box in the EAD Setup form:

```
session = (axlGetWindowSession (hiGetCurrentWindow))  
=> "fnxSession0"  
elecGetEnableEADMode(session ?historyName "Interactive.0")  
=> t
```

elecSetEnableEADMode

```
elecSetEnableEADMode(  
    t_sessionName  
    g_value  
    [ ?historyName t_historyName ]  
)  
=> t / nil
```

Description

Selects or deselects the *Enable Electrical Data Capture for EAD Flow* check box in the EAD Setup form for the specified session and history name.

Arguments

<i>t_sessionName</i>	Name of the maestro session.
<i>g_value</i>	Values that you want to set: ■ <i>t</i>: Selects the <i>Enable Electrical Data Capture for EAD Flow</i> check box in the EAD Setup form. ■ <i>nil</i>: Deselects the <i>Enable Electrical Data Capture for EAD Flow</i> check box in the EAD Setup form.
[?historyName <i>t_historyName</i>]	Name of the history.

Value Returned

<i>t</i>	If value of the check box is set according to the value specified by the argument <i>g_value</i> .
<i>nil</i>	If there is an error.

Example

The following example shows how to select the *Enable Electrical Data Capture for EAD Flow* check box in the EAD Setup form:

```
session = (axlGetWindowSession (hiGetCurrentWindow))  
=> "fnxSession0"  
elecSetEnableEADMode(session t ?historyName "Interactive.0")
```

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

=> t

The following example shows how to deselect the *Enable Electrical Data Capture for EAD Flow* check box in the EAD Setup form:

```
session = (axlGetWindowSession (hiGetCurrentWindow))  
=> "fnxSession0"  
elecSetEnableEADMode(session nil ?historyName "Interactive.0")  
=> t
```

elecGetCurrentData

```
elecGetCurrentData(  
    t_libName  
    t_cellName  
    t_viewName  
    t_dataSetName  
    t_terminalName  
    [ t_measurementType ]  
)  
=> t / nil
```

Description

Gets the electrical data for the specified terminal in the given cellview. The data is retrieved from the specified dataset.

Arguments

<i>t_libName</i>	Name of the library.
<i>t_cellName</i>	Name of the cell.
<i>t_viewName</i>	Name of the schematic or layout view
<i>t_dataSetName</i>	Name of the dataset
<i>t_terminalName</i>	Name of the terminal or instance terminal
[<i>t_measurementType</i>]	The measurement type for which you need to get the current or voltage data. If not specified, the data for all the measurement types saved in the dataset is returned.

Value Returned

<i>t_electricalData</i>	List of measurement types and their corresponding values
<i>nil</i>	The operation was unsuccessful.

Example

The following example shows how to use this function to get the electrical data:

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

; The following example code shows how to get the Idc data for terminal "/I2/I0/
; PM0:B" in the schematic view of cell 'hier_buffer'. The dataset use is
; 'check_0_I':

```
elecGetCurrentData("testLib" "hier_buffer" "schematic" "check_0_I" "/I2/I0/PM0:B"  
"Idc")  
=> ("Idc" 1.710235e-14)
```

; In the following example code measurement data for all the available types is
; returned for terminal "/I2/I0/PM0:B" in the schematic view of cell 'hier_buffer'.

```
elecGetCurrentData("testLib" "hier_buffer" "schematic" "check_0_I" "/I2/I0/PM0:B")  
=> ("Iavg" 7.77565e-07 "Idc" 1.710235e-14 "Isignal" "I2.I0.PM0:4" )
```

; The following example code shows how to get the Idc data for terminal '/PM0:B'
; in the layout view of cell 'buffer':

```
elecGetCurrentData("testLib" "buffer" "layout" "check_0_I" "/PM0:B" "Idc")  
=> ("Idc" 1.710407e-14)
```

In the following example code measurement data for all the available types is returned for
terminal '/PM0:B' in the layout view of cell 'buffer':

```
elecGetCurrentData("testLib" "buffer" "layout" "check_0_I" "/PM0:B")  
=> ("Iavg" -1.213794e-06 "Idc" 1.710407e-14 "Isignal" "I3.I0.PM0:4" )
```

The following example code shows how to get the voltage data for terminal '/I3/I5/I0/I0/
IN' in the schematic view of cell 'INV_TOP':

```
elecGetCurrentData("testLib" "INV_TOP" "schematic" "pp_1_V" "/I3/I5/I0/I0/IN")  
=> ("Vmin" 0.00401131 "Vmax" 2.9869 "Vsignal" "I3.OUT1_P\\<0\\>")
```

elecGetDataSetNames

```
elecGetDataSetNames (
    t_libName
    t_cellName
    t_viewName
    [ t_dataSetType ]
)
=> l_dataSetNames / nil
```

Description

Gets the list of available result datasets for a given design. Also see [EM Dataset](#).

Arguments

<i>t_libName</i>	Name of the library.
<i>t_cellName</i>	Name of the cell.
<i>t_viewName</i>	Name of the view. You can specify the name of a schematic or layout view.
<i>t_dataSetType</i>	Dataset type. Valid values: ■ <code>currentData</code>: Gets the list of only the current datasets. ■ <code>voltageData</code>: Gets the list of only the voltage datasets.

Value Returned

<i>l_dataSetNames</i>	List of dataset names.
<i>nil</i>	The operation was unsuccessful.

Example

The following example shows how to use this function to get the list of dataset names:

```
elecGetDataSetNames("TestLib" "buffer" "schematic" "currentData")
=> ("buffer3_0_I" "buffer3_1_I" "buffer3_2_I" "buffer3_3_I" "buffer6_I" "new_I" )
```

elecGetDataSetParamsPropValue

```
elecGetDataSetParamsPropValue(  
    t_libName  
    t_cellName  
    t_viewName  
    t_dataSetName  
    t_paramName  
)  
=> t_paramValue / nil
```

Description

Gets the value for the given parameter from the property list of the given dataset saved for the given cellview.

If you had updated the paths to the results database and the psf directory by using the [elecUpdateDataSetParamsPropValue](#) function, you can use this function to verify the paths. Also see [EM Dataset](#).

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

Arguments

<i>t_libName</i>	Name of the library.
<i>t_cellName</i>	Name of the cell.
<i>t_viewName</i>	Name of the view. You can specify the name of a schematic or layout view.
<i>t_dataSetName</i>	Name of the dataset saved for the cellview.
<i>t_paramName</i>	Name of the parameter for which you need to retrieve the value. It is recommended to use this function to get the values only for <code>simResDB</code> or <code>simResDir</code> .

Value Returned

<i>t_paramValue</i>	Value corresponding to the given parameter name.
<code>nil</code>	The operation was unsuccessful.

Example

The following example code shows how to retrieve value of parameters `simResDB` and `simResDir` for schematic view of a given cellview.

```
elecGetDataSetParamsPropValue("em_test" "route_em_test" "schematic" "myDataset"
"simResDB")
=> "/home/test/libs/em_test/route_em_test/adexl/results/data/Interactive.1.rdb"

elecGetDataSetParamsPropValue("Two_Stage_Opamp" "DiffOpAmp" "schematic"
"myDataset" "simResDir")
=> "/home/test/libs/em_test/route_em_test/adexl/results/data/psf"
```

elecMakeDefaultDatasetName

```
elecMakeDefaultDatasetName(  
    t_sessionName  
    t_historyName  
    t_dutLibName  
    t_dutCellName  
    t_dutViewName  
    t_dutInstance  
    t_cornerName  
    t_temperature  
    t_designPoint  
    t_testName  
)  
=> t_defaultDatasetName
```

Description

Returns the user-defined default name of the dataset to be displayed in the *DUT Dataset* field of the *EAD* results view.

Arguments

<i>t_sessionName</i>	Name of the session.
<i>t_historyName</i>	Name of the history.
<i>t_dutLibName</i>	Name of the DUT library.
<i>t_dutCellName</i>	Name of the DUT cell.
<i>t_dutViewName</i>	Name of the DUT view.
<i>t_dutInstance</i>	Name of DUT instance.
<i>t_cornerName</i>	Name of the corner.
<i>t_temperature</i>	Value of temperature.
<i>t_designPoint</i>	Name of the design point.
<i>t_testName</i>	Name of the test.

Value Returned

t_defaultDatasetName
Default name of the dataset.

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

Examples

The following example sets the default name of *DUT Dataset* field to *dutCellName* "_" *dutInstance* "_" *historyName* "_" *cornerName* "_" *designPoint* "_" *temperature*.

```
procedure( elecMakeDefaultDatasetName( session @key historyName dutLibName
dutCellName dutViewName dutInstance cornerName temperature designPoint testName)
    let( (retVal)
        retVal = strcat( dutCellName "_" dutInstance "_" historyName "_"
cornerName "_" designPoint "_" temperature )
        retVal
    )
)
=> buffer_I2_I1_Interactive.0_Nominal_1_27
```

The following example sets the default name of *DUT Dataset* field to *testName* "_" *topCellName* "_" *dutCellName* "_" *dutInstance* "_" *historyName* "_" *cornerName* "_" *designPoint* "_" *temperature*

```
procedure( elecMakeDefaultDatasetName( session @key historyName dutLibName
dutCellName dutViewName dutInstance cornerName temperature designPoint testName)
    let( (retVal checkPoint testSdb toolArgs topCellName)
        checkPoint = axlGetHistoryCheckpoint( axlGetCurrentHistory( session ) )
        testSdb = axlGetTest( checkPoint testName )
        toolArgs = axlGetTestToolArgs( testSdb )
        topCellName = cadr(assoc( "cell" toolArgs ))
        retVal = strcat( testName "_" topCellName " " dutCellName " "
dutInstance "_" historyName "_" cornerName "_" designPoint "_" temperature )
        retVal
    )
)
=> Creating Dataset TRAN_buffer_test_buffer_I0_Interactive.5_Nominal_1_27 on DUT
Master: "sample_test", "buffer", "schematic"
```

Where *testName* is TRAN, *topCellName* is buffer_test, and *dutCellName* is buffer.

Note the following points:

- When any of the drop-down lists is set to ALL, ALL is replaced by *\$variableName* in the DUT Dataset field, which is replaced by the corresponding values at the time of dataset creation.

Default name of the dataset in the DUT Dataset field:

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

`$dutCellName_$dutInstance_Interactive.3_$cornerName_1_$temperature`

Name of the dataset created:

`OpAmp__I0_Interactive.3_C4_0_1_M40`

- Negative temperature is represented with a prefix `M`. For example, `-40` is represented as `M40`.
- `dutInstance / I2 / I1` is represented as `_I2_I1`.

elecUpdateDataSetParamsPropValue

```
elecUpdateDataSetParamsPropValue(  
    t_libName  
    t_cellName  
    t_viewName  
    t_dataSetName  
    t_paramName  
    t_paramValue  
)  
=> t / nil
```

Description

Updates the value for a parameter/property of the given dataset saved for the given cellview. You can use this function to modify the value of a dataset property for a schematic or layout view.

This function is helpful when you need to change the paths speified in the `simResDB` and `simResDir` properties from where the datasets are read. For more details, see [EM Dataset](#).

Arguments

<i>t_libName</i>	Name of the library
<i>t_cellName</i>	Name of the cell
<i>t_viewName</i>	Name of the view. You can specify the name of a schematic or layout view.
<i>t_dataSetName</i>	Name of the dataset saved for the cellview
<i>t_paramName</i>	Name of the parameter for which you need to update the value It is recommended to use this function to update the values only for <code>simResDB</code> or <code>simResDir</code> .
<i>t_paramValue</i>	New value to be set for the parameter

Value Returned

<code>t</code>	If the parameter value was successfully set
<code>nil</code>	The operation was unsuccessful

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

Example

The following example code shows how to update the `simResDir` parameter (the simulation results directory path) for schematic view of a given cellview.

```
elecUpdateDataSetParamsPropValue("em_test" "route_em_test" "schematic"  
"myDataset" "simResDir" "em_test/route_em_test/adexl7/results/data/Interactive.1/  
1/tran/psf")  
=>t
```

elecGetVoltageData

```
elecGetVoltageData(  
    t_libName  
    t_cellName  
    t_viewName  
    t_dataSetName  
    t_terminalName  
    [ t_measurementName ]  
)  
=>
```

Description

Returns the $V_{\min}$ and $V_{\max}$ values for the specified terminal in a dataset.

To calculate the $V_{\min}$ and $V_{\max}$ values for a worst-case dataset, you must set the following environment variables before running this function:

```
envSetVal("ei.dataSet" "excludeCurrentsInCI" 'boolean nil)  
envSetVal("ei.setup" "enableWorstCaseDatasetForVoltages" 'boolean t)
```

Arguments

<i>t_libName</i>	Name of the library.
<i>t_cellName</i>	Name of the cell.
<i>t_viewName</i>	Name of the schematic view.
<i>t_dataSetName</i>	Name of the dataset.
<i>t_terminalName</i>	Name of the terminal or instance terminal.
<i>t_measurementName</i>	The measurement type for which you need to get voltage data. If not specified, data for all the measurement types saved in the specified dataset is returned.

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

Value Returned

<i>l_values</i>	$V_{\min}$ and $V_{\max}$ values of the specified terminal.
<i>nil</i>	$V_{\min}$ and $V_{\max}$ values cannot be calculated because of an error.

Examples

The following examples returns the $V_{\min}$ and $V_{\max}$ values of the instance terminal `"/I0/R0/PLUS"` in the dataset `"dataset_V"`.

```
elecGetVoltageData("em_test" "route_em_test" "schematic" "dataset_V" "/I0/R0/PLUS")  
=> ("Vmin" 6.034015 "Vmax" 28.74936)
```

The following examples returns the $V_{\min}$ and $V_{\max}$ values of the instance terminal `"/I0/R0/PLUS"` in the worst-case dataset `"data_worstcase_V"`.

```
elecGetVoltageData("em_test" "route_em_test" "schematic" "data_worstcase_V" "/I0/R0/PLUS")  
=> ("Vmin" 6.034015 "Vmax" 28.74936)
```

elecGetVsyncEnabled

```
elecGetVsyncEnabled(  
    t_adeSession  
)  
=> t / nil
```

Description

Returns whether the creation of post-simulation voltage constraints is enabled or disabled.

Arguments

<i>t_adeSession</i>	The ID of the session.

Value Returned

<i>t</i>	The constraint creation is enabled.
<i>nil</i>	The constraint creation is disabled.

Example

The following example shows how to use this function to get the status of voltage constraints:

```
elecGetVsyncEnabled (session)  
=> t
```

elecSetVsyncEnabled

```
elecSetVsyncEnabled(  
    t_adeSession  
    g_value  
)  
=> t / nil
```

Description

Enables or disables the creation of post-simulation voltage constraints.

Arguments

t_adeSession

The ID of the session.

g_value

Values that you want to set:

- *t*: Enables the creation of post-simulation voltage constraints.
- *nil*: Disables the creation of post-simulation voltage constraints.

Value Returned

t

The operation is successful.

nil

When there is an error.

Example

The following example enables the creation of post-simulation voltage constraints:

```
elecSetVsyncEnabled (session t)  
=> t
```

The following example disables the creation of post-simulation voltage constraints:

```
elecSetVsyncEnabled (session nil)  
=> t
```

elecVdrCreateConstraints

```
elecVdrCreateConstraints(  
    inputToUserFunction  
    outputFromUserFunction  
)  
=> l_constraintID / nil
```

Description

Creates a post-simulation `voltageSyncedNet` constraint at the DUT level when called from a user-defined function, which has been specified using the `customVoltageConstraintFunc` environment variable.

By default, a user-defined function can create constraints automatically using the Constraints Manager SKILL functions because these functions provide control on where to create constraints in the hierarchy. The `elecVdrCreateDeltaV` function helps in providing the same functionality.

Arguments

inputToUserFunction

A list that is passed by ADE to the user-defined function as input.

The list is a DPL with the following fields:

- `session` — An ADE session ID
- `netData` — table of nets and their waveform data
- `dutLCV` — DUT lib, cell, view
- `dutInstPath` — Path to DUT inst from top level
- `clipFrom` — The start time of the clip interval specified in the EAD Setup form.
- `clipTo` — The end time of the clip interval specified in the EAD Setup form.

Example:

```
((nil session sevSession1 netData table:vsyncTable  
dutLCV ("INV_HIER_SOU" "INV_TOP" "schematic")  
dutInstPath "I3" clipFrom 2e-09 clipTo 4e-09))
```

outputFromUserFunction

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

The DPL that is expected from ADE as the output of the user-defined function.

```
'(nil constraintsLCV "test/top/schematic"
      deltaVNets      (("netE" "netF" 2.3) ("netG"
"netH" 3.4))
      syncedNets      (("netA" "netB" ) ("netC" "netD" ))
    )
```

Value Returned

l_constraintID Constraint IDs of the constraints.

Example

The following example shows how to call the `elecVdrCreateConstraints` and `elecVdrCreateDeltaV` functions from a user-defined function `userDefinedElecFunc`, which has been specified by using the environment variable `customVoltageConstraintFunc`.

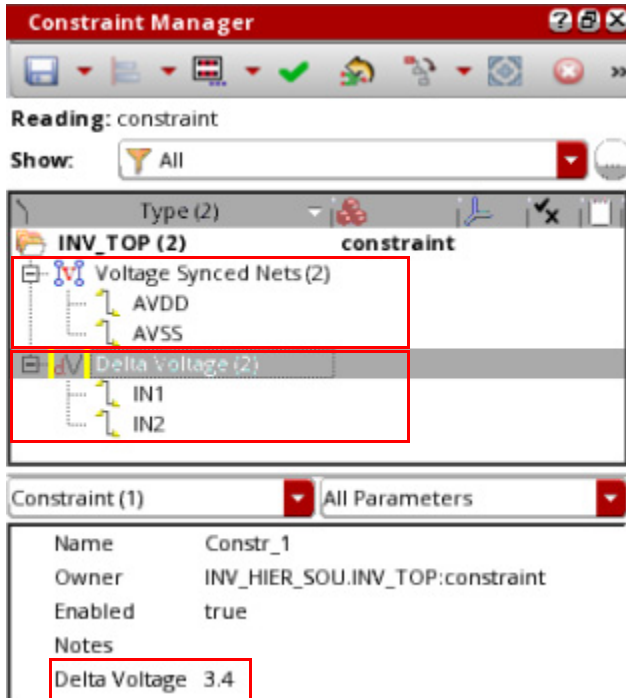
```
(envSetVal "elec.gui" "customVoltageConstraintFunc" 'string
"userDefinedElecFunc")

(define (userDefinedElecFunc @rest args)
  (printf "userDefinedElecFunc called with args :%L" args)
  (elecVdrCreateConstraints args
    (list nil 'constraintsLCV (car args)->dutLCV
      'syncdNets '(("AVDD" "AVSS"))))
  (elecVdrCreateDeltaV args
    (list nil 'constraintsLCV (car args)->dutLCV
      'deltaVNets '(("IN1" "IN2" 3.4))))
  )
```

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

When you run a simulation after adding this code in your `.cdsinit` file, the *Voltage Synced Nets* and *Delta Voltage* constraints are created at the DUT level in the Constraint Manager.



elecVdrCreateDeltaV

```
elecVdrCreateDeltaV(  
    inputToUserFunction  
    outputFromUserFunction  
)  
=> l_constraintID / nil
```

Description

Creates a post-simulation `deltaVoltage` constraint at the DUT level when called from a user-defined function, which has been specified using the `customVoltageConstraintFunc` environment variable.

By default, a user-defined function can create constraints automatically using the Constraints Manager SKILL functions because these functions provide control on where to create constraints in the hierarchy.

Note: Ensure that the *Create voltage constraints* check box is selected in the *Voltage Dependent Rules* section of the EAD Setup form before using this function.

Arguments

inputToUserFunction

A list that is passed by ADE to the user-defined function as input.

The list is a DPL with the following fields:

- `session` — An ADE session ID
- `netData` — table of nets and their waveform data
- `dutLCV` — DUT lib, cell, view
- `dutInstPath` — Path to DUT inst from top level
- `clipFrom` — The start time of the clip interval specified in the EAD Setup form.
- `clipTo` — The end time of the clip interval specified in the EAD Setup form.

Example:

```
((nil session sevSession1 netData table:vsyncTable  
dutLCV ("INV_HIER_SOU" "INV_TOP" "schematic")  
dutInstPath "I3" clipFrom 2e-09 clipTo 4e-09))
```

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

outputFromUserFunction

The DPL that is expected from ADE as the output of the user-defined function. For example,

```
'(nil constraintsLCV "test/top/schematic"
      deltaVNets      (("netE" "netF" 2.3) ("netG"
"netH" 3.4))
      syncedNets      (("netA" "netB" ) ("netC" "netD" ))
    )
```

Value Returned

l_constraintID Constraint IDs of the constraints.

Example

The following example shows how to call the `elecVdrCreateConstraints` and `elecVdrCreateDeltaV` functions from a user-defined function `userDefinedElecFunc`, which has been specified by using the environment variable `customVoltageConstraintFunc`.

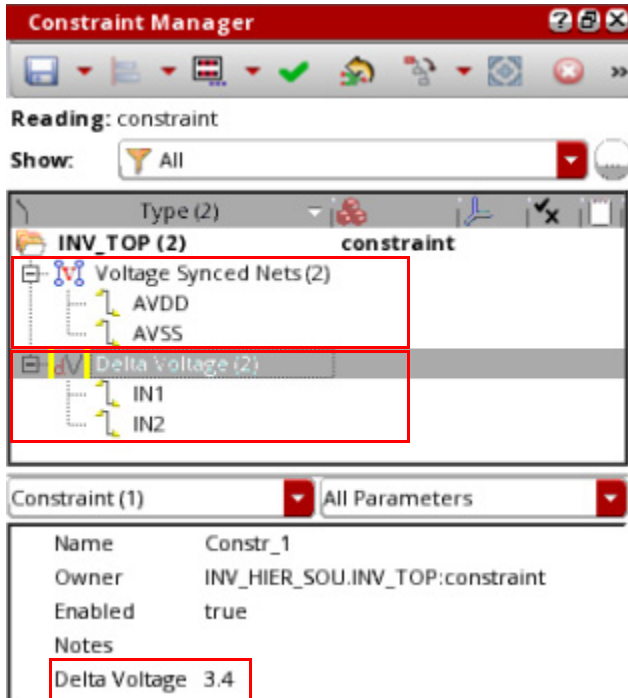
```
(envSetVal "elec.gui" "customVoltageConstraintFunc" 'string
"userDefinedElecFunc")

(define (userDefinedElecFunc @rest args)
  (printf "userDefinedElecFunc called with args :%L" args)
  (elecVdrCreateConstraints args
    (list nil 'constraintsLCV (car args)->dutLCV
      'syncedNets '(("AVDD" "AVSS"))))
  (elecVdrCreateDeltaV args
    (list nil 'constraintsLCV (car args)->dutLCV
      'deltaVNets '(("IN1" "IN2" 3.4))))
  )
```

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

When you run a simulation after adding this code in your `.cdsinit` file, the *Voltage Synced Nets* and *Delta Voltage* constraints are created in the Constraint Manager.



eadAddNetsToNetGroup

```
eadAddNetsToNetGroup(  
    x_cellViewID  
    t_groupName  
    ?netNames l_netNames  
    ?regExp t_regExp  
    ?file t_fileName  
)  
=> t / nil
```

Description

Adds the specified nets to an existing net group for the given cellview.

Arguments

<i>x_cellViewID</i>	ID of the cellview
<i>t_groupName</i>	Name of the net group to which nets are to be added
<i>?netNames l_netNames</i>	List of names of the nets to be added to the given net group
<i>?regExp t_regExp</i>	Regular or wildcard expression to identify names of nets to be added
<i>?fileName t_fileName</i>	Name of the file from where net names are to be read. All the nets whose names are given in the file are included in the net group.

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

Value Returned

<code>t</code>	When the specified nets are added to the net group
<code>nil</code>	The operation was unsuccessful

Example

The following example code shows how to add nets contained in file `test.il` to a group `groupX` in the current cellview. It also verifies the list of net names using `eadGetNetsInNetGroup`.

```
cvId = geGetEditCellView()
list netNames = nil
if(eadAddNetsToNetGroup(cvId "groupX" ?file "./test.il") then
    netNames = eadGetNetsInNetGroup(cvId "groupX")
    printf("netNames: %L\n" netNames)
else
    warn("Failed to create netGroup correctly\n")
)
```

eadCreateDataSetFromLayCSV

```
eadCreateDataSetFromLayCSV(  
    d_cvId  
    t_datasetName  
    t_fileName  
)  
=> t / nil
```

Description

Creates a user-defined dataset, with the specified name, in the given layout cellview using the electrical current values from the specified layout CSV file.

Note: This SKILL function does not refresh the EAD GUI with the newly created user-defined dataset. To view the EAD GUI refreshed with the new dataset, first save the layout cellview and then open the GUI. Ensure that EAD is initialized before opening the GUI.

Arguments

<i>d_cvId</i>	ID of the layout cellview.
<i>t_datasetName</i>	Name of the dataset you want to create.
<i>t_fileName</i>	The name of the layout CSV file containing the values of the electrical currents.

Value Returned

<i>t</i>	A user-defined dataset is successfully created.
<i>nil</i>	The operation was unsuccessful.

Examples

The following example creates a user-defined dataset, `exampleDataset`, using the electrical current values from the specified layout CSV file, `currents.csv`, for the specified layout cellview:

```
eadCreateDataSetFromLayCSV(cvId "exampleDataset" "./currents.csv")  
==> t
```

Where `cvId = geGetEditCellView(window)` and `window = hiGetCurrentWindow()`.

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

See Also: [eadCreateDataSetLayCSVTemplate](#)

eadCreateDataSetFromSchCSV

```
eadCreateDataSetFromSchCSV(  
    d_cvId  
    t_datasetName  
    t_fileName  
)  
=> t / nil
```

Description

Creates a user-defined dataset, with the specified name, in the given layout cellview using the electrical current values from the specified schematic CSV file.

Note: This SKILL function does not refresh the EAD GUI with the newly created user-defined dataset. To view the EAD GUI refreshed with the new dataset, first save the layout cellview and then open the GUI. Ensure that EAD is initialized before opening the GUI.

Arguments

<i>d_cvId</i>	ID of the layout cellview.
<i>t_datasetName</i>	Name of the dataset you want to create.
<i>t_fileName</i>	The name of the schematic CSV file containing the values of the electrical currents.

Value Returned

<i>t</i>	A user-defined dataset is successfully created.
<i>nil</i>	The operation was unsuccessful.

Examples

The following example creates a user-defined dataset, `exampleDataset`, using the electrical current values from the specified schematic CSV file, `currents.csv`, for the specified layout cellview:

```
eadCreateDataSetFromSchCSV(cvId "exampleDataset" "./currents.csv")  
==> t
```

Where `cvId = geGetEditCellView(window)` and `window = hiGetCurrentWindow()`.

eadCreateDataSetLayCSVTemplate

```
eadCreateDataSetLayCSVTemplate(  
    d_cvId  
    t_fileName  
)  
=> t / nil
```

Description

Creates a CSV template file for a given layout cellview in a format that can be used later to fill in current values and to create a user-defined EAD dataset from it.

Arguments

<i>d_cvId</i>	ID of the layout cellview.
<i>t_fileName</i>	The name of the CSV template file for the given layout cellview. A new file is created with the given name if it does not exist already.

Value Returned

<i>t</i>	When a CSV template file is successfully created.
<i>nil</i>	The operation was unsuccessful.

Examples

The following examples creates the `exampleTemplate` CSV file for the specified cellview:

```
eadCreateDataSetLayCSVTemplate(cvId "./exampleTemplate.csv")  
==> t
```

Where *cvId* = `geGetEditCellView(window)` and *window* = `hiGetCurrentWindow()`.

See Also: [eadCreateDataSetFromLayCSV](#)

eadCreateNetGroup

```
eadCreateNetGroup(  
    x_cellViewID  
    t_groupName  
    ?netNames l_netNames  
    ?regExp t_regExp  
    ?file t_fileName  
)  
=> t / nil
```

Description

Creates a new net group by the given name in the given cellview. If given, it can also contain the nets.

Arguments

<i>x_cellViewID</i>	ID of the cellview
<i>t_groupName</i>	Name of the net group to be created
<i>?netNames l_netNames</i>	List of names of nets to be added to the new net group
<i>?regExp t_regExp</i>	Regular or wildcard expression to identify names of nets to be added
<i>?fileName t_fileName</i>	Name of the file from where net names are to be read. All the nets whose names are given in the file are included in the new net group.

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

Value Returned

t_eventDescription If a net group is created successfully
nil If a net group is not created

Example

The following example code creates a new net group named `groupX` and adds two nets, `AVDD` and `AVSS`, to it:

```
cvId = geGetEditCellView()

list netNames = nil
if(eadCreateNetGroup(cvId "groupX" ?netNames list("AVDD" "AVSS")) then
    netNames = eadGetNetsInNetGroup(cvId "groupX")
    printf("netNames: %L\n" netNames)
else
    warn("Failed to create netGroup correctly\n")
)
```

eadDeleteNetGroup

```
eadDeleteNetGroup(  
    x_cellViewID  
    t_groupName  
)  
=> t / nil
```

Description

Deletes a net group from the given cellview.

Arguments

<i>x_cellViewID</i>	ID of the cellview
<i>t_groupName</i>	Name of the net group to be deleted

Value Returned

t	If the given net group is deleted successfully
nil	If the net group does not exist or deletion fails

Example

The following example code deletes the net group `groupX` from the current cellview. Then, it verifies the deletion by using the [eadGetNetsInNetGroup](#) function:

```
cvId = geGetEditCellView()  
list netNames = nil  
  
if(eadDeleteNetGroup(cvId "groupX") then  
    netNames = eadGetNetsInNetGroup(cvId "groupX")  
    ;should warn about non-existing netGroup  
else  
    warn("Failed to delete the net group.\n")  
)
```

eadExtractParasiticsForDesign

```
eadExtractParasiticsForDesign(  
    w_winId  
)  
=> t / nil
```

Description

Extracts parasitics for the specified layout window if the EAD Browser is initialized. You can configure extraction options on the EAD Options form.

Arguments

<code>w_winId</code>	Window Id of the layout.

Value Returned

<code>t</code>	Returns <code>t</code> if parasitics are extracted successfully.
<code>nil</code>	Returns <code>nil</code> if there is an error or if the EAD Browser is not initialized for the specified layout window.

Example

The following example extracts the parasitics for the layout in the current window.

```
eadExtractParasiticsForDesign(hiGetCurrentWindow())  
=> t
```

eadGetAllEMDatasets

```
eadGetAllEMDatasets(  
    )  
=> l_emDatasets / nil
```

Description

Returns a list of all EM datasets present in the current design

Arguments

None

Value Returned

<i>l_emDatasets</i>	A list of all EM datasets in the current design.
<i>nil</i>	Indicates an error.

Examples

The following examples returns a list of all EM datasets present in the current design.

```
eadGetAllEMDatasets()  
=>  
("myDataset_I" "dataSet_8_I" "dataSet_7_I" "dataSet_6_I" "dataSet_4_I"  
 "dataSet_1_I" "dataSet_0_I" "EMData"  
)
```

eadGetNetsInNetGroup

```
eadGetNetsInNetGroup(  
    t_eventName  
)  
=> t_eventDescription / nil
```

Description

Lists all the nets contained in the specified net group of the given cellview.

Arguments

<i>x_cellViewID</i>	ID of the cellview
<i>t_groupName</i>	Name of the net group for which the net names are to be displayed

Value Returned

<i>l_netNames</i>	List of names of nets contained in the given net group
<i>nil</i>	Net group is not found

Example

The following example code deletes the net group `groupX` from the current cellview. Then, it verifies the deletion by using the `eadGetNetsInNetGroup` function:

```
cvId = geGetEditCellView()  
list netNames = nil  
  
if(eadDeleteNetGroup(cvId "groupX") then  
    netNames = eadGetNetsInNetGroup(cvId "groupX")  
    ;should warn about non-existing netGroup  
else  
    warn("Failed to delete the net group.\n")  
)
```

eadGetTriggerDesc

```
eadGetTriggerDesc(  
    t_eventName  
)  
=> t_eventDescription / nil
```

Description

Provides a description about the given event and the arguments required by the SKILL triggers registered for that event.

Arguments

t_eventName Name of the event for which details are required.

Note: Currently, `PreLoadSetup` is the only valid event for which triggers can be registered. Trigger registration is not supported for any other event.

Value Returned

t_eventDescription Help string describing the event and the required arguments
nil If the description for the specified event was not found

Example

The following example code shows how the `eadGetTriggerDesc` function can be used to get the details of the `PreLoadSetup` event.

```
eadGetTriggerDesc("PreLoadSetup")  
=> "This event takes place after the design is initialized and before any EAD  
related setup is loaded. Mandatory arguments for the functions registered against  
PreLoadSetup event are WindowId and cellViewId."
```


eadGetLocCurrentLimit

```
eadGetLocCurrentLimit(  
    w_winId  
    l_point  
    t_layerName  
    [ ?direction t_direction ]  
    [ ?pulseWidth n_pulseWidth ]  
    [ ?dutyCycleRatio n_dutyCycleRatio ]  
    [ ?rmsMetalLineNum n_rmsMetalLineNum ]  
    [ ?hiEM n_hiEM ]  
)  
=> l_limit / nil
```

Description

Returns the current carrying EM limit of a resistor on the specified location. Additional arguments, such as ?pulseWidth and ?hiEM can be specified to match specific EM rules in the EAD technology file or the ICT-EM file.

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

Arguments

<code>w_winId</code>	Window Id of the layout.
<code>l_point</code>	List of points specifying the location.
<code>t_layerName</code>	Name of the layer.
<code>?direction t_direction</code>	Default direction of current.
<code>?pulseWidth n_pulseWidth</code>	Pulse width.
<code>?dutyCycleRatio n_dutyCycleRatio</code>	Ratio of the time for which the signal remains 'high' to the time period of the signal.
<code>?rmsMetalLineNum n_rmsMetalLineNum</code>	RMS metal line number for nets. This number is used to search the metal line rating tables in the EM models.
<code>?hiEM n_hiEM</code>	High EM value.

Value Returned

<code>l_limit</code>	Current extraction or EM limit.
<code>nil</code>	Returns <code>nil</code> if there is an error or if the EAD Browser is not initialized for the specified layout window.

Example

The following example returns the extraction or EM limit for the specified location in the layer `Via1` of the layout in the current window.

```
eadGetLocCurrentLimit(hiGetCurrentWindow() list(29.49 9.155) "Via1")
```

eadJobClose

```
eadJobClose(  
    x_windowID  
    t_jobConfigurationFile  
)  
=> t / nil
```

Description

Collects results from the batch jobs submitted to computer farms or dedicated servers for distributed computation. When all the distributed jobs are completed successfully, you can use this command to collect extraction results from each job. The parasitic results from individual batch run are combined into one OA database and automatically displayed in the EAD browser summary table.

Arguments

<i>x_windowID</i>	Window ID of the layout view window on which the extraction is being performed.
<i>t_jobConfigurationFile</i>	Name of the job configuration file.

Value Returned

t	When all the jobs are closed successfully
nil	In case of an error

Example

The following example shows how to run and close jobs for distributed extraction:

```
;Start extraction using the eadJobSubmit function
```

```
eadJobSubmit( hiGetCurrentWindow() "NULL" )  
=> t  
; check status using the eadJobStatus function  
eadJobStatus( hiGetCurrentWindow() "NULL" )
```

```
; After the eadJobStatus command returns t, run the following command:  
eadJobClose( hiGetCurrentWindow() "NULL" )
```

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

; The tool will collect results from all finished jobs, combine
; the results into one OA database and display results in the EAD browser.

Also see: [Distributed Computation of Extraction Process](#)

eadJobStatus

```
eadJobStatus(  
    x_windowID  
    t_jobConfigurationFile  
)  
=> t / t_jobStatus
```

Description

Returns the status of batch jobs submitted to computer farms or dedicated servers for distributed computation.

Arguments

<i>x_windowID</i>	ID of the current cellview window
<i>t_jobConfigurationFile</i>	Name of the job configuration file

Value Returned

<i>t</i>	When all the submitted jobs are complete
<i>t_jobStatus</i>	Status of all the jobs (running, waiting, or finished)

Example:

Total 5 jobs, 5 running, 0 status unknown, 0 finished

Example

The following example shows how to check the status of jobs submitted using the eadJobSubmit command:

The following example shows how to check the status of jobs submitted using the `eadJobStatus` command:

```
;Start extraction using the eadJobSubmit function  
eadJobSubmit( hiGetCurrentWindow() "NULL" )  
=> t  
; check status using the eadJobStatus function
```

```
eadJobStatus( hiGetCurrentWindow() "NULL" )  
=> t
```

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

Also see: [Distributed Computation of Extraction Process](#)

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

eadJobSubmit

```
eadJobSubmit(  
    x_windowID  
    t_jobConfigurationFile  
)  
=> t / nil
```

Description

Submits batch jobs to computer farms or dedicated servers for distributed computation.

Arguments

<i>x_windowID</i>	ID of the current cellview window
<i>t_jobConfigurationFile</i>	Name of the job configuration file

Value Returned

<i>t</i>	When the jobs are submitted successfully
<i>nil</i>	In case of an error

Example

The following example shows how to submit jobs for distributed extraction:

The following example shows how to submit jobs for distributed extraction:

```
ead_dp_net_group_number = 5  
ead_dp_batch_command = "/grid/sfi/farm/bin/bsub"  
ead_dp_batch_command_option = "-W 3:00 -q lnx64 -R \"OSREL==EE50 || OSREL==EE60  
span[hosts=1] rusage[mem=8000]\" -P IC:6.1.6:RD:QUAL"
```

; Run the following command to submit batch jobs:

eadJobSubmit(hiGetCurrentWindow() "NULL")

=> t

; The tool will automatically divide all nets into 5 groups, using command
; "/grid/sfi/farm/bin/bsub" to send each job to LSF farm,
; with command option ""-W 3:00 -q lnx64 -R \"OSREL==EE50"

Also see: [Distributed Computation of Extraction Process](#)

eadLoadSetup

```
eadLoadSetup(  
    w_winId  
    t_setupFile  
)  
=> t / nil
```

Description

Initializes the EAD Browser in the specified layout window with the given setup file.

Arguments

<i>w_winId</i>	Window Id of the layout.
<i>t_setupFile</i>	Path to the setup file to be loaded.

Value Returned

<i>t</i>	Returns <i>t</i> if setup is loaded successfully.
<i>nil</i>	Returns <i>nil</i> if there is an error or if the path to the setup file is not correct.

Example

The following example loads the EAD setup environment using the `setup_test.ini` file.

```
eadLoadSetup(hiGetCurrentWindow() ".\cadence/dfII/EAD/1/setup/setup_test.ini")  
=> t
```


eadPreEMSetUserStopCVList

```
eadPreEMSetUserStopCVList(  
    l_libraryCells  
)  
=> t / nil
```

Description

Sets all or the specified cells of user-specified libraries as stop views. The instances of the specified cells are not descended further during the pre-EM checks.

Arguments

l_libraryCells

A list of library cells that are to be set as stop views during pre-EM checks.

Value Returned

<i>t</i>	Sets the specified cellviews as stop views.
<i>nil</i>	If there is an error.

Examples

Example 1

The following example sets all the cellviews of the libraries `lib1` and `lib2` as stop views during pre-EM check:

```
eadPreEMSetUserStopCVList(list("lib1" "lib2"))
```

Example 2

The following example sets the following cellviews as stop views:

- All the cellviews of `lib 2` and `lib4` libraries
- Only `cell11` and `cell 2` cellviews of the `lib 1` and `lib3`

```
eadPreEMSetUserStopCVList(list(list("lib1" "cell11" "cell2") "lib2" list("lib3"  
"cell11" "cell2") "lib4"))
```

eadRegTrigger

```
eadRegTrigger(  
    t_eventName  
    t_callbackProcName  
    n_priority  
)  
=> t / nil
```

Description

Registers a user-defined callback SKILL function for the specified event or trigger with the given priority. You can create custom callback functions to perform necessary updates when these events are triggered.

You can register multiple triggers for an event. When the specified event occurs, the SKILL triggers registered for that event are executed in the order of their priority. Triggers registered without any priority are executed in the order of their registration.

Arguments

<i>t_eventName</i>	Name of the event. For details about the events for which you can register callbacks, refer to the Events table.
<i>t_callbackProcName</i>	Name of the callback SKILL procedure to be registered for the given event or trigger.
<i>n_priority</i>	(Optional) Priority level at which the registered trigger is to be executed when the given event occurs. You can use an integer value to specify the priority level. A lower value indicates a higher priority.

Table C-1 Events

Event	Description
PreLoadSetup	Occurs before any EAD-related setup is loaded.
PostDatasetTransfer	Occurs after the dataset transfers to the layout view are complete. The callback function for this event takes two inputs, the cellview id of the layout design and the list of transferred datasets.

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

Value Returned

t	The trigger function is successfully registered for the given event
nil	Registration of trigger was unsuccessful

Examples

Example 1:

The following example registers four triggers for the `PreLoadSetup` event. `trigger2` gets the highest priority followed by `trigger1`, `trigger3`, and `trigger4` in the given order.

```
eadRegTrigger("PreLoadSetup" "trigger1" 10)
=> t
eadRegTrigger("PreLoadSetup" "trigger2" 5)
=> t
eadRegTrigger("PreLoadSetup" "trigger3")
=> t
eadRegTrigger("PreLoadSetup" "trigger4")
=> t
;; registration of a trigger for an unsupported event shows a warning message

eadRegTrigger("LoadSetup" "trigger6")
=> *WARNING* Specified event "LoadSetup" is not supported.
```

Example 2:

The following example code registers a callback for the `PostDatasetTransfer` event:

```
eadRegTrigger("PostDatasetTransfer" "CDNS_PostDatasetTransfer_Update")

procedure( CDNS_PostDatasetTransfer_Update(cvId datasets)
  printf("Target Layout : '%s/%s/%s'.\n" cvId->libName cvId->cellName cvId->viewName)
  foreach( ds datasets
    printf("Dataset Transferred : '%s'.\n" ds)
  )
)
```

eadRegisterCustomEMVariable

```
eadRegisterCustomEMVariable(  
    t_varName  
)  
=> t / nil
```

Description

Registers a variable as a custom variable for the purpose of EM calculation. By using the [eadSetEMVariable](#) function, you can vary the value of the registered variable before running EM checks. If a variable is not registered, modifying its values does affect EM calculations.

Related Topic: [Using Custom Variables in ICT-EM Files](#)

Arguments

<i>t_varName</i>	Name of the variable to be registered.

Value Returned

<i>t</i>	The variable is successfully registered.
<i>nil</i>	Registration of variable was unsuccessful.

Example

The following example code register a custom variable named `myCustomVariable`:

```
eadRegisterCustomEMVariable("myCustomVariable")
```

eadRemoveNetsFromNetGroup

```
eadRemoveNetsFromNetGroup(  
    x_cellViewID  
    t_groupName  
    ?netNames l_netNames  
    ?regExp t_regExp  
    ?file t_fileName  
)  
=> t / nil
```

Description

Removes the given nets from the specified net group of the given cellview.

Arguments

<i>x_cellViewID</i>	ID of the cellview
<i>t_groupName</i>	Name of the net group from which nets are to be deleted
<i>?netNames l_netNames</i>	List of names of nets to be removed from the given net group
<i>?regExp t_regExp</i>	Regular or wildcard expression to identify names of nets to be removed
<i>?fileName t_fileName</i>	Name of the file from where net names are to be read. All the nets whose names are given in the file are removed from the net group.

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

Value Returned

<code>t</code>	If the nets were removed from net group successfully.
<code>nil</code>	If the net group does not exist or deletion of nets from the net group fails.

Example

The following example code uses a regular expression to identify and remove nets from net group `groupX` in the current cellview. After removal, it verifies the list of net names using the `eadGetNetsInNetGroup` function:

```
cvId = geGetEditCellView()
list netNames = nil
if(eadRemoveNetsFromNetGroup(cvId "groupX" ?regExp "AV*") then
    netNames = eadGetNetsInNetGroup(cvId "groupX")
    printf("netNames: %L\n" netNames)
else
    warn("Failed to create netGroup correctly\n")
)
```

eadRemoveParasitics

```
eadRemoveParasitics(  
    t_libName  
    t_cellName  
    t_viewName  
)  
=> t / nil
```

Description

Deletes the parasitic information associated with the specified layout design and saves the design (provided there were no other unsaved changes prior to calling the function).

Note: If you run this function interactively from the CIW, the EAD Browser tables are not refreshed.

Arguments

<i>t_libName</i>	Name of the library containing the layout design.
<i>t_cellName</i>	Name of the cell for which parasitics are to be deleted.
<i>t_viewName</i>	Name of the view for which parasitics are to be deleted.

Value Returned

<i>t</i>	Parasitic information was removed from the specified cellview.
<i>nil</i>	The operation was unsuccessful.

Example

The following example removes parasitic information from a layout design called `via_em_test` in library `em_test`.

```
eadRemoveParasitics("em_test" "via_em_test" "layout")  
=> t
```

eadSetEMVariable

```
eadSetEMVariable(  
    g_varName  
    [ g_varValue ]  
    [ g_layerName ]  
    [ g_analysisPointName ]  
)  
=> t / nil
```

Description

Sets the value of a custom variable registered by [eadRegisterCustomEMVariable](#). The value is used by the EM limit calculation expressions in the ICT-EM file.

Arguments

<i>g_varName</i>	Name of the custom variable
<i>g_varValue</i>	Value to be set for the variable
<i>g_layerName</i>	Name of the layer for which this variable is to be used.
<i>g_analysisPointName</i>	Name of the analysis point or corner
<i>e</i>	

Value Returned

<i>t</i>	The value is successfully set for the variable.
<i>nil</i>	The value is not set successfully.

Example

The following example sets the value for a custom variable `myCustomVar` for layer `mt3`:

```
eadRegisterCustomEMVariable("myCustomVar" "10.3" "mt3" "nom")  
=> t
```


elecTransferDataSets

```
elecTransferDataSets(  
    t_libName  
    t_cellName  
    t_viewName  
    [ ?dataSetList l_dataSetList ]  
    [ ?dataSetType t_dataSetType ]  
    [ ?schLCV g_schLCV ]  
    [ ?needToSave g_needToSave ]  
)  
=> t / nil
```

Description

Transfers datasets from schematic to layout. By default, the function transfers all the datasets from schematic to layout. You can use the key arguments to transfer specific datasets.

Note: If you run this function interactively from the CIW, the EAD Browser tables are not refreshed.

Arguments

<i>t_libName</i>	Name of the library of the source view.
<i>t_cellName</i>	Name of the cell of the source view.
<i>t_viewName</i>	Name of the layout view to which the datasets are to be transferred from the schematic view of the same cell.
[?dataSetList <i>l_dataSetList</i>]	List of the names of the datasets to be transferred. If you do not specify any list, all the datasets for the given cellview are deleted.
[?dataSetType <i>t_dataSetType</i>]	Type of the datasets to be transferred. Valid values: ■ "currentData": Only the datasets that contain the current data are transferred. ■ "voltageData": Only the datasets that contain the voltage data are transferred.

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

[?schLCV *g_schLCV*]

List containing the library, cell, and view name to be considered as the schematic source or the dbObject ID of a schematic view that is already opened in Virtuoso.

Datasets are transferred from this schematic to the layout view of the library and cell specified by the *t_libName* and *t_cellName* arguments.

[?needToSave *g_needToSave*]

Boolean value to specify if the layout view is to be saved after transferring the datasets from schematic to layout.

Set this value to `t` if you need to save the datasets in the layout view and use them in future sessions. Set this value to `nil` if you need to use the transferred datasets only in the current session, without saving them in the layout view.

Value Returned

<code>t</code>	The dataset is successfully transferred.
<code>nil</code>	The operation was unsuccessful.

Example

The following examples show different ways in which datasets can be transferred.

```
elecTransferDataSets("test1" "buffer" "layout")
```

```
=> t
```

```
elecTransferDataSets("test1" "buffer" "layout" ?dataSetList list("buffer_0_I" "buffer_1_V"))
```

```
=> t
```

```
elecTransferDataSets("test1" "buffer" "layout" ?dataSetType "currentData")
```

```
=> t
```

```
elecTransferDataSets("test1" "buffer" "layout" ?schLCV list("test1" "buffer" "schematic"))
```

```
=> t
```

```
elecTransferDataSets("test1" "buffer" "layout" ?dataSetList list("buffer_0_I" "buffer_1_V") ?schLCV list("test1" "buffer" "schematic"))
```

```
=> t
```

; The following example transfers the datasets that store current data from the schematic view of test2:buffer to the layout view of test1:buffer:

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

```
elecTransferDataSets("test1" "buffer" "layout" ?dataSetType "currentData" ?schLCV  
list("test1" "buffer" "schematic"))  
=> t
```

```
elecTransferDataSets("test1" "buffer" "layout" ?dataSetList list("buffer_0_I"  
"buffer_1_V") ?dataSetType "currentData" ?schLCV list("test1" "buffer"  
"schematic"))  
=> t
```

; The following example transfers the datasets that store current data using the object ID of a schematic view:

```
elecTransferDataSets("test1" "buffer" "layout" ?dataSetList list("buffer_0_I"  
"buffer_1_V") ?dataSetType "currentData" ?schLCV dbOpenCellViewByType("test2"  
"buffer" "schematic"))  
=> t
```

eadUnregTrigger

```
eadUnregTrigger(  
    t_eventName  
    t_triggerName  
)  
=> t / nil
```

Description

Removes the entry for the specified SKILL trigger from the registered list of triggers for the given event. When the specified event occurs, the remaining registered triggers, if any, for that event get executed in their order of priority.

Arguments

<i>t_eventName</i>	Name of the event
<i>t_triggerName</i>	Name of the SKILL trigger to be unregistered

Value Returned

<i>t</i>	The trigger is successfully unregistered for the given event.
<i>nil</i>	The operation was unsuccessful.

Example

The following example removes `trigger1` from the list of registered triggers for the `PreLoadSetup` event.

```
eadUnregTrigger("PreLoadSetup" "trigger1")  
=> t
```

eadInitializeDesign

```
eadInitializeDesign(  
    t_layoutDesignCellViewId  
    t_eadSetupFile  
    t_cornerName  
)  
=> t / nil
```

Description

Initializes the specified layout design with the specified EAD setup and corner name.

Arguments

<i>t_layoutDesignCellViewId</i>	Cellview ID of the layout design.
<i>t_eadSetupFile</i>	Path to the EAD setup file.
<i>t_cornerName</i>	Name of the corner.

Value Returned

<i>t</i>	EAD setup is initialized for the specified layout design.
<i>nil</i>	EAD setup cannot be initialized because of an error.

Example

Opens the cellview in append mode.

```
cv = (dbOpenCellViewByType "Two_Stage_Opamp" "OpAmp" "layout_user" "maskLayout"  
"a")  
=> db:0xb697de9a
```

Initializes the design specified in the cellview with the specified EAD setup file for the nominal corner:

```
eadInitializeDesign(cv ".cadence/dfII/EAD/1/setup/gpdk090.ini" "nominal")  
=> t
```

eadExtractParasitics

```
eadExtractParasitics(  
    t_layoutDesignCellViewId  
    [ l_netNames ]  
)  
=> t / nil
```

Description

Extracts parasitics from the specified layout design.

Before extracting parasitics using the `eadExtractParasitics` function, ensure that:

- Cellview must be open in append mode.
- The layout design is initialized using the `eadInitializeDesign` function.

Arguments

t_layoutDesignCellViewId

Cellview ID of the layout design.

l_netNames

List of nets for which parasitics are to be extracted.

This is an optional argument. If you do not specify this argument, parasitics are extracted for all nets in the specified design.

Value Returned

t

Parasitics are extracted.

nil

Parasitics cannot be extracted because of an error.

Example

Opens the cellview in append mode.

```
cv = (dbOpenCellViewByType "Two_Stage_Opamp" "OpAmp" "layout_user" "maskLayout"  
"a")  
=> db:0xb697de9a
```

Extracts parasitics for nets `net14` and `Ibias` in the specified layout design:

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

```
eadExtractParasitics(cv "net14" "Ibias")  
=> t
```

Extracts parasitics for all nets in the specified layout design.

```
eadExtractParasitics(cv)  
=> t
```

eadRemoveDesignParasitics

```
eadRemoveDesignParasitics(  
    t_layoutDesignCellViewId  
)  
=> t / nil
```

Description

Removes the parasitics extracted from the specified layout design.

Before removing parasitics using the `eadRemoveDesignParasitics` function, ensure that:

- Cellview must be open in append mode.
- The layout design is initialized using the `eadInitializeDesign` function.

Arguments

t_layoutDesignCellViewId

Cellview ID of the layout design.

Value Returned

<code>t</code>	Parasitics are removed.
<code>nil</code>	Parasitics cannot be removed because of an error.

Example

Opens the cellview in append mode.

```
cv = (dbOpenCellViewByType "Two_Stage_Opamp" "OpAmp" "layout_user" "maskLayout"  
"a")  
=> db:0xb697de9a
```

Removes parasitics from the specified layout design.

```
eadRemoveDesignParasitics(cv)  
=> t
```


Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

eadRunEM

```
eadRunEM(  
    d_cellViewID  
    [ l_netNames ]  
)  
=> t / nil
```

Description

Runs EM analysis on all nets or on a list of specified nets in the specified layout design.

Arguments

<i>t_cellViewID</i>	Cellview ID of the layout design.
<i>l_netNames</i>	A list of net names for which you want to run EM Analysis.

Value Returned

<i>t</i>	EM analysis is run successfully.
<i>nil</i>	EM analysis cannot be run because of an error.

Examples

The following example runs EM analysis on the nets, *net14*, *net17*, and *net019* in the specified design.

```
cvID=dbOpenCellViewByType("Two_Stage_Opamp" "DiffOpAmp"  
"mostly_routed_backup_00001")  
; Opens the specified cellview  
eadRunEM(cvID list("net14" "net17" "net019"))  
;Runs the EM analysis on the nets, net14, net17, and net019  
=> db:0x3fa3169a  
=> t
```

The following example runs EM analysis on all the nets in the specified design.

```
cvID=dbOpenCellViewByType("Two_Stage_Opamp" "DiffOpAmp"  
"mostly_routed_backup_00001")  
; Opens the specified cellview  
eadRunEM(cvID)  
;Runs EM analysis on all nets
```

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

=> db:0x3fa3169a

=> t

eadSaveSummaryTableToCSV

```
eadSaveSummaryTableToCSV(  
    t_csvFileName  
)  
=> t / nil
```

Description

Exports data from the Net Summary table of the EAD Browser to the specified CSV file.

Arguments

<i>t_csvFileName</i>	Path to the CSV file to which data from the Net Summary table is to be exported.

For example:

```
"/home/user/export/myNetSummaryTable.csv"
```

If you specify only the name of the CSV file, the CSV file is saved in the current working directory.

Value Returned

<i>t</i>	Data is successfully exported to the specified CSV file.
<i>nil</i>	Data cannot be exported because of an error.

Examples

The following example exports data from the Net Summary table of the EAD Browser to the CSV file at the specified location.

```
eadSaveSummaryTableToCSV("/home/user/export/myNetSummaryTabletoCSV.csv")  
=> t
```

The contents of the CSV file are:

```
Net,R Count,EM Viol,J/Jmax,Max Drop  
AVDD,338,0,0,0  
AVSS,141,,,  
Ibias,402,0,0,0  
inm,32,0,0,0  
inp,36,,,
```

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

```
net14,276,9,2.615,4.055e-3
net17,36,0,0,0
net019,39,21,5.702,2.746e-3
net021,76,0,0,0
net22,240,23,6.084,19.94e-3
net035,0,,
net036,189,0,0,0
net044,243,24,6.084,19.96e-3
outm,123,43,15.41,36.76e-3
outp,122,41,12.24,37.23e-3
```

The following example exports data from the Net Summary table of the EAD Browser to the specified CSV file in the current working directory:

```
eadSaveSummaryTableToCSV("myNetSummaryTabletoCSV.csv")
=> t
```

eadSetCurrentDataset

```
eadSetCurrentDataset(  
    d_cellViewID  
    t_datasetName  
)  
=> t / nil
```

Description

Sets the specified dataset as the active electrical dataset in the EAD Browser of the specified layout cellview.

Arguments

<i>d_cellViewID</i>	ID of the layout cellview.
<i>t_datasetName</i>	Name of the dataset to be set as the active electrical dataset.

Value Returned

<i>t</i>	The specified dataset is set as the active electrical dataset.
<i>nil</i>	The dataset cannot be set as the active electrical dataset because of an error.

Examples

The following example sets the dataset "dataSet_7_I" as the active electrical dataset in the EAD Browser of the specified cellview.

```
cvID=dbOpenCellViewByType("Two_Stage_Opamp" "DiffOpAmp"  
"mostly_routed_backup_00001")  
; Opens the specified cellview  
eadGetAllEMDatasets()  
;Returns a list of all available EM datasets in the current cellview  
eadSetCurrentDataset(cvID "dataSet_7_I")  
; Sets the dataset "dataSet_7_I" as the active electrical dataset in the EAD Browser  
of the current cellview  
=> db:0x3fa3169a  
=>  
("myDataset_I" "dataSet_8_I" "dataSet_7_I" "dataSet_6_I" "dataSet_4_I"  
 "dataSet_1_I" "dataSet_0_I" "EMData"  
)  
=> t
```

Virtuoso Electrically Aware Design Flow Guide

Electrically Aware Design Functions

The following example sets the dataset "All" as the active electrical dataset in the EAD Browser of the specified cellview.

```
cvID=dbOpenCellViewByType("Two_Stage_Opamp" "DiffOpAmp"
"mostly_routed_backup_00001")
; Opens the specified cellview
eadGetAllEMDatasets()
;Returns a list of all available EM datasets in the current cellview
eadSetCurrentDataset(cvID "All")
; Sets the dataset "All" as the active electrical dataset in the EAD Browser of the
current cellview
=> db:0x3fa3169a
=>
("myDataset_I" "dataSet_8_I" "dataSet_7_I" "dataSet_6_I" "dataSet_4_I"
"dataSet_1_I" "dataSet_0_I" "EMData"
)
=> t
```

Generating an EAD Technology File

An EAD technology file, named `eadTechFile`, is a binary encrypted file that contains electrical process parameters for RC extraction, halo distances, capacitance models, and the electromigration rules. While preparing the Layout EAD setup before running parasitic extraction and electromigration checks, you need to specify an EAD technology file in the *Process Settings* section on the Layout EAD Options form.

Virtuoso provides the `eadModelGen` executable utility under `$CDSHOME/tools/catena/bin` in the Virtuoso installation area. You can use this utility to generate an EAD technology file.

Note: You would require a separate EAD technology file for every combination of process technology, layer stack, and process corner.

The following sections provide the details on how to use the `eadModelGen` utility to create or update an EAD technology file:

- Prerequisites
- Generating an EAD Technology File
- Example Scenarios of EAD Technology File Generation
- Updating an EAD Technology File

Prerequisites

Before running the `eadModelGen` utility, ensure that the following are available:

- **A process description file in Cadence-standard text format:** You can either provide `qrcTechFile` or an ICT file that describes all the process parameters required for RC extraction.

If using `qrcTechFile`, ensure that the file contains the RCGEN models.

If using an ICT file, you can also add rules for electromigration checking to the ICT file using the `em_model` statements. It is mandatory that the file is provided in the Cadence-standard text format that has been defined in conjunction with various foundries. For details on how to create an ICT file, refer to the *EXT Techgen Reference Manual* in QRC installation.

- **A multi-core machine with large RAM:** The `eadModelGen` is a resource-intensive utility, so it is recommended to use a multi-core machine, preferably with more than 16 cores and 64GB of RAM.

Generating an EAD Technology File

To run the `eadModelGen` utility, run the following command:

```
$ eadModelGen <ictfilename> | qrcTechFile
```

The utility reads the specified input ICT file or `qrcTechFile` to create a technology file and saves it at the current location. The input file is usually an ASCII file in the ICT format that contains the process description, but a binary `qrcTechFile` may also be used if an ICT file is not available.

For the `<ictfilename>` argument, you can provide the name of a technology file (`ictfile`, `eadTechfile`, or any other custom name) or a symbolic link to a technology file.

Note: If the input file is an ICT file, EAD first looks for `eadTechFile` in the same directory. If found, loads it to update the content. Otherwise, loads the ICT file.

Depending on your requirements, you can also use one or more of the following optional command-line arguments:

Command-Line Argument	Purpose
<code>-h</code> <code>-help</code>	Prints help for the utility
<code>-v</code> <code>-version</code>	Prints the tool and library version information

Virtuoso Electrically Aware Design Flow Guide

Generating an EAD Technology File

Command-Line Argument	Purpose
<code>-output_file</code>	Saves the output in a file specified using this argument. If not specified, an output file with the default name, <code>eadTechFile</code> , is created.
<code>-log_file</code>	Saves output to the given log file
<code>-dry_run</code>	Displays the run information and options without actually running the command
<code>-threads</code> <code>-multi_cpu</code>	Runs the model generation jobs using the given number of threads
<code>-update</code>	Updates the complete ICT file contained in <code>eadTechFile</code> with the given ICT file. You cannot update the <code>eadTechFile</code> using an ICT-EM file or <code>emDataFile</code> that is read only using the EAD Browser.
<code>-update_em <EM-ircx-file></code> <code>-ircx_file <RC-ircx-file></code>	Updates an existing <code>eadTechFile</code> with EM rules from an EM <code>ircx</code> file. While updating the EM rules, the command looks for matching layer names in the <code>eadTechFile</code> and the given EM <code>ircx</code> file. However, when the layer names in these files do not match, you must provide an RC <code>ircx</code> file along with the EM <code>ircx</code> file by using the optional <code>-ircx_file</code> argument. The RC <code>ircx</code> file provides the missing layer mapping information. You cannot update the <code>eadTechFile</code> using an ICT-EM file or <code>emDataFile</code> that is read only using the EAD Browser. Note: This command-line option updates only an existing EAD technology file. It cannot be used to update a QRC technology file.

Virtuoso Electrically Aware Design Flow Guide

Generating an EAD Technology File

Command-Line Argument

Purpose

`-lsf_command`

Runs the model generation jobs using LSF. Use this argument to provide a command string that defines how to submit jobs to an LSF queue in your specific LSF environment. For example:

```
-lsf_command "bsub -q lnx64 -m pto_opt -err err  
-o out " -lsf_number 100
```

Note that the LSF command must be enclosed in double quotes. You also need to specify the `-lsf_number` argument. It is also recommended that you use a unique project name (`-P`) when submitting your jobs so that you can easily query the job status later.

Important

Depending on the process and the number of conductor layers, the LSF jobs might take a long time to generate models. Therefore, after submitting jobs to the LSF server, the `eadModelGen` command exits. When you find that LSF jobs have completed the run, you must execute the `eadModelGen` command again, with the same options except `-lsf_command` and `-lsf_number`, to compile all the models into a binary technology file.

The following example steps show how to generate a technology file:

1. `eadModelGen -lsf_command "bsub -q lnx64" -lsf_number 8 ictfile`
2. Confirm that all the LSF jobs have completed their runs
3. `eadModelGen ictfile`

If you want the command to wait for the completion of jobs before exiting, use the `-lsf_wait` and `-lsf_wait_interval` options.

The `eadModelGen` command internally uses the `bjobs` command to check the status of the submitted LSF jobs. If `bjobs` fails to return the status of the jobs, the command exits and shows an error message.

Virtuoso Electrically Aware Design Flow Guide

Generating an EAD Technology File

Command-Line Argument	Purpose
<code>-lsf_number</code>	Specifies how many LSF jobs to run in parallel. This argument is used with the <code>-lsf_command</code> argument. For example: <pre>-lsf_command "bsub -q lnx64 -m pto_opt -err err -o out " -lsf_number 100</pre> You can use the <code>-threads</code> with this argument for multi-threaded execution in LSF.
<code>-lsf_wait</code>	Forces the <code>eadModelGen</code> utility to wait until all the LSF jobs have completed and <code>eadTechFile</code> has been created. This option prevents the command to exit immediately after running the LSF commands. Note: ■ This argument is useful only when used with <code>-lsf_command</code> and <code>-lsf_wait_interval</code>. ■ If you use this command, you do not need to rerun the <code>eadmodelGen</code> utility after completion of LSF jobs.
<code>-lsf_wait_interval <no-of-seconds></code>	Specifies the duration (in seconds) after which the <code>eadModelGen</code> utility needs to repetitively check the completion status of LSF jobs. After all the jobs are complete, the <code>eadModelGen</code> utility exits. Note: This argument is useful only when specified with <code>-lsf_command</code> and <code>-lsf_wait</code>.
<code>-mail <email_address></code>	Sends an email to the given address after all model generation jobs have completed and the EAD technology file has been created. After receiving the completion mail, you must execute the <code>eadModelGen</code> command again, with the same options except <code>-lsf_command</code> and <code>-lsf_number</code>, to compile all the models into a binary technology file.
<code>-overwrite</code>	Forces overwrite of the existing <code>eadTechFile</code> if one already exists in the output directory.
<code>-split</code>	Splits the components of an existing <code>eadTechFile</code> into the given directory

Virtuoso Electrically Aware Design Flow Guide

Generating an EAD Technology File

Command-Line Argument	Purpose
<code>-info</code>	Prints the summary information about the specified ICT file, EAD technology file, or <code>qrcTechFile</code> . Provide the complete directory path to the file for which you want to print information.
<code>-comments</code>	Prints the comments embedded in the ICT file in addition to the summary information printed by <code>-info</code> . Note: Use this argument along with <code>-info</code> .
<code>-no_cap_models</code>	Creates an <code>eadTechFile</code> without capacitance models. The resulting <code>eadTechFile</code> can be used only for the R extraction mode, EM checking, and high precision C extraction, but cannot be used for fast C extraction. Note: The time required to generate this partial <code>eadTechFile</code> is much less than the time required to create the default file that contains the capacitance models.  Tip An efficient way can be to get started using EAD very quickly using this option, that is with the R-only extraction, EM checking, and HPC. In parallel, you can generate the full <code>eadTechFile</code> that includes capacitance models too.
<code>-no_load_models</code>	When using an existing <code>eadTechFile</code> , instead of an ICT file, as the input to <code>eadModelGen</code> for model generation, you can specify this option. This will avoid loading the existing capacitance models in the <code>eadTechFile</code> , which will not be used anyway.
<code>-use_ead2dc</code>	Use the native <u>EAD 2D</u> solver instead of the default <u>FEM 2D</u> solver to generate the capacitance models for all the layers. This option may be necessary if model generation fails to complete with the default FEM 2D solver.

Virtuoso Electrically Aware Design Flow Guide

Generating an EAD Technology File

Command-Line Argument	Purpose
<code>-use_ead2dc_for_device_layers</code>	Use the native <u>EAD 2D</u> solver instead of the default <u>FEM 2D</u> solver to generate the capacitance models for the local interconnect and device-level layers. The default FEM 2D solver will be used for all other layers. This option may be necessary if model generation with the FEM 2D solver fails to complete for these layers.
<code>-no_cap_3d_models</code>	Creates an eadTechFile without any 3D correction capacitance models. As a result, EAD will not consider any 3D corner or crossover effects for normal capacitance extraction. However, this does not affect extraction using the High Precision C Solver.
<code>-skip_layer_type</code>	Skips generating capacitance models involving layers of the given type. For example, <code>diffusion</code> , <code>poly</code> , <code>lil</code> , or <code>metal</code> .
<code>-lithobias_file</code>	Specifies the lithoBias rules file to be considered for the generation of eadTechFile. This file should be in the standard OpenDFM format provided by Si2. The lithoBias rules are added to eadTechFile. Note: If the input file for the eadModelGen command is a QRC technology file that contains lithoBias rules, these rules are automatically included in the output eadTechFile. You can use the <code>eadModelGen -info</code> command to view the names of layers that include lithoBias rules.
<code>-combination_layers</code>	Limits the number of capacitance models to be generated to the set of layer combinations specified by the <code>combination_layers</code> section in the ICT file.
<code>-protected_process</code>	Protects the process information in the resulting eadTechFile so that the ICT process description cannot be extracted from the file using the <code>-split</code> option.
<code>-max_spacing_factor</code>	Overrides the default halo spacing distance per layer, which is the distance that EAD uses to search for neighboring shapes. Specify a factor to be multiplied by the <code>min_spacing</code> value in the ICT file. This option may be necessary in order to generate capacitance models for processes that define a very thick substrate, or thick oxide layers below the first conductor layer.

Virtuoso Electrically Aware Design Flow Guide

Generating an EAD Technology File

Command-Line Argument	Purpose
<code>-max_simulation_widths</code>	Specifies that if the ICT file contains <code>manual_simulation_points</code> , use the <code>simulation_widths</code> table to define the maximum width per conductor layer. This limits the width ranges used while generating models.
<code>-max_width_file <file-name></code>	Specifies an optional file that defines the maximum width for each conductor layer in the ICT file. Other than the comments marked with #, each line in the given file should consist of a layer name followed by a maximum width value (in microns). Note: It is required that the maximum width value should be at least 9 times the minimum width value for the given layer.
<code>-status</code>	Returns the status of model generation jobs for the given ICT file.

Important Points to Note

- If the `eadModelGen` utility does not generate an `eadTechFile` due to some network or disk space issues, you can execute the command again. It will continue to run from the point where it stopped in the previous run.
- The time required to generate an `eadTechFile` with capacitance model depends on the hardware configuration of the computer on which it is running, the number of threads, and the number of layers for which model information is to be generated.
- The `eadModelGen` utility also works with OpenLava, the open-source equivalent of LSF.
- An `eadTechFile` generated using a newer version of Virtuoso is not compatible with an earlier version of Virtuoso.
- It is recommended to use `-lsf_wait` with `-lsf_interval` in LSF mode to ensure that the `eadModelGen` command does not exit after submitting the LSF jobs and compiles all the models generated by different jobs into a common `eadTechFile`.

Example Scenarios of EAD Technology File Generation

This section provides some of the possible example scenarios in which an `eadTechFile` can be generated with different requirements.

Virtuoso Electrically Aware Design Flow Guide

Generating an EAD Technology File

Scenario 1: You need to update an existing `eadTechFile` with a new ICT file

You can update an existing `eadTechFile` with a new ICT file only if there is no change in the ICT file that would affect the capacitance models. For example, in cases where the existing `eadTechFile` does not contain the EM rules, but now you need to include the EM rules added to the ICT file. In this case, do the following:

1. Change the name of the new ICT file to a name other than `ictfile`.
2. Run the `eadModelGen` utility with this following command:

```
$ eadModelGen -update <new_ict_file_name> eadTechFile
```

Scenario 2: You need to extract all the components contained in the given `eadTechFile`.

You can do this by executing the following command:

```
$ eadModelGen -split <tmp_dir> eadTechFile
```

If you run this command on a protected `eadTechFile`, the specified directory will contain all the contents of `eadTechFile` except the ICT File.

The following examples describe how the `eadTechFile` is split in different cases:

- If the `eadTechFile` was generated for R-only extraction, `<tmp_dir>` will contain the VHALOS file.
- If the `eadTechFile` was generated for RC extraction, `<tmp_dir>` will contain VHALOS and all the model files, but not the ICT File.

Scenario 3: You need to protect the process information in the `eadTechFile`.

You can use the `-protected_process` command-line argument, as shown below.

```
$ eadModelGen -protected_process -threads 16 ictfile
```

Note: If the `eadTechFile` is protected, it is not possible to extract the ICT file using the `-split` command-line argument.

Scenario 4: The generation of the complete `eadTechFile` with capacitance models takes a very long time (a few hours), so you need to generate a partial `eadTechFile` without capacitance models.

You can do this by executing the following command:

```
$ eadModelGen -no_cap_models -threads 16 ictfile
```

Scenario 5: Your ICT file contains an explicit specification of the layer combinations required for model generation in the `manual_simulation_combinations` section. You need to

Virtuoso Electrically Aware Design Flow Guide

Generating an EAD Technology File

include only those layer combinations in the `eadTechFile` instead of automatically generating all the possible layer combinations.

To do this, run the following command to limit the model generation to only the specified layer combinations:

```
$ eadModelGen -combination_layers -threads 16 ictfile
```

Scenario 6: You need to verify some details for an existing `eadTechFile`.

You can do this using the `-info` command-line argument, as shown below:

```
$ eadModelGen -info -log_file info.log eadTechFile
eadModelGen version 7.2.6r910, built 2014/03/31 23:01:34, platform opteron (64bit)
Current time is Tue Apr 01 11:19:49 2014
Loading technology file : eadTechFile
Technology File : ./eadTechFile
Process Name      : cln28hp_lp10m_7x2r_typical
Model version     : 6
Created on        : 2013/08/11 14:39:47
Created by        : eadModelGen version 7.2.6r543 (built 2013/08/05 23:41:52)
Layers            : 23
Diffusion         : 1 (active)
Poly              : 1 (poly)
Conductors        : 10 (M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, C1, C2, B1, B2)
lithoBias         : 0
Vias              : 11 (odCont, polyCont, VIA1, VIA2, VIA3, VIA4, VIA5, VIA6, VIA7, VIA8, VIA9)
EM Models         : 0
Cap models        : 2944
```

Scenario 7: You have a QRC technology file, which contains a copy of the ICT file (including the EM rules). You need to use this for `eadTechFile` generation.

In this case, you can run the following command to generate `eadTechFile` from `qrcTechFile`:

```
$ eadModelGen -threads 16 qrcTechFile
```

Scenario 8: You have an EAD technology file, but the EM rules are missing. You need to update the `eadTechFile` to include EM rules from an iRCX file.

You can do this by performing the following steps:

1. Update the EAD technology file using the `eadModelGen` utility, as shown below.

```
$ eadModelGen -update_em <iRCX file> <eadTechFile>
```

2. (Optional) Confirm that the EAD technology file has EM models by using the `eadModelGen` utility, as shown below.

```
$ eadModelGen -info eadTechFile
```

The EM rules get added to the existing `eadTechFile` in lines beginning with `EM models`.

Virtuoso Electrically Aware Design Flow Guide

Generating an EAD Technology File

Scenario 9: Run `eadModelGen` with LSF using 200 jobs, wait for the LSF jobs to complete (checking every 10 minutes), and then receive an email when `eadTechFile` has been created.

You can do this by submitting an LSF command and using the supporting options, as shown below.

```
eadModelGen -lsf_command "/grid/sfi/farm/bin/bsub -P EAD:6.1.8:RD:gpd090 -R
2015_lnx86_test -W 48:00 -q lnx64" -lsf_number 200 -lsf_wait -lsf_wait_interval
600 -mail abc@mailaddress.com ictfile
```

The LSF command string will depend on your specific LSF installation. It is recommended to use a unique project name (`-P`) when submitting your jobs so that you can query the job status later using that name. In the above example command, a project name has been provided using `-P`. You can query its status as shown below.

```
$ bjobs -P EAD:6.1.8:RD:gpd090
```

Updating an EAD Technology File

As the foundries update the ICT and processes on a regular basis, you might also need to update the EAD models in the `eadTechFile`. For this, you can use the `-update` command-line argument to update the `eadTechFile` without regenerating the capacitance models.

Some examples of changes in ICT that **do** require generation of new EAD models are given below.

- When there is a change in parameters that affect the capacitance, including layer heights, thickness, widths, spacings, bias (top/bottom enlargement), WEE tables, dielectrics, layout scaling.
- When a new `eadTechFile` will need to be generated.

You might also need to update the `eadTechFile` if some modeling improvements have been done.

Some examples of changes in ICT that **do not** require generation of new EAD models are given below.

- Changes to the `sub_conductor` device layers. This is because currently, EAD does not use these layers.
- Changes to resistance parameters, for example, rho tables, and `wire_edge_enlargement_r`.
- Addition or changes in the EM rules

After upgrading Virtuoso, you need not update or regenerate the `eadTechFile` because it is backward compatible.

EM Rule Support in EAD

This section lists the differences between EM rules supported in EAD and Voltus-Fi:

- The EM parameter `em_recover` is supported by the parser but it is not used during EM analysis.
- The following operators are supported in lowercase in EAD:
 - ❑ `sqrt`
 - ❑ `log`
 - ❑ `exp`
 - ❑ `abs`
 - ❑ `min`
 - ❑ `max`
 - ❑ `normsinv`
 - ❑ `*` `/` `+` `-`
 - ❑ `()`
 - ❑ `e`
 - ❑ `^`
 - ❑ `expr1?expr2: expr3`

Voltus-Fi-XL supports parameters for only limit-based EM analysis. The following parameters can be used to specify EM rules in the ICT EM file. The EM rules might also be embedded in the ICT file or the `qrcTechFile`.

Parameter Name	Description
<code>em_vewidth</code> <code><value></code>	Optional parameter for <code>via</code> layers that defines the minimum size via that is used by the software to identify single square via. This parameter is similar to the <code>viaWidthList</code> parameter in <code>emDataFile</code> .

Virtuoso Electrically Aware Design Flow Guide

Generating an EAD Technology File

<code>em_W_n</code>	Optional parameter that specifies the wire width in microns below which a line is considered narrow. All lines are considered wide unless you specify otherwise using this parameter Note: It helps to select the rule between <code>em_jmax_*_w</code> and <code>em_jmax_*_n</code>. If <code>w < em_W_n</code>, then <code>em_jmax_*_n</code> is selected, otherwise <code>em_jmax_*_w</code> selected.
<code>em_jmax_dc_rms_metal_line_factor</code>	Defines the relationship between the value of power on metal lines and the <code>Irms</code> rating values for DC RMS analysis.
<code>em_jmax_ac_rms_metal_line_factor</code>	Defines the relationship between the value of power on metal lines and the <code>Irms</code> rating values for AC RMS analysis.
<code>em_ac_rms_context_aware_pitch_num</code>	Context-aware relaxation keyword. Specifies the pitch count values for RMS relaxation for AC RMS analysis in the context-aware RMS Relaxation flow. For example, <pre>em_ac_rms_context_aware_pitch_num 5</pre> Note: The context-aware RMS Relaxation flow is foundry-specific. For more information, contact your Cadence representative.
<code>em_dc_rms_context_aware_pitch_num</code>	Context-aware relaxation keyword. Specifies the pitch count values for RMS relaxation for DC RMS analysis in the context-aware RMS Relaxation flow. For example, <pre>em_dc_rms_context_aware_pitch_num 5</pre> Note: The context-aware RMS Relaxation flow is foundry-specific. For more information, contact your Cadence representative.

Virtuoso Electrically Aware Design Flow Guide

Generating an EAD Technology File

`em_jmax_ac_rms_context_aware_factor` Additional factor to be multiplied for EM limits to take into consideration the context-aware RMS relaxation flow.

For example,

```
em_jmax_ac_rms_context_aware_factor 1.5
srms_0>= 0.008 srms_1> 0.0004
```

Where,

`srms_0` is the worst `srms` value in the neighboring area within the context-aware region of the parasitic.

`srms_1` is the second worst `srms` value in the neighboring area within the context-aware region of the parasitic.

Note: The context-aware RMS Relaxation flow is foundry-specific. For more information, contact your Cadence representative.

`em_jmax_dc_rms_context_aware_factor` Specifies the benefit factor for RMS relaxation for DC RMS analysis in the context-aware RMS Relaxation flow.

Additional factor to be multiplied for EM limits to take into consideration the context-aware RMS relaxation flow.

For example,

```
em_jmax_dc_rms_context_aware_factor 1.5
srms_0>= 0.008 srms_1> 0.0004
```

Where,

`srms_0` is the worst `srms` value in the neighboring area within the context-aware region of the parasitic.

`srms_1` is the second worst `srms` value in the neighboring area within the context-aware region of the parasitic.

Note: The context-aware RMS Relaxation flow is foundry-specific. For more information, contact your Cadence representative.

Virtuoso Electrically Aware Design Flow Guide

Generating an EAD Technology File

<code>em_segment_length</code>	A process definition parameter that determines how to calculate the length of a wire segment. This parameter determines whether the segment length is of the type perimeter, centerline, or total. The difference between the different length types is as follows: ■ <code>perimeter</code>: specifies that the length calculation is along the wire's outer edge boundary. This value is used for 28nm process nodes. ■ <code>centerline</code>: specifies that the length is extended to the edge of the segment along the centerline path. This value is used for 20nm process nodes. ■ <code>total</code>: specifies that the segment length is calculated by adding the path lengths of all the resistors in the segment.
<code>em_recover</code>	This parameter is supported in the parser and in the EM process, but it is not used during EM analysis. Optional parameter that specifies the recovery factor used for calculating the average current density with recovery. You can set the recovery factor to -1.0 when you want to account for the average of the absolute current (<code>avg(abs(current))</code>) so that the negative current does not cancel the positive current values in arriving at the average current result. <i>Default:</i> 1 <i>Range:</i> A positive integer or floating point number

Current Density (JMAX) Keywords

The table below lists the Current Density (JMAX) keywords that can be specified in the ICT EM file.

Keyword	Description
<code>em_jmax_dc_avg*</code>	Optional keyword that specifies the DC AVG analysis.
<code>em_jmax_ac_avg*</code>	Optional keyword that specifies the AC AVG analysis.
<code>em_jmax_dc_peak*</code>	Optional keyword that specifies DC PEAK analysis.
<code>em_jmax_ac_peak*</code>	Optional keyword that specifies AC_PEAK analysis.
<code>em_jmax_dc_rms*</code>	Optional keyword that specifies DC RMS analysis.

Virtuoso Electrically Aware Design Flow Guide

Generating an EAD Technology File

<code>em_jmax_ac_rms*</code>	Optional keyword that specifies AC RMS analysis.
<code>em_jmax_ac_rec</code>	Optional keyword that specifies the maximum recovery AC current analysis.

Where star '*' implies that '_w' and '_n' rules are also supported.

For example,

- `em_jmax_ac_peak_w` is an optional parameter that is essentially the same as `em_jmax_ac_peak`, with the exception that it is specified for wide wires (as defined by `em_W_n`).
- `em_jmax_ac_peak_n` is an optional parameter that is essentially the same as `em_jmax_ac_peak`, with the exception that it is specified for narrow wires (as defined by `em_W_n`).

Rules for Defining EM Parameters

An example of the EM rule for defining the `em_jmax_dc_avg` parameter is provided below:

```
[em_jmax_dc_avg <value> | <value_1> <area_1/width_1> [...] | EQU <fn(E)>]
[jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
[jmax_life <life1> <scale1> [<life2> <scale2>.....]]
[current_direction up | down | both]
[conditions]
[single]
[power_rail/power_grid]
[priority==<priority no.>]
[device=="<device model names>"]
[sub_conductor=="<subconductor names>"]
[color=="<list of color no.>"]
[mask==<mask no.>]
[via_range <value>]
```

All other EM parameters are defined with the same structure but apply to different characteristics of the `em_model` (peak current versus average for instance). The following rules apply to all the limit-based `em_model` parameters:

■ Single Value Current Density

Specifying a single value or an equation is an enhancement for width or area dependent parameters (such as `em_jmax_ac_avg`).

Note: For narrow/wide wire parameters (such as `em_jmax_ac_avg_n` or `em_jmax_ac_avg_w`) you can only specify a single value. It will not accept either a PWL table or an equation.

■ Piece-wise Linear (PWL) Interpolation

In a width-dependent model for a conductor layer, or for an area-dependent model for a via layer, PWL interpolation uses a series of data points that are points on a graph of X versus Y, where X can be either a value or an expression and Y is a value. For example, current density value and area `<value1> <area1>`, current density value and width `<value1> <width1>`, current density expression and area `<expression1> <area1>`, or current density expression and width `<expression1> <width1>`.

The following syntax are supported:

```
em_jmax_dc_avg PWL <value1> <area1/width1> <value2> <area1/width1>  
em_jmax_dc_avg PWL (<value1> <area1/width1> <value2> <area1/width1>)  
em_jmax_dc_avg PWL (<expression1> <area1/width1> <expression2> <area1/  
width1>)
```

For example:

```
em_jmax_dc_avg PWL 0.3582 0.001444 0.5694 0.003364 0.7164 0.0038  
em_jmax_dc_avg PWL (0.3582 0.001444 0.5694 0.003364 0.7164 0.0038)  
em_jmax_dc_avg PWL (0.05*0.5 0.002 0.1*0.5 0.004 0.2*0.5 0.008)
```

■ Equations in EM Model

The EQU keyword denotes the definition of an equation for current density.

An EOL or one of the qualifiers (`jmax_factor`, `current_direction`, or `single`) marks the end of an equation specifying the Current Density limit. If the equation resolves into two separate equations, then this will result in an error.

For EQU, the following operators (case-sensitive) are supported:

- ❑ `sqrt`
- ❑ `log`
- ❑ `exp`
- ❑ `abs`
- ❑ `min`
- ❑ `max`
- ❑ `normsinv`
- ❑ `*` `/` `+` `-`

- ❑ `()`
- ❑ `e`
- ❑ `^`
- ❑ `expr1? expr2: expr3`

Where, `expr1` is a conditional expression. If `expr1` evaluates to `true`, then `expr2` is evaluated, else `expr3` is evaluated.

For example:

```
Life_factor= ((A < 5 and B > 2)? 3:4)
```

```
em_jmax_dc_avg EQU 3.0*w * Life_factor jmax_factor 50 1.1 110 1.0  
125 0.92 L > 5
```

Note: Square brackets `[]` are not supported. A comma `,` implies multiple equations, and results in an error. The period `.` is ambiguous and should not be used to indicate multiplication (use `*` instead).

■ Spaces are permitted to make an equation more readable.

Variables within equations, such as `deltaT`, `w`, and `l`, are populated by the analysis tool at runtime. Use the `em_variables` parameter in the process definition to globally declare variables that are used within the various `em_models`.

The equation definition ends when reaching the end of line (EOL) or another keyword like:

The equation definition ends when reaching the end of line (EOL) or another keyword like:

- ❑ `/* Limit-based EM equations (EQU), terminated by any of the following:`
- ❑ `// EOL`
- ❑ `// jmax_factor`
- ❑ `// jcdf_factor`
- ❑ `// rating_factor`
- ❑ `// current_direction`
- ❑ `// single`
- ❑ `// power_rail`

Virtuoso Electrically Aware Design Flow Guide

Generating an EAD Technology File

```
❑ // power_grid
❑ // priority
❑ // mask
❑ // color
❑ // VO_
❑ // Itolerance
❑ // via_range
❑ // jmax_life
❑ // conditions [L/W/Lu/Wu/Lb/Wb/Lv2v/Wv/Lv/Vg/N/a == | < | <=
| > | >= <value>
❑ // device
❑ // sub_conductor
❑ // sub_via
❑ // bridge_via
❑ // MD_AND_OD_OVERLAPPING_REGION
❑ // Sh
❑ // hi_em
❑ // em_ (the next em_ rule)
❑ // } (the end of the model)
```

List of supported variables are provided below:

- ❑ a: area of the resistor
- ❑ w: width
- ❑ l: length
- ❑ La: length of pseudo via
- ❑ La_x: length of pseudo via along the x-axis
- ❑ La_y: length of pseudo via along the y-axis
- ❑ N: number of vias

- ❑ `deltaT`: rise in temperature in degree Celsius
- ❑ `r`: duty ratio
- ❑ `Tref`: refers to `em_tref`, the nominal temperature for EM analysis defined in the process section of the ICT file. If `em_tref` is not defined, then default value(110C) is set.
- ❑ `cdf_percentage`: This variable lets you use the value specified using the keyword `cdf_percentage` in the emir config file or command file, in the EM rules.

■ Order Dependency

The actual Current Density limit (value, piece-wise linear or PWL pairs, or equation) should follow immediately after the Current Density keyword (`em_jmax_dc_avg`). Any of the qualifiers (`jmax_factor`, `current_direction`, or `single`) will come after that. The order of the qualifiers is not important.

■ Units of Values

The unit should be specified in accordance with the setting defined by the `em_conductor_unit`, `em_via_unit`, and `em_via_area_unit` parameters defined in the process definition. Units are specified as:

- ❑ For a single value or an equation:
 - A/cm^2 or mA for metal layers (width based)
 - A/via or mA for via layers (per via)
- ❑ For a PWL table:
 - A/cm^2 or mA, with width specified in microns for metal layers
 - A/cm^2 or mA, with area specified in square microns for via layers

■ JMAX Factor

`jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]`

`jmax_factor` is the optional scaling factor to use at different temperatures compared to the reference temperature (defined by `em_tref` in the process definition). The temperature for `jmax_factor` should be specified in degrees Celsius.

Scaling factor is a positive integer: >1 to scale up, <1 to scale down, or 1 for no scale effect.

If the specified temperature, T , falls between the defined minimum and maximum temperatures, the software will calculate a new scale factor (RT) using the formula, $TTF=A \cdot J^{(-n)} \cdot \exp(Ea/kT)$.

Virtuoso Electrically Aware Design Flow Guide

Generating an EAD Technology File

For example:

$$RT = RT1 * \exp((Ea/kT) * (1/T - 1/T1))$$

where

$$Ea/kT = (T1 * T2 / (T1 - T2)) * \ln(RT2/RT1)$$

■ Format of the Conditions

Condition = [L/W/Lu/Wu/Lb/Wb/Lv2v/Sv/Td/r/a/N < | <= | > | >= | == | != <value in microns/micro seconds> | ground_net | supply_net | Vu_current_direction {up | down}] | [Itolerance < | <= | > | >= | == | != <value in percentage>]

Conditions = [condition]*

The **L** parameter specifies the length-based jmax values in microns.

The **W** parameter specifies the width-based jmax values in microns.

L and **W** are applicable for both metals and vias.

Lu, **Lb**, **Wu**, and **Wb** are applicable for vias only.

Lv2v and **Itolerance** are applicable only for the **power_rail** rule. **Itolerance** is not supported in EAD and will be ignored.

Where,

L = length

W = Width

Lu = Upper metal length

Lb = Bottom metal length

Wu = Upper metal width

Wb = Lower metal width

Lv2v = Distance between terminal vias

Itolerance = Difference in the current values of terminal vias. For example, if a difference of up to 5 percent is acceptable, then specify '**Itolerance** <= 0.05'.

Td = Time duration in micro second or total 'On Time' period

r = Duty ratio

a = Area of the single cut in the via or in the via array

N = Number of vias in the via array

Voltus-Fi-XL supports metal length/width rules above and below vias, where, the dimensions to be checked for the rules are different for the metal above and the metal below. In this case, **Lu/Wu** refers to dimensions of the metal above the via and **Lb/Wb** refers to dimensions of the metal below the via.

■ Single

Virtuoso Electrically Aware Design Flow Guide

Generating an EAD Technology File

The keyword, `single` can be specified to differentiate between a single square cut via and square viaarray. The following example shows how to use the `single` keyword:

```
em_jmax_dc_avg PWL 0.022 0.01 jmax_factor 105 1.1 110 1.0 115
0.9 120 0.8 125 0.7 130 0.6 140 0.5 150 0.4 current_direction up
Lb > 4 Wb >= 0.05 current_direction up single
```

■ JMAX Life

The keyword, `jmax_life` provides the ability to set the scaling factor that applies to the Current Density limits for different lifetimes. The syntax is as follows:

```
jmax_life <life1> <scale1> [<life2> <scale2>.....]
```

Note: The software will take the unit of lifetime from the `em_lifetime_units` parameter specified in the process section of the ICT file. In case the specified `life` value does not match any of the `life` values provided using `jmax_life`, the software performs linear interpolation to calculate the `Tlife` factor.

■ Current Direction

`current_direction` [`up` | `down` | `both`] can be specified for via current direction. The syntax of `current_direction` is:

- ❑ `up` - means the direction of the current is from bottom to top and specifies to use the length and width of the metal below `Lb/Wb` for the VIA Jmax factor
- ❑ `down` - means the direction of the current is from top to bottom and specifies to use the length and width of the metal above `Lu/Wu` for VIA Jmax factor
- ❑ `both` - This rule is applicable only when the direction of the current is uncertain. This rule is unlikely to be applied since the software calculates the current direction

■ Power Rail/Power Grid

The keyword `power_rail` or `power_grid` can be specified to enable the particular EM rule for power-rail analysis. These rules are not supported in EAD and will be ignored during parsing.

■ ApplyR

The keyword is not supported in EAD EM.

■ Priority

These rules are not supported in EAD and will be ignored during parsing.

■ Device

This keyword is used to specify different EM rules for device resistors, which are subsets of other layers. The syntax is as follows:

Virtuoso Electrically Aware Design Flow Guide

Generating an EAD Technology File

```
device=="device model names"
```

For example, the following keyword specifies that the particular EM rule is applicable only to device with the model names, “devRA devRB”.

```
device=="devRA devRB"
```

■ Subconductor

This keyword is used to specify different EM rules for subconductors, which are subsets of poly or other layers. The syntax is as follows:

```
sub_conductor=="subconductor names "
```

For example, the following keyword specifies that the particular EM rule is applicable only to subconductor layers, ppoly and gpoly.

```
sub_conductor=="ppoly gpoly"
```

■ bridge_via

This keyword is used to specify that the particular EM rule will apply only to bridge vias.

■ Color

This keyword is used to specify that the particular EM rule is applicable only to the resistor with the specified color number.

For example,

`color=="2 4 5"` specifies that the particular EM rule is applicable only to the resistor with property, \$M=2 OR \$M=4 OR \$M=5.

■ Masks

This keyword is used to specify that the particular EM rule is applicable only to the resistor with the specified mask number.

For example,

`mask==2` specifies that the particular EM rule is applicable only to the resistor with property, \$M=2.

Rules for Specifying Via Area

You can specify PWL in either one of the following ways:

■ Provide PWL for a specific area:

```
em_jmax_* PWL Value_1 area_1 ... Value_N areaN
```

For example,

```
em_jmax_* PWL 1.3 2.0 1.7 3.0
```

■ **Provide PWL for a specified area range:**

```
em_jmax_* EQU <equation with 'a'> conditions
```

For example,

```
em_jmax_* EQU (1.3 * a)/2.0 a>=0.0 a<2.5
```

```
em_jmax_* EQU (1.7 * a)/3.0 a>=2.5 N>=3
```

```
em_jmax_* EQU (1.9 * a)/3.0 a>=2.5 N<3
```

EM Rule Selection Priority

This section details the order of priority in which the EM rules are applied by Voltus-Fi-XL based on the specified keywords. The order of priority of keywords in descending order is provided below:

1. device --> **highest priority**
2. sub_conductor
3. bridge_via
4. priority
5. power_grid/power_rail
6. color
7. mask
8. current_direction
9. Conditions
10. Area matching in case of PWL
11. Base rule
12. Optimistic/pessimistic rules --> **lowest priority**

These are detailed below.

■ **device**

When a device is specified in the EM rule file, the software first looks for the `device` keyword and applies the EM rules to the specified device.

Virtuoso Electrically Aware Design Flow Guide

Generating an EAD Technology File

For example, for a device with model name, `devRA`, if the following rules are specified in the ICT file:

1. `em_jmax* ... device=="devRA"`
2. `em_jmax* ... device=="devRB"`
3. `em_jmax* ...`

Then the software follows the order of priority provided below:

- ☐ Match rule 1 because it has a matching `device` keyword
- ☐ Match rule 3 if rule 1 does not match
- ☐ Skip rule 2 because it is for device, `devRB`
- ☐ KPNS: CCMPR02674181

■ `sub_conductor`

When a `sub conductor` is specified in the EM rule file, the software first looks for the `sub_conductor` keyword and applies the EM rules to the specified sub conductor.

For example, for a resistor on layer, `ppoly`, if the following rules are specified in the ICT file:

1. `em_jmax* ... sub_conductor=="ppoly"`
2. `em_jmax* ... sub_conductor=="gpoly"`
3. `em_jmax* ...`

Then the software follows the order of priority provided below:

- ☐ Match rule 1 because it has a matching `sub_conductor` keyword
- ☐ Match rule 3 if rule 1 does not match
- ☐ Skip rule 2 because it is for sub conductor, `gpoly`
- ☐ KPNS: CCMPR02674181

■ `bridge_via`

When a `bridge_via` is specified in the EM rule file, the software first looks for the `bridge_via` keyword and applies the EM rules having `bridge_via` keyword for the bridge via. In case, the via is not bridge via, rules having `bridge_via` keyword will be ignored.

For example, for a bridge via, if the following rules are specified in the ICT file:

1. `em_jmax* ... bridge_via`
2. `em_jmax* ...`

Then the software follows the order of priority provided below:

- ☐ Match rule 1 because it is a bridge via
- ☐ Match rule 2 if rule 1 does not match

In case, the via is not bridge via, then the software will match rule 2.

■ **priority**

This keyword is not supported in EAD.

■ **power_rail/power_grid**

This keyword is not supported in EAD.

■ **color**

When the `color` keyword is specified in the EM rule file, in the following manner:

1. `em_jmax* ... color=="1 3 5"`
2. `em_jmax* ...`

For example, if a resistor has color number 5, specified using `$M=5`, then the software follows the order of priority provided below:

- ☐ Match rule 1 for the specified color number
- ☐ Match rule 2 if rule 1 does not match

■ **mask**

When the `mask` keyword is specified in the EM rule file, in the following manner:

1. `em_jmax* ... mask==2`
2. `em_jmax* ...`

For example, if a resistor has mask number 2, specified using `$M=2`, then the software follows the order of priority provided below:

- ☐ Match rule 1 for the specified mask number
- ☐ Match rule 2 if rule 1 does not match

■ **current_direction**

When the `current_direction` keyword is specified in the EM rule file, in the following manner:

1. `em_jmax* ... current_direction up`
2. `em_jmax* ... current_direction down`
3. `em_jmax* ...`

Then, for a resistor with `current_direction up`, the software follows the order of priority provided below:

Virtuoso Electrically Aware Design Flow Guide

Generating an EAD Technology File

- ☐ Match rule 1 because it has a matching `current_direction` keyword
- ☐ Match rule 3 if rule 1 does not match
- ☐ Skip rule 2 because the current direction is not matching

■ Conditions

When conditions are specified in the EM rule file, in the following manner:

```
1. em_jmax* ... L <=5
2. em_jmax ... L > 5
3. em_jmax* ...
```

Then the software follows the order of priority provided below:

- ☐ Match rules 1 and 2 because conditions are specified
- ☐ Match rule 3 if rules 1 and 2 are not matching

■ Area Matching from the PWL

When the PWL for specific via areas are specified in the EM rule file, in the following manner:

```
1. em_jmax* ... PWL VAL_1 AREA_1 VAL_2 AREA_2 L <=5
2. em_jmax ... PWL VAL_3 AREA_3 VAL_4 AREA_4 L <=5
```

Then the software follows the order of priority provided below:

- ☐ Match the exact areas provided in the rule file
- ☐ The software does not interpolate for unmatched areas (other than `AREA_1`, `AREA_2`, `AREA_3`, and `AREA_4` in above example)

■ Base Rule Selection

When the PWL for specific via area is specified in the EM rule file, **without any conditions** in the following manner:

```
em_jmax* ... PWL VAL_1 AREA_1 VAL_2 AREA_2
```

Then the software follows the order of priority provided below:

- ☐ Match the exact areas from the base rule
- ☐ The software does not interpolate for unmatched areas (other than `AREA_1` and `AREA_2` in above example)

When multiple rules match for a conductor/via, then maximum/minimum amongst them is returned depending upon the environment variable `layoutEAD.em useMaxRule`.

For more information about EM rules, see [*Voltus-Fi Custom Power Integrity Solution L User Guide*](#) and [*Voltus-Fi Custom Power Integrity Solution XL User Guide*](#).

Solvers for Generation of Capacitance Models

`eadModelGen` can use different solvers for capacitance model generation. Each solver has its own advantages and limitations. Depending on your accuracy requirements, choose a solver by using appropriate command options.

The solvers available for capacitance model generation are listed below:

- **FEM 2D solver:** Finite Element Method (FEM) 2D solver it is the default solver. It runs model generation on regular interconnect Back End of the Line (BEOL) layers. The BEOL is the second part of IC fabrication where the individual devices get interconnected with the wiring on the wafer. Both contribute towards the maximum rise in temperature that causes the self-heating effect.

The FEM 2D solver provides the best performance on the BEOL layers in the layer stack with high accuracy. It works best on those BEOL layers that are within some range of typical conductor dimensions for the given process. In general, for these types of BEOL layers, FEM 2D is much faster than EAD 2D with virtually the same accuracy, but in some cases, the performance might not be so good. Some of those cases are listed below:

- ☐ For the top layers in the stack, also referred to as RDL layers, which are very thick or wide relative to the other layers in the stack, and are typically used for global power or ground distribution. In FEM 2D, the entire process layer stack is meshed for solving, so these very thick and wide layers can result in a very large mesh, which may consume a large amount of memory and increase the time to solve. In extreme cases, FEM 2D can actually be slower than EAD 2D on these layers.
 - ☐ For the very thin layers. For example, the highly resistive RH_TN layers in the MEOL. These layers require using a very small mesh unit size. The presence of these very thin layers can result in a finer, and therefore, larger mesh for the layer stack, which increases the amount of memory required and decreases the performance.
- **EAD 2D solver:** Runs model generation for most layers, including the regular interconnect (BEOL) layers down to the device and local interconnect Front End of Line (FEOL) and Middle End of Line (MEOL) layers. FEOL is the first part of IC fabrication where the individual devices such as transistors, capacitors, and resistors are patterned in the semiconductor. FEOL covers everything up to, but not including, the deposition of metal interconnect layers. MEOL, which is also called as MOL, comprises contacts that connect the separate transistor and interconnect pieces.

EAD 2D provides a good balance of speed and accuracy. Unlike FEM 2D, the EAD 2D solver does not mesh the interior of the conductors. It only meshes the boundaries of the conductors and the dielectrics in between them. Therefore, it is not susceptible to the same issues that FEM 2D has when there are large variations in the conductor widths or thicknesses. In general, it is slower than FEM 2D.

Note: On some processes, particularly for the FEOL and MEOL layers, the default FEM 2D solver may not provide the best option for accuracy. In such cases, you can run the EAD 2D solver by using the `-use_ead2dc` command-line option. You can also run a hybrid mix of these solvers by using the `-use_ead2dc_for_device_layers` command-line option. With this option, EAD automatically runs the EAD 2D solver for any models involving FEOL or MEOL layers, and invokes the default solver for the BEOL layers.

Guidelines for Using a Solver

- It is recommended to use FEM 2D in general.
- Use EAD 2D only on those layers that either require higher accuracy (at the expense of speed), or for which FEM 2D shows performance issues.
- Use EAD 2D only for specific layers, not for all layers. For example, the very thin or thick layers for which FEM 2D shows performance issues. Specify the layer names using the `-use_ead2dc_for_device_layers` option. eadModelGen always picks up where it left, so you can run EAD 2D for selected layers, and then run it using FEM 2D to complete the remaining models.