

Virtuoso[®] Layout Suite XL User Guide

Product Version 6.1.5

January 2011

© 1993–2011 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.

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.

Patents: Cadence Product Virtuoso Layout Suite XL, described in this document, is protected by U.S. Patents

5,790,436; 5,812,431; 5,859,785; 5,949,992; 6,493,849; 6,278,964; 6,300,765; 6,304,097; 6,414,498; 6,560,755; 6,618,837; 6,693,439; 6,826,736; 6,851,097; 6,711,725; 6,832,358; 6,874,133; 6,918,102; 6,954,908; 6,957,400; 7,003,745; 7,003,749

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.

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

<u>Preface</u>	23
<u>Licensing in Layout XL</u>	23
<u>Related Documents for Layout XL</u>	24
<u>Installation, Environment, and Infrastructure</u>	24
<u>Technology Information</u>	24
<u>Virtuoso Tools</u>	24
<u>Relative Object Design and Inherited Connections</u>	25
<u>Third Party Tools</u>	25
<u>Typographic and Syntax Conventions</u>	26
1	
<u>Getting Started with Layout XL</u>	27
<u>Starting Layout XL</u>	27
<u>Starting Layout XL from a Schematic View</u>	29
<u>Starting Layout XL from a Layout View</u>	41
<u>Starting Layout XL from a Physical Configuration View</u>	43
<u>Automatic Constraint Transfer during Layout XL Startup</u>	44
<u>Specifying the Constraint Group</u>	45
<u>Layout XL Desktop</u>	46
<u>Customizing the Desktop</u>	46
<u>Layout XL Window</u>	47
<u>Menu Access Keys</u>	48
<u>Bindkeys</u>	48
<u>Canvas</u>	52
<u>Dockable Assistants</u>	52
<u>Toolbar</u>	53
<u>Workspaces</u>	54
<u>Setting Environment Variables</u>	58
<u>Setting Environment Variables in a .cdsenv or .cdsinit File</u>	59
<u>Setting Environment Variables in the CIW</u>	59
<u>Setting Environment Variables in the Layout XL Options Form</u>	60

Virtuoso Layout Suite XL User Guide

<u>Display and Selection Controls</u>	60
<u>Setting Display Colors</u>	61
<u>Editing Entry Layers</u>	61
<u>Selecting Objects in Layout XL</u>	64
<u>Deselecting Objects in Layout XL</u>	65
<u>Viewing Multiple Cellviews in Layout XL</u>	65
<u>Displaying Options Forms Automatically</u>	65
<u>Printing to the Command Interpreter Window</u>	65

2

<u>Technology File Requirements for Layout XL</u>	67
<u>Layout Editor Rules</u>	68
<u>Layer Definitions</u>	68
<u>Layer Rules</u>	68
<u>Specifying Electrical Equivalence</u>	69
<u>Specifying Via Layers</u>	69
<u>Constraint Groups</u>	69
<u>Specifying Information Required by the Layout XL Connectivity Extractor</u>	69
<u>Specifying Information Required by the Virtuoso Chip Assembly Router</u>	75
<u>Devices</u>	75
<u>Specifying Vias</u>	76
<u>Specifying Multipart Paths</u>	76
<u>Physical Rules</u>	76
<u>Connectivity Rules</u>	77
<u>Layer Functions Table</u>	77
<u>Valid Layers</u>	80
<u>Valid Vias</u>	80
<u>Via Definitions</u>	81
<u>Equivalent Layers</u>	84

3

<u>Preparing Your Connectivity Source</u>	85
<u>Schematic Design Elements</u>	85
<u>Design Variables</u>	86
<u>Design Constraints</u>	88

Virtuoso Layout Suite XL User Guide

<u>One-to-Many Mapping</u>	88
<u>Defining One-to-Many Mapping with Iterated Instances and Bus Pins</u>	88
<u>Defining Quick Cells for Use in Layout XL</u>	89
<u>Creating a New Technology Library for Use with QCells</u>	90
<u>Using an Existing Technology Library with QCells</u>	97
<u>Setting Units</u>	97
<u>Inserting a QCell Device</u>	98
<u>Optimizing the Layout</u>	99
<u>Properties in Layout XL</u>	99
<u>Pins and Pin Names in Layout XL</u>	100
<u>Extra Pins in the Symbol or Layout Views</u>	100
<u>External Connections</u>	101

4

<u>Configuring the Physical Hierarchy</u>	103
<u>LAM Files and Configuration Views</u>	103
<u>Converting Data to Use the IC 6.1 Layout XL Schema</u>	104
<u>Do I Need to Convert Data to the Layout XL IC 6.1 Schema</u>	105
<u>How Data is Converted to the Layout XL IC 6.1 Schema</u>	106
<u>Converting Schematic Library and Design Data</u>	107
<u>Converting Layout Designs</u>	110
<u>Troubleshooting the Conversion</u>	111
<u>Launching Configure Physical Hierarchy</u>	111
<u>Configure Physical Hierarchy Window</u>	113
<u>Hierarchy Configuration Mode</u>	119
<u>Hierarchy Configuration Instances Table</u>	121
<u>Hierarchy Configuration Cells Table</u>	125
<u>Mapping Logical Cellviews to Physical Cellviews</u>	126
<u>Mapping Parameter and Terminal Names</u>	129
<u>Ignoring Components</u>	130
<u>Ignoring Parameters and Terminals</u>	138
<u>Rounding Parameter Values</u>	140
<u>Splitting Mfactored Devices</u>	143
<u>Controlling CPH with Schematic Properties</u>	144
<u>Component Types Mode</u>	145

Virtuoso Layout Suite XL User Guide

<u>Component Types Cells Table</u>	147
<u>Defining a Design-Level Component Type</u>	149
<u>Modifying a Component Type</u>	151
<u>Removing a Component Type</u>	152
<u>Soft Block Mode</u>	152
<u>Soft Block Instances Table</u>	154
<u>Setting the Cell Type</u>	158
<u>Defining Soft Block Parameters</u>	161
<u>Specifying the Boundary</u>	163
<u>Specifying I/O Pins</u>	168
<u>Specifying Obstructions</u>	172
<u>Removing Soft Block Parameters</u>	175
<u>Creating a Physical Binding</u>	176
<u>Loading Soft Block Parameters from Another Cellview</u>	176
<u>Saving Soft Block Parameters to a Floorplan File</u>	178
<u>Loading Soft Block Parameters from a Floorplan File</u>	178
<u>Generating the Physical Hierarchy</u>	178

5

<u>Device Abutment</u>	181
<u>Requirements for Abutment in Layout XL</u>	183
<u>Setting Up Cells for Abutment</u>	184
<u>Creating CMOS Pcells to Use with Abutment</u>	190
<u>Abutting MOS Devices without a User-Defined Function</u>	193
<u>Adding Automatic Abutment Properties</u>	193
<u>Abutting Multiple Pins</u>	193
<u>Snapping Instances in the Direction Perpendicular to the Abutment</u>	194
<u>Additional Pins on Ignored Instances</u>	194
<u>Manual Device Abutment</u>	194
<u>Chaining Devices Interactively</u>	195
<u>Chaining Devices when Generating Selected From Source</u>	198
<u>Moving Chained Transistors</u>	202
<u>Automatic Device Abutment</u>	203
<u>Chaining Transistors Automatically during Layout Generation</u>	203
<u>Chaining Transistors Automatically when Updating Components and Nets</u>	204

Virtuoso Layout Suite XL User Guide

<u>Abutting Parameterized Cells and Quick Cells</u>	205
<u>Recognizing a QCell</u>	205
<u>Understanding QCell Parameters</u>	206
<u>Specifying Whether to Use QCell or Pcell Rules</u>	209
<u>Sample userAbutWithQCell Function</u>	210
<u>Abutment and the Virtuoso Chip Assembly Router</u>	214

6

<u>Generating a Layout</u>	215
<u>Naming Conventions</u>	215
<u>Constraint Transfer</u>	218
<u>Place and Route Boundary</u>	220
<u>Setting the Default Size for the Place and Route Boundary</u>	220
<u>Creating a Place and Route Boundary Manually</u>	220
<u>Stretching, Moving and Deleting the Place and Route Boundary</u>	221
<u>Moving Components Inside the Place and Route Boundary</u>	221
<u>Updating the Placement Status Automatically</u>	221
<u>Generating All Components from Source</u>	222
<u>Specifying the Components to be Generated</u>	225
<u>Specifying the I/O Pins to be Generated</u>	227
<u>Specifying the Place and Route Boundary</u>	235
<u>Preserving Floorplanning Objects</u>	238
<u>Generating a Layout from Source Using SKILL</u>	238
<u>Generating Selected Components from Source</u>	239
<u>Generating Components Together</u>	240
<u>Generating Individual Instances</u>	243
<u>Generating Individual Pins</u>	247
<u>Listing Unplaced Components</u>	250
<u>Generating Clones</u>	252
<u>Generating a Clone using the Generate Clones Form</u>	256
<u>Cloning a Group of Components</u>	261
<u>Cloning between Multiple Cellviews</u>	262
<u>Cloning Mfactored Components</u>	264
<u>Generating Synchronous Clones</u>	272
<u>Generating a Family of Synchronous Clones</u>	275

Virtuoso Layout Suite XL User Guide

<u>Editing a Clone in a Synchronized Family</u>	276
<u>Updating Clone Families</u>	278
<u>Generating Synchronous Copy</u>	286
<u>Generating a Synchronous Copy</u>	287
<u>Synchronous Copy with syncClone Source Group</u>	290
<u>Synchronous Copy with Other Source Groups</u>	293
<u>Synchronous Copy with Different Source Objects</u>	294
<u>Clone Non-Checked Objects</u>	295
<u>SKILL Functions</u>	296
<u>Loading Physical Information from Another Cellview</u>	297
<u>Creating a Source Cellview</u>	297
<u>Loading a Source Cellview</u>	298
<u>Modifying a Source Cellview</u>	302
<u>How Cellview Information is Handled</u>	303

7

<u>Editing the Layout</u>	311
<u>Constraint-Aware Editing</u>	312
<u>Enabling and Disabling Constraint-Aware Editing Mode</u>	312
<u>Objects, Commands, and Constraints Supported in Constraint-Aware Editing Mode</u>	313
<u>Notes on Constraint-Aware Editing</u>	314
<u>Moving Objects</u>	315
<u>Moving Objects Manually</u>	315
<u>Moving Objects Using the Move Form</u>	316
<u>Moving Generated Components into the Design Boundary</u>	318
<u>Swapping Components</u>	319
<u>Locking and Unlocking Devices</u>	320
<u>Adding Components</u>	323
<u>Adding an Instance</u>	323
<u>Assigning Instance Pins to a Net</u>	325
<u>Adding a Pin</u>	326
<u>Transistor Folding</u>	328
<u>Folding a Transistor</u>	329
<u>Ignoring Schematic Mfactor during Folding</u>	336

Virtuoso Layout Suite XL User Guide

<u>Pin Permutation</u>	341
<u>Permuting Pins Manually</u>	342
<u>Permuting Pins Automatically</u>	343
<u>Checking Permutation Information</u>	345
<u>Updating Permuted Pins</u>	345
<u>Disabling Permutation for an Instance</u>	345
<u>Working with Incomplete Nets</u>	347
<u>Showing and Hiding All Incomplete Nets</u>	350
<u>Showing and Hiding Current Incomplete Nets</u>	351
<u>Showing and Hiding Selected Incomplete Nets</u>	353
<u>Searching for Specific Incomplete Nets</u>	360
<u>Automatically Showing Newly-Created Incomplete Nets</u>	364
<u>Hiding All Incomplete Nets</u>	366
<u>Assigning Colors to Incomplete Nets</u>	367
<u>Zooming In on Incomplete Nets</u>	367
<u>Editing In Place</u>	368
<u>Editing Soft Blocks</u>	369
<u>Changing the Block Type</u>	369
<u>Changing the Boundary</u>	371
<u>Changing I/O Pin Attributes</u>	376
<u>Changing Obstructions</u>	379
<u>Enabling Level-1 Editing Mode</u>	384
<u>Moving an Object in Level-1 Editing Mode</u>	387
<u>Stretching an Object in Level-1 Editing Mode</u>	390
<u>Reshaping an Object in Level-1 Editing Mode</u>	398
<u>Chopping an Object in Level-1 Editing Mode</u>	400
<u>Finding and Replacing Nets</u>	407
<u>Discarding Edits</u>	409
<u>Saving Hierarchical Cellviews under a Different Name</u>	409
<u>Soft Block Pins</u>	410
<u>Creating a Soft Pin</u>	411
<u>Updating Parameters for a Soft Pin</u>	415
<u>Deleting a Soft Pin</u>	415
<u>Creating a Feedthrough Terminal Pin</u>	416
<u>Updating Parameters for a Feedthrough Terminal Pin</u>	420
<u>Deleting a Feedthrough Terminal Pin</u>	420

Virtuoso Layout Suite XL User Guide

<u>Snapping Soft Block Pins to Grid</u>	420
<u>Setting the Pin Connectivity Model</u>	423
<u>Creating a Strong Connect Group</u>	424
<u>Creating Separate Weak Connect Groups</u>	424
<u>Creating a Single Weak Connect Group</u>	425
<u>Creating a Must Connect Group</u>	425
<u>Deleting a Must Connect Group</u>	425

8

Preparing Your Design for Routing

<u>Connectivity Extraction</u>	428
<u>Connectivity Extraction Use Model</u>	428
<u>Connections Recognized by the Extractor</u>	430
<u>Extracting a Top-Level Design</u>	436
<u>Extracting Hierarchical Cellviews</u>	438
<u>Enabling and Disabling Incremental Extraction</u>	441
<u>Avoiding False Overlap and Short Markers</u>	441
<u>Assigning Connectivity to a Via</u>	441
<u>Connectivity Extraction from Substrate and Well Layers</u>	442
<u>Pseudoparallel Connections</u>	464
<u>Defining a Pseudoparallel Connection</u>	469
<u>Resetting a Pseudoparallel Connection</u>	470
<u>Generating a Layout without Chaining and Folding on Pseudoparallel Nets</u>	471
<u>Generating a Layout with Chaining and Folding on Pseudoparallel Nets</u>	473
<u>Generating Selected Components with Pseudoparallel Nets</u>	474
<u>Working With Nets</u>	475
<u>Selecting and Deselecting Nets</u>	475
<u>Routing a Net</u>	477
<u>Deleting Routing on a Net</u>	480
<u>Locking and Unlocking Nets</u>	481
<u>Editing Net Attributes and Properties</u>	486
<u>Creating and Editing Net Constraints</u>	488
<u>Making Net Connections</u>	493
<u>Probing Nets</u>	493
<u>Layer Selection</u>	493

Virtuoso Layout Suite XL User Guide

<u>Selecting a Layer in the Palette Assistant</u>	494
<u>Selecting a Layer Automatically when Creating a Wire</u>	496
<u>Selecting a Layer Automatically when Creating a Path or Shape</u>	496
<u>Flight Lines</u>	496
<u>Markers</u>	497
<u>Finding Markers</u>	497
<u>Explaining Markers</u>	498
<u>Deleting a Single Marker</u>	498
<u>Deleting All Markers</u>	498

9

Checking Design Data

<u>Probing</u>	502
<u>Probing Objects using the Right Mouse Button</u>	504
<u>Probing Objects by Clicking in a Window</u>	511
<u>Probing Objects using the XL Probe Form</u>	512
<u>Probing a Hierarchical Design</u>	514
<u>Removing Probes</u>	517
<u>Exiting the Probe Command</u>	517
<u>Checking Shorts, Incomplete Nets, and Illegal Overlaps</u>	517
<u>Check Against Source</u>	519
<u>Checking a Layout Against a Schematic</u>	520
<u>Suppressing the Check Against Source Form</u>	522
<u>Ignoring Components with no Schematic Counterpart</u>	523
<u>Checking Manufacturability</u>	525
<u>Match and Fix Flow</u>	525
<u>Litho Fixing Flow</u>	530
<u>Litho/LDE Analysis</u>	530

10

Updating Design Data

<u>Updating Components and Nets</u>	536
<u>Updating Selected Layout Components</u>	538
<u>Updating Nets and Instance Name Mismatches Only</u>	540
<u>Specifying the Components to be Updated</u>	542

Virtuoso Layout Suite XL User Guide

<u>Specifying the I/O Pins to be Updated</u>	545
<u>Specifying How the PR Boundary is Updated</u>	546
<u>Updating Components and Nets Using SKILL</u>	546
<u>Updating Layout Constraints</u>	547
<u>Updating Layout Parameters</u>	547
<u>Updating Layout Parameter Values in Preselection Mode</u>	552
<u>Updating Layout Parameter Values in Postselection Mode</u>	552
<u>Updating Parameter Values for Folded Devices</u>	553
<u>Excluding a Specific Parameter from the Update</u>	554
<u>Updating Schematic Parameters</u>	554
<u>Updating Schematic Parameter Values in Preselection Mode</u>	559
<u>Updating Schematic Parameter Values in Postselection Mode</u>	559
<u>Updating Device Correspondence</u>	560
<u>Defining a One-to-One Device Correspondence</u>	563
<u>Defining a One-to-Many Device Correspondence</u>	565
<u>Defining a Many-to-One Device Correspondence</u>	566
<u>Defining a Many-to-Many Device Correspondence</u>	566
<u>Assigning Layout Instance Terminals to Nets</u>	567
<u>Deleting a Device Correspondence</u>	569
<u>Define Device Correspondence SKILL API</u>	569
<u>Updating the Connectivity Reference</u>	570
<u>Updating the Schematic View</u>	570
<u>Updating the Physical Configuration View</u>	571

11

<u>Troubleshooting Layout XL</u>	573
<u>Troubleshooting Cloning</u>	575
<u>Structure Already Exists in the Layout</u>	575
<u>Connectivity Structure is Different</u>	576
<u>Master Cells are Different</u>	576
<u>More Source Components are Selected than Target Components</u>	577
<u>Properties are Different</u>	578
<u>Automatic Parameter Update Causes Different Submasters (Message LX-2149)</u>	579
<u>Troubleshooting Configure Physical Hierarchy</u>	580
<u>Schematic Properties Not Present in physConfig</u>	580

Virtuoso Layout Suite XL User Guide

<u>NLP Expressions Cannot be Converted to the New Schema</u>	580
<u>Global or Inherited Nets Beyond Physical Leaf Cells are not Detected</u>	581
<u>Troubleshooting Connectivity Extraction</u>	582
<u>Connections Not Made</u>	582
<u>Layout XL Does Not Recognize Connected Pins and Nets</u>	582
<u>Markers for Nonexistent Overlaps and Shorts Appear</u>	582
<u>Path Ends Not Accepted</u>	583
<u>Placement and Routing Do Not Run</u>	583
<u>Layout XL Does Not Recognize Physical Vias</u>	583
<u>Moving Software Executables To a New Location</u>	584
<u>Unmatched Instance Terminals are Automatically Removed</u>	584
<u>Layout XL Commands Requiring Schematic Extraction</u>	584
<u>Troubleshooting Constraint Migration</u>	584
<u>CMX Legacy Net Classes</u>	584
<u>Troubleshooting Editing</u>	585
<u>Components Move Slowly</u>	585
<u>Extra Probes Appear</u>	585
<u>Generate Layout Form Does Not Keep Values from the Last Entry</u>	585
<u>Parameters Not Updated</u>	585
<u>Schematic Not Editable</u>	585
<u>Warning to Update Your Design Appears at Startup</u>	586
<u>Troubleshooting Generating Selected Components From Source</u>	586
<u>Hierarchical Designs</u>	586
<u>Global Terminals and Implicit Inherited Connection Terminals</u>	587
<u>Troubleshooting Parameter Evaluation</u>	587
<u>Netlisting Mode</u>	587
<u>Evaluating CDF Callbacks by Default</u>	587
<u>Troubleshooting Performance</u>	588
<u>Design Fails to Open or Takes a Long Time to Open</u>	588
<u>Layout XL Is Slow</u>	588
<u>Troubleshooting Startup</u>	589
<u>Invalid Markers from Previous Software Versions</u>	589
<u>Options Form Does Not Appear</u>	590

A

<u>Layout XL Environment Variables</u>	591
<u>List of Layout XL Environment Variables</u>	592
<u>autoAbutment</u>	598
<u>autoArrange</u>	599
<u>autoMirror</u>	601
<u>autoMirrorChains</u>	603
<u>autoSpace</u>	606
<u>autoZoomIsFixed</u>	608
<u>bndLargeBinderNetThreshold</u>	609
<u>capacitanceParamNames</u>	610
<u>chainDummyFlexBothEndNets</u>	612
<u>chainExtendSelection</u>	613
<u>chainFolds</u>	614
<u>chainPreserveExistingChains</u>	615
<u>chainUseDeviceOrder</u>	616
<u>checkMissingParamsOrProps</u>	617
<u>checkParamsOnly</u>	618
<u>checkStateConfirmModeChange</u>	620
<u>ciwWindow</u>	621
<u>cloningDoExactMatch</u>	622
<u>cloningAutomaticUpdateLayoutParameters</u>	623
<u>colorDraglines</u>	624
<u>constraintAwareEditing</u>	626
<u>cphPromptConfigOrSchematic</u>	628
<u>cphStopLogicalElabAtPhysLeaf</u>	629
<u>createBoundaryLabel</u>	631
<u>createImplicitBusTerminals</u>	632
<u>crossSelect</u>	634
<u>deleteConfirmModeChange</u>	635
<u>deleteUnmatchedInsts</u>	636
<u>deleteUnmatchedPins</u>	637
<u>deviceExtractType</u>	638
<u>disableCASOptionsPopUp</u>	640
<u>distanceMFactorExpansion</u>	641

Virtuoso Layout Suite XL User Guide

<u>drdUseNetName</u>	644
<u>extractShowMustConnectMarkers</u>	646
<u>extractShowUnimplementedInstTerms</u>	647
<u>extractShowWeakMarkers</u>	648
<u>flightLineEnable</u>	649
<u>foldOptimizeSupplyAndGroundNets</u>	651
<u>hideDraglinesForGlobalNets</u>	652
<u>hideMarkersWhenBrowserHidden</u>	654
<u>hierarchyDepth</u>	655
<u>highlightedIsSelectable</u>	656
<u>incNetHiliteLayer</u>	657
<u>inductanceParamNames</u>	658
<u>infoWindow</u>	660
<u>infoWindowPos</u>	661
<u>initAspectRatio</u>	662
<u>initAspectRatioOption</u>	663
<u>initCreateBoundary</u>	664
<u>initCreateInstances</u>	665
<u>initCreateMTM</u>	666
<u>initCreatePadPins</u>	668
<u>initCreatePins</u>	669
<u>initCreateSnapBoundary</u>	670
<u>initDoFolding</u>	671
<u>initDoStacking</u>	672
<u>initEstimateArea</u>	673
<u>initGlobalNetPins</u>	674
<u>initIOLabelType</u>	675
<u>initIOPinLayer</u>	677
<u>initIOPinName</u>	679
<u>initPinHeight</u>	680
<u>initPinMultiplicity</u>	682
<u>initPinWidth</u>	683
<u>initPrBoundaryH</u>	685
<u>initPrBoundaryW</u>	686
<u>initUtilization</u>	687
<u>labelOrient</u>	688

Virtuoso Layout Suite XL User Guide

<u>layoutWindow</u>	690
<u>lswWindow</u>	691
<u>lxAllowPseudoParallelNets</u>	692
<u>lxChainAlignNMOS</u>	694
<u>lxChainAlignPMOS</u>	696
<u>lxDeltaWidth</u>	698
<u>lxEvalCDFCallbacks</u>	699
<u>lxFingeringNames</u>	701
<u>lxGenerationOrientation</u>	703
<u>lxGenerationTopLevelOnly</u>	705
<u>lxGenerationTopLevelOnly</u>	706
<u>lxGenerationTopLevelOnly</u>	707
<u>lxGenerateInBoundary</u>	709
<u>lxGenerationTopLevelOnly</u>	710
<u>lxGetSignifDigits</u>	711
<u>lxGroundNetNames</u>	713
<u>lxInitResetSource</u>	714
<u>lxLocalAbutment</u>	715
<u>lxPositionMinSep</u>	717
<u>lxPositionPinsOnBoundary</u>	718
<u>lxRetainFoldOrient</u>	719
<u>lxSchematicDefaultApp</u>	720
<u>lxSchExtractTopLevelOnly</u>	721
<u>lxStackMinimalFolding</u>	722
<u>lxStackPartitionParameters</u>	723
<u>lxSupplyNetNames</u>	725
<u>lxUpdateFoldedWidth</u>	726
<u>lxUseLibList</u>	728
<u>lxValidateXLParameterEvaluation</u>	730
<u>lxWidthTolerance</u>	732
<u>mfactorNames</u>	733
<u>mfactorSplit</u>	734
<u>moveAsGroup</u>	735
<u>openConnRef</u>	736
<u>openConnRefTab</u>	737
<u>openLocation</u>	738

Virtuoso Layout Suite XL User Guide

<u>openMode</u>	739
<u>paramsToIgnore</u>	740
<u>paramsToIgnoreForCheck</u>	742
<u>paramTolerance</u>	743
<u>pinTextSamePurpose</u>	745
<u>prefixLayoutInstNamesWithPipe</u>	746
<u>preserveAreaBoundary</u>	747
<u>preserveBlockages</u>	748
<u>preserveClusters</u>	749
<u>preserveClusterBoundaries</u>	750
<u>preserveRows</u>	751
<u>preserveTrackPattern</u>	752
<u>probeCycleHilite</u>	753
<u>probeDevice</u>	754
<u>probeDuringCreate</u>	755
<u>probeHiliteLayer</u>	756
<u>probeInfoInCIW</u>	757
<u>probeNet</u>	758
<u>probePin</u>	759
<u>propsUsedToIgnoreObjs</u>	760
<u>propsUsedToIgnoreObjsForCheck</u>	762
<u>resistanceParamNames</u>	763
<u>schematicWindow</u>	765
<u>scopeLevel</u>	766
<u>setPPConn</u>	768
<u>sfactorNames</u>	769
<u>showDraglinesForDistantConns</u>	771
<u>snapCpaToPlacementGrid</u>	773
<u>stopList</u>	774
<u>suppressExpansion</u>	776
<u>transistorWidthParamNames</u>	778
<u>turnInfixOffWhenSmartSnapping</u>	780
<u>updateLayoutParameters</u>	781
<u>updateNetSigType</u>	782
<u>updateNetsOnly</u>	783
<u>updateOneToOneMappings</u>	784

Virtuoso Layout Suite XL User Guide

<u>updatePlacementStatus</u>	785
<u>updatePlacementStatusInBoundary</u>	786
<u>updateReplacesMasters</u>	787
<u>updateSelectedComponents</u>	789
<u>updateWithMarkers</u>	790
<u>viewList</u>	791
<u>GUI Options and Corresponding Environment Variables</u>	793
<u>Hidden Environment Variables with No Corresponding GUI Option</u>	800

B

<u>Command Quick Reference</u>	803

C

<u>Layout XL Properties</u>	809
<u>Setting Layout XL Properties</u>	811
<u>abutAccessDir</u>	813
<u>abutClass</u>	814
<u>abutCondInclusion</u>	816
<u>abutFunction</u>	817
<u>abutGateNet</u>	820
<u>abutMosStretchMat</u>	821
<u>abutStretchMat</u>	825
<u>abutOffset</u>	827
<u>ignore</u>	829
<u>lvslgnore</u>	830
<u>lxAutoAbut</u>	832
<u>lxAutoSpace</u>	833
<u>lxBlockExtractCheck</u>	834
<u>lxBlockOverlapCheck</u>	836
<u>lxCombination</u>	838
<u>lxMfactorSplit</u>	841
<u>lxNetNamePrefix</u>	842
<u>lxParamsToIgnore</u>	845
<u>lxParamsToIgnoreForCheck</u>	847
<u>lxRemoveDevice</u>	848

Virtuoso Layout Suite XL User Guide

<u>lxRounding</u>	849
<u>lxSeriesTerms</u>	850
<u>lxStickyNet</u>	851
<u>lxStopList</u>	853
<u>lxUseCell</u>	854
<u>lxViewList</u>	855
<u>mfactor</u>	856
<u>permuteRule</u>	859
<u>sfactor</u>	862
<u>vxInstSpacingDir</u>	865
<u>vxInstSpacingRule</u>	868

D

<u>Layout XL Forms</u>	871
<u>Add A New Pin</u>	874
<u>Add Soft Pin</u>	875
<u>Assign Layout Instance Terminals</u>	876
<u>Assign Nets</u>	877
<u>Block Parameters using Physical View</u>	878
<u>Check Against Source</u>	879
<u>Chop</u>	880
<u>Configure Physical Hierarchy</u>	881
<u>Convert Libraries to Use Physical Configuration Views</u>	904
<u>Copy</u>	905
<u>Create Cluster Boundary</u>	906
<u>Create Custom Placement Area</u>	909
<u>Create Feed Through Terminal Block Pins</u>	910
<u>Create Instance</u>	912
<u>Create Path</u>	913
<u>Create Physical Configuration View</u>	915
<u>Create Polygon</u>	916
<u>Create Rectangle</u>	917
<u>Define Device Correspondence</u>	918
<u>Design Summary</u>	921
<u>Device List</u>	922

Virtuoso Layout Suite XL User Guide

<u>Edit Net Attributes</u>	923
<u>Edit Soft Blocks</u>	924
<u>Extract Layout</u>	932
<u>Generate Chained Devices</u>	934
<u>Generate Clones</u>	935
<u>Generate Folded Devices</u>	938
<u>Generate Layout</u>	941
<u>Generate Selected Components</u>	947
<u>Generate Selected Components - Pin Options</u>	950
<u>Instance/Pin List</u>	951
<u>Layout XL Options</u>	952
<u>LDE Analysis</u>	974
<u>Litho Fixing</u>	976
<u>Load Physical View</u>	982
<u>Match And Fix</u>	992
<u>Modify Correspondence</u>	999
<u>Move</u>	1000
<u>Move Cells</u>	1003
<u>Open Physical Configuration View</u>	1004
<u>Permutation Information</u>	1005
<u>Repeat Copy</u>	1006
<u>Reshape</u>	1007
<u>Selection Options</u>	1009
<u>Set Net Order</u>	1014
<u>Set Pin Label Text Style</u>	1015
<u>Snap Pins</u>	1017
<u>Soft Block Global Options</u>	1018
<u>Startup Option</u>	1020
<u>Stretch</u>	1022
<u>Update Clone Families</u>	1025
<u>Update Components and Nets</u>	1029
<u>Update Connectivity Reference</u>	1036
<u>XL Probe</u>	1038

E

<u>Library and Attributes Mapping File Syntax</u>	1041
<u>LAM File Locations</u>	1042
<u>Inserting Comments in a LAM File</u>	1043
<u>Basic LAM File Structure</u>	1044
<u>LAM File Logical Section</u>	1046
<u>LAM File Physical Section</u>	1051
<u>LAM File Component Type Section</u>	1052
<u>LAM File Mappings Section</u>	1057
<u>Sample LAM File</u>	1064

F

<u>Layout XL Assistants</u>	1067
<u>Annotation Browser</u>	1069
<u>Annotation Browser Graphical User Interface</u>	1070
<u>Annotation Browser Context Menus</u>	1083
<u>Annotation Browser Forms</u>	1086
<u>Grouping Markers in the Annotation Browser</u>	1090
<u>Sorting Markers in the Annotation Browser</u>	1096
<u>Filtering Markers in the Annotation Browser</u>	1098
<u>Assigning Colors to Markers</u>	1107
<u>Zooming In on Markers</u>	1108
<u>Constraint Manager</u>	1110
<u>Updating Constraints From Schematic to Layout</u>	1112
<u>Comparing Constraints Between Schematic and Layout</u>	1113
<u>Overriding Process Rules</u>	1115
<u>Creating a Cluster Boundary Using the Constraint Manager</u>	1115
<u>Navigator</u>	1119
<u>Viewing Pin, Cell, and Signal Types in the Navigator Assistant</u>	1120
<u>Customizing the Navigator Filters</u>	1122
<u>Viewing Ungenerated Instances</u>	1126
<u>Viewing XL Status</u>	1129
<u>Property Editor</u>	1136
<u>Search</u>	1137

Virtuoso Layout Suite XL User Guide

<u>World View</u>	1138

Preface

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

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

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

- The Virtuoso design environment and application infrastructure mechanisms supporting consistent operations between all Cadence tools
- The applications for designing and developing integrated circuits in the Virtuoso design environment, notably the Virtuoso Layout Suite L layout editor
- Virtuoso technology data
- Component description format (CDF), which lets you create and describe your own components for use with Layout XL
- The design and use of Quick Cells and other types of parameterized cells
- The procedures used to translate designs for routing with the Virtuoso Chip Assembly Router

Licensing in Layout XL

For information on licensing in the Virtuoso design environment, see the [Virtuoso Software Licensing and Configuration Guide](#).

Related Documents for Layout XL

This document does not contain information on all the functions and commands enabled in Layout XL.

Commands that are implemented in a lower tier of the Virtuoso Layout Suite are covered in the documentation for the relevant application. For example, the *Virtuoso Layout Suite L User Guide* describes the basic *Move* command; the current document describes only its extended functionality.

Many of the advanced features available in Layout XL are described in dedicated user guides. Where this is the case, you will find specific references to the documents that contain the most detailed information.

The following documents contain more information on topics of interest to the Layout XL user.

Installation, Environment, and Infrastructure

- For information on installing Cadence products, see the *Cadence Installation Guide*.
- For information on the Virtuoso design environment, see the *Virtuoso Design Environment User Guide*.
- For information on database SKILL functions, including data access functions, see the *Virtuoso Design Environment SKILL Reference*.
- For information on library structure, the library definitions file, and name mapping for data shared by multiple Cadence tools, see the *Cadence Application Infrastructure User Guide*.

Technology Information

- For information on how to create and maintain a technology file and display a resource file, see the *Virtuoso Technology Data User Guide* and the *Virtuoso Technology Data ASCII Files Reference*.
- For information on how to access the technology file using SKILL functions, see the *Virtuoso Technology Data SKILL Reference*.

Virtuoso Tools

- For information on how to perform design tasks with the Virtuoso Layout Suite L layout editor, see the *Virtuoso Layout Suite L User Guide*

Virtuoso Layout Suite XL User Guide

Preface

- For information on design-rule-driven editing, see the [*Virtuoso Design Rule Driven Editing User Guide*](#).
- For information on a collection of general layout tools designed to streamline command sequences and solve specific issues common to custom mask designers, see the [*Virtuoso Toolbox User Guide*](#).
- For information on how to use the Virtuoso Layout Suite wire editing capability, see “Interactive Wire Editing” in the [*Virtuoso Space-based Router User Guide*](#).
- For information on how to use the automatic custom digital placer to place your design components, see the [*Virtuoso Custom Digital Placer User Guide*](#).
- For information on creating parameterized cells using the graphical user interface or low-level SKILL functions, see the [*Virtuoso Parameterized Cell Reference*](#).
- For information on Quick Cells, see the [*Virtuoso Quick Cells User Guide*](#).
- For information on Component Description Format, see the [*Component Description Format User Guide*](#).
- For information on how to route a design, see the [*Virtuoso Chip Assembly Router Guide*](#) and the [*Virtuoso Space-based Router User Guide*](#).
- For information on how to stream mask data, see the [*Design Data Translator’s Reference*](#).
- For information on custom layout SKILL functions, see the [*Virtuoso Layout Suite SKILL Reference*](#).

Relative Object Design and Inherited Connections

- For information on using relative object design (ROD) functions, see the [*Virtuoso Relative Object Design User Guide*](#).
- For information on connectivity and naming conventions for inherited connections, and how to add and edit net expressions in a schematic or symbol cellview, see [*Virtuoso Schematic Editor L User Guide*](#).

Third Party Tools

To view the videos included in this documentation, you need

- A Cadence Online Support login.

- A Flash-enabled web browser, for example, Internet Explorer 5.0 or later, Netscape 6.0 or later, or Mozilla Firefox 1.6 or later. Alternatively, you can download Flash Player (version 6.0 or later) directly from the [Adobe](#) website.
- Speakers and a sound card for your computer (for videos with narration).

Typographic and Syntax Conventions

The table below lists the typographic and syntax conventions used in this document.

<code>text</code>	Indicates text you must type exactly as it is presented.
<code><i>z_argument</i></code>	Indicates text that you must replace with an appropriate argument. The prefix (in this case, <code>z_</code>) indicates the data type the argument can accept. Do not type the data type or underscore.
<code>[]</code>	Denotes an optional argument. When used with vertical bars, they enclose a list of choices from which you can choose one.
<code>{ }</code>	Used with vertical bars, they denote a list of choices from which you must choose one.
<code> </code>	Separates a choice of options.
<code>...</code>	Indicates that you can repeat the previous argument.
<code>=></code>	Precedes the values returned by a Cadence [®] SKILL language function.
<code>/</code>	Separates the possible values that can be returned by a Cadence SKILL language function.
<code><i>text</i></code>	Indicates names of manuals, menu commands, form buttons, and form fields.

Getting Started with Layout XL

This section tells you how to start the Virtuoso® Layout Suite XL layout editor (Layout XL) and introduces the main features of the Layout XL desktop (i.e., the four windows that make up a Layout XL session) and the Layout XL window (i.e., the layout window that lets you view and change your layout views).

It also explains how to set up Layout XL to suit your preferences and work habits. You can use environment variables to change the value of many aspects of your environment either for an individual design session or permanently until you change the value of the variable again.

Subsequent chapters in this guide provide information on the following topics.

- [Technology File Requirements for Layout XL](#) on page 67
- [Preparing Your Connectivity Source](#) on page 85
- [Configuring the Physical Hierarchy](#) on page 103
- [Device Abutment](#) on page 181
- [Generating a Layout](#) on page 215
- [Editing the Layout](#) on page 311
- [Preparing Your Design for Routing](#) on page 427
- [Checking Design Data](#) on page 501
- [Updating Design Data](#) on page 535
- [Troubleshooting Layout XL](#) on page 573

For information on Layout XL SKILL functions, see [Layout XL SKILL Functions](#) in the *Virtuoso Layout Suite SKILL Reference*.

Starting Layout XL

To use Layout XL, you must have access to one of the following design environments:

Virtuoso Layout Suite XL User Guide

Getting Started with Layout XL

- Virtuoso Physical Design Environment (`layoutPlus`)
- Virtuoso Front to Back Design Environment (`virtuoso`)

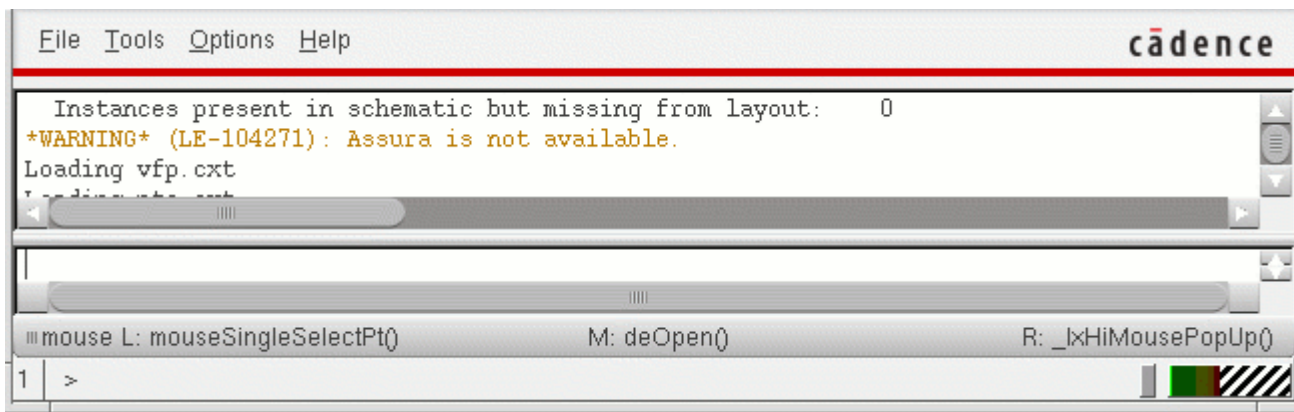
To start a design environment that includes the Layout XL software:

1. In a terminal window, move into the top-level directory of the design you want to edit.

The top-level directory is the directory that contains the library definitions (`cds.lib`) file for the design in question. The library definitions file defines the physical locations of the reference and design libraries.

2. Type in the executable name of the design environment you want to launch; for example, `virtuoso`.

The Command Interpreter Window (CIW) is displayed.



You use the CIW to launch applications on specific cellviews; perform general, non-tool-specific operations; enter SKILL commands; and display messages and warnings.

After you have started the design environment, you can then launch Layout XL from

- A schematic view, with or without a physical configuration view. See [Starting Layout XL from a Schematic View](#).
- A layout view, with or without a connectivity reference. See [Starting Layout XL from a Layout View](#).
- A physical configuration view. See [Starting Layout XL from a Physical Configuration View](#).

Starting Layout XL from a Schematic View

A schematic cellview is the only valid connectivity reference for a Layout XL session. If you do not have a schematic view of your design, you first need to generate one. For more information, see [Generating a Schematic View from a CDL Netlist](#).

When you start Layout XL from a schematic, you need to specify

- Whether to create a new layout view or use an existing one.
- Whether the layout view is opened in the context of a *new* physical configuration view, an *existing* physical configuration view, or an *automatic* physical configuration view.

Note: If you do not need to use any of the features provided by the *Configure Physical Hierarchy* command, you can choose Automatic mode. This ensures that Layout XL functions the same way as in previous releases, handling all operations related to the physical configuration view automatically.

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

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

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



Tip

For more information on physical configuration views, see [Configuring the Physical Hierarchy](#)

For more information on starting Layout XL from a schematic view, see

- [Starting Layout XL with an Automatic Physical Configuration View](#)
- [Starting Layout XL with a New Physical Configuration View](#)
- [Starting Layout XL with an Existing Physical Configuration View](#)

Generating a Schematic View from a CDL Netlist

If your connectivity reference is a CDL netlist, you cannot import it directly into Layout XL. You must first generate a schematic from the netlist.

To generate a schematic view from a CDL netlist,

1. From the CIW, select *File – Import – Spice*.
2. Set the *Netlist Language* to *CDL* and type the *Netlist File* name in the field provided.
3. Click *OK*.
4. Start Layout XL using one of the methods described.

For more information on importing CDL, see [Netlist import Using Spice In](#) in the *Design Data Translator's Reference*.

Starting Layout XL with an Automatic Physical Configuration View

If you do not need to use any of the features provided by the Configure Physical Hierarchy command, you should launch Layout XL in Automatic mode. This ensures that Layout XL handles all operations related to the physical configuration view automatically.

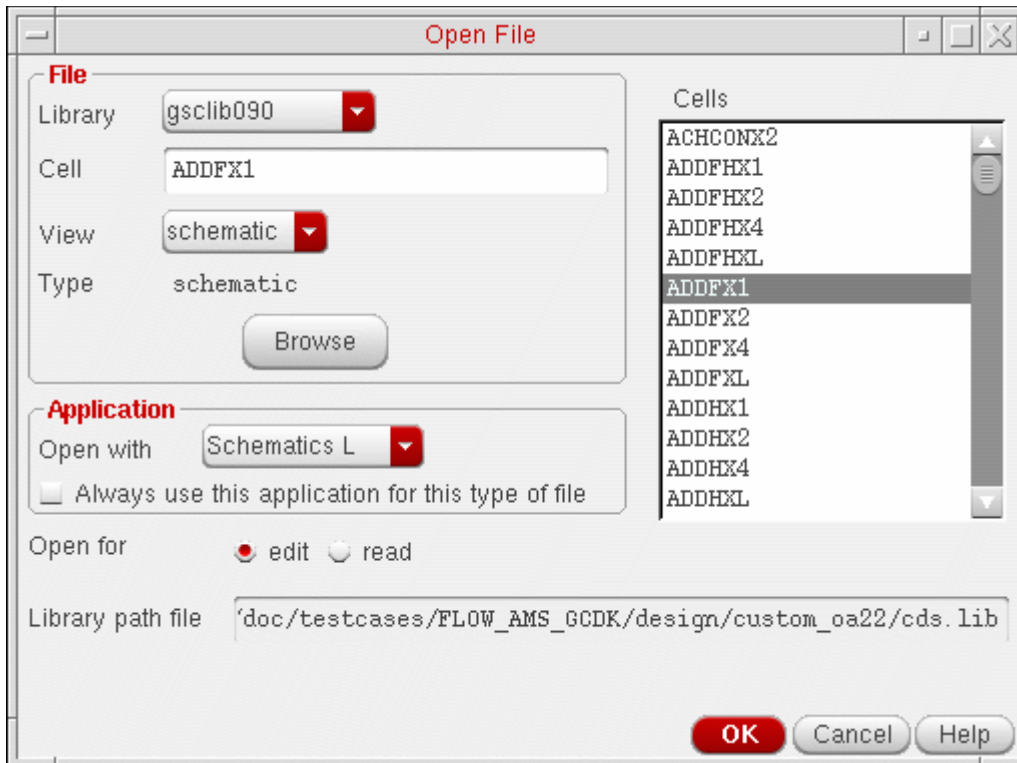
To start Layout XL from a schematic view in Automatic mode,

1. From the CIW, choose *File – Open*.

Virtuoso Layout Suite XL User Guide

Getting Started with Layout XL

The Open File form is displayed.



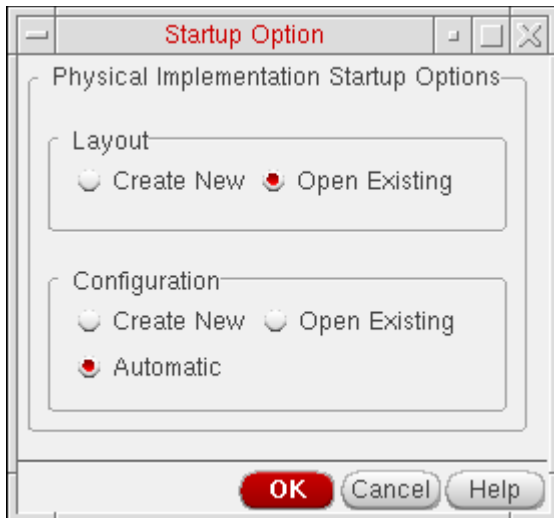
2. Specify the *Library* and *Cell* names, choose *schematic* from the *View* cyclic field and click *OK*.

The Virtuoso Schematic Editor L Editing window appears displaying the cellview you specified.

3. From the schematic window, choose *Launch – Layout XL*.

Note: *Layout XL* appears in the *Launch* menu only if the Layout XL software is installed. If you do not see *Layout XL* in the menu, check with your system administrator.

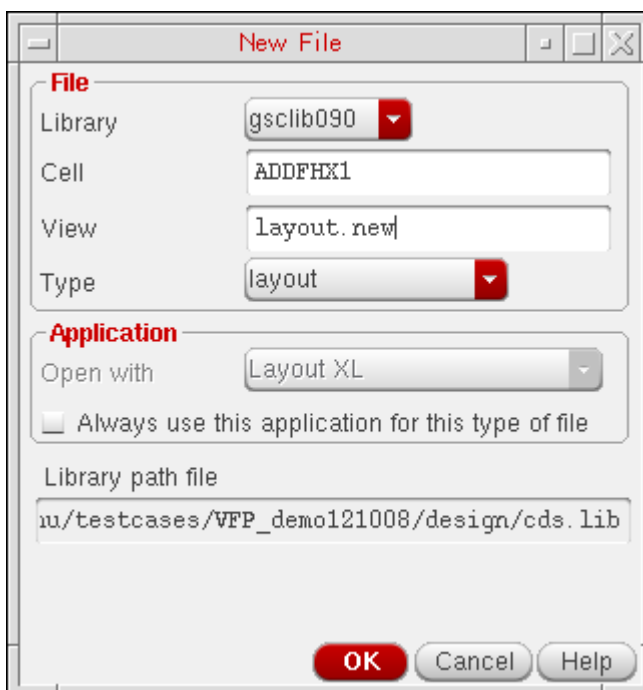
The Startup Option form is displayed.



4. In the *Layout* group box, choose whether you want to create a new layout cellview or open an existing one.
5. 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.



Virtuoso Layout Suite XL User Guide

Getting Started with Layout XL

The default view name is `layout`.

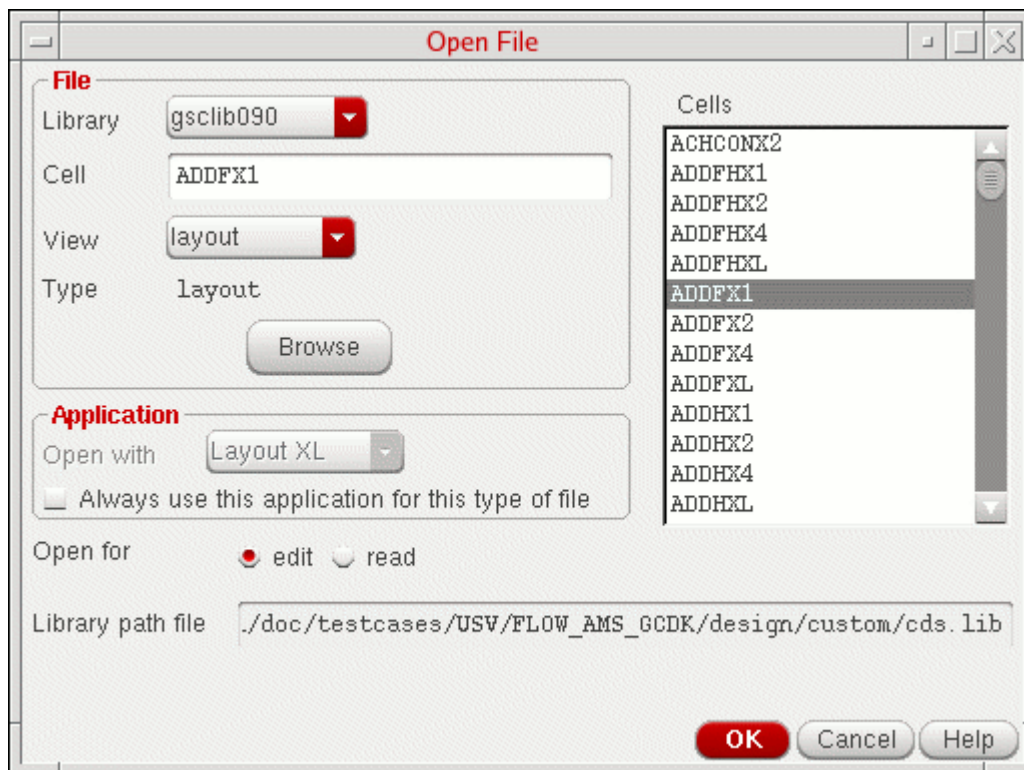
Choose a *Library* name from the cyclic field and type the *Cell* and *View* names in the fields provided.

Click *OK* to create the new cellview.

The software automatically creates the new layout cellview and looks for an existing physical configuration view called either `physConfig` or `physConfig_schName`, which might have been created in a previous Layout XL session or when Converting Data to Use the IC 6.1 Layout XL Schema.

If neither view exists, it creates a new one called `physConfig` or `physConfig_schName`.

- ❑ If you chose to open an existing layout view, the Open File form is displayed again.



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

Note: You cannot change setting for the *Open with* option in the *Application* group box, regardless of the value set for the `maskLayoutDefaultApp` environment variable.

Click *OK* in the form to open the cellview.

Virtuoso Layout Suite XL User Guide

Getting Started with Layout XL

If there is an existing physical configuration view associated with the layout cellview, the software opens it in the background. If there is no existing physical configuration, the software creates a default one called `physConfig` or `physConfig_schName`.

The default Layout XL desktop configuration appears: the schematic window, the layout window, the CIW, and the Layer Selection Window (LSW). For more information, see [Layout XL Desktop](#).

The schematic view is re-opened in the context of the automatic physical configuration view, which might be different from the configuration it was using previously. The schematic window banner updates to indicate the name of the automatic physical configuration view.

Note: One effect of this is that you might need to re-extract the schematic view in order to take into account the settings in the physical configuration view. The system will inform you if this extraction is required. To avoid the need for re-extracting the schematic view, and to have the tool automatically copy the physconfig settings from the previous view, you must copy the cellview using the Library Manager. For more information, see [Copying a View](#).

The Configure Physical Hierarchy window is not displayed in Automatic mode. However, you can invoke it at any time by choosing *Launch – Configure Physical Hierarchy* from the layout window menu bar.

Note: Even in Automatic mode, you still need to use the Configure Physical Hierarchy window if you want to change the switch, stop, and constraint view lists, or make changes to component type for soft block definitions. For more information, see [Configuring the Physical Hierarchy](#)



Tip

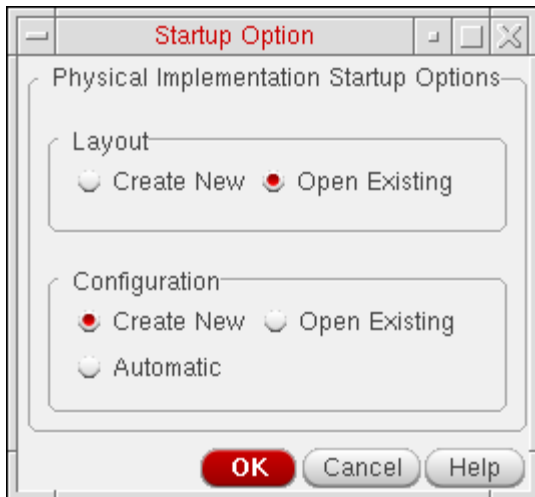
If Layout XL does not initialize and instead you see the Layout XL Constraint Group Setting dialog, see [Specifying the Constraint Group](#).

Starting Layout XL with a New Physical Configuration View

To start Layout XL from a schematic view in the context of a new physical configuration view,

1. Complete steps 1 through 4 described in [Starting Layout XL with an Automatic Physical Configuration View](#).

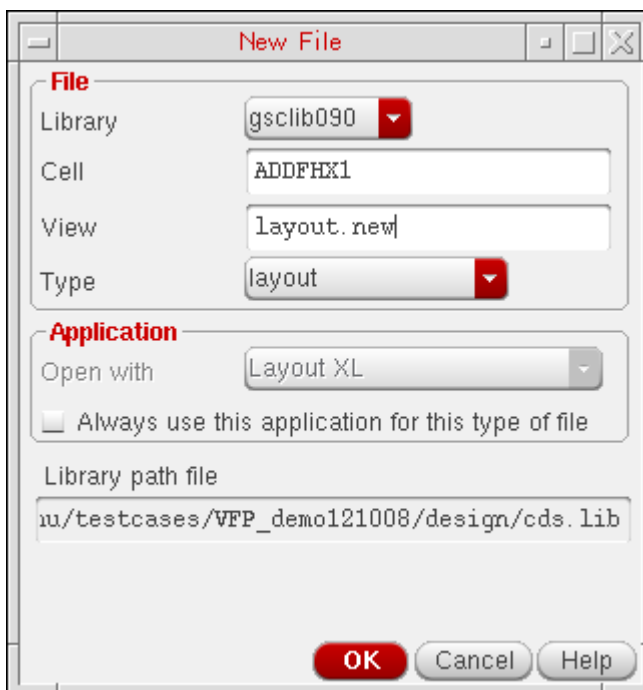
The Startup Option form is displayed.



2. Choose whether you want to create a new layout cellview or open an existing one, make sure that the *Configuration* option is set to *Create New*, and then click *OK*.

One of the following happens.

- ☐ If you chose to create a new layout cellview, the New File form is displayed.

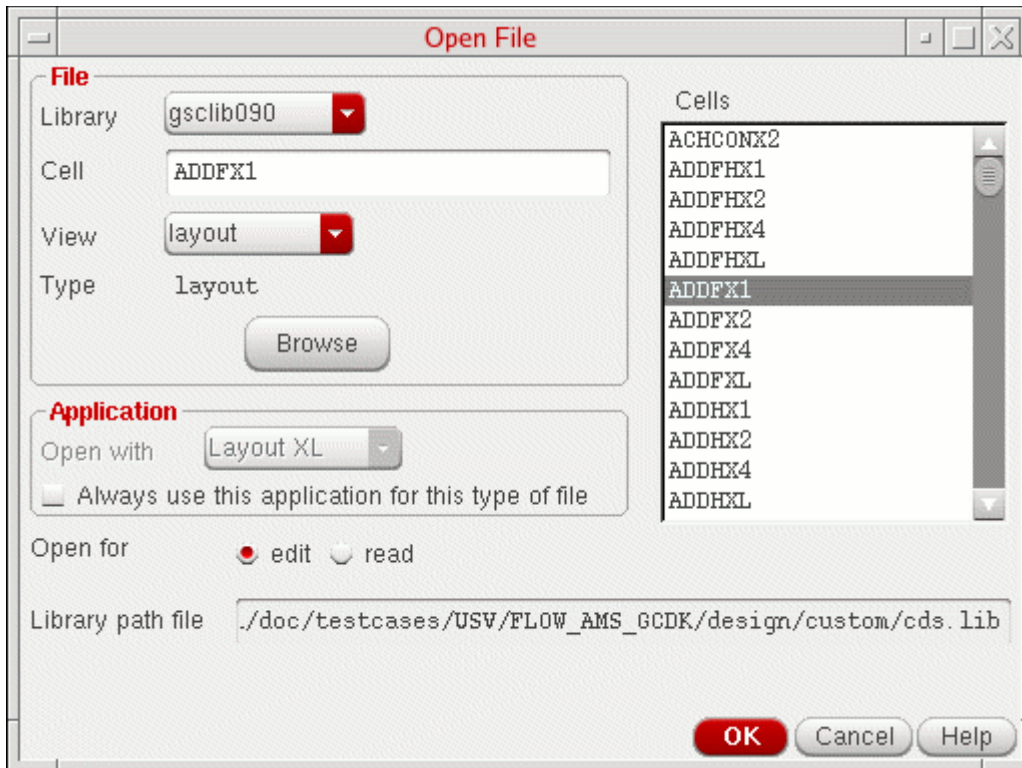


Choose a *Library* from the cyclic field and type the *Cell* and *View* names in the fields provided.

Virtuoso Layout Suite XL User Guide

Getting Started with Layout XL

- ❑ If you chose to open an existing layout view, the Open File form is displayed again.

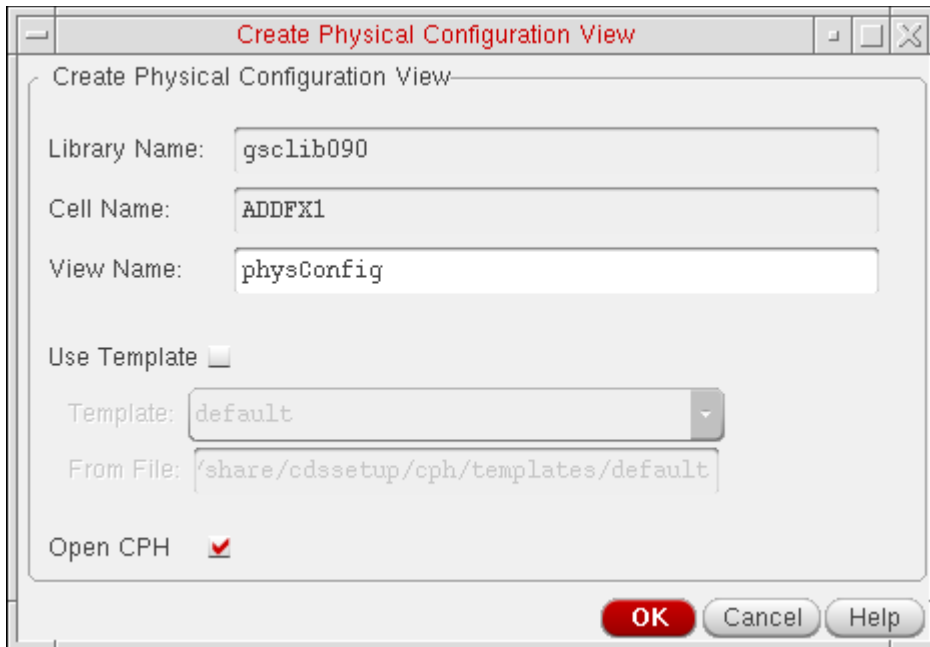


Specify the *Library* and *Cell* names and choose the *View* you want to open from the cyclic field.

Note: You cannot change setting for the *Open with* option in the *Application* group box, regardless of the value set for the `maskLayoutDefaultApp` environment variable.

3. Click *OK* to create the new layout cellview or to open an existing one.

The Create Physical Configuration View form is displayed.



The layout *Library* and *Cell* names are seeded automatically.

4. Specify the *View Name* for the physical configuration view.

To seed the new view with predefined values, check *Use Template* and either choose one of the predefined templates from the drop-down list or specify a file stored in another location. If you do not specify a template, the corresponding environment variables are used to obtain initial values.

Check the *Open CPH* box to open the Configure Physical Hierarchy window automatically.

5. Click *OK*.

The default Layout XL Desktop configuration appears: the schematic window, the layout window, the CIW, and the LSW.

The schematic view is re-opened in the context of the new physical configuration view, which might be different from the configuration it was using previously. The schematic window banner updates to indicate the name of the new physical configuration view. The Configure Physical Hierarchy window opens to display the physical configuration view you created.

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



Tip

If Layout XL does not initialize and instead you see the Layout XL Constraint Group Setting dialog, see [Specifying the Constraint Group](#).

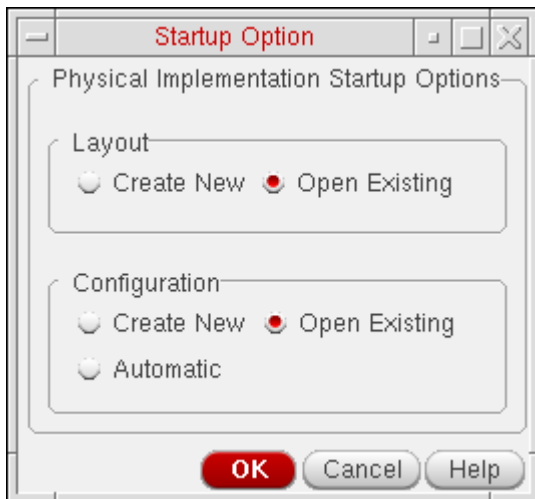
Starting Layout XL with an Existing Physical Configuration View

To start Layout XL from a schematic view in the context of an existing configuration view,

1. Complete steps 1 through 4 described in [Starting Layout XL with an Automatic Physical Configuration View](#).

The [Startup Option](#) form is displayed.

2. Choose whether you want to create a new layout cellview or open an existing one and make sure that the *Configuration* option is set to *Open Existing*.



3. Click *OK*.

One of the following happens.

Virtuoso Layout Suite XL User Guide

Getting Started with Layout XL

- ❑ If you chose to create a new layout cellview, the New File form is displayed.

New File

File

Library: gsclib090

Cell: ADDFHX1

View: layout.new

Type: layout

Application

Open with: Layout XL

☐ Always use this application for this type of file

Library path file: u/testcases/VFP_demo121008/design/cds.lib

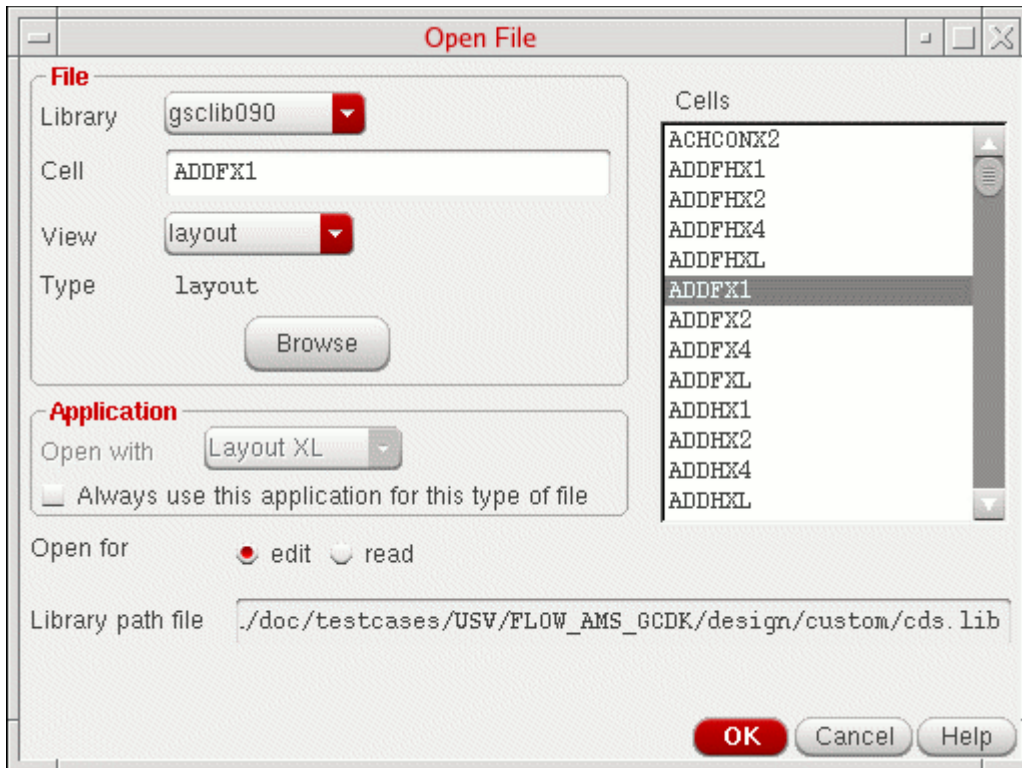
OK Cancel Help

Choose a *Library* from the cyclic field and type the *Cell* and *View* names in the fields provided.

Virtuoso Layout Suite XL User Guide

Getting Started with Layout XL

- ❑ If you chose to open an existing layout view, the Open File form is displayed again.

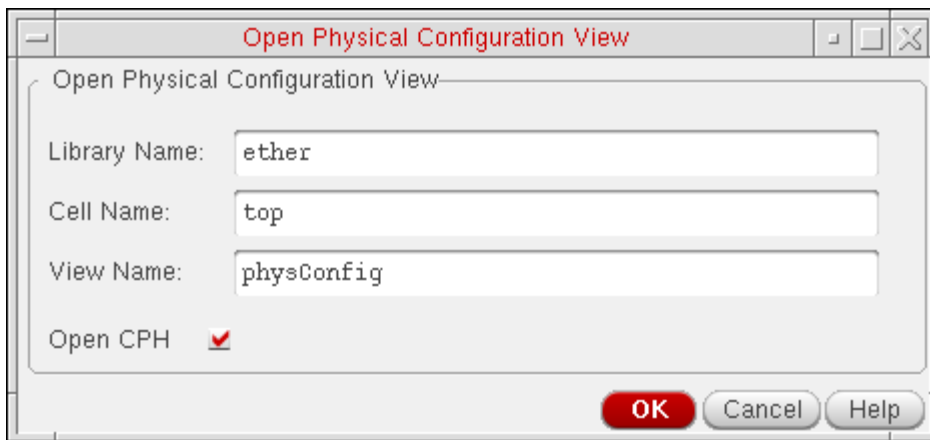


Specify the *Library* and *Cell* names and choose the *View* you want to open from the cyclic field.

Note: You cannot change setting for the *Open with* option in the *Application* group box, regardless of the value set for the `maskLayoutDefaultApp` environment variable.

4. Click *OK* to create the new layout cellview or to open an existing one.

The Open Physical Configuration View form is displayed.

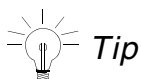


5. Specify the *Library*, *Cell*, and *View* names and check the *Open CPH* box to open the Configure Physical Hierarchy window automatically.
6. Click *OK*.

The default Layout XL desktop configuration appears: the schematic window, the layout window, the CIW, and the LSW.

The schematic view is re-opened in the context of the specified physical configuration view, which might be different from the configuration it was using previously. The schematic window banner updates to indicate the name of the specified physical configuration view and the Configure Physical Hierarchy window opens.

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



If Layout XL does not initialize and instead you see the Layout XL Constraint Group Setting dialog, see [Specifying the Constraint Group](#).

Starting Layout XL from a Layout View

To start Layout XL from a layout view,

1. From the CIW, choose *File – Open*.

The Open File form is displayed.

Virtuoso Layout Suite XL User Guide

Getting Started with Layout XL

2. Specify the *Library* and *Cell* names and choose *schematic* from the *View* cyclic field.
3. Click *OK*.

The Virtuoso Layout Suite L Editing window appears displaying the cellview you specified.

4. From the layout window, choose *Launch – Layout XL*.

The default Layout XL Desktop configuration appears: the schematic window, the layout window, the CIW, and the LSW. The schematic view is opened in the context of the physical configuration view being used for the session, which might be different from the configuration it uses when it is opened standalone.

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



Tip

By default, the schematic editor launches in the XL mode. To launch the editor in the L mode instead, select the *Schematics L* option for the *Open in* field on the Layout XL Options form.

The corresponding environment variable is lxSchematicDefaultApp.

Notes

- *Layout XL* appears in the *Launch* menu only if the Layout XL software is installed. If you do not see *Layout XL*, check with your system administrator.
- If Layout XL does not initialize and instead you see the Layout XL Constraint Group Setting dialog, see Specifying the Constraint Group.

Starting Layout XL with no Connectivity Reference

If the layout you opened has no schematic associated with it, the Update Connectivity Reference form is displayed when you start Layout XL. If you see the Update Connectivity Reference form, you can do either of the following:

- Use Layout XL without a connectivity reference
- Specify a source to use as the connectivity reference for the layout

Using Layout XL with no Connectivity Reference

To use Layout XL with no connectivity reference:

1. In the Update Connectivity Reference form, uncheck the *Use schematic view* option.
2. Click *OK*.

Layout XL is launched, but if you choose a command that requires information from a connectivity source, the Update Connectivity Reference form pops up automatically and you cannot proceed with that command until you specify a schematic as the connectivity reference.

When using Layout XL without a connectivity reference, the connectivity model stipulates that the nets of pins and instance terminals cannot be changed, but the nets of all other shapes can be changed if they connect to a pin or an instance terminal.

You can specify a connectivity reference at any time by choosing *Connectivity – Update – Connectivity Reference* from the layout window menu bar.

Specifying a Connectivity Reference

To specify a schematic as the connectivity reference for a layout view,

1. In the Update Connectivity Reference form, check the *Use schematic view* option and do one of the following.
 - ☐ Type the *Library*, *Cell*, and *View* names of the schematic in the fields provided.
You can type in the name of a different library. The default view name is `schematic`.
 - ☐ Use *Sel by Cursor* and click on an open schematic window
 - ☐ Use the *Browse* button to browse your file system for the schematic you want.
2. Switch on *Update layout cellviews hierarchically* to update the library setting for any lower-level cellviews in a hierarchical cellview.
3. Click *OK*.

The default Layout XL Desktop appears: the schematic window, the layout window, the CIW, and the LSW.

Starting Layout XL from a Physical Configuration View

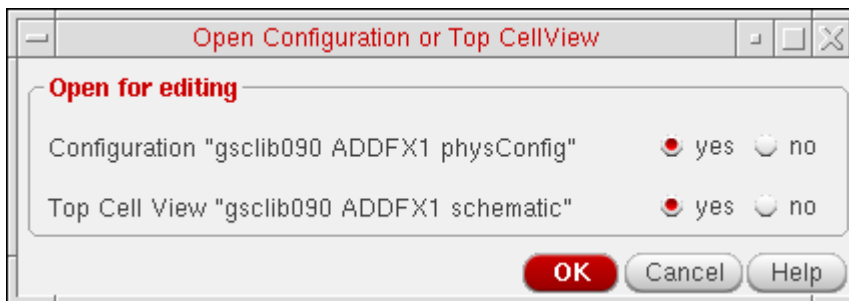
To start Layout XL directly from a physical configuration view,

1. From the CIW, choose *File – Open*.

The Open File form is displayed.

2. Specify the *Library* and *Cell* names and choose *physConfig* from the *View* cyclic field.
3. Click *OK*.

The Open Configuration or Top Cellview form is displayed.



4. Set both *Configuration* and *Top Cellview* to *yes* to open both the physical configuration view and the schematic view.

Note: You can set defaults for this form using the following environment variables.

```
envSetVal("ddserv.he" "hierEditor" 'boolean t)
envSetVal("ddserv.he" "schEditor" 'boolean t)
```

5. Click *OK*.

The schematic view is opened in a Virtuoso Schematic Editor window and the Configure Physical Hierarchy window is displayed showing the physical configuration view you specified. The system checks out a Layout XL license if there is not one currently checked out.

6. Start Layout XL as detailed in Starting Layout XL from a Schematic View.

Automatic Constraint Transfer during Layout XL Startup

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

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

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

For more information about constraint transfer, see [Constraint Transfer](#).

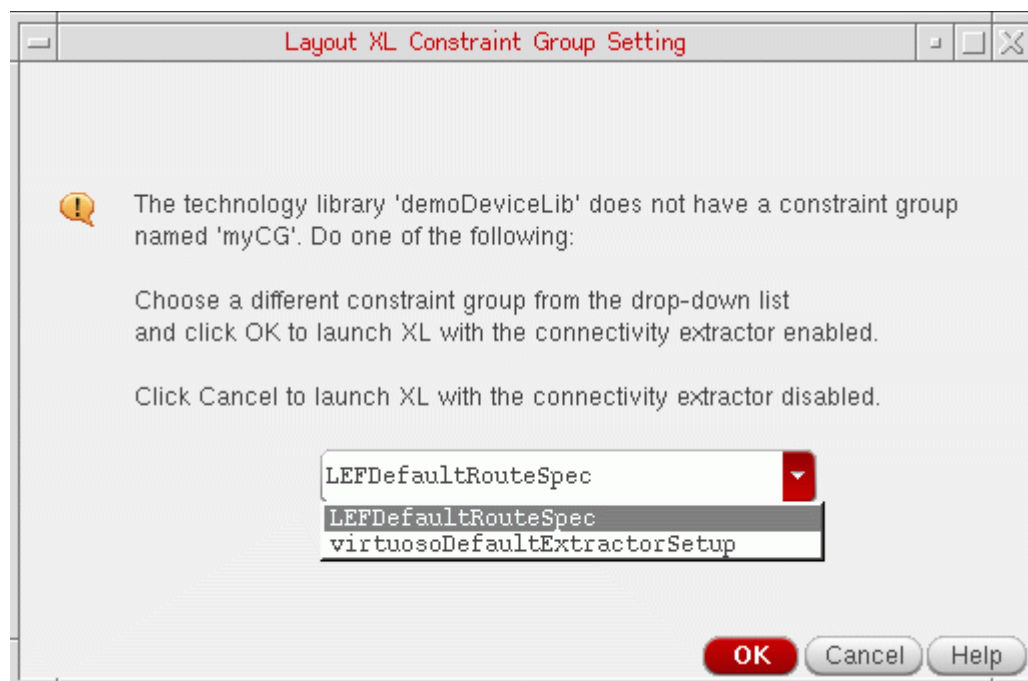
Specifying the Constraint Group

The Layout XL connectivity extractor requires a constraint group containing information on valid layers, valid vias, and error layers. You specify the default constraint group using a layout environment variable called [setupConstraintGroup](#).

For more information on using the environment variable in Soft Block mode, see [Specifying Obstructions](#).

If the specified constraint group does not contain the information required by Layout XL, during initialization the software pops up a dialog inviting you to either:

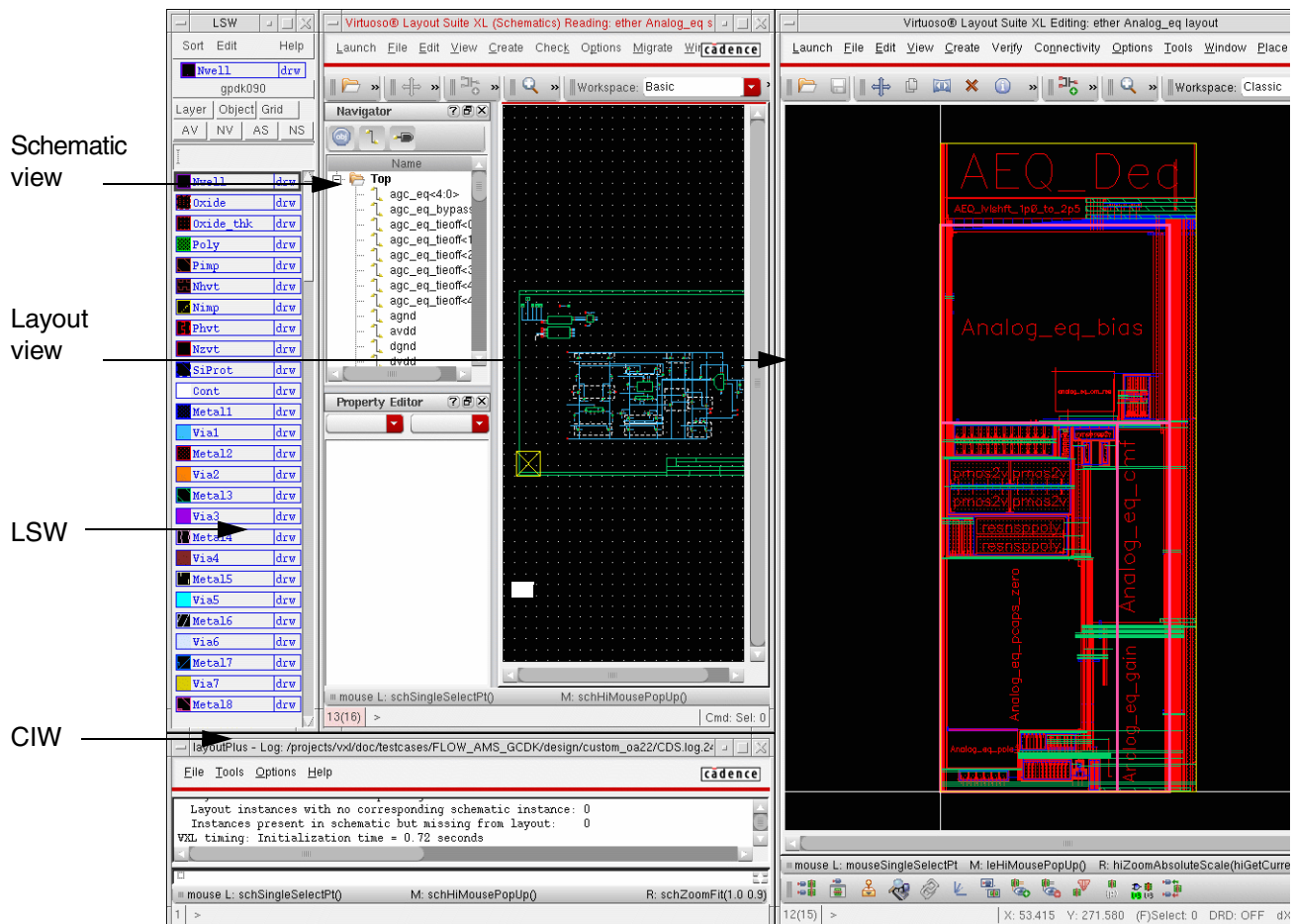
- Choose the constraint group you want from the ones specified in the technology file.
- Launch XL with the connectivity extractor disabled.



For more information on constraint groups and Layout XL, see [Constraint Groups](#).

Layout XL Desktop

When you start Layout XL, the four windows are rearranged into the default window configuration on your desktop. The layout window is on the right with “Virtuoso® Layout Suite XL...” in the banner; the schematic that corresponds to the layout is in the schematic window on the left. The layer selection window (LSW) and the command interpreter window (CIW) are placed to the left and bottom of the screen respectively.



You can move, resize, and iconify any of the four windows.

Customizing the Desktop

You can save and reuse your customized desktop layout in one of the following ways.

- Using the *Options – Save Defaults* command from the CIW.

Virtuoso Layout Suite XL User Guide

Getting Started with Layout XL

When you use this command, Layout XL saves the size and position of the schematic and layout windows as a property on the current layout cellview. It also sends the LSW and CIW window size and location information to the `.cdsenv` file.

When you reopen the layout and schematic cellviews, they appear in the same configuration as they had when you used the *Save Defaults* command.

For more information, see [Saving and Recalling Default Settings](#) in the *Virtuoso Design Environment User Guide*.

- By adding the appropriate environment variables to your `.cdsenv` file; for example, to prevent Layout XL from rearranging or resizing windows, add the following line to the `.cdsenv` file:

```
envSetVal("layoutXL" "autoArrange" 'boolean nil)
```

For more information, see [Layout XL Environment Variables](#).

Note: If you save the current window positions and form settings to a file other than `.cdsenv` (for example, `.envFileName`), you can restore the settings by adding the following line to your `.cdsinit` file or typing it in the CIW.

```
envLoadFile("layoutXL" ".envFileName")
```

Layout XL Window

The Layout XL window extends the basic functionality available in the Layout L window by adding functions that allow you to edit soft blocks and component types, create soft pins and feedthrough pins, edit options settings specific to Layout XL functions, and place pins.

It also adds a raft of connectivity-based commands that let you generate, check, and update your layout components based on the components present in the schematic view. These commands are described in detail in the remaining chapters of this user guide and are summarized in [Command Quick Reference](#)

The Layout XL window comprises a menu bar, a canvas, in which you can display and manipulate your design graphically, and a number of toolbars and dockable assistants to help you with your design tasks. This guide describes only those toolbars and assistants as they relate to Layout XL design tasks.

- For information on the individual components of the layout window, see the [Virtuoso Layout Suite L User Guide](#).
- For information on the individual components of the schematic window, see the [Virtuoso Schematic Editor L User Guide](#).

- For detailed information on the new design environment, see the *Virtuoso Design Environment User Guide*.

Menu Access Keys

Menu access keys provide keyboard access to application menus and the functionality they contain without the need to use mouse selections. For example, if you press **Alt-F** on your keyboard, the software displays the contents of the *File* banner menu.

The menu access key for an Layout XL menu, submenu, or command is the underlined letter in the menu command label. For example, to display the Layout XL Options form using only menu access keys,

1. With the cursor in the Layout XL window, press **Alt-O** to display the contents of the *Options* banner menu.
2. Press **Alt-X** to display the Layout XL Options form.

Bindkeys

Bindkeys are distinct from menu access keys in that they provide direct access to a function without any manipulation of the graphical user interface. A bindkey is a macro that assigns a menu command to a key you choose from the keyboard or a mouse button. When your cursor is in the layout window, you can use all the bindkeys that are loaded for the applications you are running.



Tip

For detailed information on bindkeys in the Virtuoso design environment, see *Bindkeys and Access Keys* the *Virtuoso Design Environment User Guide*.

Default Bindkeys Defined for Layout XL

There is a list of default bindkeys for Layout XL at the following location.

`your_install_dir/samples/local/lxBindKeys.il`

Where `your_install_dir` is the directory in which you store Cadence software.

Virtuoso Layout Suite XL User Guide

Getting Started with Layout XL

This file defines the following bindkeys for Layout XL.

Key	SKILL	Menu
F10	<u>nclToggleCAEMode()</u>	<i>Options – Layout XL – General tab; Constraint-aware editing option</i>
Ctrl+h	<u>geDSAFreeze(hiGetCurrentWindow() hiGetPoint(hiGetCurrentWindow()))</u>	<i>Freeze DSA and Unfreeze DSA from the RMB context menu</i>
Ctrl+l	<u>lxProbeRemoveAll()</u>	
Ctrl+m	<u>lxHiSwapComps()</u>	<i>Edit – Advanced – Swap Components</i>
Shift+l	<u>lxHiProbe()</u>	<i>Connectivity – XL Probe</i>
RMB	<u>lxCmdOptions()</u>	
Shift+RMB	<u>lxCmdShiftOptions()</u>	
Shift+Ctrl+h	<u>lxToggleShowAllIncompleteNets()</u>	<i>Show Hide All Incomplete Nets</i>
h	<u>lxToggleShowIncompleteNets()</u>	<i>Show Hide Current Incomplete Nets</i>
Shift+h	<u>lxShowHideIncompleteNets()</u>	<i>Show/Hide Selected Incomplete Nets</i>

Displaying Bindkeys

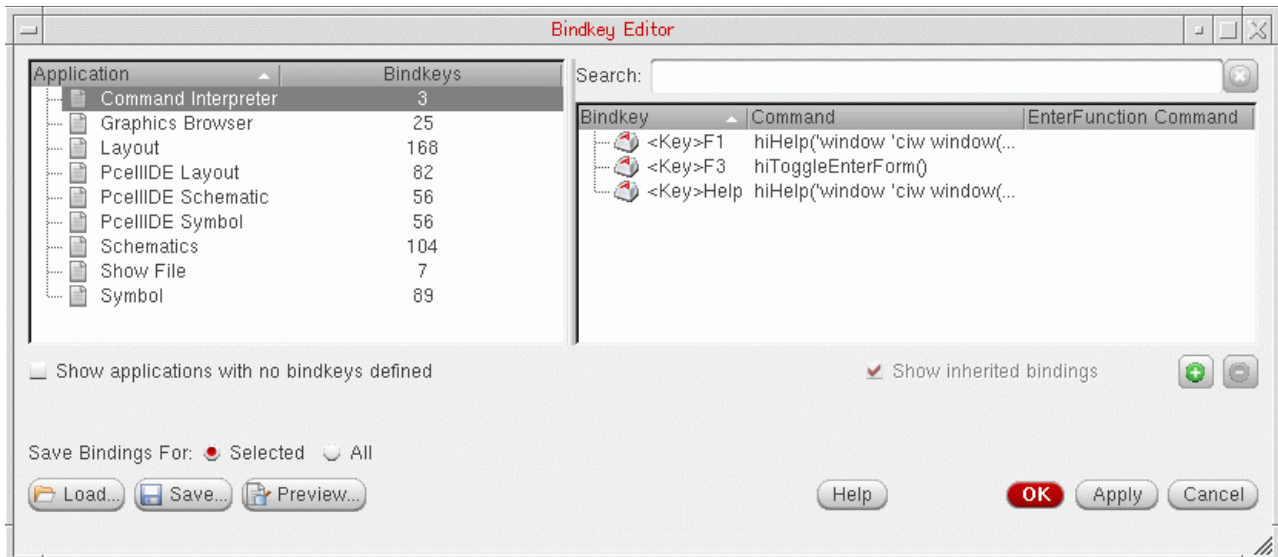
To display the list of bindkeys for Layout XL,

1. From the CIW menu bar, choose *Options – Bindkeys*.

Virtuoso Layout Suite XL User Guide

Getting Started with Layout XL

The Bindkey Editor form is displayed.



Note: The *Application Tree* to the left of the form, by default, initially displays all the applications for which bindkeys have been defined.

2. In the *Application Tree*, select the application whose bindkeys you want to view.

Note: The *BindKey Table*, to the right of the form, will update to list the bindkeys associated with the currently selected application.

3. Optionally, use the *Search* field to filter the bindkey listing.
4. To view the bindkey settings for the currently selected application, either:
 - ☐ Scroll through the listed bindkeys in the *BindKey Table* to view the current settings for one or more bindkeys, or

Virtuoso Layout Suite XL User Guide

Getting Started with Layout XL

- ❑ Select the *Preview* button to display a listing of all of the bindkeys associated with the selected application in a text window.



The screenshot shows a window titled "Layout Application Bindkeys" with a menu bar containing "File" and "Help". The main content area displays a list of bindkeys for the "Layout" application, organized into two sections separated by dashed lines. The first section, "Bindkeys for 'Layout'", lists various applications that inherit the bindkeys. The second section, "hiSetBindKeys('Layout' list(", lists specific keybindings and their corresponding actions.

```
;;-----  
;; Bindkeys for 'Layout'  
;; Inherited by:  
;; * Dracula Interactive  
;; * NC-Verilog-MaskLayout  
;; * Other-Layout  
;; * Other-MaskLayout  
;; * Other-Symbolic  
;; * Pcell  
;; * Power IR/EM  
;; * SiP-Layout  
;; * Spectre-Layout  
;; * Spectre-MaskLayout  
;; * Spectre-Symbolic  
;; * UltraSim-Layout  
;; * UltraSim-MaskLayout  
;; * UltraSim-Symbolic  
;; * VLS-GXL  
;; * Virtuoso CE  
;; * Virtuoso XL  
;; * adeqxl-maskLayout  
;; * adexl-maskLayout  
;; * parasitics-MaskLayout  
;;-----  
hiSetBindKeys( "Layout" list(  
  list("<DrawThru1>" "leSelBoxOrStretch()" "geSingleSelectBox()")  
  list("<DrawThru2>" "" "_cmdOptionDrawThru2()")  
  list("<DrawThru3>" "hiZoomIn()" "hiZoomIn()")  
  list("<Key>#" "hiToggleAnchorMagnifier()")  
  list("<Key>'" "hiUpdateMagOptions()")  
  list("<Key>." "leToggleMagnifier()")  
  list("<Key>A" "leHiQuickAlign()")  
  list("<Key>B" "leReturnToLevel()")  
  list("<Key>BackSpace" "deletePoint()" "deletePoint()")  
  list("<Key>C" "leHiCopy()")  
  list("<Key>D" "geSubSelectPoint()")  
  list("<Key>Delete" "leHiDelete()" "leHiDelete()")  
  list("<Key>Down" "geScroll(nil \"s\" nil)")  
  list("<Key>E" "leHiEditDisplayOptions()")  
  list("<Key>Escape" "" "cancelEnterFun()")  
  list("<Key>F" "hiZoomAbsoluteScale(hiGetCurrentWindow() 0.9)")  
  list("<Key>F1" "hiHelp('window deGetAppInfo(deGetEditViewType(hiGetCurrentWindow()")  
  list("<Key>F11" "deToggleAssistants()")  
  list("<Key>F2" "geSave()")  
  list("<Key>F3" "geSaveAs()" "hiToggleEnterForm()")  
  list("<Key>F4" "geTogglePartialSelect()")  
  list("<Key>F5" "deFileOpen()")  
  list("<Key>F6" "leToggleMaintainConnections()")  
  list("<Key>F7" "leHiEditDRDRuleOptions()")  
  list("<Key>F8" "leHiEditDRDRuleOptions()")  
)
```

16

Loading Bindkeys

To load a file of bindkey definitions every time you run Layout XL,

- Add the following line to your `.cdsinit` file.

```
load(prependInstallPath("path_to_bindkey_file"))
```

For example, the following line loads the list of default bindkeys for Layout XL.

```
load(prependInstallPath("samples/local/lxBindKeys.il"))
```

Canvas

In the canvas, you create and edit objects: paths, polygons, and other shapes for your physical layout. The canvas is always visible.

For more information, see [Layout Editor L Basics](#) in the *Virtuoso Layout Suite L User Guide*.

Dockable Assistants

Layout XL is shipped with either dockable assistants. You show and hide these using the layout window menu bar by choosing *Windows – Assistants* and selecting the name of the assistant you want.

- [Navigator](#)
- [Constraint Manager](#)
- [Search](#)
- [Property Editor](#)
- [World View](#)
- [Annotation Browser](#)

For more information, see [Layout XL Assistants](#).

Virtuoso Layout Suite XL User Guide

Getting Started with Layout XL

Toolbar

The Layout XL toolbar lets you access the main Layout XL commands directly without opening any menus. As with all toolbars, you can use the handle on the left hand side of the toolbar to reposition it anywhere within the layout window.



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

Icon	Command	Form
	<i>Generate All From Source</i> <u>Generating All Components from Source</u>	<u>Generate Layout</u>
	<i>Generate Selected From Source</i> <u>Generating Selected Components from Source</u>	<u>Generate Selected Components</u>
	<i>Generate Clones</i> <u>Generating Clones</u>	<u>Generate Clones</u>
	<i>Update Clone Families</i> <u>Updating Clone Families</u>	<u>Update Clone Families</u>
	<i>Generate Chained Devices</i> <u>Chaining Devices Interactively</u>	<u>Generate Chained Devices</u>
	<i>Generate Folded Devices</i> <u>Folding a Transistor</u>	<u>Generate Folded Devices</u>
	<i>Place As In Schematic</i> <u>Moving Components Inside the Place and Route Boundary</u>	N/A
	<i>Update Components And Nets</i> <u>Updating Components and Nets</u>	<u>Update Components and Nets</u>
	<i>Update Layout Parameters</i> <u>Updating Layout Parameters</u>	N/A
	<i>Update Schematic Parameters</i> <u>Updating Schematic Parameters</u>	N/A
	<i>Define Device Correspondence</i> <u>Updating Device Correspondence</u>	<u>Define Device Correspondence</u>

Virtuoso Layout Suite XL User Guide

Getting Started with Layout XL

Icon	Command	Form
	<i>Extract Layout</i> <u>Extracting a Top-Level Design</u>	<u>Extract Layout</u>
	<i>Windows – Assistants – Annotation Browser</i> <u>Annotation Browser</u>	<u>Annotation Browser Graphical User Interface</u>
	<i>Connectivity – Nets – Show/Hide Selected Incomplete Nets</i> <u>Working with Incomplete Nets</u>	N/A

Workspaces

You can rearrange the canvas, assistants, and toolbars to suit your individual work preferences. A customized configuration of the toolbars and assistants makes a *workspace*.



Tip

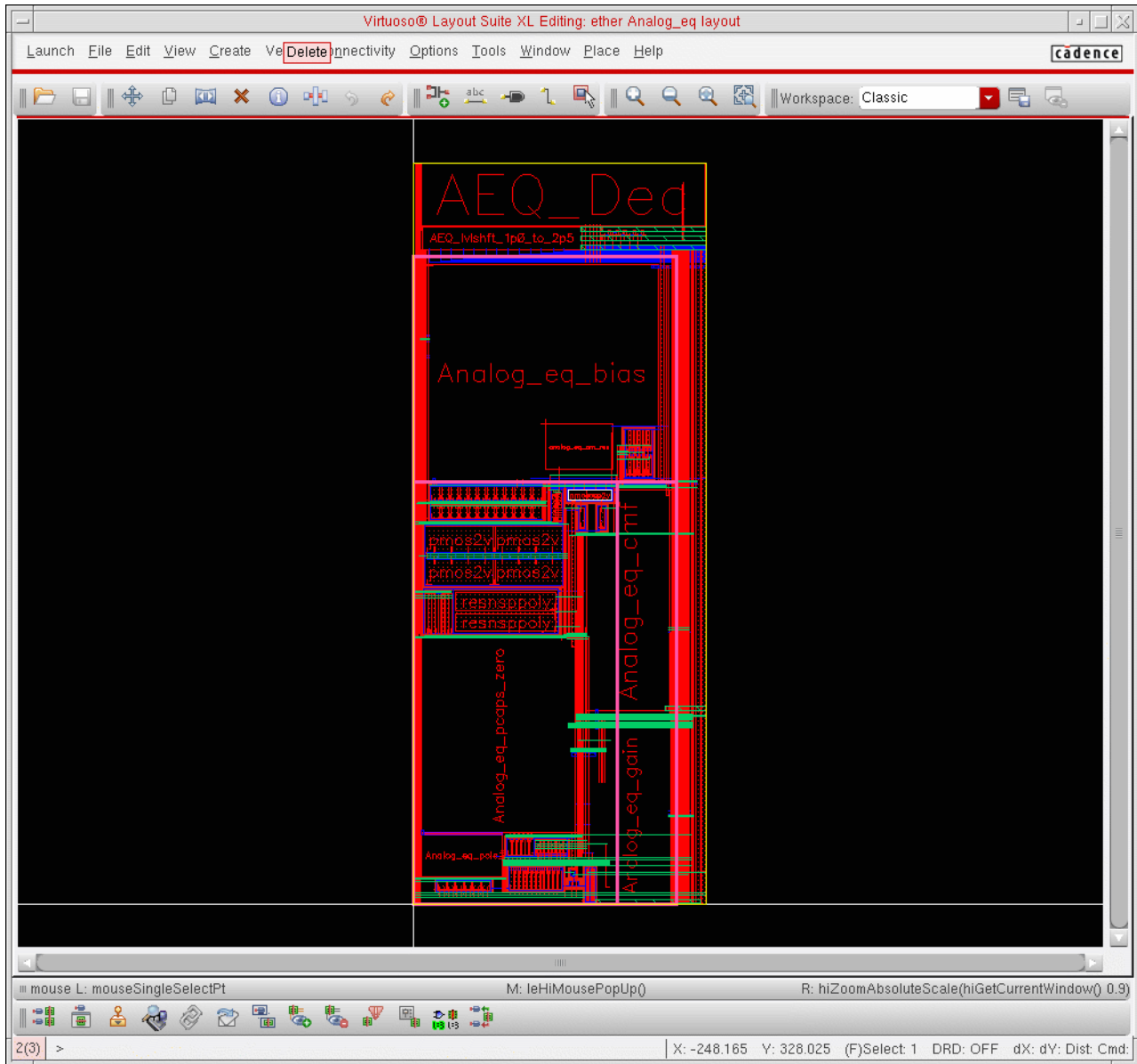
This section describes only the default workspaces defined for Layout XL. For detailed information on workspaces and how you define them, see Getting Started with Workspaces and Working with Workspaces in the *Virtuoso Design Environment User Guide*.

Layout XL ships with the following default workspaces.

Virtuoso Layout Suite XL User Guide

Getting Started with Layout XL

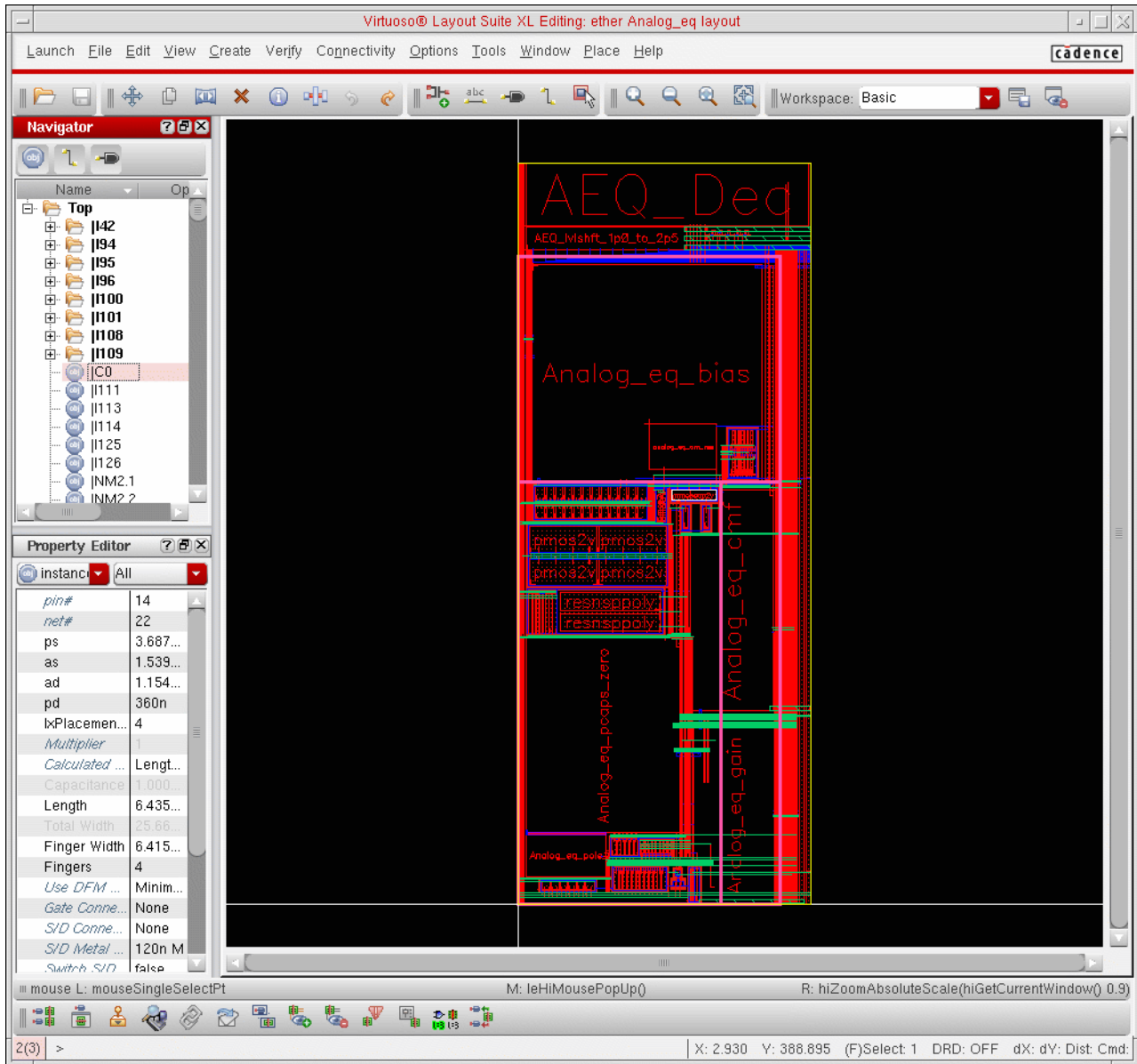
- *Classic* (the default), which shows only the canvas and the toolbars. This workspace mimics the appearance of the application in the previous releases.



Virtuoso Layout Suite XL User Guide

Getting Started with Layout XL

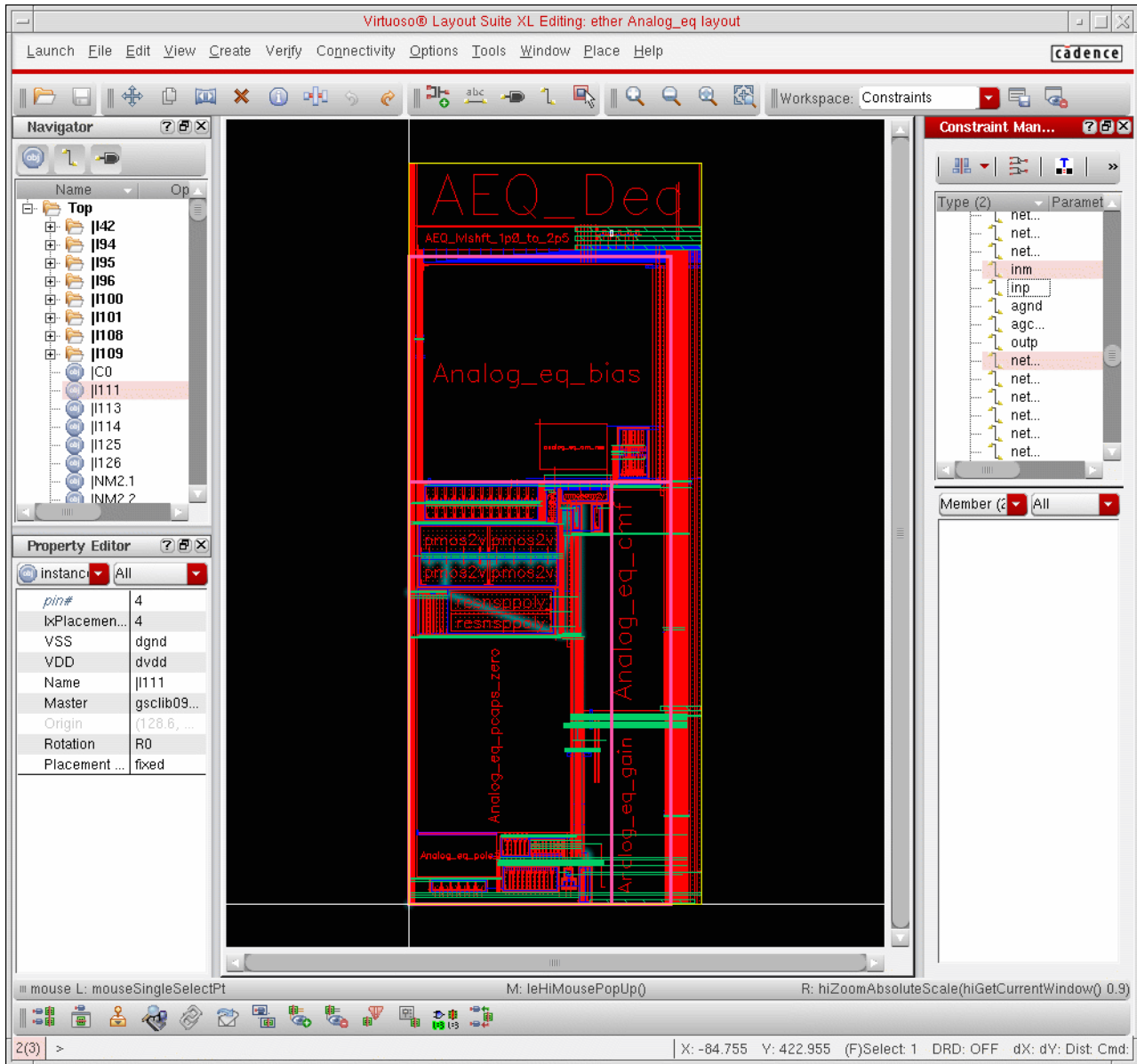
- **Basic**, which shows the canvas, the toolbars, and the Navigator and the Property Editor assistants.



Virtuoso Layout Suite XL User Guide

Getting Started with Layout XL

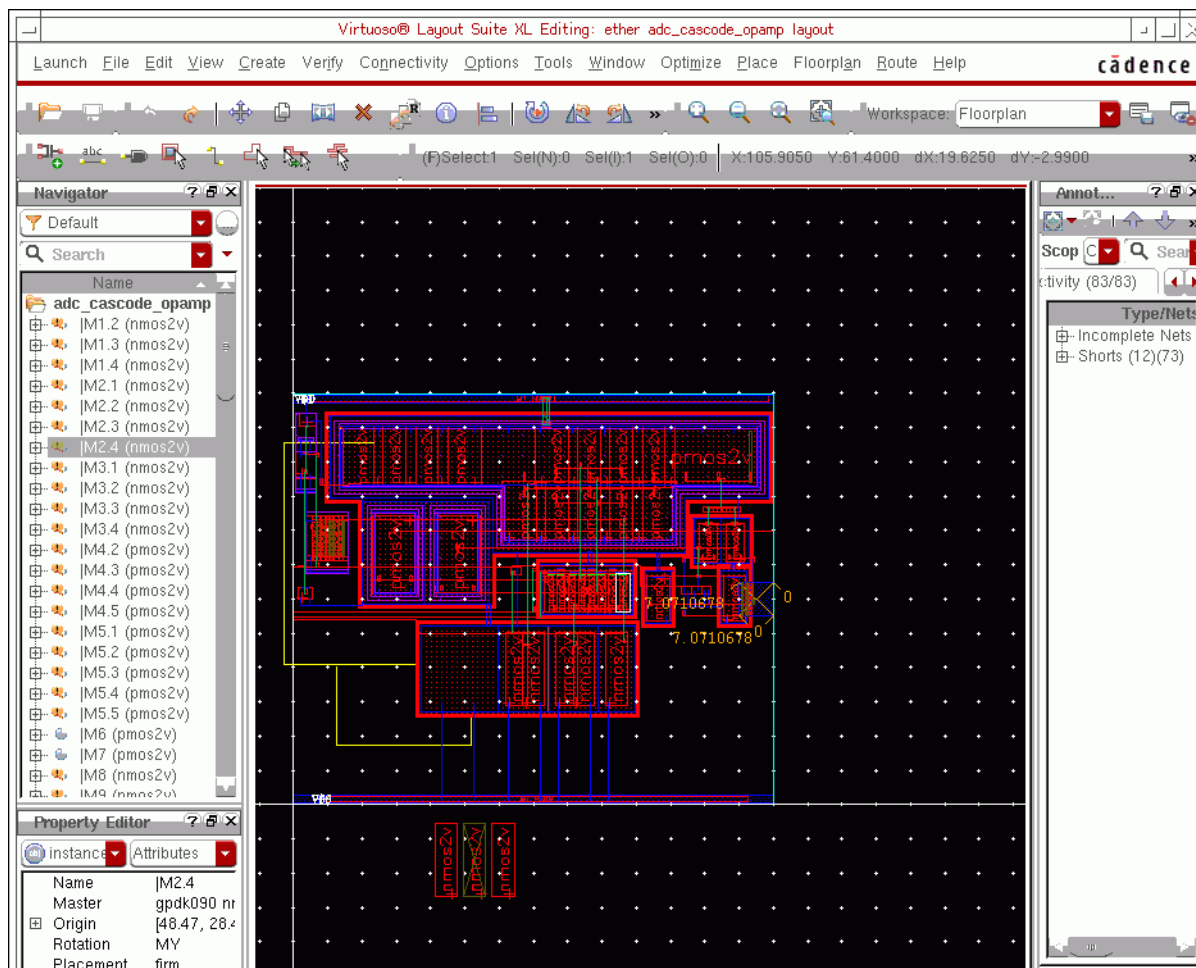
- *Constraints*, which shows the canvas, the toolbars, and the Navigator, Property Editor, and the Constraint Manager assistants.



Virtuoso Layout Suite XL User Guide

Getting Started with Layout XL

- **Floorplan**, which shows the canvas, the toolbars, and the Navigator, Property Editor, and the Annotation Browser assistants.



Setting Environment Variables

Environment variables control the values of the Layout XL options. For a list of all the Layout XL environment variables and their values, see [Layout XL Environment Variables](#).

There are three ways in which you can set environment variables:

- To set an environment variable that is applied every time you start Layout XL, add the setting to your `.cdsenv` or `.cdsinit` file. For more information, see [Setting Environment Variables in a .cdsenv or .cdsinit File](#).

Virtuoso Layout Suite XL User Guide

Getting Started with Layout XL

- To set an environment variable that is applied for the duration of the current session, use the `envSetVal()` command in the CIW. For more information, see [Setting Environment Variables in the CIW](#).
- To set an environment variable from the layout window, use the Layout XL commands or, if appropriate, the form associated with the command you are using. For more information, see [Setting Environment Variables in the Layout XL Options Form](#).

Note: The graphical user interface provides access only to specific environment variables.

Setting Environment Variables in a `.cdsenv` or `.cdsinit` File

To have your environment variable settings set automatically when you start Layout XL, do one of the following.

- Include the environment variables in the `.cdsenv` file in your home directory; for example,

```
layoutXL chainFolds boolean t
```

- Include an `envSetVal()` command in your `.cdsinit` file

```
envSetVal("layoutXL" "chainFolds" 'boolean t)
```

For more information on the `.cdsenv` and `.cdsinit` files, see [Environment Variables](#) in the *Virtuoso Layout Suite L User Guide*.

Setting Environment Variables in the CIW

If you use any environment variable values consistently and do not want to set these values each time you use a command, you can set the variables to the value you normally use in the CIW and it will remain valid for the duration of the current session.

To set environment variables for a single session, do one of the following.

- Include `envSetVal()` in any other Cadence SKILL file you load.
- Type `envSetVal()` in the CIW.

For example, to set the Layout XL `chainFolds` variable, which causes Layout XL to abut the folds of newly folded devices into chains, type the following in the CIW or include it in a setup file.

```
envSetVal("layoutXL" "chainFolds" 'boolean t)
```

To determine the current value of any Layout XL environment variable, type the following in the CIW.

```
envGetVal("layoutXL" "chainFolds")
```

For more information on using `envSetVal()` and `envGetVal()`, see [Layout Editor L Basics](#) in the *Virtuoso Layout Suite L User Guide*.

Setting Environment Variables in the Layout XL Options Form

To set environment variables in the Layout XL Options form,

1. From the layout window menu bar, choose *Options – Layout XL*.

The [Layout XL Options](#) form is displayed.

2. Select one of the following.

- ☐ *Global* to apply options to the current cellview and all other views of the current cell.
- ☐ *Cellview* to apply options only to the current cellview. Selecting this option inactivates the *Extraction*, *Generation*, and *Parameters* tabs, along with certain options on the *General* and *Display* tabs.

3. Select the options you want to apply to your design.

4. In the *Load/Save* group box at the bottom of the *General* tab, choose whether you want to save the options settings to the current cellview, the design library, the technology library, or a file.

5. Click *OK* or *Apply*.

The environment variable values you choose are enforced for all subsequent design sessions until you change the value of the variables again.

Note: To load previously-saved options into this form, type the name of a file and click *Load from*. To delete settings from a cell or a library, click *Delete from*. You cannot delete environment values from a file.

Display and Selection Controls

This section describes how to customize the display settings for the layout canvas and explains the selection and cross-selection mechanisms in Layout XL.

Setting Display Colors

You can change the colors and characteristics of the display layers that Layout XL uses for the following functionality.

- The Annotation Browser assistant uses the `y0` through `y9` layers to show flight lines.
For more information, see Highlighting Incomplete Nets.
- The *XL Probe* command uses the `(hilite drawing)` through `(hilite drawing9)` layers to probe pins, nets, and devices.
For more information, see Highlighting Probes.
- The *Generate Selected From Source* command uses the `(hilite drawing1)` layer to highlight schematic symbols.
- The *Check Against Source* command uses the `(hilite drawing9)` entry layer to identify missing devices.
- The *Highlight* command for placement constraints uses the `(annotate drawing3)` entry layer to highlight in the layout the components bound by a particular placement constraint.

Editing Entry Layers



Tip

Although you can edit layers from the LSW, it is not recommended.

To edit entry layers,

1. In the CIW, choose *Tools – Display Resource Manager*.
2. In the Display Resources Tool Box, click *Edit*.

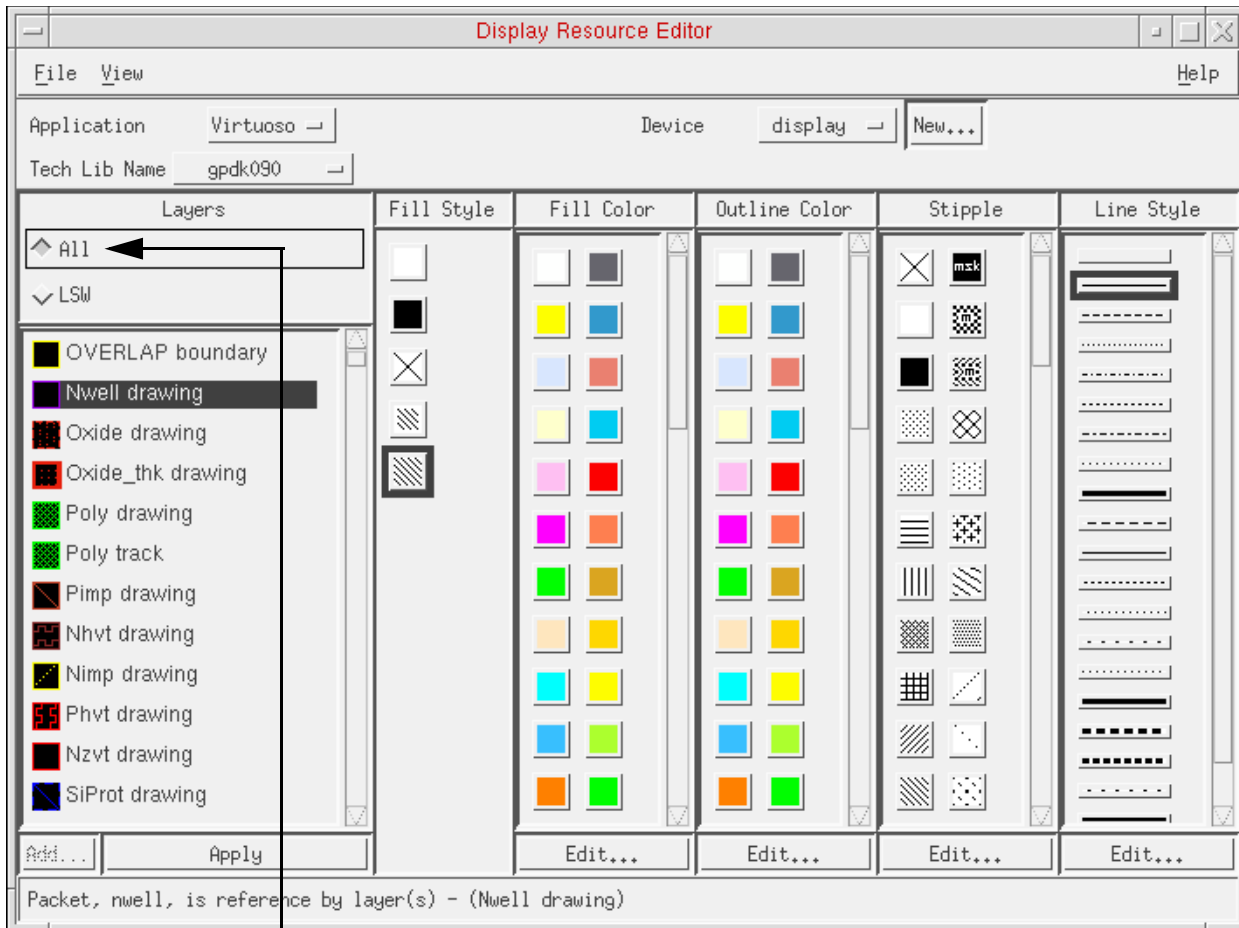
The Display Resource Editor window appears.

For more information, see Display Resource Editor (DRE) Form in the *Virtuoso Technology Data User Guide*.

Virtuoso Layout Suite XL User Guide

Getting Started with Layout XL

3. Click *All*, so that you can see all the available layers.



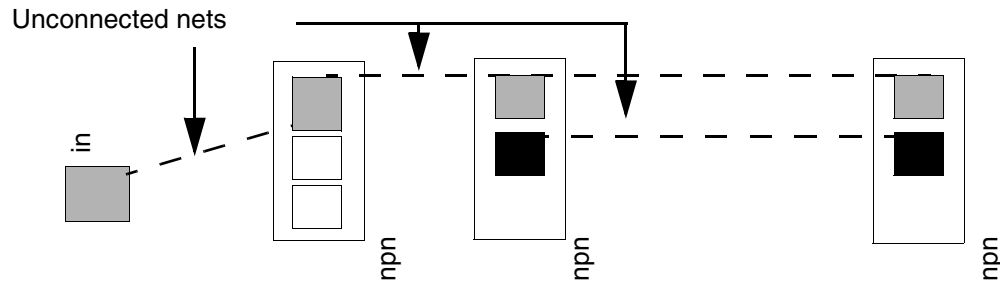
Click on *All*

4. In the *Layers* column, choose the name of the layer to change.
5. Turn on the *Fill Style*, *Fill Color*, *Outline Color*, *Stipple*, and *Line Style* options you want to use for that layer.
6. Click *Apply*.

The next time you use a command that displays the layer you changed, the color or pattern changes in the design window.

Highlighting Incomplete Nets

To highlight incomplete connections with flight lines, Layout XL cycles through the `y0` through `y9` layers. For more information, see [Working with Incomplete Nets](#).



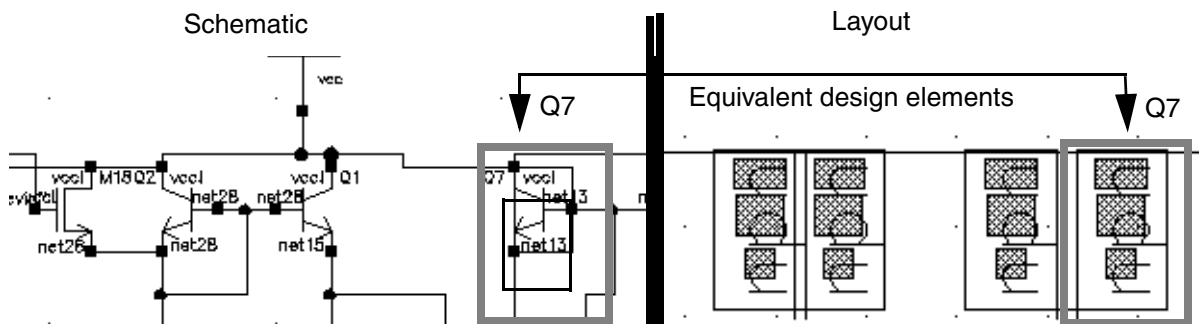
Layout XL displays the flight lines of each net in a different color. If there are more than 10 flight lines, the cycle is repeated unless you have specified certain layers for specific nets. For more information, see [Assigning Colors to Incomplete Nets](#).

When assigning new colors to the flight lines,

- Use colors that do not closely resemble each other or the colors you use for paths and components.
- Avoid solid fill so you can see what is underneath.

Highlighting Probes

Layout XL uses the (`hilite drawing`) through (`hilite drawing9`) entry layers for probes, which identify equivalent design elements on the schematic and the layout.



The *Check Against Source* command uses the (`hilite drawing9`) layer to identify components in the schematic that are not in the layout (and vice versa).

If you change the colors of the (`hilite drawing`) layers,

- Use thick lines so that you can see device pins.
- Use bright colors to make probes and highlights easy to find.



Tip

Do not make the (`hilite drawing`) layer and the (`hilite drawing2`) layer solid fill because these layers are used to indicate selected components and to manipulate shapes.

Selecting Objects in Layout XL

You can select objects either before or after you start a command.

Preselect Mode

If you select objects before starting a command, the command is said to be operating in *preselect mode*. In preselect mode, the command operates only on the objects that were selected when the command was started. When the command is completed, the selected objects remain in the selected set.

Note that in preselect mode,

- Editing commands do not automatically repeat, even if the *Repeat Commands* option is switched on in the Layout Editor Options form.
- Some editing commands prompt you for a starting point, called the reference point, for the edit.

Postselect Mode

If you select the command and then the objects, the command is said to be operating in *postselect mode*. In postselect mode, you make your selection within the command and when the command is completed, nothing is selected.

Note that in postselect mode,

- Editing commands automatically repeat if *Repeat Commands* is on (the default).
- Editing commands do not necessarily prompt you for a reference point. If you click to select the first object, the editor uses that point as the reference point.

Deselecting Objects in Layout XL

- To deselect an object:
 - ➔ Invoke the *(De)Select Under Cursor* context menu and from the *De(select)* submenu, hover over the object you want to deselect. The object gets highlighted.

For more information on object selection, see [Editing Objects](#) in the *Virtuoso Layout Suite L User Guide*.

Viewing Multiple Cellviews in Layout XL

You can open more than one pair of schematic-layout windows in Layout XL and work on each pair independently.

However, when you open multiple cellviews in Layout XL, the second and subsequent sets of windows are not automatically configured in the same way as the first set.

- If you open a schematic and two copies of the same layout, the *XL Probe* command and the commands in the *Check* and *Update* submenus apply to both layouts.
- If you open a schematic and two different layouts, the *XL Probe* command and the commands in the *Check* and *Update* submenus apply only to the layout from which you selected the command.

Displaying Options Forms Automatically

Options forms let you change command settings while you are running commands. Many options forms do not appear automatically if you have *Options Displayed When Commands Start* turned off in the *Options - User Preferences* in the CIW. In this situation, you must press **F3** to see the command forms.

Printing to the Command Interpreter Window

To print process information to the CIW instead of to a separate Layout XL Info window,

- Add the following line to your `.cdsenv` file or type it in the CIW

```
envSetVal("layoutXL" "infoWindow" 'boolean nil)
```

Virtuoso Layout Suite XL User Guide

Getting Started with Layout XL

Technology File Requirements for Layout XL

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

This section describes the Virtuoso technology data requirements for using Layout XL.

- [Layout Editor Rules](#) on page 68
- [Layer Definitions](#) on page 68
- [Layer Rules](#) on page 68
- [Constraint Groups](#) on page 69
- [Devices](#) on page 75
- [Physical Rules](#) on page 76
- [Connectivity Rules](#) on page 77

Further Reading

Technology information specific to Layout XL stored in an existing CDB technology file is automatically updated to its new location in the OpenAccess 2.2 technology file when the database is converted using the `cdb2oa` translator.

- For information about the differences in technology information on CDB and OpenAccess 2.2, see [OpenAccess Technology Data Changes](#) in the *Virtuoso Design Environment Adoption Guide*.
- For information about creating and editing technology files, see
 - [Virtuoso Technology Data User Guide](#)
 - [Virtuoso Technology Data ASCII Files Reference](#)

- You can also use Cadence SKILL functions to query or update the technology file. For more information, see
 - ❑ [Virtuoso Technology Data SKILL Reference](#)
 - ❑ [Virtuoso Layout Suite SKILL Reference](#) (Chapter 3)

Layout Editor Rules

The `leRules` section of the technology file specifies the layers that are listed in the layer selection window (LSW) and the order in which they are listed. For more information, see “Technology File LSW Layers Specification” in the [Virtuoso Technology Data ASCII Files Reference](#).

Layer Definitions

The `layerDefinitions` section of the technology file defines the layers used in the current technology. All the layers used in your design must be defined in this section. For more information, see “Technology File Layer Definitions” in the [Virtuoso Technology Data ASCII Files Reference](#).

Layer Rules

The `layerRules` section defines rules for the layers used in the current technology.

Layout XL requires the `functions` table, which specifies the layer function and assigns a `maskNumber` to each layer; for example,

```
functions (
; ( layer          function          [maskNumber] )
; ( -----          -)
  ( NWELL          "nwell"           1          )
  ( PWELL          "pwell"           2          )
  ( NDIFF          "ndiff"           3          )
  ( NPLUS          "nplus"           4          )
  ( PPLUS          "pplus"           5          )
  ( NIMP           "nplus"           6          )
  ( PIMP           "pplus"           7          )
  ( DIFF2          "pdiff"           8          )
  ( DIFF           "diff"            9          )
  ( POLY           "poly"            10         )
  ( LI             "li"              11         )
  ( CONT           "cut"              12         )
  ( METAL1         "metal"           13         )
  ( VIA1           "cut"              14         )
  ( METAL2         "metal"           15         )
)
```


`);functions`

This section replaces the `layerFunctions` section used in previous releases and is used by the connectivity extractor when it derives the connectivity rules for the current technology. For more information, see [Connectivity Rules](#).

Specifying Electrical Equivalence

If two or more layers need to be considered electrically equivalent, use the `equivalentLayers` section. For information, see [Connectivity Rules](#).

Specifying Via Layers

In previous releases, Layout XL also required the `viaLayers` section to be defined. This section is no longer required because the extractor now derives the list of via layers from the `validVias` constraint, which lists the `viaDefs` contained in the technology file. For more information, see [Connectivity Rules](#).

Constraint Groups

A constraint group lets you specify the different sets of design constraints to be applied under different design circumstances or for different design objects. This allows the flexibility to experiment with less or more stringent process rules at different stages of the design process.

You can group together any set of constraints into a constraint group, which is then applied to any design associated with the technology library. For more information, see **Technology File Constraint Groups and Constraints** in the *Virtuoso Technology Data ASCII Files Reference*.

Specifying Information Required by the Layout XL Connectivity Extractor

The Layout XL connectivity extractor requires a constraint group containing information about the following to build the connectivity rules for the technology you are using:

- [Valid Layers](#)
 - [Stop Layers](#)
 - [Bulk Area Layers](#)
- [Valid Vias](#)

■ Error Layers



Tip

For detailed information about the connectivity rules derived from a constraint group, see [Connectivity Rules](#).

A typical constraint group definition is shown below.

```
constraintGroups (
; ( group                                [override] )
; ( ----                                - )
  ( "virtuosoDefaultExtractorSetup"      nil
; layer constraints
  interconnect (
    ( validLayers      ( METAL1 METAL2 METAL3 POLY ) )
    ( validVias        ( VIA01 VIA12 VIA23 ) )
    ( errorLayer       noOverlapLayer1 )
    ( errorLayer       noOverlapLayer2 )
  );interconnect
  );virtuosoDefaultExtractorSetup
);constraintGroups
```

Valid Layers

The `validLayers` constraint specifies all the physical layers that you want to extract through. The layers may be directly listed in the constraint or used as an operand for a top or an intermediate derived layer. Each layer must also be defined in the `layerRules` section.

Here is an example of a `validLayers` constraint definition.

```
(D1 10001 (M1 'select drawing))
(D2 10002 (M2 'select pin))
(D3 10003 (M2 'select drawing))
(D4 10004 (D2 'or D3))
(validLayers (M3 D4 D1))
```

The constraint definition indicates that M3, (M1 drawing), (M2 pin), and (M2 drawing) are extractable; where M3 indicates that all the purposes of the layer, M3, are extractable.

Note: If you want to use the folding, chaining, and abutment functions, you must also include the diffusion layers in your `validLayers` constraint.

You can list the layers in any order. Layout XL establishes the correct sequence using the `maskNumber` in the `functions` section and the order of the via layers. All the purposes of the specified layers are considered electrically connected.

In addition, you can list the cut layers or other non-metal layers in this constraint and specify the layer purposes. For information about how to specify the various purposes of a layer, see [Specifying Layer-Purpose Pairs in the validLayers Constraint](#).

Stop Layers

You can also use the `validLayers` constraint to specify the layers that “stop” another layer. The Stop layers are defined using a NOT operation between two layers.

Let us consider the following derived layer definitions in the `validLayers` constraint:

```
(D1 10001 (Poly 'select drawing))
(D2 10002 (Oxide 'select pin))
(D3 10003 (Oxide 'select drawing))
(D4 10004 (D2 'or D3))
(Stop 10005 (D4 'not D1))
(validLayers (Stop))
```

The constraint definition indicates that the:

- (Poly drawing), (Oxide pin), and (Oxide drawing) are extractable layers.
- (Oxide pin) and (Oxide drawing) are being cut by (Poly drawing).

Let us consider another example where a layer-purpose pair “cuts” another purpose pair on the same layer:

```
(D1 10001 (Oxide 'select pin))
(D2 10002 (Oxide 'select drawing))
(Stop 10003 (D2 'not D1))
(validLayers (Stop))
```

The constraint definition indicates that the:

- (Oxide pin), and (Oxide drawing) are extractable layers.
- (Oxide drawing) is being “cut” by (Oxide pin).

Note: In the example above, since the layer-purpose pair (Oxide pin) “stops” another layer-purpose pair (Oxide drawing) on the same layer, the layer-purpose pair (Oxide pin) is not considered electrically connected to any other layer.

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

Let us consider an example:

```
(Stop 10001 (substrate 'not Nwell))  
(StopWithInsideConstraint 10002 (Stop 'inside DNwell))  
(validLayers (StopWithInsideConstraint))
```

The constraint definition indicates that the:

- Substrate, Nwell, and DeepNwell (DNwell) are extractable layers
- Substrate is being cut by the Nwell layer
- Stopped regions of the substrate must be enclosed within the specified DeepNwell layer

Note that pin figures of instantiated masters can be stopped by top-level shapes on stop layers. This allows you to isolate overlapping or abutting pin figures of two different instantiated masters. Since the pin figures of instantiated masters are considered to be at the top level, they are extractable even at the default extract stop level of 0.

It is also worth noting that shapes of fluid guard rings can stop layers at the top level. Therefore, a fluid guard ring can be used for isolation without having to increase the extract stop level for the whole design. In general, you can use a layer at hierarchical level “N” to “stop” the corresponding area at the top level, provided the extract stop level is sufficient to make the hierarchical level “N” visible to the extractor from the top level.

Bulk Area Layers

In addition to using the `validLayers` constraint for defining extractable and stop layers, you can use the constraint to define the area of a shapeless bulk terminal. The area of a shapeless bulk terminal is determined using an AND operation between the intersection of the shapes on one layer with the shapes on the other layer.

Let us consider the following `validLayers` constraint definition:

```
(D1 10001 (Poly 'select drawing))  
(D2 10002 (Oxide 'select pin))  
(D3 10003 (Oxide 'select drawing))  
(D4 10004 (D2 'or D3))  
(Bulk 10005 (D1 'and D4))  
(validLayers (Bulk))
```

The constraint definition indicates that the:

- (Poly drawing), (Oxide pin), and (Oxide drawing) are extractable layers.
- Intersections of (Poly drawing) and (Oxide drawing), and of (Poly drawing) and (Oxide pin) define the area of shapeless bulk terminals.

Note: If you want to extract the other purposes of the Poly and Oxide layers, you must update the `validLayers` constraint with the corresponding layer purpose information. For example, you can use `(validLayers (Bulk (Poly pin) Oxide))` to additionally specify that `(Poly pin)` is extractable and that all the purposes of the Oxide layer are also extractable.

Valid Vias

You can use the `validVias` constraint to define the layers which the extractor uses. Layout XL determines the extractable layers from the list of via definitions (`viaDefs`) provided. All the vias specified in the `validVias` constraint must be defined in the `viaDefs` section of the technology file.

Note: Although it is possible for Layout XL to determine connectivity from the `validVias` constraint alone, Cadence recommends that you always provide the `validLayers` constraint. When there are no extractable layers in the constraint group, Layout XL will be launched with the connectivity extractor disabled.

Error Layers

The `errorLayer` constraint replaces the `lxNoOverlapLayers` property to list the pairs of layers that should not overlap. The “no-overlap” relationship is defined using an AND operation between two layers. However, for the no-overlap relationship to be honored by the extractor, you must define the corresponding physical layers as extractable.

Let us consider the following `errorLayer` and `validLayers` constraint definition:

```
(D1 10001 (Poly 'select drawing))
(D2 10002 (Oxide 'select pin))
(D3 10003 (Oxide 'select drawing))
(D4 10004 (D2 'or D3))
(NoOverlap 10005 (D1 'and D4))
(validLayers (D1 D4))
(errorLayer NoOverlap)
```

The constraint definition indicates that the:

- `(Poly drawing)`, `(Oxide pin)`, and `(Oxide drawing)` are extractable layers.
- `(Poly drawing)` and `(Oxide drawing)` are no-overlap layers.
- `(Poly drawing)` and `(Oxide pin)` are no-overlap layers.

The details about the pairs of layers that cannot overlap is included in the `layerDefinitions` section. For example,

Virtuoso Layout Suite XL User Guide

Technology File Requirements for Layout XL

```
layerDefinitions(  
  techDerivedLayers(  
    ; ( DerivedLayerName      Layer#      Composition      )  
    ; ( -----              - - - - -  
    ( noOverlapLayer1        10004        ( POLY 'and DIFF )    )  
    ( noOverlapLayer2        10009        ( POLY 'and DIFF2 )    )  
  );techDerivedLayers  
);layerDefinitions
```

In the example above, the number range for derived layers starts from 10000 and 'and' indicates that the layers are ANDed together.

If you are working with a hierarchical design, you can set properties on instances at the lower levels of the design to avoid false overlap and short markers.

The lxBLOCKOverlapCheck property, defined on a shape, instance, or instance master, tells the extractor not to check whether a no-overlap layer of this shape or instance is touching a nonoverlap layer at the current cellview level.

For example, if you define the lxBLOCKOverlapCheck property on a shape in cellview “A”, you will not see any markers when overlapping the shape with an instance. However, if cellview “A” is placed inside cellview “B”, and a *noOverlap* layer is drawn over the shape in cellview “A”, then a marker is displayed. This happens even though the shape in cellview “A” has the lxBLOCKOverlapCheck property defined.

The lxBLOCKExtractCheck property, defined on a shape instance, or instance master, tells the extractor not to check the connectivity between this shape and others it touches.

Composing Layers

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

“The ‘validLayers’ constraint contains derived layer ‘%s’ with unsupported operation ‘%s’ in constraint group ‘%s’ of technology library ‘%s’. This layer has been skipped. Its definition should be corrected or it should be removed from the constraint.”

Let us consider the following definition in the `techDerivedLayers` section of a technology file.

```
(D1 10001 (M1 'select drawing))  
(D2 10002 (M2 'select pin))
```

Virtuoso Layout Suite XL User Guide

Technology File Requirements for Layout XL

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

Here:

- D5 represents the list of physical layers: (M1 drawing), (M2 pin), and (M2 drawing)
- D5 is directly listed in a constraint read by the extractor
- D4, D3, D2, and D1 are “intermediate derived layers”

Note: An intermediate derived layer only supports the OR or SELECT operations. If an intermediate derived layer with an AND or NOT operation is processed by the extractor, the following warning message is issued:

“The ‘validLayers’ constraint contains derived layer ‘%s’ with unsupported operation ‘%s’ in an intermediate derived layer ‘%s’ in constraint group ‘%s’ of technology library ‘%s’. This layer has been skipped. Its definition should be corrected or it should be removed from the constraint.”

Specifying Information Required by the Virtuoso Chip Assembly Router

If you are using the Virtuoso Chip Assembly Router, you must define a `virtuosoDefaultSetup` constraint group. For example,

```
constraintGroups (
; ( group                                [override] )
; ( ----                                ----- )
  ( "virtuosoDefaultSetup"              nil
; layer constraints
  interconnect(
    ( validLayers      ( METAL1 METAL2 METAL3 ) )
    ( validVias        ( M1_M2 M2_M1 ) )
  );interconnect
  );virtuosoDefaultSetup
);constraintGroups
```

The list of layers and vias specified here must be defined in the `iccRules` section of the technology file.

Devices

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

Specifying Vias

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

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



Tip

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

Specifying Multipart Paths

Multipart path templates are defined in the `multipartPathTemplates` section in the technology file. For more information, see “Technology File Devices” in the [Virtuoso Technology Data ASCII Files Reference](#).

Physical Rules

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

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

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

Note: The `physicalRules` section is still used for the `mfgGridResolution` which is required if you are using QCells. For example,


```
mfgGridResolution(  
    ( 0.0050000 )  
);mfgGridResolution
```

Connectivity Rules

The connectivity stack is an internal data structure listing the extractable layers and associated electrical connections used by the Layout XL connectivity extractor. It is derived from the following sections of the technology file.

Technology section	For information, see...
functions	Layer Functions Table
validLayers	Valid Layers
validVias	Valid Vias
viaDefs	Via Definitions
equivalentLayers	Equivalent Layers

Layer Functions Table

The `functions` table in the `layerRules` section of the technology file specifies the function and assigns a `maskNumber` for each layer. For example:

Virtuoso Layout Suite XL User Guide

Technology File Requirements for Layout XL

```

layerRules(

functions(
;( layer      function      [maskNumber])
;( -----)
( DNwell      "nwell"      0      )
( PWdummy     "pwell"      1      )
( Nburied     "recognition" 2      )
( Nwell       "nwell"      3      )
( Oxide       "diff"       4      )
( Oxide_thk   "recognition" 5      )
( Poly        "poly"       6      )
( Nhvt        "nplus"      7      )
( Nimp        "nplus"      8      )
( Phvt        "pplus"      9      )
( Pimp        "pplus"     10      )
( Nzvt        "nplus"     11      )
( Nlvt        "nplus"     12      )
( Plvt        "pplus"     13      )
( SiProt      "pplus"     14      )
( Cont        "cut"       15      )
( Metal1      "metal"     16      )
( Via1        "cut"       17      )
( Metal2      "metal"     18      )
( Via2        "cut"       19      )
( Metal3      "metal"     20      )
( Via3        "cut"       21      )
( Metal4      "metal"     22      )
( Via4        "cut"       23      )
( Metal5      "metal"     24      )
( Via5        "cut"       25      )
( Metal6      "metal"     26      )
( Via6        "cut"       27      )
( Metal7      "metal"     28      )
( Via7        "cut"       29      )
( Metal8      "metal"     30      )
( Via8        "cut"       31      )
( Metal9      "metal"     32      )
( Via9        "cut"       33      )
( Metal10     "metal"     34      )
( Via10       "cut"       35      )
( Metal11     "metal"     36      )
( Bondpad     "passivationCut" 37      )
);functions

```

The `function` is used by Layout XL to identify the cut layers in vias and the `maskNumber` to order the layers in the connectivity stack.

Valid Layers

The `validLayers` section lists all the layers you want to extract through and is specified in the `setupConstraintGroup` currently set for the design. Each layer referenced must be defined in the `layerRules` section.

You can list the layers in any order. Layout XL establishes the correct sequence using the `maskNumber` in the `functions` table and the order of the via layers. All the purposes of the specified layers are considered electrically connected.

For more information, see [Specifying Information Required by the Layout XL Connectivity Extractor](#).

Valid Vias

The `validVias` section is specified in the `setupConstraintGroup` currently set for the design. It is used to limit the list of via definitions considered when building the connectivity stack.

Although it is possible for Layout XL to determine extractable layers from the `validVias` section, Cadence recommends that you use the `validLayers` section instead.

- When the `validVias` section is present, each layer in each of the via definitions listed is set as extractable, including any cut, substrate, and implant layers.
- When the `validVias` section is not present, a layer is considered extractable only if it is listed in the `validLayers` section.

Specifying Layer-Purpose Pairs in the `validLayers` Constraint

A constraint group can have multiple purposes defined for a layer, forming multiple layer-purpose pairs. You can directly add layer-purpose pairs in the `validLayers` constraint.

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

Via Definitions

Vias are defined using `standardViaDefs`, which are equivalent to the `symContactDevices` used in previous releases; and `customViaDefs`, which reference a via cellview. For example,

```
viaDefs (

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

    (PTAP pdiff metall ("contact" 1.0 1.0)
    (1 1 (1.0 1.0))
    (1.0 1.0) (0.5 0.5)
    (0.0 0.0) (0.0 0.0) (0.0 0.0)
    nil nil nil nil
    substrate
    )

    (NTAP ndiff metall ("contact" 1.0 1.0)
    (1 1 (1.0 1.0))
    (1.0 1.0)
    (0.5 0.5)
    (0.0 0.0) (0.0 0.0) (0.0 0.0)
    nil nil nil nil
    nwell
    )
  );standardViaDefs

  customViaDefs (
    ;( viaDefName   libName   cellName   viewName   layer1   layer2   resistPerCut)
    ;( -----   -----   -----   -----   -----   -----   -----)
    ( M2_M1       sample    M2_M1      via         METAL1   METAL2   0.5          )
  );customViaDefs

);viaDefs
```

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

The connectivity extractor derives the following information from via definitions.

- Via Layer Connectivity
- Substrate Layer Connectivity

■ Implant Layer Connectivity

Via Layer Connectivity

Cut and local interconnect (`li`) layers always connect to the associated pair of metal layers. `li` layers also connect the two metals directly. For example, for a via `M1_M2`,

- `cut` layer; two groups of connected layers (`M1 cut`) and (`M2 cut`)
- `li` layer; single group of connected layers (`M1 M2 li`)

For `standardViaDefs`, the cut layer is specified as a parameter of the via definition. For `customViaDefs`, the cut layer is read directly from `functions` table.

Zero-level `cut` and `li` shapes are extracted like any other shapes

Substrate Layer Connectivity

In the Virtuoso design environment, a `standardViaDef` can have a fourth layer—the substrate layer—which is specified using the `well/substrate` construct shown in the `standardViaDef` syntax below.

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

The `well/substrate` construct can target the substrate layer or any physical well layer.

There is no physical shape on the substrate layer in the database. It is only when a substrate via targets a well layer that it connects to physical shapes on that well layer. Otherwise, a substrate via placed over the design is considered connected to the substrate simply by definition. Again, there is no physical shape on the substrate layer in the via (which means that there is no interoperability issue), unless the substrate via targets a well layer, in which case it will have a shape on that well.

The connectivity of the substrate is given by the first substrate via found by the extractor in the design. Therefore the connectivity of the substrate depends on the substrate vias being added, removed, or edited in the design.

As with all vias, the connectivity of a substrate via is given by the net attached to it. There are no opens between substrate vias because they are shorted through the substrate, provided that these vias are on the same net as the substrate. Otherwise, short markers are created between the substrate vias and the substrate itself, which can, in turn, lead to open markers being created between the vias. For substrate vias targeting a well layer, the maintenance of shorts and opens follows the natural definition based on the wiring of the corresponding nets on the corresponding physical layers.

Implant Layer Connectivity

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

```
standardViaDefs(  
  (M1 NWELL Oxide Metall ("Cont" 0.12 0.12)  
    (1 I (0.16 0.16))  
    (0.07 0.07) (0.06 0.06) (0.0 0.0) (0.0 0.0) (0.0 0.0)  
    nimp (0.15 0.15) nwell (0.24 0.24)S)  
);standardViaDefs
```

The implant layers are `nimp` (for Oxide) and `nwell` (for Metall). Often the second implant is used not as a real implant definition, but instead to target a well layer. (This legacy use can be replaced by new `well/substrate` construct described previously.)

In the example shown above,

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

```
Metall -> Cont -> ( Oxide & nimp ) -> nwell
```

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

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

```
Metall -> Cont -> ( Oxide & pimp ) -> pwell
```

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

In both the cases, a cut shape will connect to an oxide shape only if it is implanted with an n-type or a p-type implant.

Layout XL enforces these interconnections not only inside vias, but also for shapes at the top level. For example, a top-level diffusion shape must be implanted by an n-type material in order to connect to a top-level n-well shape.

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

Equivalent Layers

The `equivalentLayers` section lists the layers that need to be set as electrically equivalent. For example, two layers that correspond to two different voltages of a metal are considered as electrically-connected and, therefore, must be listed in the `equivalentLayers` section.

```
equivalentLayers( ((M1a drawing) (M1b drawing) M1c) )
```

This indicates that `(M1a drawing)`, `(M1b drawing)`, and all the purposes of `M1c` are electrically-connected. When a shape on the `(M1a drawing)` layer touches a shape on the `M1c` layer, the two layers form a connection.

Note: The `equivalentLayers` rule does NOT support derived layers. If a derived layer is included in the rule definition, the following message is displayed:

“The ‘equivalentLayers’ rule contains derived layer ‘%s’ in the constraint group ‘%s’ of the technology library ‘%s’. Derived layers are not supported in this rule, so the layer has been skipped. The rule should be corrected to avoid this warning to be displayed.”

Preparing Your Connectivity Source

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

This chapter covers the following topics.

- [Schematic Design Elements](#) on page 85
- [Design Variables](#) on page 86
- [One-to-Many Mapping](#) on page 88
- [Defining Quick Cells for Use in Layout XL](#) on page 89
- [Properties in Layout XL](#) on page 99
- [Pins and Pin Names in Layout XL](#) on page 100

Schematic Design Elements

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

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

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

- Use consistent parameter defaults for comparable symbols.

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

 Important

Layout XL does not pass to the layout instance CDF parameters that are not evaluated (i.e., that are not AEL expressions) and have `cdfParamRec->storeDefault=nil`. These are master parameters and are expected to match for both schematic and layout master. If you require these parameters to be passed, set `cdfParamRec->storeDefault` to `t` for the schematic master parameters.

Note: When transferring information from the schematic to the layout, Layout XL flattens the schematic (i.e., expands schematic symbols into corresponding devices) if you provide an [IxViewList](#) or an [IxStopList](#) to tell Layout XL which view of the lower-level instances to use.

Layout XL also supports

- Design variables. For more information, see [Design Variables](#)
- One-to-many mapping. For more information, see [One-to-Many Mapping](#) on page 88.
- Many-to-many and many-to-one mapping.

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

Design Variables

You can use the following design variables with Layout XL.

- [Netlist Processor Expressions](#)
- [Analog Expression Language Expressions](#)
- [Simulation Design Variables](#).

Netlist Processor Expressions

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

For more information, see the [Open Simulation System Reference](#).

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

Analog Expression Language Expressions

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

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

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

Netlisting Mode

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

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

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

- Type the following commands in the CIW.

```
setShellEnvVar("CDS_Netlisting_Mode=Analog")  
cdsSetNetlistMode()
```

To check which mode is currently set,

- Type the following command in the CIW.

```
cdsGetNetlistMode()
```

For more information on `CDS_Netlisting_Mode`, see [Customizing the Simulation Environment](#) in the *Open Simulation System Reference*.

For more information on AEL expressions, see [Scope of Parameters](#) in the *Virtuoso Analog Design Environment L User Guide*.

Simulation Design Variables

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

For more information about simulation design variables, see [Design Variables and Simulation](#) in the *Virtuoso Analog Design Environment L User Guide*.

Design Constraints

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

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

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

For more information about constraint transfer, see [Constraint Transfer](#) on page 218.

One-to-Many Mapping

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

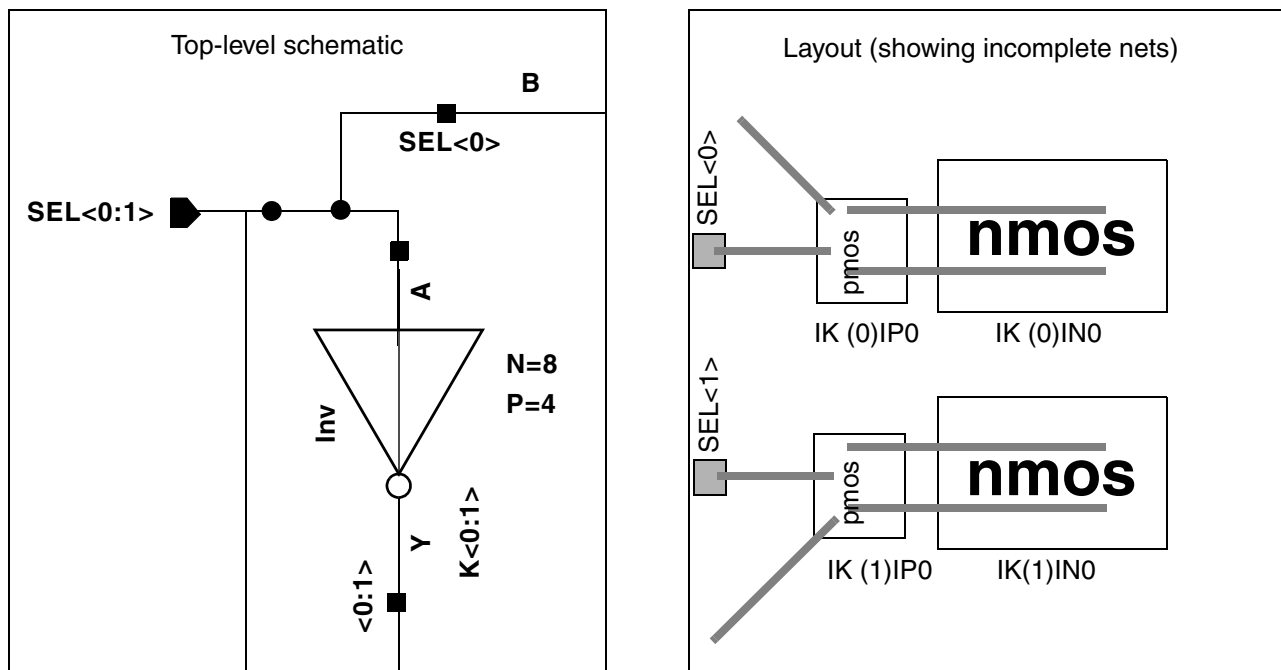
- [Defining One-to-Many Mapping with Iterated Instances and Bus Pins](#)
- The multiplication factor ([mfactor](#))
- The series-connected factor ([sfactor](#))
- [Defining a One-to-Many Device Correspondence](#)

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

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

For example, the figure below shows

- Bus pin $SEL<0:1>$ in the schematic mapped to bus pins $SEL<0>$ and $SEL<1>$ in the layout.
- Iterated instance $K<0:1>$ mapped to instances $IK(0) | P0$, $IK(0) | N0$, $IK(1) | P0$, and $IK(1) | N0$ in the layout.



Note: For more information on iterated instances, see [Adding Instances Using an Iterative Expression](#) in the *Virtuoso Schematic Editor L User Guide*.

Defining Quick Cells for Use in Layout XL

Quick Cells (QCells) are parameterized cells that are installed in a technology file using the Install QCell form, which lets you specify names, design rules, stretch handles, and layer definitions for devices. The form prompts you graphically for process design rule values that are not already defined in the technology file.

After you have created one device, it becomes much easier to create different variants because you need to supply only process rule values that do not already exist in the technology file.

For more information, see the [Virtuoso Quick Cells User Guide](#).

Creating a New Technology Library for Use with QCells

To create a new technology library for use with QCells,

1. Create a technology library from an existing technology library.
For more information, see [Creating a Library](#) on page 90.
2. Copy the symbols you need from an existing technology library.
For more information, see [Copying Symbols to the New Library](#) on page 91.
3. Remaster the symbol instances so that they match the names of the MOS devices you are going to create using Quick Cells.
For more information, see [Remastering Symbol Instances](#) on page 91.
4. By default, Layout XL places the layout view with the same cell name as the symbol in the schematic. If you want to use a layout view with a different name, you must specify the name to use in the Configure Physical Hierarchy window.
For more information, see [Specifying the Mapping for a Cell](#).
5. Install your QCells into the new technology library.
For more information, see [Installing a QCell](#) on page 92.

Creating a Library

To create a technology library,

1. Choose an ASCII technology file to be loaded into the new library by doing one of the following.
 - ☐ From the CIW menu bar, choose *Tools – Technology File Manager* and use the Technology File Toolbox to dump out an ASCII technology file from an existing library.
 - ☐ Use an existing ASCII technology file.
2. Remove from the technology file any references to layout cellviews from another library and rename any original layout views that you want to preserve.



Caution

When you install QCells into a new or existing technology library, the software either creates a new layout view or overwrites an existing layout view. To preserve any original layout views, Cadence recommends that you rename them before you begin to install QCells; for example, rename `nmoslayout` to `nmoslayout.orig`.

3. In the CIW, select *Tools – Technology File Manager* and click the *New* button.
4. Specify a name for the new technology library, turn on the *Load ASCII Technology File* option, and type in the name of the ASCII technology file you want to use.
5. Click *OK* to create the new technology library.

Copying Symbols to the New Library

To copy symbols from an existing technology library into your new library,

1. From the CIW, choose *Tools – Library Manager* and select the symbol to copy.
2. Choose *Edit – Copy* and type the name of the new library into the *To Library* field.
3. If the *Views To Copy* field is visible, make sure it is empty. You will create your layouts by installing QCells instead.
4. Switch on *Update Instances*, choose *Of New Copies Only* from the cyclic field, and click *OK*.

The Copy Problems dialog box appears, highlighting potential problems with the copy operation and letting you decide what action you want to take.

5. Click *Fix Errors* and then *OK*.

The cell is copied to your new library.

6. Repeat steps 1-5 for the other symbols to copy.

Remastering Symbol Instances

The names of the schematic symbols should match the names of the MOS devices you create using Quick Cells.

To remaster symbol instances so that they have the correct names,

1. From the CIW, choose *File – Open* and select the schematic you copied.

2. In the layout window, choose *Tools – Search*.
3. In the *Search for* field, select *inst* and click *Add Criteria*.

The *Search for* pane updates to display a number of options to let you limit the search.

4. Select *inst name* from the cyclic field and type in the name of the current symbol.
5. In the *Replace* field, select *inst name* from the cyclic field and type in the name of the MOS device to be created.
6. Click *Replace*.

The symbol master is updated.

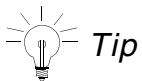
Specifying a Layout Cellview with a Different Name

By default, Layout XL places the layout view with the same cell name as the symbol in the schematic. If you want to use a layout view with a different name, you must specify the name to use in the Configure Physical Hierarchy window. For more information, see [Specifying the Mapping for a Cell](#).



Cadence recommends that you do not use differently-named cells for mixed-signal designs. For digital designs, the cells can be in different libraries but for mixed-signal designs, you must use CDF to define the parameters for the devices. Therefore the symbol and layout views must be in the same library and cell.

Installing a QCell



Tip

This section provides a brief introduction on how to install QCells into a technology library. For more information, see the [Virtuoso Quick Cells User Guide](#).

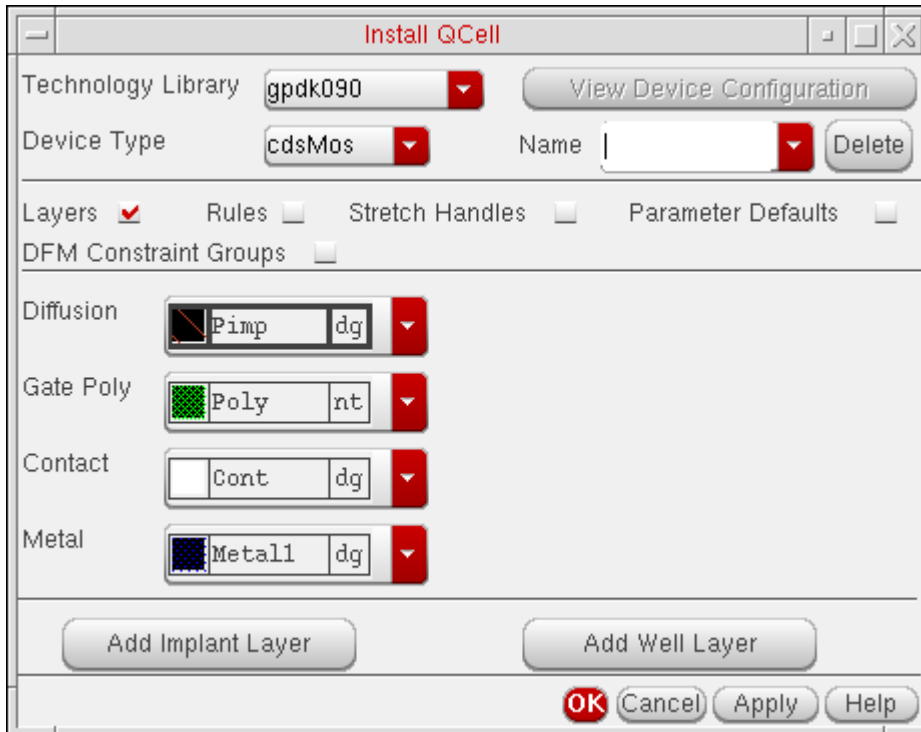
To install a QCell into the new technology library,

1. From the CIW menu bar, choose *Tools – Technology File Manager* and click *QCell*.
The Install QCell form is displayed.
2. Select your *Technology Library* from the cyclic field at the top of the form and type in the *Name* of the MOS device.

Virtuoso Layout Suite XL User Guide

Preparing Your Connectivity Source

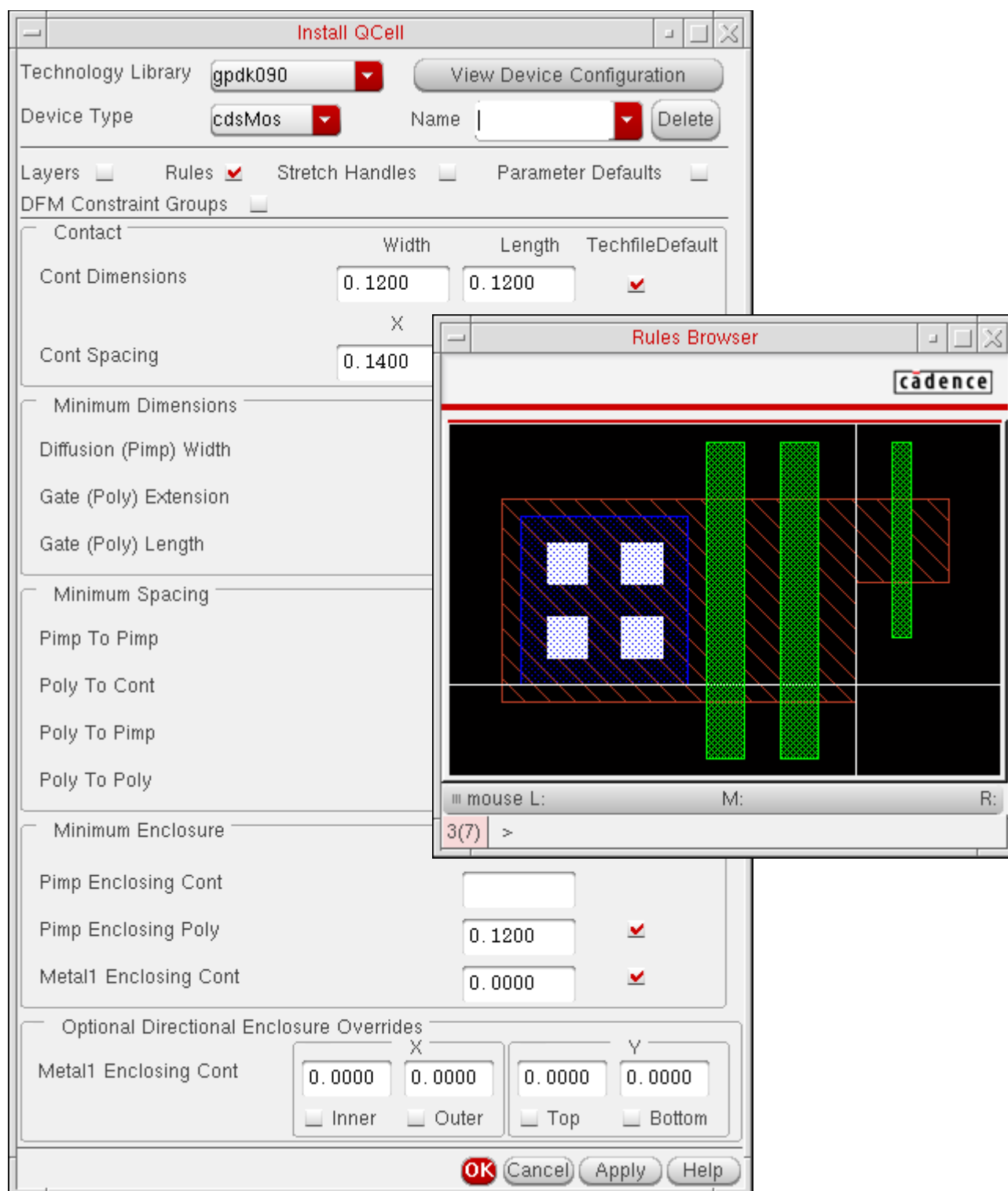
3. The *Layers* check box is checked by default. Select *Diffusion*, *Gate Poly*, *Contact*, and *Metal* layers using the cyclic fields provided.



Virtuoso Layout Suite XL User Guide

Preparing Your Connectivity Source

4. Check the *Rules* box.



The form displays the different rules options and opens a Rules Browser showing a graphical representation of a generic device using the layers you specified in the *Layers* section. The rules are taken from the technology file attached to your library.

5. If there are any rules missing, add them in the *Rules* section of the Install QCell form.

When you position the cursor in the rules fields, the Rules Browser displays an arrow to show the physical area to which the rule applies.

6. In the Install QCell form, check the *Stretch Handles* box and select the stretch handles to use.

The Rules Browser displays the stretch handles you selected.

7. In the Install QCell form, check *Parameter Defaults*.

- a. Set the *Abutment Class* to either *nmos* or *pmos* depending on whether you are creating an NMOS or PMOS device. Alternatively, you can specify a user-defined abutment class.

Note: The abutment class is the device name.

- b. Set the *Component Class* to either *NMOS* or *PMOS*.

The information from the *Component Class* field updates the component type for that device. The purpose of component types is to identify the NMOS and PMOS transistor cells and set their parameters for device chaining and folding.

Virtuoso Layout Suite XL User Guide

Preparing Your Connectivity Source

c. Specify the *MOS Type*. The entire form is now editable.

Install QCell

Technology Library: gpdK090 View Device Configuration

Device Type: cdsMos Name: NMOS Delete

Layers ☐ Rules ☐ Stretch Handles ☐ Parameter Defaults ☒

DFM Constraint Group ☐

Component Parameters

Abutment Class:

Component Class: -

MOS Type:

Fold Threshold: 0 Meter

Terminal Names

Gate: G Source: S Drain: D

CDF Parameters

Name	Type	Default	Unit
Total Width: w	string	12u	meter
Length: l	string	0.1u	meter
Fingers: fingers	string	1	

Contact Cut Spacing Method

☒ Distribute

☐ Minimum

Rows: ☒ Centered ☐ Bottom ☐ Top

Columns: ☒ Centered ☐ Towards Gate ☐ Away From Gate

Create Single Contact only ☐

OK Cancel Apply Help

The *CDF Parameters* section contains either default CDF parameters or defined CDF if present.

 Important

You can edit the *Parameter Defaults* or units in the Install QCell form, but you cannot change the parameter types once they have been applied. Use the CIW's *Tools – CDF – Edit* command to change component parameters.

- d. Set the *Unit* cyclic field to either `meter` or `micron`.

The CDF unit `meter` = `lengthMetric` and `micron` = don't use.

 Caution

You cannot reset or edit the *Unit* field.

- e. Set the values for `w`, `l`, and `fingers` as needed (e.g., `w=2u`, `l=.18u`, `fingers=1`).

8. Click *Apply* to create the QCell.
9. Repeat steps 1-8 for the other devices that need to be created.
10. Click *Save* in the Technology File Tool Box to save the technology library.

Using an Existing Technology Library with QCells

If you are using an existing technology library, remember that

- The symbols must contain CDF parameters.
- When you install QCells into a technology library, the software overwrites any existing layout of the same name. To preserve the original versions, rename the existing NMOS and PMOS layout cellviews to a different view name. For example, rename `nmoslayout` to `nmoslayout.orig`.

You can also specify a differently-named layout cellview using the Configure Physical hierarchy window. For more information, see [Specifying the Mapping for a Cell](#) on page 126.

- If layout devices are being created directly from the technology file, remove or rename them.

Setting Units

If the symbol view corresponding to the QCell has dimension units defaulting to `lengthMetric` (i.e., meter rather than micron), you must enter the values in the schematic

with the appropriate extension (e.g., 1u or 1000n or 1e-06 instead of simply 1, which would be interpreted as 1m).

The same applies when you are setting the *Folding Threshold* in the Configure Physical Hierarchy window in Component Types mode. However, using the Property Editor form to edit QCells lets you enter values both with and without the unit extension.

Note: Most PDKs have callbacks to prevent you from entering a value such as this erroneously in the schematic.

Inserting a QCell Device

To insert a QCell into your design,

1. From the layout window menu bar, choose *Create – Device*.
2. Click on the *Mos Pattern* icons to specify the type of device you want to insert.
3. Move the cursor into the layout window and click to place the device in the design.
4. Choose *Connectivity – Update – Device Correspondence*.

Important

When you insert a QCell device into your design, you must run *Connectivity – Update – Device Correspondence* to update the connectivity, otherwise the QCell you inserted will have no connectivity. For more information, see [Updating Device Correspondence](#) on page 560.

The schematic window prompts you to click on one or more symbols.

5. In the schematic window, click on the symbols whose connectivity you want to update in the layout.
6. With the cursor still in the schematic window, press `Return`.

The layout window prompts you to select the corresponding objects in the layout window.

7. In the layout window, click on the objects to associate with the selected symbols.
8. With the cursor in the layout window, press `Return`.

Important

The correspondence is defined only when you press `Return`.

A message is displayed in the CIW to indicate that the correspondence has been made.

Optimizing the Layout

Virtuoso Layout Optimize is a compaction tool designed to facilitate the hand layout of cells. Optimize is a limited version of Virtuoso Layout Migrate.

For more information, see the [*Virtuoso Layout Migrate User Guide*](#).

Properties in Layout XL

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

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

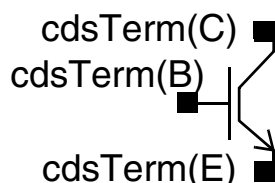
Pcells are often the most effective because you can assign the dimensions of the device at the time you generate the layout and vary the sizes of a contact each time you place the cell. For more information about pcells, see the [*Virtuoso Parameterized Cell Reference*](#).

For more information on the properties used by Layout XL, see [*Layout XL Properties*](#) on page 809.

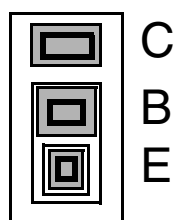
Pins and Pin Names in Layout XL

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

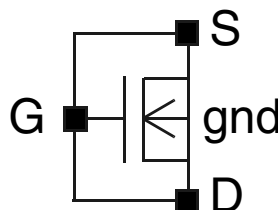
analoglib library style pin symbol



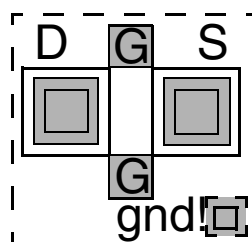
analoglib library style pin layout



basic library style pin symbol



basic library style pin layout



Extra Pins in the Symbol or Layout Views

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

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

To determine which net to connect to such a pin, Layout XL first tries to resolve the inherited net expression on the layout pin. If there is no net expression, the software looks for a property named either `sub`, `sub_inh`, or `bn` on the layout master, uses the value of the property as the substrate net name, and connects the extra terminal to that net.

Note: Do not place pins where you do not want to make a connection; for example, on a polysilicon layer that covers the gate area of a FET. For more information, see [Adding a Pin](#) on page 326.

External Connections

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

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

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

For more information about must-connect pins, strongly connected pins, weakly connected pins, and pseudoparallel connected pins, see [Using Connectivity](#) in the *Virtuoso Layout Suite L User Guide*.

To permute pins, see [permuteRule](#) on page 859.

If you are using parameterized cells and want to give them the capability for abutment, see [Device Abutment](#) on page 181.

Virtuoso Layout Suite XL User Guide

Preparing Your Connectivity Source

Configuring the Physical Hierarchy

Configure Physical Hierarchy is a physical hierarchy configuration utility that lets you specify how a layout implementation is generated from a specific schematic design. It has three modes of operation:

- *Hierarchy Configuration* mode controls how the physical hierarchy is generated from your logical design, including which logical components are to be ignored in the physical implementation and which physical views are used to implement the logical components.
- *Component Types* mode lets you view the library-level component types. In addition, this mode lets you create, edit, and remove the design-level component types, which identify the NMOS and PMOS transistor cells and set the parameters for device chaining and folding.
- *Soft Block* mode lets you configure and specify bindings for the soft blocks that will be created by the *Floorplan – Generate Physical Hierarchy* command. You select the instances to be included and define the boundary, pin, and blockage parameters for the block you want to create. You can also remove the parameters from an existing soft block to prevent it from being created when the physical hierarchy is generated.

Note: To edit an existing soft block that has already been generated in the layout view, use the *Edit Soft Blocks* command. For more information, see [Editing Soft Blocks](#).

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

LAM Files and Configuration Views

Underlying the Configure Physical Hierarchy functionality is a new Layout XL Schema in IC 6.1. It comprises two new data formats.

- For schematic and layout library data, there is a **library and attributes mapping (LAM) file** to store the symbol properties that drive layout generation, component type

information for a given library, and library and cell-level mapping information. This file is called `cph.lam` and is stored in the following location:

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

There is no graphical user interface to help you create or edit a LAM file. For more information on the LAM file syntax, see [Library and Attributes Mapping File Syntax](#).

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

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

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

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

The [Configure Physical Hierarchy](#) window is the interface used to create and edit this view.

Converting Data to Use the IC 6.1 Layout XL Schema

This section explains how you decide whether you need to convert your existing data to use the new Layout XL schema in IC 6.1; what happens to the data during the conversion; and how to run the conversion commands. It contains the following sections.

- [Do I Need to Convert Data to the Layout XL IC 6.1 Schema](#)
- [How Data is Converted to the Layout XL IC 6.1 Schema](#)
- [Converting Schematic Library and Design Data](#)
- [Converting Layout Designs](#)
- [Troubleshooting the Conversion](#)

Do I Need to Convert Data to the Layout XL IC 6.1 Schema

If your schematic library or layout design data uses any of the following features then you must convert it to the new IC 6.1 Layout XL schema.

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

```
lxMfactorSplit  
lxParamsToIgnore  
lxParamsToIgnoreForCheck  
lxRemoveDevice  
lxRounding  
lxStopList  
lxUseCell  
lxViewList
```

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

Note: These properties will not be converted if the variable `CPH_USE_SCHEM_PROP` is turned on. For more details see [Controlling CPH with Schematic Properties](#)

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

For information on how this data is converted, see [Figure 4-1](#) on page 107.

Migration Path

The migration path depends on the format of your original data.

- For **CDB** data from an **IC 5.1.41** or **previous** release, use the `cdb2oa` translator to convert your data to OpenAccess 2.2. By default, `cdb2oa` automatically converts your data to use physical configuration views.

Note: If you run `cdb2oa` with the `-disablecphuprev` option, your data is not converted to the new schema.

- For **OpenAccess 2.0** data from an **ICOA 5.1.41** release, use the `dfIIoa20222` translator to convert your data to OpenAccess 2.2. By default, `dfIIoa20222` automatically converts your data to use physical configuration views.

Note: If you run `dfIIoa20222` with the `-disableCphUpPrev` option, your data is not converted to the new schema.

- For **OpenAccess 2.2** data from an **IC 5.2.51** or **IC 6.1EA (Early Adopter)** release, you must convert your schematic library data manually using the utilities described in this section. Your layout design data is converted automatically when you open a design for the first time in Layout XL.

Automatic Mode

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

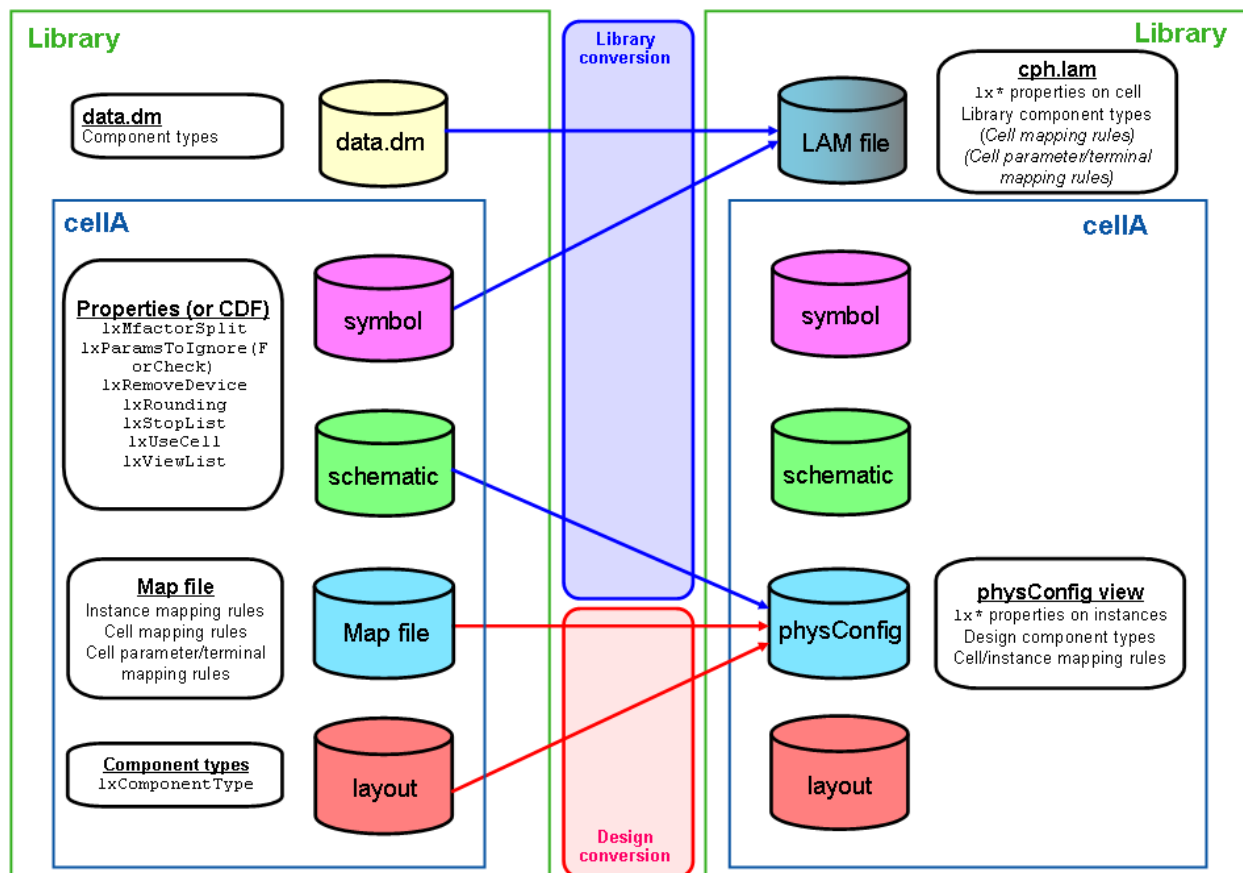
When you start Layout XL in IC 6.1, set the *Configuration* option to *Automatic* in the Startup Option form. Layout XL creates a temporary physical configuration view, handles all operations related to the physical configuration view automatically, and removes the temporary view again when it is no longer required; i.e., when you convert your data to use physical configuration views or edit the automatic configuration and save it.

For more information, see [Starting Layout XL with an Automatic Physical Configuration View](#).

How Data is Converted to the Layout XL IC 6.1 Schema

The picture below shows how legacy data is converted to use the new IC 6.1 Layout XL schema.

Figure 4-1 How Data is Converted



Data is converted in two separate operations.

- A schematic library conversion (indicated by the top three arrows in the picture above).
This can be run automatically by the translators described in [Migration Path](#), or you can run it manually using the utility described in [Converting Schematic Libraries and Designs](#).
- A layout design conversion (indicated by the bottom two arrows in the picture above).
This runs automatically when you load your design into Layout XL in IC 6.1, or you can run it manually using the [cphUprevDesign](#) SKILL function.

Converting Schematic Library and Design Data

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

The schematic library conversion converts

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

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

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

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

- The properties listed in [Figure 4-1](#) on page 107 (or their CDF equivalents) set on a symbol view for a cell.

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

- The properties listed in [Figure 4-1](#) on page 107 (or their CDF equivalents) set on a schematic view for a cell.

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

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

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

Important

If the properties for a particular schematic instance do not appear in the physical configuration view, it could be because there were errors due to cell dependencies during the `cdb2oa` translation.

If this is the case, Cadence recommends that you perform the translation to OpenAccess 2.2 first, and then convert the data to use the IC 6.1 Layout XL schema as a separate step.

You can convert schematic libraries manually using one of the following methods.

- *The **Convert Libraries to Use physConfigs** utility in the Conversion Tool Box.*
For more information, see [Converting Schematic Libraries and Designs](#).
- *The `cphUprevLibrary` SKILL function.*
For more information, see `cphUprevLibrary` in the *Virtuoso Layout Suite SKILL Reference*.

Converting Schematic Libraries and Designs

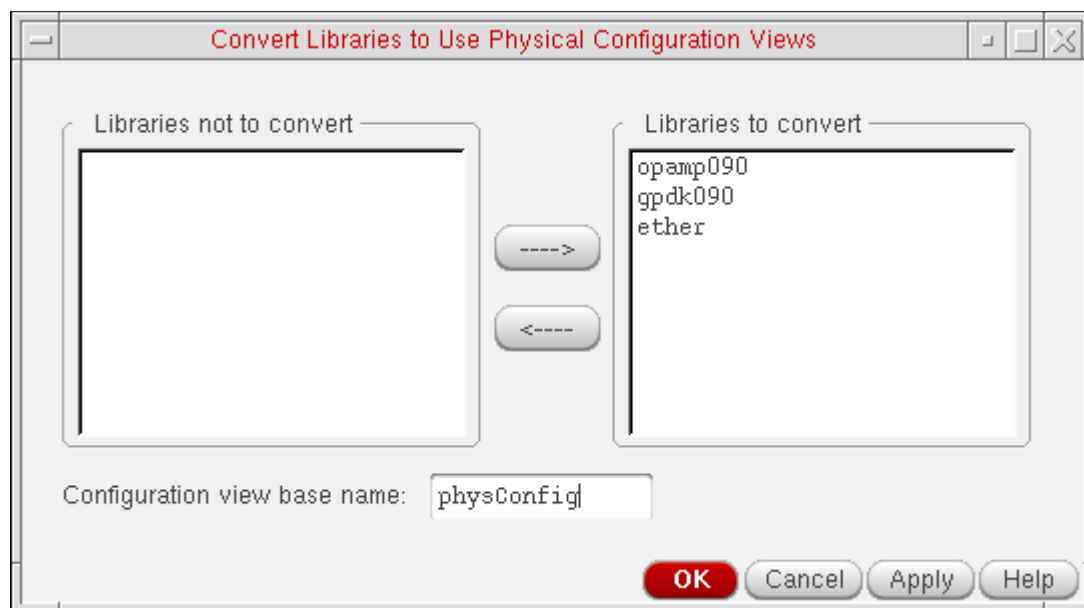
To convert your schematic libraries and designs,

1. From the CIW, choose *Tools – Conversion Toolbox*.

The Conversion Tool Box window appears.

2. Click *Convert Libraries to Use physConfigs*.

The Conversion Tool Box window appears.



The *Libraries to convert* pane lists the libraries that will be converted when you click **OK**.

The software automatically moves all the libraries defined in the current library definitions (`cds.lib`) file into this field. However, reference libraries shipped with Cadence software and libraries for which you do not have write permission are not listed.

3. Use the left arrow key to move any library you do not want to convert into the *Libraries not to convert* pane.

4. Specify the base name of the physical configuration views to be generated.

If there are multiple schematic views to be converted, the base name you specify is prepended to each schematic view name. For example, if your cell has the following schematic views before conversion:

```
schematic  
schematic1
```

Then it will have the following physical configuration views after conversion:

```
physConfig  
physConfig_schematic1
```

5. Click *OK*.

Messages issued in the CIW monitor the progress of the conversion.

Converting Layout Designs

Layout designs are converted automatically the first time you load the design into Layout XL in the IC 6.1 release. The layout design conversion converts

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

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

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

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

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

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

Important

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

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

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

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

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

Note: The following environment variables remain valid after conversion:

```
mfactorSplit  
paramsToIgnore  
paramsToIgnoreForCheck  
propsUsedToIgnoreObjs  
propsUsedToIgnoreObjsForCheck
```

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

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

Converting Layout Designs Manually

You can convert a layout design manually using the `cphUprevDesign` SKILL command. For more information, see [cphUprevDesign](#) in the *Virtuoso Layout Suite SKILL Reference*.

Troubleshooting the Conversion

For more information, see [Troubleshooting Configure Physical Hierarchy](#).

Launching Configure Physical Hierarchy

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

- From the Layout XL (or GXL) window, choose *Launch – Configure Physical Hierarchy*.

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

- When launching Layout XL (or GXL) from a schematic, check the *Open CPH* option in either the Create Physical Configuration View or Open Physical Configuration View dialog.
- From the CIW, choose *File – New* or *File – Open* and create or open a cellview of type `physConfig`.

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

For more information, see [Starting Layout XL](#).

Note on Performance

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

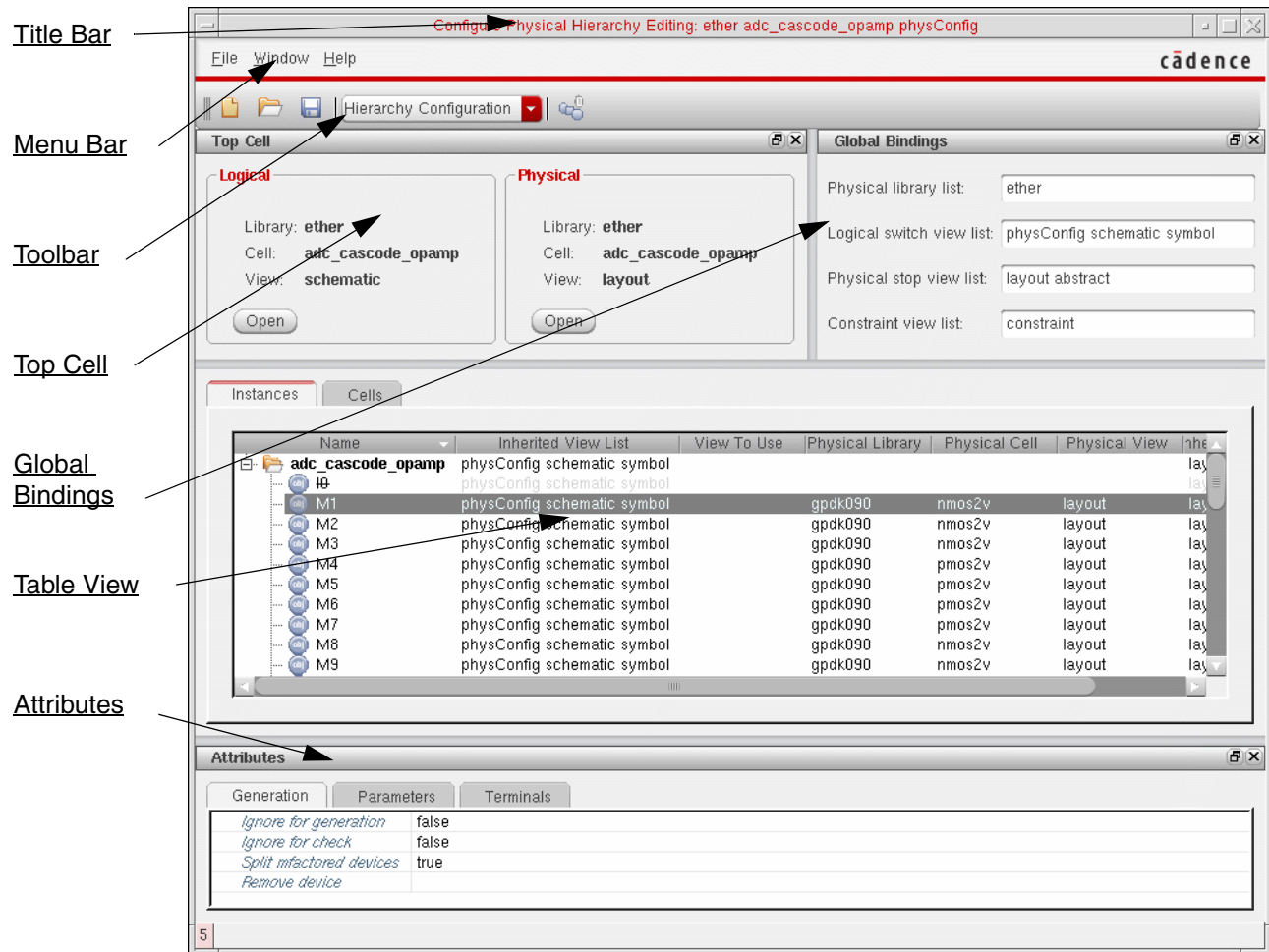
You can use the physical stop view list to limit the logical elaboration by switching on the `cphStopLogicalElabAtPhysLeaf` environment variable. This stops the elaboration of the logical design hierarchy when a node is reached which maps to one of the view names specified in the *Physical stop view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

Important

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

Configure Physical Hierarchy Window

The Configure Physical Hierarchy window contains a number of different elements. This section explains the basics; for more information, see [Configure Physical Hierarchy](#).



License Requirements

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

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

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy



Tip

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

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

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

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

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

Title Bar

The window banner indicates the library, cell, and view names of the currently loaded physical configuration view. For example,

Configure Physical Hierarchy Editing: STDLIB COMPARATOR physConfig

Menu Bar

The menu bar displays the Configure Physical Hierarchy menus.

Command	Function
File Menu	
<i>New</i>	Creates a new physical configuration view in the current window.
<i>Open</i>	Opens an existing physical configuration view.

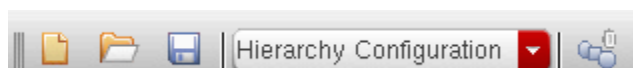
Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

Command	Function
<i>Save</i>	Saves the current physical configuration view.
<i>Save As</i>	Saves the current physical configuration view under a different name.
<i>Load Floorplan File</i>	Loads soft block parameters from a floorplan file.
<i>Save Floorplan File</i>	Saves the current soft block parameters to a floorplan file, which you can then re-use in a different Virtuoso session.
<i>Exit</i>	Closes the window. If the view contains any unsaved changes, you are prompted to save them before the window closes.
Window Menu	
<i>Panes – Top Cell</i>	Toggles the display of the Top Cell pane.
<i>Panes – Global Bindings</i>	Toggles the display of the Global Bindings pane.
<i>Panes – Attributes</i>	Toggles the display of the Attributes pane.
<i>Toolbars – File</i>	Toggles the display of the File toolbar.
Options Menu	
<i>Soft Block Options</i>	Opens the Soft Block Global Options form where you specify global settings for the pin labels in the soft blocks of your design.
Help Menu	
<i>Configure Physical Hierarchy</i>	Opens this document
<i>Cadence Documentation</i>	Opens the Cadence documentation library for the current release.

Toolbar

The *File* toolbar lets you access certain Configure Physical Hierarchy commands directly without opening any menus. As with all toolbars, you can use the handle to reposition it anywhere within the Configure Physical Hierarchy window.



Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

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

Icon	Command
	<i>File – New</i>
	<i>File – Open</i>
	<i>File – Save</i>
	<i>Choose Mode</i>
	<i>Toggle Occurrence Mode</i> <u>Instance Mode and Occurrence Mode</u>

Top Cell

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

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

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

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

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

Global Bindings

The settings in the Global Bindings pane control how the logical design is traversed and how the logical to physical correspondence for leaf cells (stop points) is made. It is a dockable window, which you can dock on either side of the main Configure Physical Hierarchy window, or leave undocked as a floating window.

- *Physical library list* is the list of libraries that are searched to find the corresponding physical cell for a given logical cell. By default, the library containing the logical cell is always searched first.
- *Logical switch view list* specifies the view names that are used to descend into a hierarchical design to find schematic views.
- *Physical stop view list* specifies the view names that are used to determine the corresponding physical view for a given logical view. When traversing a hierarchy, Configure Physical Hierarchy uses the first view it encounters with one of the specified names.

Note: The connectivity extractor does not stop at the stop view but descends into the hierarchy until it reaches the leaf symbol. This ensures that Layout XL finds all the required pins, including global nets, which are sometimes defined on the schematic view rather than on the symbol view.

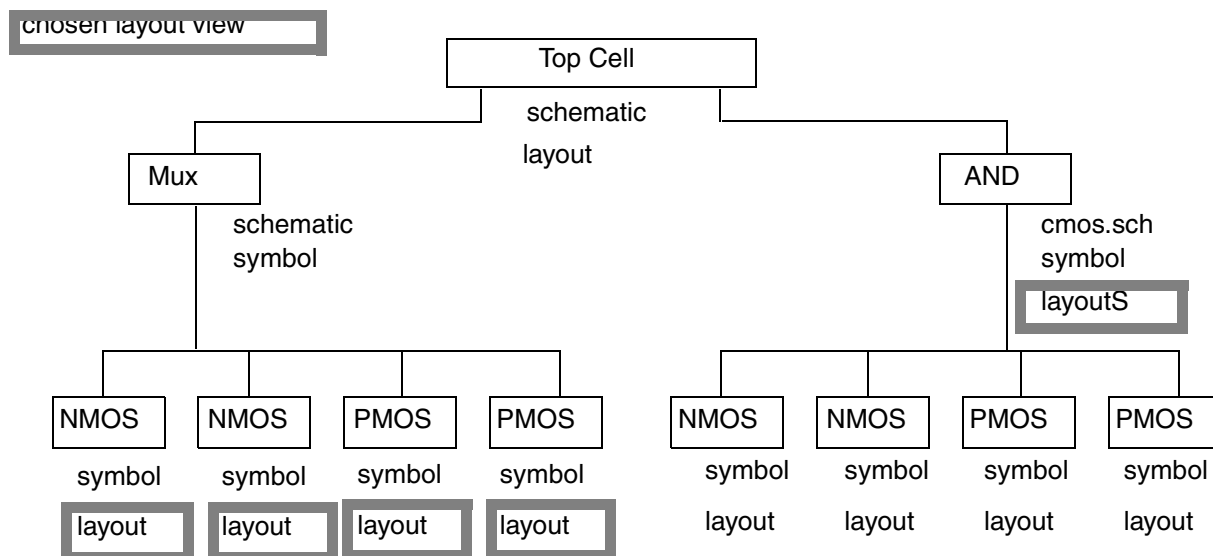
- *Constraint view list* lists the names of the views containing constraint data. This list is passed to the Virtuoso Schematic Editor when it is opened in the context of a physical configuration view, allowing it to determine whether the contents of the Constraint Manager must be updated.

Hierarchy Traversal

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

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

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



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

Table View

The table view lists the devices in the design in a format appropriate to the mode in which you are operating. It is not dockable.

- In Hierarchy Configuration mode you can choose between an *Instances* view and a *Cells* view. The former shows a hierarchical representation of the logical design – i.e., the schematic top-level cell – and how each instance in the logical design is mapped to a layout view. The latter shows a table view of the cells in the logical design.

For more information on these views, see [Hierarchy Configuration Instances Table](#).

- In Component Types mode, the table view lists the physical cells grouped by component type. If a cell is not assigned to a particular component type, it is put in the *No component type* folder.

For more information on this view, see [Component Types Cells Table](#).

- In Soft Block mode, the table view lists the instances in the design organized by cell type into *Core*, *Custom*, *Hard Blocks*, *I/Os*, and *Soft Blocks*. Instances with no physical view are also grouped together.

For more information on this view, see [Soft Block Instances Table](#).

Important

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

Attributes

Lists the attributes of the component or components currently selected in the table view. The attributes and controls are different depending on the mode in which you are operating. For more information, see

- [Hierarchy Configuration Generation Attributes](#)
- [Hierarchy Configuration Parameter Attributes](#)
- [Hierarchy Configuration Terminal Attributes](#)
- [Component Types Attributes](#)
- [Soft Block Boundary Attributes](#)
- [Soft Block I/O Pins Attributes](#)
- [Soft Block Obstructions Attributes](#)

Hierarchy Configuration Mode

Use Configure Physical Hierarchy in Hierarchy Configuration mode

- To define how the layout is generated from the schematic.

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

- To make any required changes before you update the layout from the schematic.
- To produce alternative physical implementations of an existing design, by using a different logical to physical configuration as a “what-if” experiment.

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

For more information on the graphical user interface and use model, see

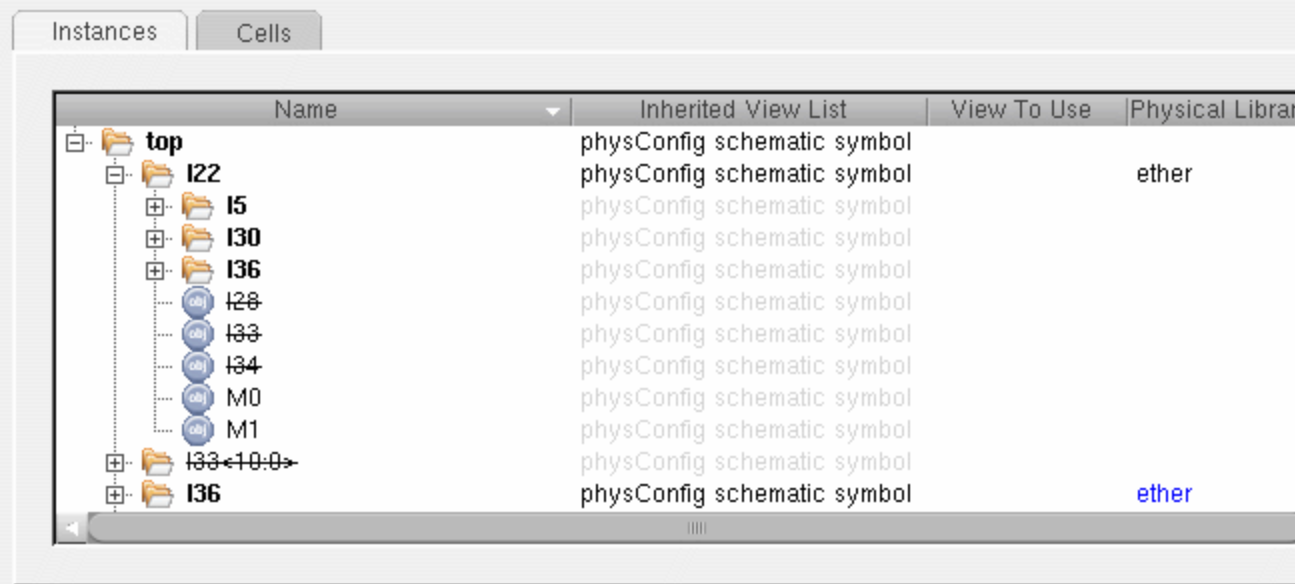
- [Hierarchy Configuration Instances Table](#)
- [Hierarchy Configuration Cells Table](#)
- [Hierarchy Configuration Generation Attributes](#)
- [Hierarchy Configuration Parameter Attributes](#)
- [Hierarchy Configuration Terminal Attributes](#)

For more information on the tasks in Hierarchy Configuration mode, see

- [Mapping Logical Cellviews to Physical Cellviews](#)
- [Mapping Parameter and Terminal Names](#)
- [Ignoring Components](#)
- [Ignoring Parameters and Terminals](#)
- [Rounding Parameter Values](#)
- [Splitting Mfactored Devices](#)

Hierarchy Configuration Instances Table

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



Name	Inherited View List	View To Use	Physical Library
top	physConfig schematic symbol		
I22	physConfig schematic symbol		ether
I5	physConfig schematic symbol		
I30	physConfig schematic symbol		
I36	physConfig schematic symbol		
I28	physConfig schematic symbol		
I33	physConfig schematic symbol		
I34	physConfig schematic symbol		
M0	physConfig schematic symbol		
M1	physConfig schematic symbol		
I33<10:0>	physConfig schematic symbol		
I36	physConfig schematic symbol		ether

For each instance in the design, the table shows the view list used to reach the schematic instance and the corresponding layout cellview found.

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

- To sort the tree alphabetically by column within each hierarchical node, click on the header of the column you want to sort. Alternate clicks sort in ascending and descending order.
- To edit an entry in a table field directly, double-click the text you want to change and choose a value from the drop-down list or type in the new setting manually.
- To edit multiple entries for a particular column,
 - a. Select the rows you want to edit.
 - b. Click the right mouse button on the column you want to change and choose *Edit multiple <fieldName>*.
 - c. Choose a value from the drop-down list or type in the new setting manually and press **Return**.

The value is updated for all the selected rows.

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

- To make an instance a stop point in the hierarchy traversal, click the right mouse button and choose *Set/clear stop point*. A 'STOP' icon against the schematic instance indicates that it is to be regarded as a leaf instance. To clear the stop point, select the instance and choose *Set/clear stop point* again.
- To force hierarchy traversal to proceed beyond a leaf instance, click the right mouse button and choose *Set/remove force descend*. A 'GO' icon indicates that the selected instance will now be traversed even if it would normally be a leaf instance. To clear the force descend, select the instance and choose *Set/remove force descend* again.

Note: You can set the force descend only on instances that have a physical binding.

- To specify how the data is presented in the tree, click the right mouse button and choose *Options*. For information on the Options form, see [The Navigator Options Form](#) in the *Virtuoso Schematic Editor XL User Guide*.

For more information on the functionality and display of this table, see

- [Columns in the Hierarchy Configuration Instances Table](#)
- [Color Coding in the Hierarchy Configuration Instances Table](#)
- [Context Menu for the Hierarchy Configuration Instances Table](#)

Instance Mode and Occurrence Mode

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

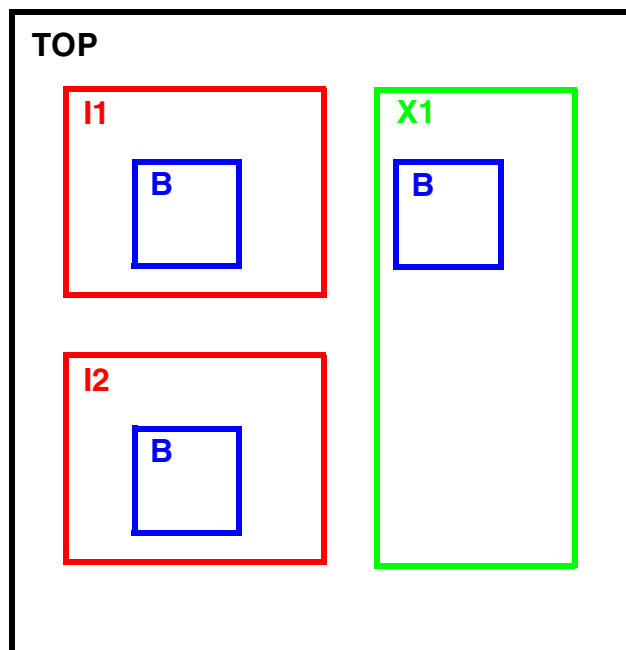
- Use Instance mode to change a particular setting (for example, the physical binding) or attribute for all identical instances by changing the setting on just one of the instances.
- Use Occurrence mode to change a setting or attribute for only a single occurrence of a particular instance; for example,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

You switch between them using the toggle button in the toolbar.



When the icon is depressed, you are working in Occurrence mode. When it is not, you are working in Instance mode. (You can check the current mode by hovering over the button to display a tooltip.)

Consider the following simple example.



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

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

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

Note: To make the same change to all three instances of **B**, edit the setting for the master cellview in the *Cells* table. See [Hierarchy Configuration Cells Table](#) for more information.

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

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

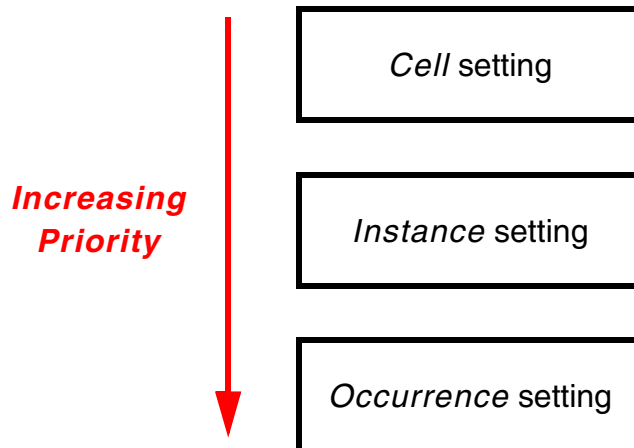
The physical binding is not applied to either **I2:B** or to **X1:B** because these are different occurrences of instance **B**.

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

Inheritance of Settings

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



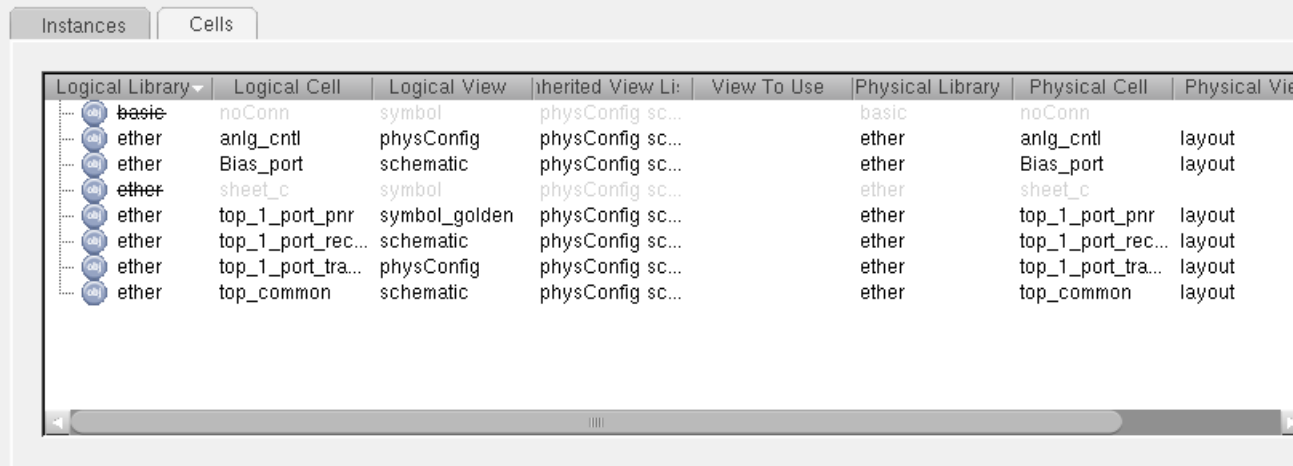
If you delete a higher priority setting, the item in question inherits the setting from the next level up the chain.

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

Instances		Cells				
Name	Inherited View List	View To Use	Physical Library	Physical Cell	Physical View	Inherited Stop Lis
I9	physConfig schematic		CellsLib	INV_C	layout	layout
I10	physConfig schematic		CellsLib	INV_C	abstract	layout abstract
I11	physConfig schematic		CellsLib	INV_C	layout	layout abstract
I12	physConfig schematic		CellsLib	NAND3_B	layout	layout abstract
I13	physConfig schematic		CellsLib	NAND2_C	layout	layout abstract
I16	physConfig schematic		design	block	layout	layout abstract
I17	physConfig schematic		design	block1	layout	layout abstract
I18	physConfig schematic		design	block3	layout	layout abstract

Hierarchy Configuration Cells Table

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



The screenshot shows the 'Cells' tab in the Virtuoso interface. It displays a table with columns: Logical Library, Logical Cell, Logical View, Inherited View Li..., View To Use, Physical Library, Physical Cell, and Physical Vie. The table lists various cellviews and their corresponding physical layout cellviews.

Logical Library	Logical Cell	Logical View	Inherited View Li...	View To Use	Physical Library	Physical Cell	Physical Vie
basic	noConn	symbol	physConfig sc...		basic	noConn	
ether	anlg_cntl	physConfig	physConfig sc...		ether	anlg_cntl	layout
ether	Bias_port	schematic	physConfig sc...		ether	Bias_port	layout
ether	sheet_c	symbol	physConfig sc...		ether	sheet_c	
ether	top_1_port_pnr	symbol_golden	physConfig sc...		ether	top_1_port_pnr	layout
ether	top_1_port_rec...	schematic	physConfig sc...		ether	top_1_port_rec...	layout
ether	top_1_port_tra...	physConfig	physConfig sc...		ether	top_1_port_tra...	layout
ether	top_common	schematic	physConfig sc...		ether	top_common	layout

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

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

- To sort the tree alphabetically by column within each hierarchical node, click on the header of the column you want to sort. Alternate clicks sort in ascending and descending order.
- To edit an entry in a table field directly, double-click the text you want to change and choose a value from the drop-down list or type in the new setting manually.
- To edit multiple entries for a particular column,
 - a. Select the rows you want to edit.
 - b. Click the right mouse button on the column you want to change and choose *Edit multiple <fieldName>*.
 - c. Choose a value from the drop-down list or type in the new setting manually and press **Return**.

The value is updated for all the selected rows.

For more detailed information on the functionality and display of this table, see

- [Columns in the Hierarchy Configuration Cells Table](#)
- [Color Coding in the Hierarchy Configuration Cells Table](#)
- [Context Menu for the Hierarchy Configuration Cells Table](#)

Mapping Logical Cellviews to Physical Cellviews

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

- [Specifying the Mapping for a Cell](#)
- [Specifying the Mapping for an Instance or Occurrence](#)

Specifying the Mapping for a Cell

To specify the logical to physical mapping for a cell,

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

The Configure Physical Hierarchy window is displayed.

Note: For alternative ways of opening the Configure Physical Hierarchy window, see [Launching Configure Physical Hierarchy](#).

2. In the Configure Physical Hierarchy window, click on the *Cells* tab.
3. Click on the row containing the cell for which you want to change the mapping.

The row is highlighted.

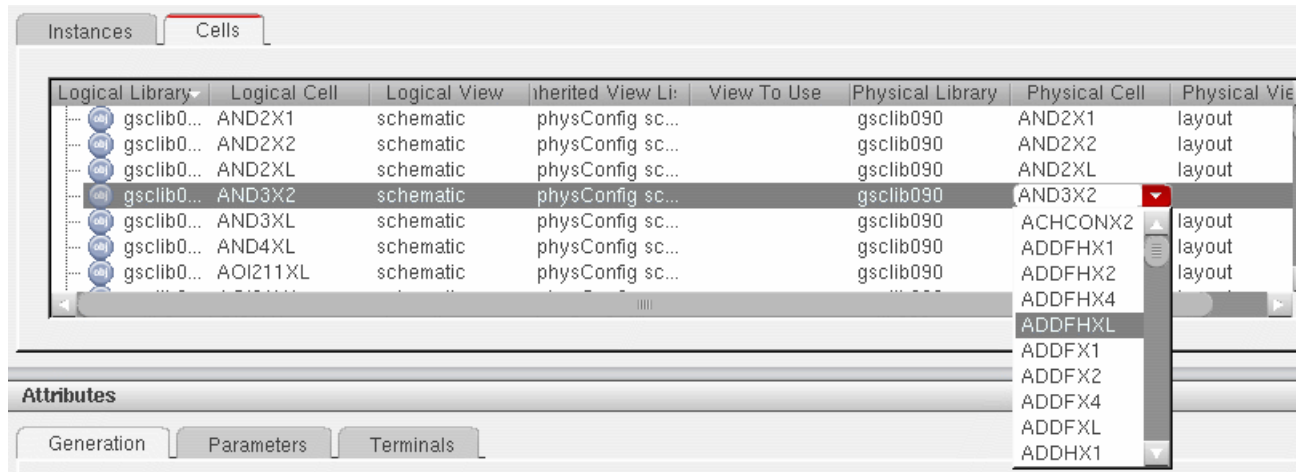
4. Double-click on the text in the *Physical Cell* column.

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

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

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



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

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

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

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

For information on how to override this setting for an individual instance, see [Specifying the Mapping for an Instance or Occurrence](#).

Specifying the Mapping for an Instance or Occurrence

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

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

The row is highlighted.

3. Do one of the following

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



Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

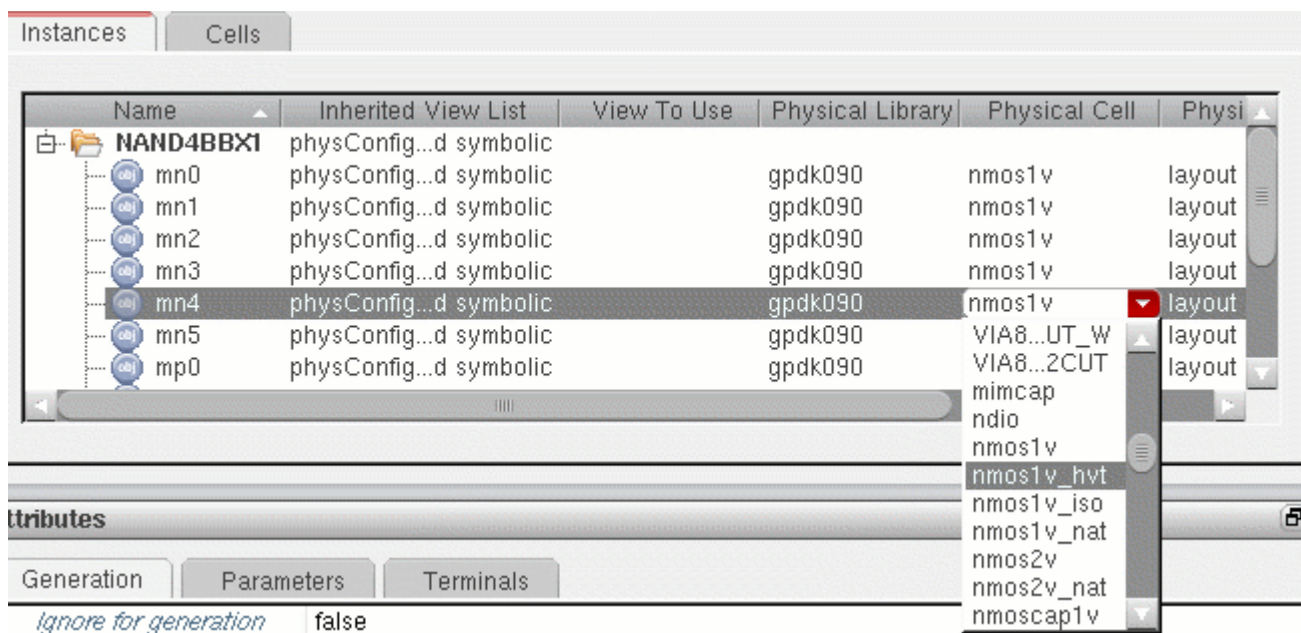
- ❑ To specify the mapping for an individual occurrence, make sure Occurrence mode is turned on.

For more information on Occurrence mode, see [Instance Mode and Occurrence Mode](#).

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

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

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

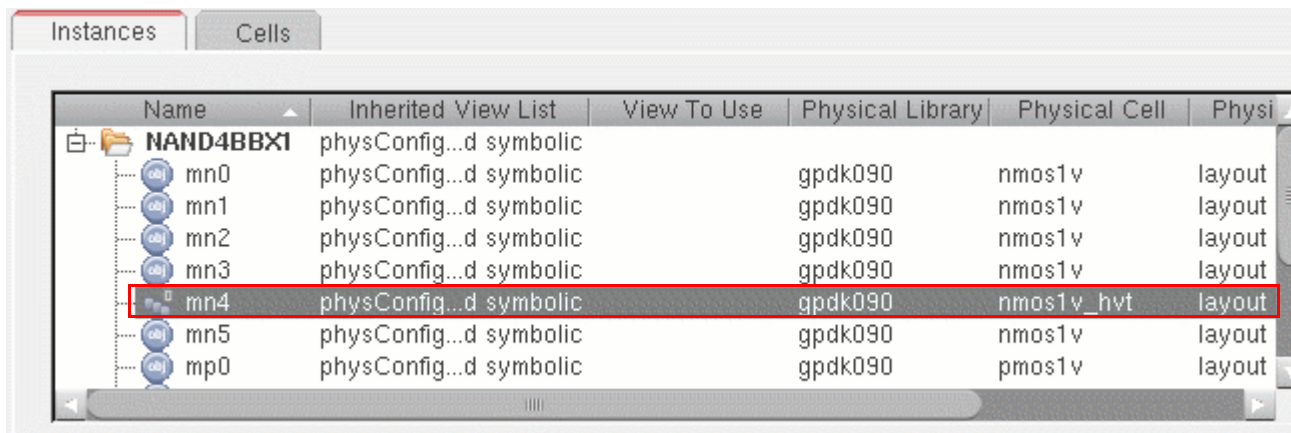


6. In the same way, select the *Physical View* to use and press `Return` to confirm the setting.

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

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



Name	Inherited View List	View To Use	Physical Library	Physical Cell	Physi
NAND4BBX1	physConfig...d symbolic				
mn0	physConfig...d symbolic	gpdK090	nmos1v	layout	
mn1	physConfig...d symbolic	gpdK090	nmos1v	layout	
mn2	physConfig...d symbolic	gpdK090	nmos1v	layout	
mn3	physConfig...d symbolic	gpdK090	nmos1v	layout	
mn4	physConfig...d symbolic	gpdK090	nmos1v_hvt	layout	
mn5	physConfig...d symbolic	gpdK090	nmos1v	layout	
mp0	physConfig...d symbolic	gpdK090	pmos1v	layout	

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

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

Mapping Parameter and Terminal Names

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

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



Tip

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

Mapping a Parameter Name

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

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

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

3. In the *Parameters* tab, type the mapping you want into the *Name mapping* field.

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

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

```
l L ; w W ;
```



Attributes	
Generation	Parameters
<i>Ignore for generation</i>	
<i>Ignore for check</i>	
Name mapping	l L ; w W ;
<i>Rounding</i>	

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

The settings are saved for all instances of the cell in question. The specified parameter names are mapped accordingly the next time you update the layout view.

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



Attributes	
Generation	Parameters
<i>Ignore for generation</i>	param1 param2
<i>Ignore for check</i>	param0
Name mapping	l L ; w W ;
<i>Rounding</i>	

Revert to default

Ignoring Components

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

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

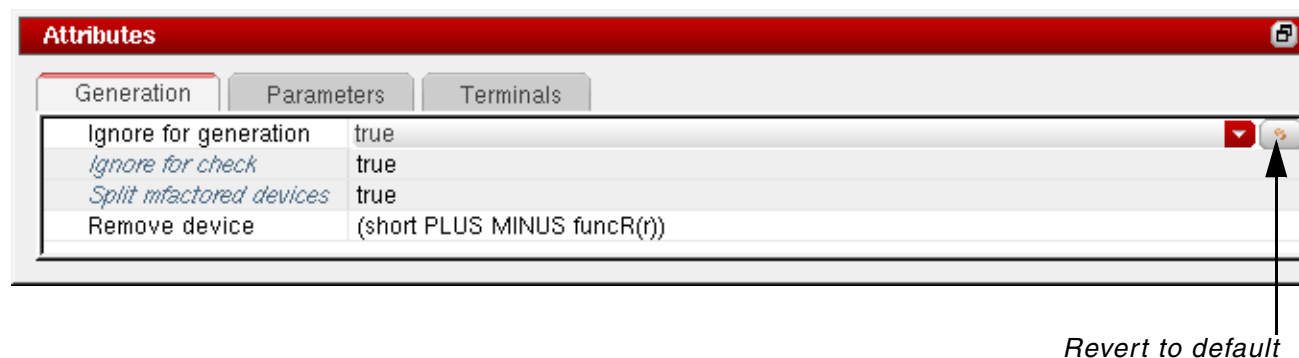
Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

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

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

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



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

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

Inheritance Rules for Ignore Properties

Cell	Instance	Instance is ignored?
No ignore property	<code>ignore = true</code>	Yes.
No ignore property	<code>ignore = true</code> <code>lvsIgnore = false</code>	Yes, because <code>lvsIgnore</code> OR <code>ignore => true</code>
<code>ignore = true</code>	No ignore property	Yes, because it inherits the <code>ignore</code> property setting from the cell

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

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

For more information, see

- [Ignoring a Cell for Generation or Check](#)
- [Ignoring an Instance or Occurrence for Generation or Check](#)

Ignoring a Cell for Generation or Check

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

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

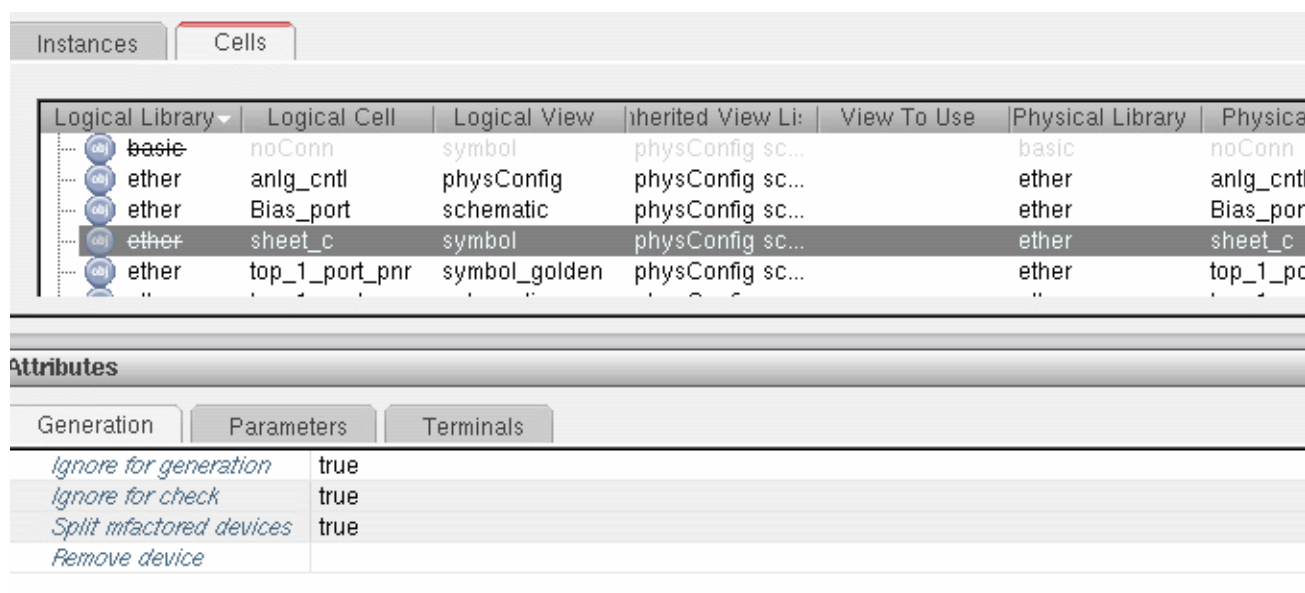
The row is highlighted and the options in the Attributes pane are enabled.

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

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

The cell name is now displayed with a strikethrough.



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

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

The ignore properties for the cell in question are saved.

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

For information on the inheritance rules for ignore values on cells and instances, see [Inheritance Rules for Ignore Properties](#).

Ignoring an Instance or Occurrence for Generation or Check

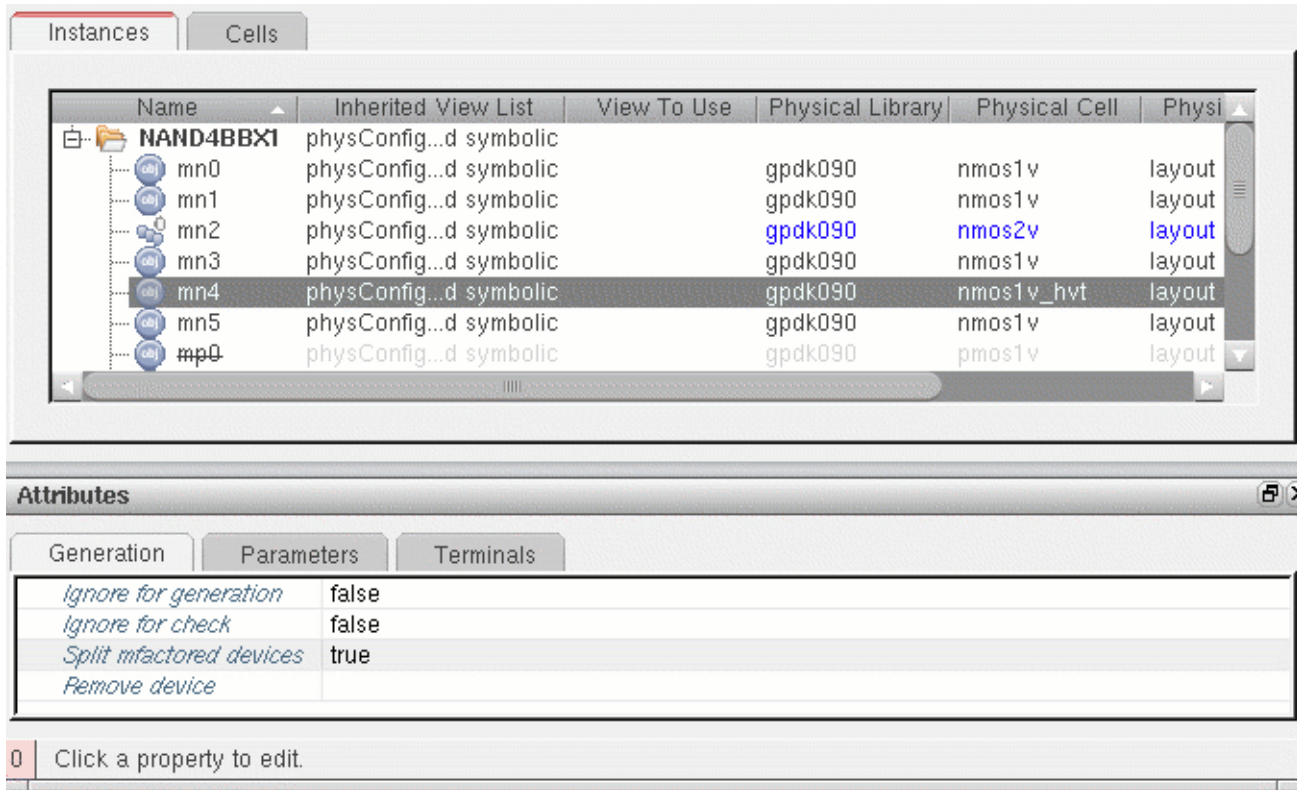
To specify that an instance is not to be generated in the layout view or checked by the *Check Against Source* command,

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

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

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



3. Do one of the following.

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



- ☐ To specify the ignore property for an individual occurrence, make sure Occurrence mode is turned on.

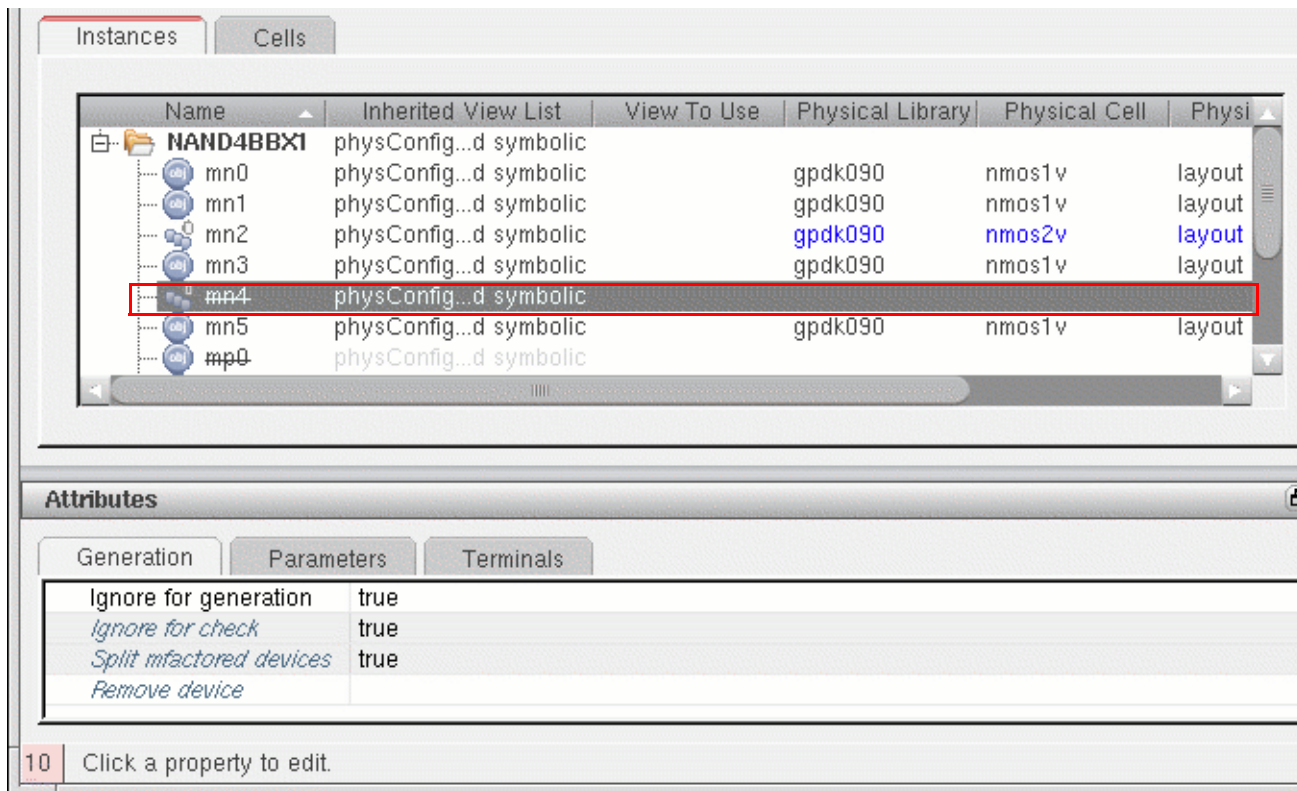
For more information on Occurrence mode, see [Instance Mode and Occurrence Mode](#).

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

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

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



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

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

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

For information on the inheritance rules for ignore values on cells and instances, see [Inheritance Rules for Ignore Properties](#).

Ignoring Parasitic Devices

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

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

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

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



If the Diva LVS `removeDevice` rule and the `Remove device` option are set on the same instance or master, running Diva LVS causes an error.

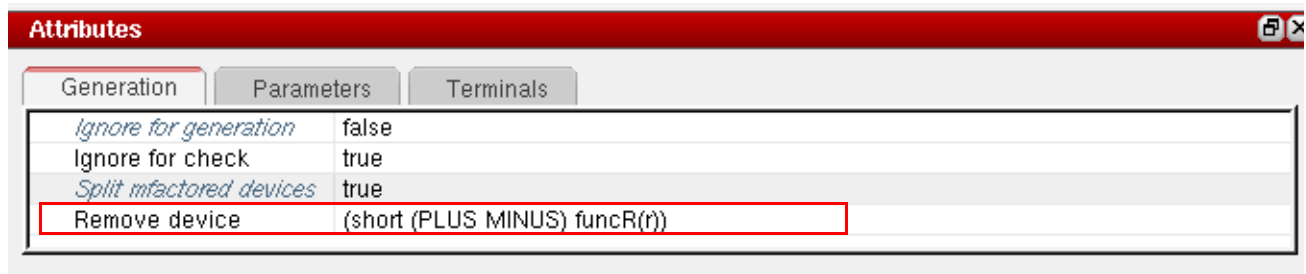
To ignore parasitic devices,

1. In the Configure Physical Hierarchy window, click on the *Instances* tab.

2. Click on the row containing the instance you want to ignore.

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

3. In the *Generation* tab, specify the *Remove device* argument in the text field provided.



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

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

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

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

The `funcR` SKILL function must already be defined in your in your `.cdsinit` or `libInit.il` file. Its first parameter is `r` and must be an instance or cell parameter

of the instance that you are testing for removal (defining, for example, the resistance value for a resistor).

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

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

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

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

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

Rules for Remove Device Handling

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

Rules for Net Merging during Remove Device

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

1. Global nets or nets connected to I/O pins always survive.
2. If there are no global or external nets, the net at a higher level of hierarchy survives; for example, `net1` survives over `/I1/net2`.
3. If the nets are at the same level of hierarchy, the net with the shortest name survives; for example, `netA` survives over `netBCD`.
4. If the nets are at the same level of hierarchy and their names are of the same length, alphabetical order is used; for example, `netM` survives over `netN`.

Ignoring Parameters and Terminals

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

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

- [Ignoring a Parameter on a Cellview](#)
- [Ignoring a Parameter on an Instance or Occurrence](#)

Ignoring a Parameter on a Cellview

To ignore a specified cellview parameter,

1. In the Configure Physical Hierarchy window, click on the *Cells* tab.
2. Click on the row containing the cell with the parameters you want to ignore.
The row is highlighted and the options in the Attributes pane are enabled.
3. In the *Parameters* tab, type the name of any parameter to be ignored by layout generation in the *Ignore for generation* field.



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

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

For information on how to override this setting for an instance or occurrence, see [Ignoring a Parameter on an Instance or Occurrence](#).

Ignoring a Parameter on an Instance or Occurrence

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

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

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

3. Do one of the following.

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



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

For more information on Occurrence mode, see [Instance Mode and Occurrence Mode](#).

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



Attributes		
Generation	Parameters	Terminals
Ignore for generation	param1 param2	
Ignore for check	param0	
Rounding	(w 0.05 round)	

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

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

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

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

Rounding Parameter Values

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

- [Rounding Cellview Parameter Values](#)
- [Rounding Instance or Occurrence Parameter Values](#)



Tip

In previous releases, you could use the equivalent `lxRounding` property to set a tolerance when Layout XL compared parameter values during the *Generate All From Source*, *Check Against Source*, *Update Layout Parameters*, and *Update Schematic Parameters* commands. To do this in the current release, use the `paramTolerance` environment variable instead.

Rounding Cellview Parameter Values

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

1. In the Configure Physical Hierarchy window, click on the *Cells* tab.
2. Click on the row containing the cell with the parameter value to be rounded.
The row is highlighted and the options in the Attributes pane are enabled.
3. In the *Parameters* tab, type the *Rounding* argument you need in the field provided.

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



For more information on the syntax, see [Rounding Syntax](#).

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

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

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

For information on how to override this setting for an individual instance, see [Rounding Instance or Occurrence Parameter Values](#).

Rounding Instance or Occurrence Parameter Values

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

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

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

3. Do one of the following.

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

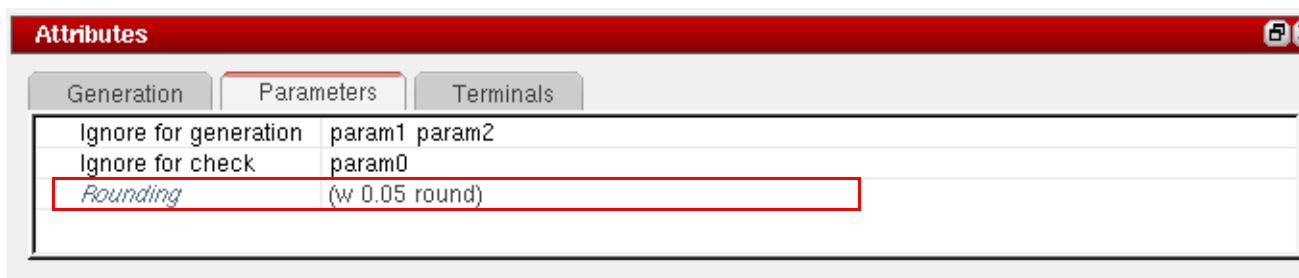


- ☐ To specify the rounding option for an individual occurrence, ensure Occurrence mode is turned on.

For more information on Occurrence mode, see [Instance Mode and Occurrence Mode](#).

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

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



For more information on the syntax, see [Rounding Syntax](#).

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

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

Rounding Syntax

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

paramName

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

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

value

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

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

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

keyword

Specifies the rounding method used. Allowed values are

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

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

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

Use the following identifiers to specify the rounding unit.

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

Splitting Mfactored Devices

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

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

To set or unset this option,

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

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

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



The screenshot shows the 'Attributes' dialog box with the 'Generation' tab selected. The 'Split mfactored devices' option is checked, and the 'Remove device' option is highlighted with a red box.

Attribute	Value
Ignore for generation	false
Ignore for check	false
Split mfactored devices	true
Remove device	

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

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

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

Controlling CPH with Schematic Properties

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

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

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

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

Note: The default value of “CPH_USE_SCHEM_PROP” is subject to change in future releases.

Note: If the evaluation of property fails, the unevaluated property will be used.

Impact in handling incremental changes in schematic properties

- Changes to the `lxProperties` made through the property editor, the db skill interface or CDF callbacks impact the active CPH/VXL session.

- For the CDF Editor, only the base CDF Parameter modification impact the active CPH when they are saved.

Impact on subConfiguration

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

Impact of changes in dependent properties

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

Impact on lxParamsToIgnore and lxParamsToIgnoreForCheck look-up

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

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

Component Types Mode

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

- *Component class* (one of *PMOS*, *NMOS*, *STDCELL*, *STDSUBCONT*, or *FILLER*)
- *Width parameter name*, *Folding threshold*, *Active layer* and *View* names
- Names defined for *Source*, *Gate*, *Drain*, and *Bulk* terminals

For more information on these attributes, see [Component Types Cells Table](#).

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

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

Application	Operation	Comments
Layout XL	Device folding	<i>Generate All From Source</i> needs the folding threshold to be set for instances <i>Generate All From Source</i> , <i>Generate Selected From Source</i> , and interactive folding require PMOS and NMOS transistors to be identified as such.
	Generate layout	<i>Generate All From Source</i> , <i>Generate Selected From Source</i> , and interactive chaining require PMOS and NMOS transistors to be identified as such.
	Determination of pseudoparallel nets	Uses the component type to access the width parameter, which in turn is used to confirm that the instances are identical.
Custom Digital Placer	Placement of components in rows	Uses the <i>Component class</i> to correctly identify standard cells (<i>STDCELL</i>), substrate contacts (<i>STDSUBCONT</i>), and filler cells (<i>FILLER</i>) and place them accordingly in the rows.

Library Component Types and Design Component Types

There are two categories of component types.

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

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

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

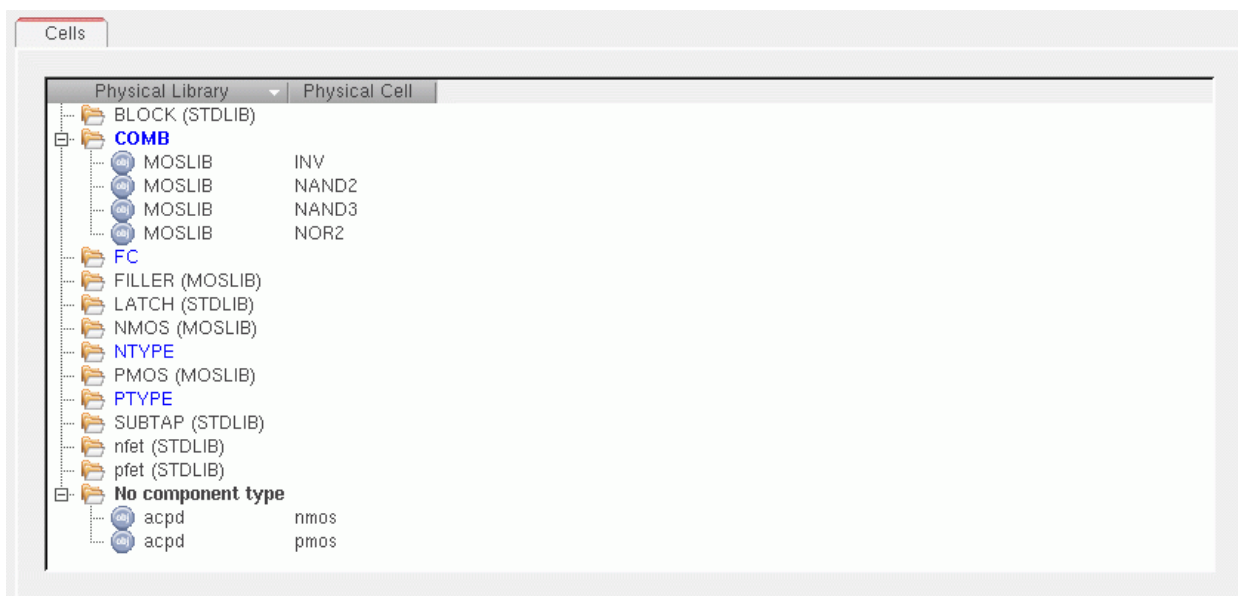
To define a new library-level component type, or modify or remove an existing one, you need to manually edit the LAM file for the design. For more information on LAM files, see [LAM Files and Configuration Views](#).

- A *design-level* component type applies only to the cells in your design. Where there is a conflict, a design-level component type overrides the library-level component type. You can add, modify, and remove design-level component types using the *Edit Component Types* command. This brings up the Configure Physical Hierarchy window in [Component Types Cells Table](#) mode.

For information on the attributes you can set for each component type, see [Component Types Attributes](#).

Component Types Cells Table

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



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

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

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

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

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

- To add a new component type, click the right mouse button and choose *Add component type*. Use the Create Component Type dialog to specify a name for the new component type. See [Defining a Design-Level Component Type](#).
- To remove a component type, select the component type you want to remove, click the right mouse button and choose *Remove component type*. You can remove only design-level component types. Evicted cellviews are put into the *No component type* folder. See [Removing a Component Type](#).
- To move cells between component types, select the cells to move, click the right mouse button and choose *Move cells*. Use the Move Cells dialog to select a new component type for the selected cells. See [Defining a Design-Level Component Type](#).

Resolving Conflicts in Component Type Names

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

```
lib list: lib1, lib2, lib3

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

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

Conflicting component type settings are resolved as indicated below.

```
lib list: lib1, lib2, lib3

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

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

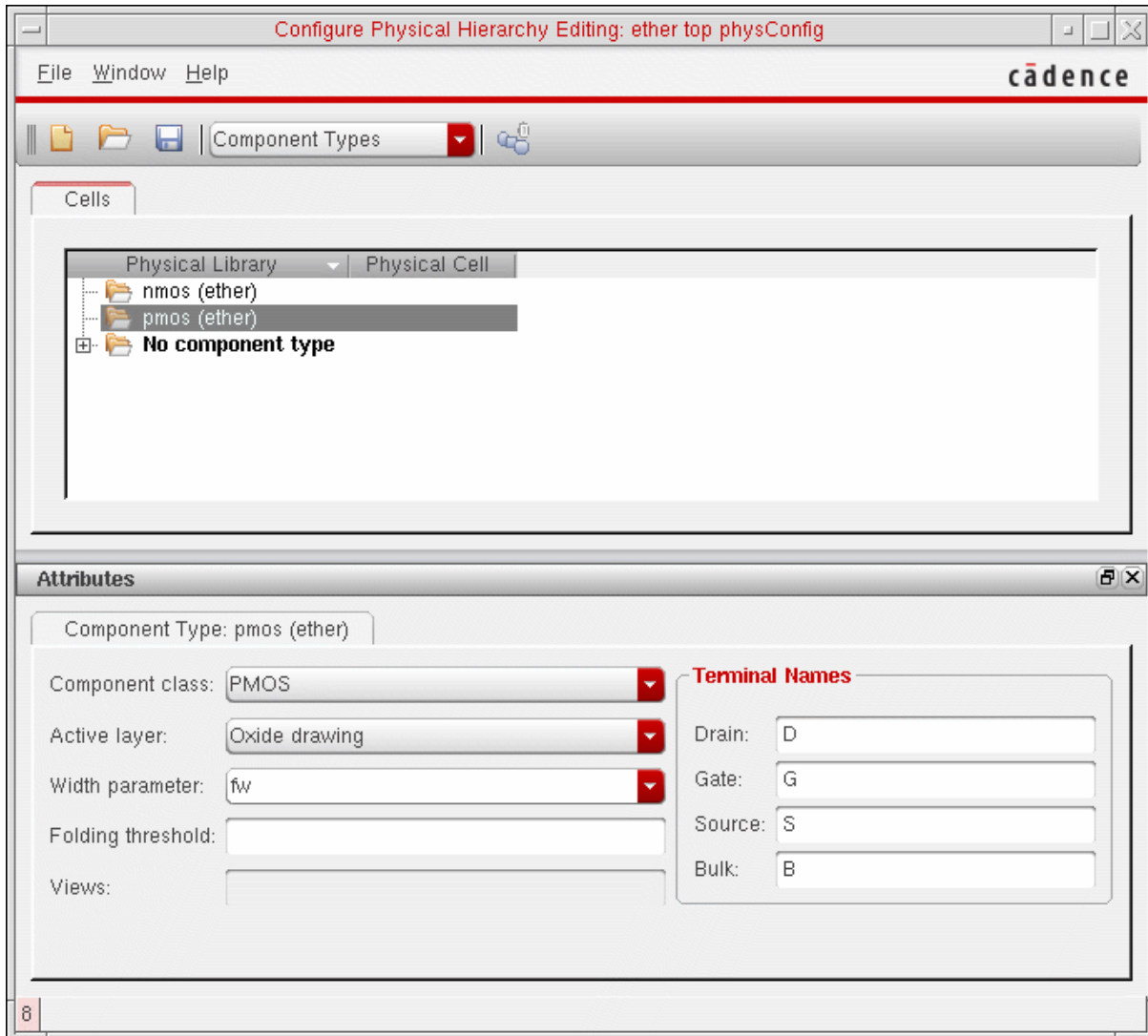
To edit the attributes for a particular component type, you must select that component type in the table view. The settings you make in the Attributes pane are applied to all the cells assigned to that component type.

Defining a Design-Level Component Type

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

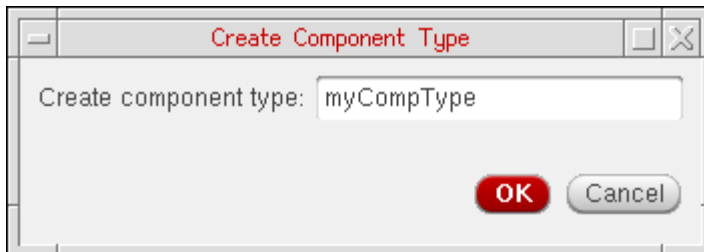
1. From the layout window menu bar, choose *Edit – Component Types*.

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



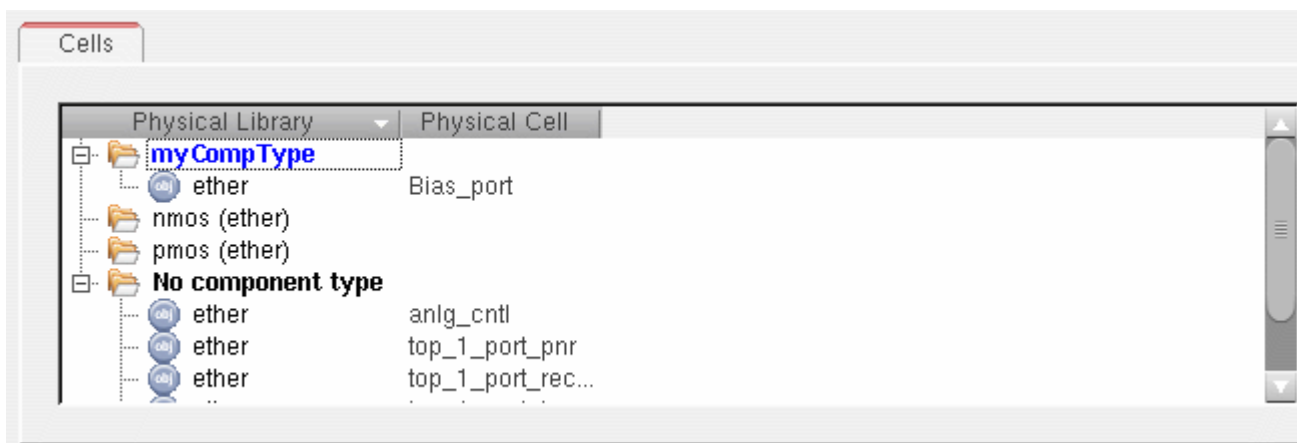
2. In the *Cells* table, click the right mouse button and choose *Add component type*.

The Create Component Type dialog is displayed.



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

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



4. Select the new component type in the table.

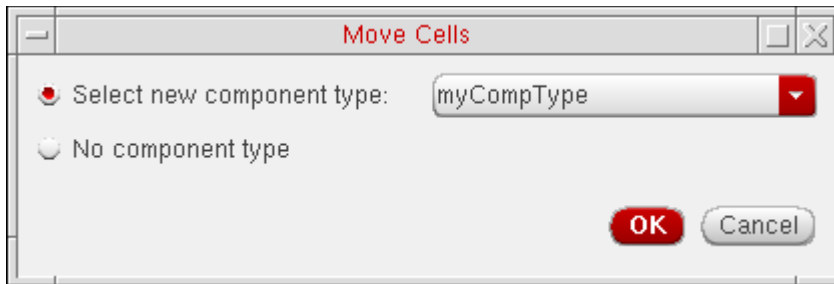
The options in the Attributes pane are enabled.

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

For more information, see [Component Types Attributes](#).

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

The Move Cells dialog is displayed.



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

The selected cells are moved to the specified component type.

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

Modifying a Component Type



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

To modify a component type,

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

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

2. Select the component type you want to modify.

The options in the Attributes pane are enabled.

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

For more information, see [Component Types Attributes](#).

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

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

Removing a Component Type

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

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

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

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

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

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

Soft Block Mode

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

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

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

- ☐ Running `verilog2oa`
- ☐ When importing another verilog design into SOC Encounter

For more information on the environment variable, see [`createImplicitBusTerminals`](#).

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

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

For more information on bus annotation in abstract generator, see [Bus Annotation in Abstract Generator](#).

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

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

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

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

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

Important

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

For more information on how to define soft blocks in the Configure Physical Hierarchy window, see

- [Setting the Cell Type](#)
- [Defining Soft Block Parameters](#)
 - [Specifying the Boundary](#)
 - [Specifying I/O Pins](#)

□ Specifying Obstructions

■ Loading Soft Block Parameters from Another Cellview

■ Saving Soft Block Parameters to a Floorplan File

■ Generating the Physical Hierarchy

You can also create and clear a manual physical binding for a selected component. For more information, see

■ Creating a Physical Binding

Soft Block Instances Table

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

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

Column Name	Description
<i>Inherited View List</i>	Shows the logical switch view list used to derive the leaf cells in the logical hierarchy.
<i>Physical Library</i>	Shows the name of the library containing the physical representation of the logical leaf cell.
<i>Physical Cell</i>	Shows the physical equivalent of the logical leaf cell.
<i>Physical View</i>	Shows the name of the physical view.

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

<i>Block Type</i>	Specifies the type of block to be generated, either CUSTOM or DIGITAL. This influences how the place and route boundary and pins in the design are snapped during level-1 editing. ■ CUSTOM specifies that the pin edges and the place and route boundary are snapped to the manufacturing grid. ■ DIGITAL specifies that the pin centers are snapped to the routing grid and the place and route boundary is snapped to the placement grid.
<i>Type</i>	Specifies whether the view that will be generated for a soft block is a LAYOUT (the default) or an ABSTRACT. ■ LAYOUT specifies a layout view that contains instances, connectivity, pins, place and route boundary and other detailed information, generated from the schematic view by the <i>Generate Physical Hierarchy</i> command. Use this setting if there are physical representations for the logical cells at lower levels of the hierarchy. ■ ABSTRACT specifies a special type of layout view containing only pins and place and route boundary based on symbol information, generated using the <i>Generate Abstract From Symbol</i> command. Use this setting if you do not have detailed physical views for the lower levels of the hierarchy but you want to start the floorplan by generating abstract views. Note: This field is blank for all instances other than soft blocks that exist already or instances for which soft block parameters have been defined.
<i>Inherited Stop List</i>	Shows the physical view list used to determine the corresponding physical view for a given logical view.

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

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

- To sort the tree alphabetically by a particular column within each node, click on the header of the column you want to sort. Alternate clicks sort in ascending and descending order.

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

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

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

Soft Block Table Folders

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

Instances

Name	Inherited View List	Physical Library	Physical Cell	Physical View	Block Type	Type
Top	physConfig schematic symbol design	top_level	layout	CUSTOM	LAY...	
Core						
Custom						
Hard blocks						
IO						
No Physical						
Soft blocks						
I18	physConfig schematic symbol design	block3	layout	CUSTOM	LAY...	
Core						
Custom						
Hard blocks						
IO						
No Physical						
Soft blocks						
I17	physConfig schematic symbol design	block1	layout	CUSTOM	LAY...	
I16	physConfig schematic symbol design	block	layout	CUSTOM	LAY...	

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

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

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

In the *No Physical* folder,

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

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

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

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

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

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

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

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

Soft Block Table View State Indicators

The following icons are used to indicate different states in the soft block table view.

Folder Name	Icon	Description
<i>Core</i> <i>IO</i> <i>Hard block</i>		Indicates a physical leaf layout.

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

<i>Soft block</i>		With the hierarchy symbol (+), indicates a schematic logical node that has a corresponding physical view which is a soft block. Without the hierarchy symbol, indicates a symbol logical node that has a corresponding physical view which is a soft block. Note: You cannot see the hierarchy for the logical node.
		Indicates that the soft block parameters were redefined.
<i>No Physical</i>	<i>Hierarchy symbol (+) AND File Folder</i>	Indicates that there is hierarchy underneath the node; for example, a schematic logical node.
	<i>Octagon symbol</i>	Indicates a logical node.
		Indicates that the soft block parameters are defined.

Setting the Cell Type

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

<i>Soft Block</i>	<i>blockBlackBox</i>	type ABSTRACT
	<i>softMacro</i>	type LAYOUT

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

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

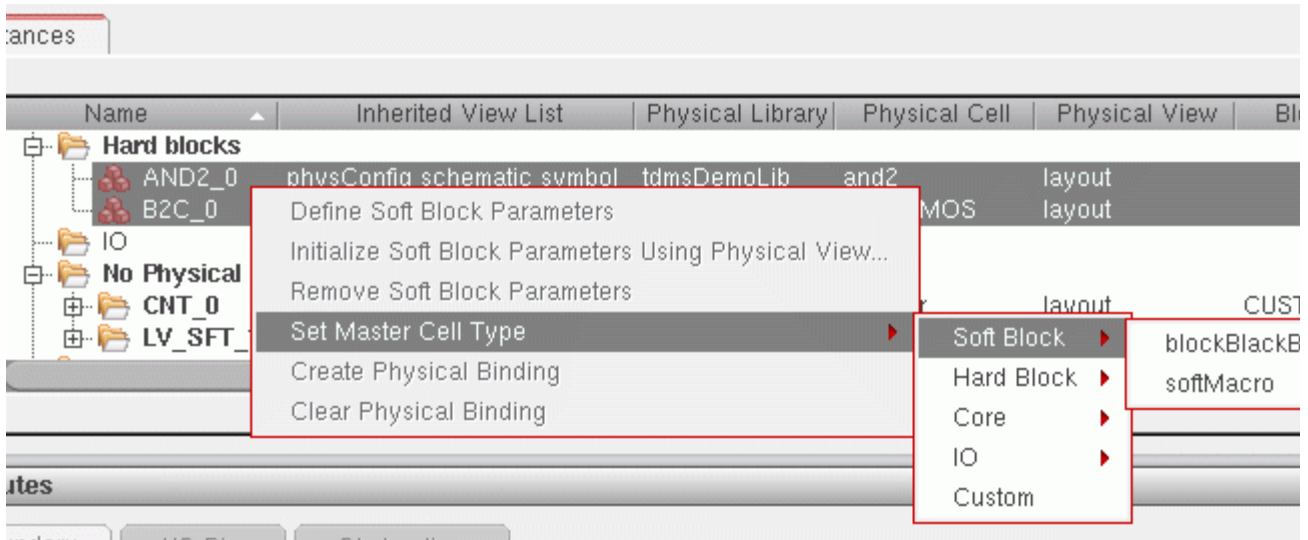
Note: To select multiple instances, keep the `Ctrl` key pressed while you select the instances.

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

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

A submenu opens showing the cell types you can set.



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

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

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

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

Setting the Cell Type from the CIW

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

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

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

The Set Cell Type form is displayed.

Library Name	Cell Name	View Name	Cell Type
"ether"	"top_common"	"layout"	"blockBlackBox"
"ether"	"top_buff"	"layout"	"none"
"ether"	"top_1_port_transmit"	"layout"	"softMacro"
"ether"	"top_1_port_receive"	"layout"	"block"
"ether"	"top_1_port_pnr"	"layout"	"block"
"ether"	"top_1_port_2p5_to1p0"	"layout"	"none"
"ether"	"top_1_port_1p0_to_2p5"	"layout"	"none"
"ether"	"top"	"layout"	"softMacro"

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

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

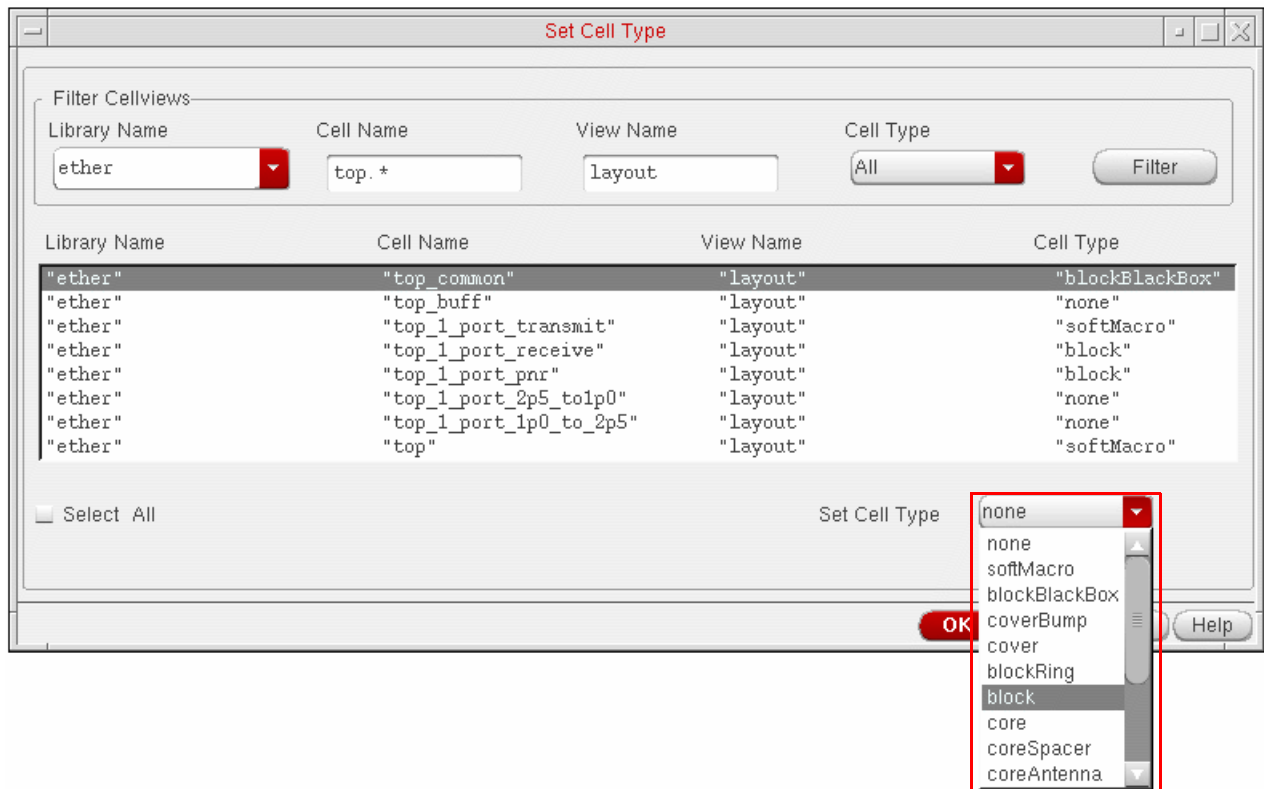
Note: For performance reasons, the *View Name* field does not accept regular expressions.

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

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

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



4. Click *Apply* or *OK*.

The specified cell type is set for the selected cells.

Defining Soft Block Parameters

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

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

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

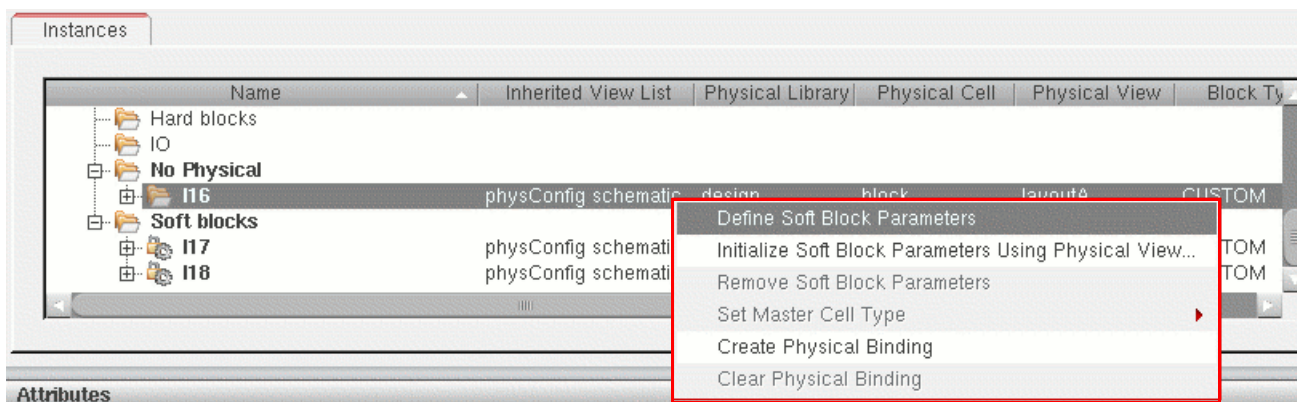
Important

To make changes to existing soft blocks that do not require the physical hierarchy to be regenerated, use the *Edit Soft Blocks* command instead. For more information, see [Editing Soft Blocks](#).

To define soft block parameters,

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

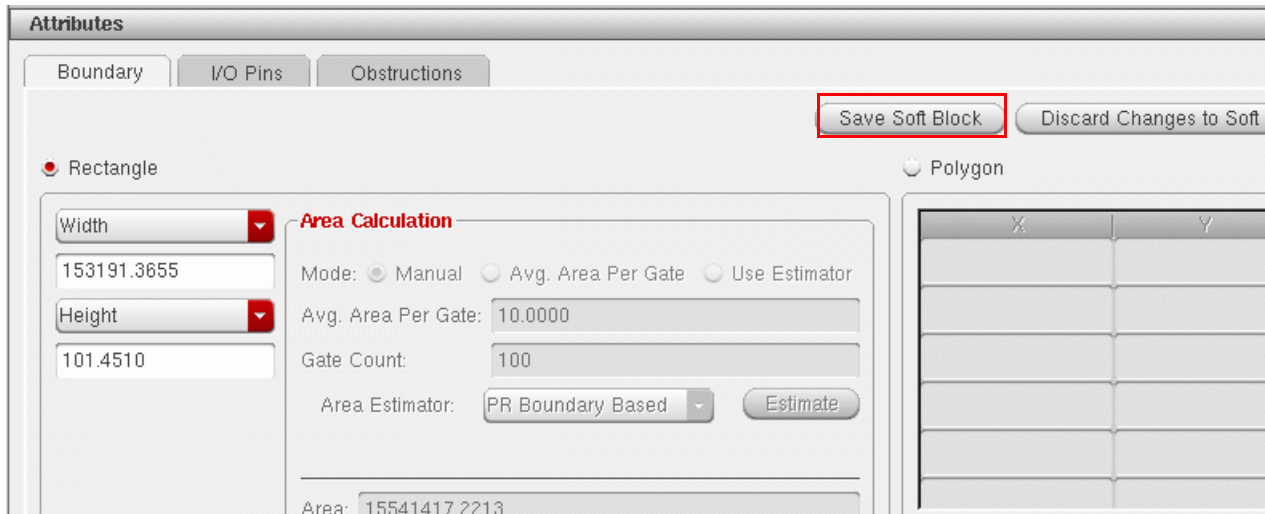
If the selected component already has soft block parameters defined, the command is labeled *Edit Soft Block Parameters*.



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

3. Edit the parameters you require for your soft block. For more information, see
 - ❑ [Specifying the Boundary](#)
 - ❑ [Specifying I/O Pins](#)
 - ❑ [Specifying Obstructions](#)

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



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

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

5. Repeat steps 1-4 for all the soft blocks you want to define.
6. From the Configure Physical Hierarchy menu bar, choose *File – Save Floorplan File* to save all your settings to a floorplan file for later re-use.

For more information, see [Saving Soft Block Parameters to a Floorplan File](#).

Specifying the Boundary

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

- [Creating a Fixed Rectangular Boundary](#)
- [Creating a Non-Fixed Rectangular Boundary](#)
- [Creating a Polygonal Boundary](#)

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

- [Creating and Registering a User Defined Area Estimation Function](#)

For general information on the controls in the *Boundary* tab, see:

- Soft Block Boundary Attributes.

Creating a Fixed Rectangular Boundary

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

To create a fixed rectangular boundary:

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

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

The screenshot shows the 'Attributes' dialog box with the 'Boundary' tab selected. The 'Rectangle' radio button is chosen. The 'Width' field is set to 200.0000. The 'Height' dropdown menu is open, showing options: 'Height', 'Aspect Ratio (W/H)', and 'Utilization (%)'. The 'Area Calculation' groupbox is disabled (grayed out). Inside this groupbox, the 'Mode' section has three radio buttons: 'Manual' (selected), 'Avg. Area Per Gate', and 'Use Estimator'. The 'Avg. Area Per Gate' field is set to 10.0. The 'Gate Count' field is set to 100. The 'Area Estimator' dropdown is set to 'DEFAULT', and there is an 'Estimate' button. At the bottom, the 'Area' field displays 100.0.

3. Type the value you require into each of the fields provided.
4. Click *Save Soft Block*.

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

Creating a Non-Fixed Rectangular Boundary

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

To create a non-fixed rectangular boundary,

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

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

2. Choose either *Width* or *Aspect Ratio (W/H)* from the first cyclic field and *Utilization* from the second cyclic field.

The screenshot shows the 'Attributes' dialog box with the 'Boundary' tab selected. The 'Rectangle' shape is chosen. The 'Area Calculation' group box is highlighted with a red border. Inside this group box, the 'Mode' is set to 'Manual'. The 'Avg. Area Per Gate' is 10.0, and the 'Gate Count' is 100. The 'Area Estimator' is set to 'DEFAULT'. The 'Area' field shows 1000.0000. The 'Width' is 2000 and 'Utilization (%)' is 60.0000.

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

3. Choose the area estimation method you prefer:

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

You can also register your own area estimation functions and use them, as appropriate. For more information, see [Creating and Registering a User Defined Area Estimation Function](#).

4. (Optional) To modify any user-defined parameters for the selected area estimation function, click *Estimate*.
5. Click *Save Soft Block*.

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

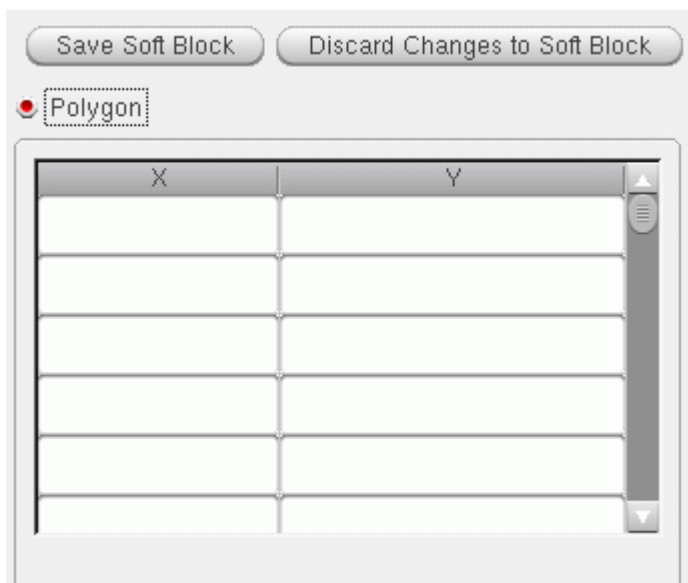
Creating a Polygonal Boundary

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

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

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

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



The screenshot shows the 'Boundary' tab in the Virtuoso Layout Suite XL. At the top, there are two buttons: 'Save Soft Block' and 'Discard Changes to Soft Block'. Below them is a radio button labeled 'Polygon' which is selected. Underneath the radio button is a table with two columns, 'X' and 'Y'. The table has six rows for entering coordinates. A vertical scrollbar is on the right side of the table.

X	Y

3. Click *Save Soft Block*.

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

Creating and Registering a User Defined Area Estimation Function

You might write a simple area estimator as follows.

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

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

```
        area = area + ins~>master~>prBoundary~>area * contactArea
    )
    return(area)
)
)
```

You register the function as shown below.

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

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

Alternatively, you can register the function as shown below.

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

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

Once registered, the area estimator function appears in the *Area Estimator* list in the *Boundary* tab.

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

Specifying I/O Pins

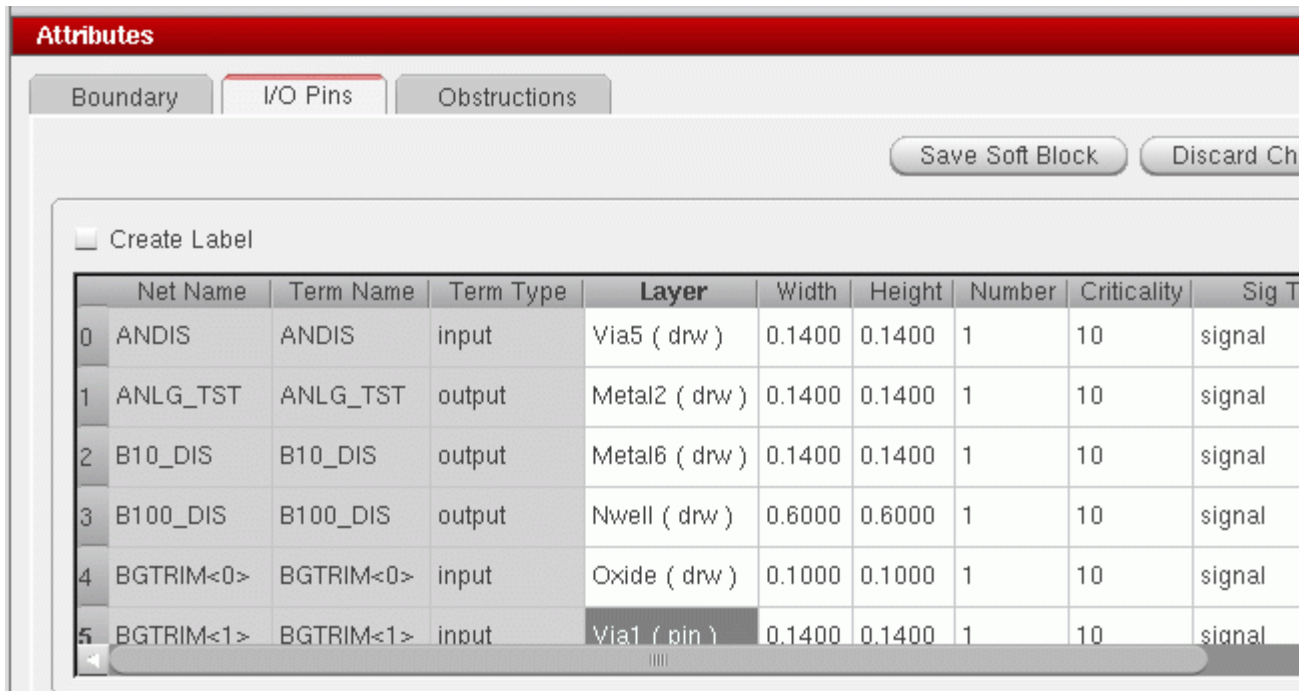
In addition to defining layer-based rules, you can now define layer purpose based rules while defining the soft block parameters in the *I/O Pins* tab.

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

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

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



All the fields with a white background are editable.

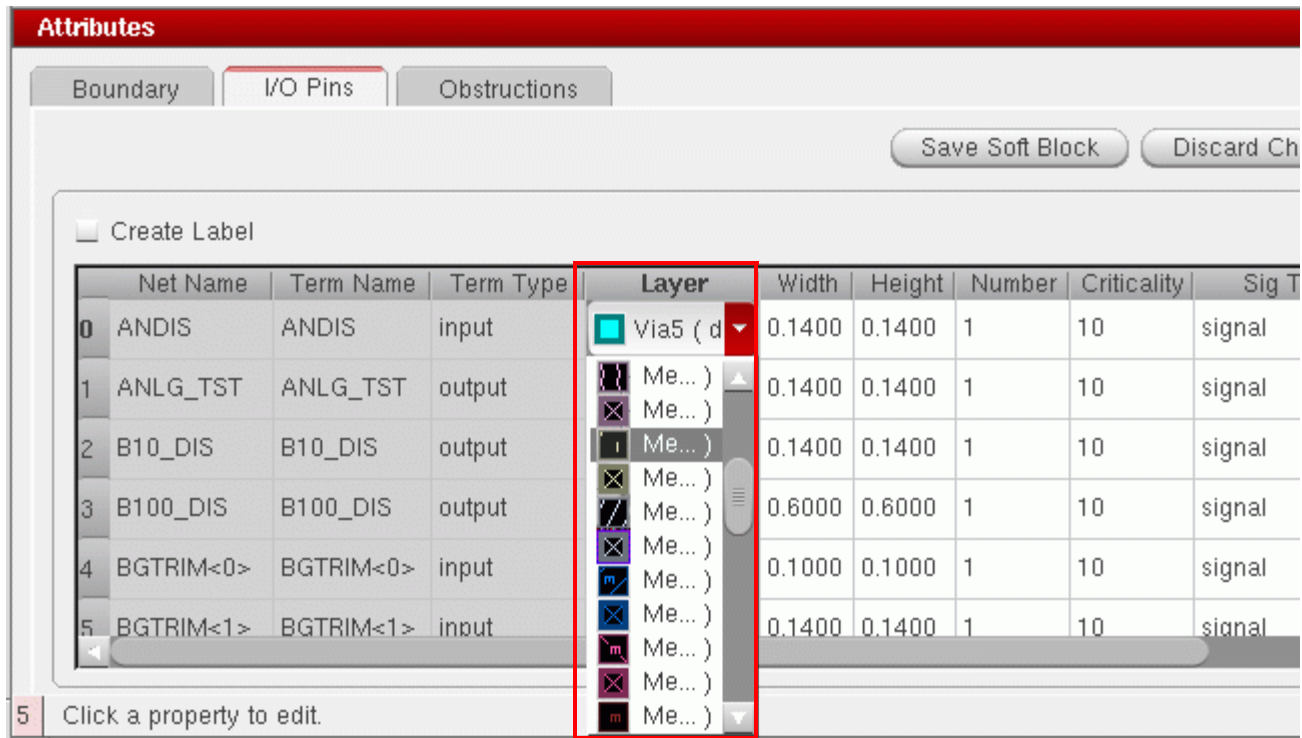
2. To change the value for an individual cell:
 - a. Double-click the cell. If a drop-down arrow appears, select a new value from the drop-down list. If the value in the cells gets highlighted, type in a new value directly.

Note: Alternatively, you can select a cell and press the space bar to display the value list box for the column. If the selected cell is a text field, pressing the space bar makes the text editable. You can then type in the new value directly.
 - b. Click any where within the editable area of the tab page.

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

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



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

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

Note: If you want to select all the cells in the column, select the column header.

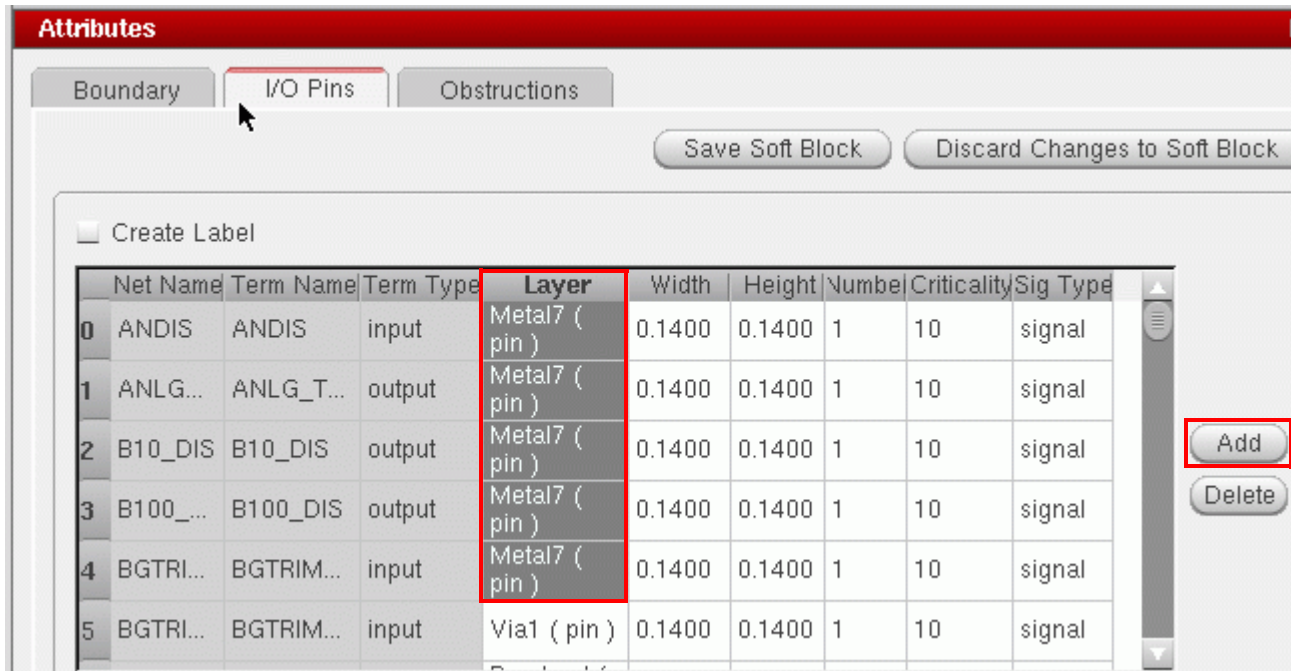
- b. With the **Ctrl** key pressed, double-click one of the selected cells and choose the new value from the value list box that appears.

Note: Alternatively, release the **Ctrl** key and press the space bar to display the value list box for the column. Then, choose an appropriate value to apply. If the selected cell is a text field, pressing the space bar makes the text editable. You can then type in the new value directly.

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

- c. Click anywhere within the editable area of the tab page. The value of the selected cells changes to the new value you selected.



4. To add a new pin:

- d. Click *Add*.

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

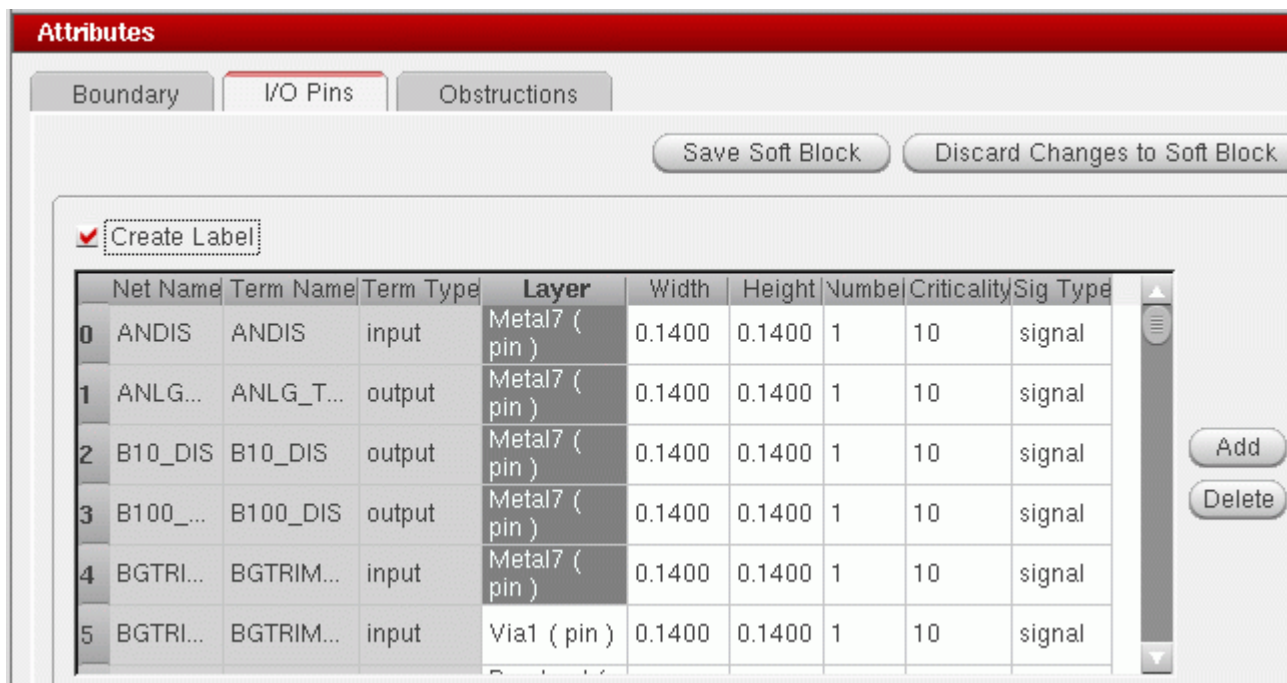
- e. Edit the values in the other fields as required.

Note: If the soft block has already been generated in the layout, the *Width* and *Height* fields show the actual dimensions of the pin shape in the layout. If the soft block has not been generated, the default minimum width and height specified in the technology file are shown.

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

The selected pin is removed from the list.

6. To generate labels for all the pins in the selected soft block, switch on the *Create Label* option.



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

- a. Choose *Options – Soft Block Options* from the Configure Physical hierarchy menu bar.
 - b. Check the *Create Label* check box in the Soft Block Global Options form.
 - c. Select appropriate label creation options under the *Pin Label Text Style* group box.
7. Click *Save Soft Block*.

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

Specifying Obstructions

You use the *Obstructions* tab to create routing and placement obstructions in your soft block. A soft block obstruction is implemented using a halo object instead of a blockage object, because there is no need to manually remove the halo prior to routing.

- To define a *routing*, *fill*, *slot*, *pin*, *feedthru*, or *screen* obstruction, you need to set the *Blockage Type*, and specify the *Layer* and *Offsets*. There is also a *Material* filter which

you can use to select layers based on the material defined for that layer in the technology file.

Note:

- ❑ The list of extractable layers is retrieved from the constraint group set for the current layout cellview. If there is no constraint group set for the current layout cellview, the software retrieves the layers from the constraint group specified by the `setupConstraintGroup` environment variable. The CPH soft block reads the environment variable and looks for the constraint group carrying the required information in the associated technology file. The technology file may be an Incremental Technology Database file (ITDB) or a design techfile.
 - ❑ If there are no layers defined in either location, it issues a message and disables the soft block definition.
- To define a *placement* obstruction, you need only set the *Offsets*. You can define only one placement obstruction per block.

To specify parameters for obstructions,

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

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

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

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

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

Attributes

Boundary I/O Pins **Obstructions**

Save Soft Block Discard Changes to Soft Block

Add Blockages

Blockage Type: placement, **routing**, fill, slot, pin, feedthru, screen

Material: all

Layer: Cont, **Metal1**, Metal2, Metal3, Metal4, Metal5, Metal6

Offsets

Top: 5.0000 Bottom: 5.0000

Right: 5.0000 Left: 5.0000

Type	Layer	Top	Bottom	Right	Left
1 routing	Metal1	5.0000	5.0000	5.0000	5.0000

Add Delete

3. To define a placement obstruction:

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

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

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

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

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

Attributes

Boundary I/O Pins **Obstructions**

Save Soft Block Discard Changes to Soft Block

Add Blockages

Blockage Type: routing, fill, slot, pin, feedthru, screen

Material: all

Layer: Cont, Metal1, Metal2, Metal3, Metal4, Metal5, Metal6

Offsets

Top: 5.0000 Bottom: 5.0000
Right: 5.0000 Left: 5.0000

Type	Layer	Top	Bottom	Right	Left
1 routing	Metal1	5.0000	5.0000	5.0000	5.0000
2 place...		5.0000	5.0000	5.0000	5.0000

Add Delete

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

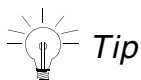
The obstruction is deleted and removed from the table.

Removing Soft Block Parameters

To remove soft block parameters,

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

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



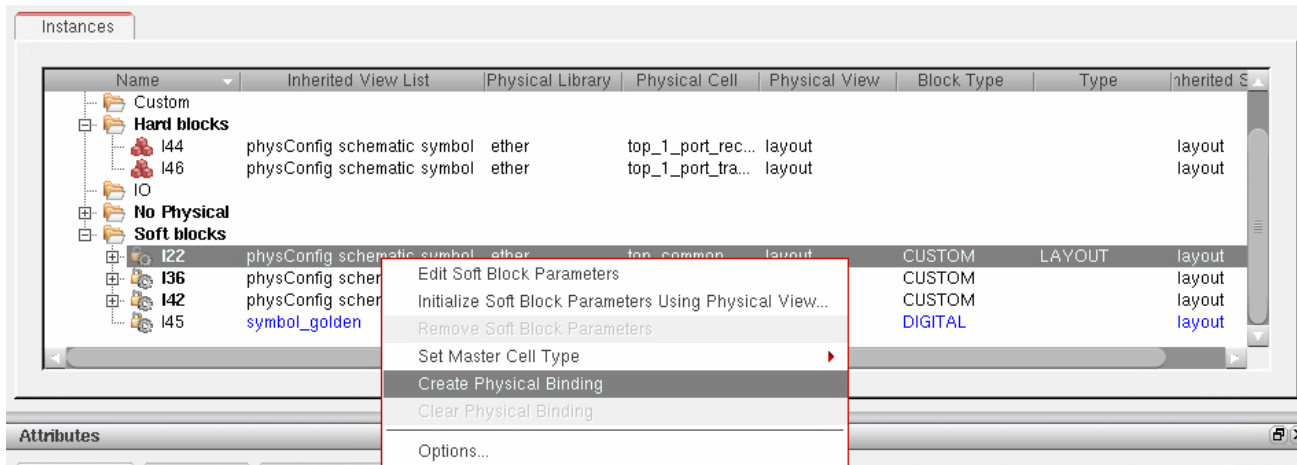
Tip

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

Creating a Physical Binding

To create a physical binding,

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



The physical binding is created.

To remove a physical binding,

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

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



Tip

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

Loading Soft Block Parameters from Another Cellview

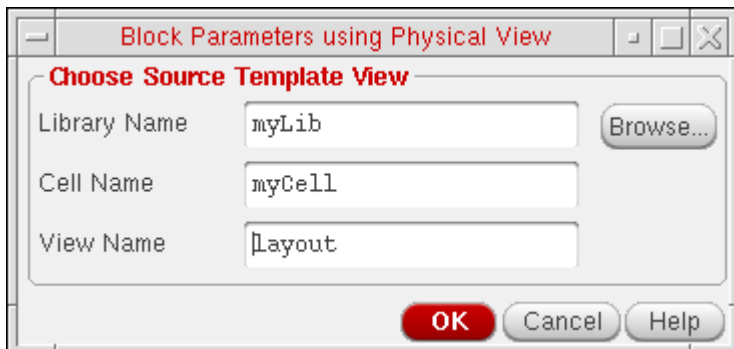
The *Initialize Soft Block Parameters Using Physical View* command imports soft block parameters from an existing layout cellview. For more information, see [Loading Physical Information from Another Cellview](#).

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

To load predefined soft block parameters from another cellview,

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

The Block Parameters using Physical View dialog is displayed.



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

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

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

- ☐ [Specifying the Boundary](#)
- ☐ [Specifying I/O Pins](#)
- ☐ [Specifying Obstructions](#)

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

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

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

6. Repeat steps 1-5 for all the soft blocks you want to define in this way.
7. From the Configure Physical Hierarchy menu bar, choose *File – Save Floorplan File* to save all your settings to a floorplan file for later re-use.

For more information, see [Saving Soft Block Parameters to a Floorplan File](#).

Saving Soft Block Parameters to a Floorplan File

You can save the current block parameters to a floorplan file, which you can then re-use in a different Virtuoso session.

To do this,

1. Define the soft block parameters you require using the procedures described in [Defining Soft Block Parameters](#).
2. From the Configure Physical Hierarchy menu bar, choose *File – Save Floorplan File*.
3. Specify a name for the file and click *OK*.

The floorplanning parameters are saved to the specified file.

Loading Soft Block Parameters from a Floorplan File

To load parameters from a floorplan file,

1. From the Layout XL window, choose *Launch – Configure Physical Hierarchy*.
The Configure Physical Hierarchy window appears.
2. Change to Soft Block mode using the drop down list in the toolbar.
3. From the Configure Physical Hierarchy menu bar, choose *File – Load Floorplan File*.
The Load Floorplan File dialog appears.
4. Specify the file you want to load and click *OK*.

The floorplanning parameters defined in the file are loaded into the current configuration view.

Generating the Physical Hierarchy

Use the *Generate Physical Hierarchy* command to generate the components in your design while maintaining the hierarchy levels you have defined in Soft Block mode.



Tip

This section describes only how to start the *Generate Physical Hierarchy* command. For information on what it does, see the [Virtuoso Floorplanner User Guide](#).

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

To generate the physical hierarchy,

1. In the Configure Physical Hierarchy window, define or load the soft block parameters you require. For information on how to do this, see
 - ❑ [Defining Soft Block Parameters](#)
 - ❑ [Loading Soft Block Parameters from a Floorplan File](#)
2. From the layout window menu bar, choose *Launch – Layout GXL* to switch into GXL mode.
3. From the layout window menu bar, choose *Floorplan – Generate Physical Hierarchy*.

The physical hierarchy is generated in the layout window, including the soft blocks defined in the Configure Physical Hierarchy window.

Virtuoso Layout Suite XL User Guide

Configuring the Physical Hierarchy

Device Abutment

This chapter describes the Virtuoso® Layout Suite XL layout editor's (Layout XL) abutment capability.

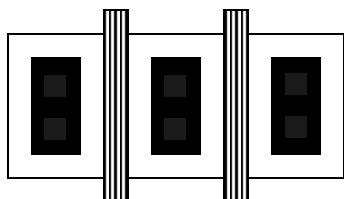
This chapter discusses the following topics.

- [Requirements for Abutment in Layout XL](#) on page 183
- [Setting Up Cells for Abutment](#) on page 184
- [Manual Device Abutment](#) on page 194
- [Automatic Device Abutment](#) on page 203
- [Abutting Parameterized Cells and Quick Cells](#) on page 205
- [Abutment and the Virtuoso Chip Assembly Router](#) on page 214

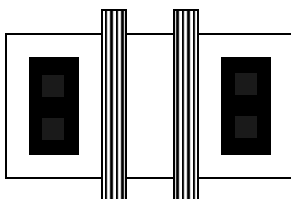
Abutment allows cells to be automatically overlapped, aligned, and electrically connected without introducing a design rule violation or connectivity error. Abutment reduces both the

area occupied by a circuit and the length of the interconnect wiring. You can use abutment during interactive layout generation.

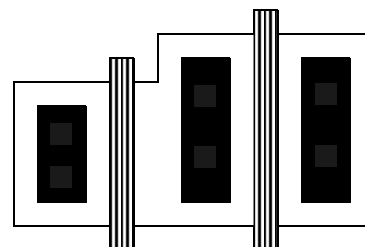
MOS transistors sharing a diffusion



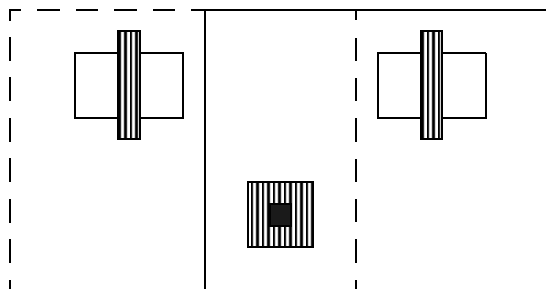
A. Same size, terminals with external connection on same net



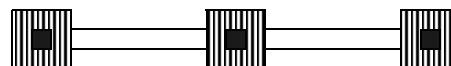
B. Same size, no other connections on the same net



C. Different size, terminals with external connection on the same net



Wells sharing part of their surface and bias contacts



Resistors sharing terminals

Mirroring and Pin Permutation During Abutment

If an attempt to abut two devices causes two pins on the same net to touch, a short violation is created. Layout XL tries to resolve this violation automatically by first mirroring the device or chain in question. If that fails, it then attempts to resolve it by permuting pins.

This behavior is controlled by two environment variables, both of which are switched on by default.

- **autoMirror** which specifies that if a short violation is created during abutment, then Layout XL first mirrors the device in question in order to resolve the short. Only if this is unsuccessful does Layout XL attempt to resolve the short by permuting the pins.

The GUI equivalent is the *Mirror transistors* option on the Layout XL Generation Options form.

Automatic mirroring is temporarily disabled when constraint-aware editing is on and constraints exist between the instances and chains under consideration.

- autoMirrorChains which specifies that device chains are mirrored subject to certain conditions being satisfied. This environment variable is honored only if `auto Mirror` is set to `t`. It has no GUI equivalent.

Important

If one of the devices is in any type of group (including a synchronous clone), the other device must be in the same group for automatic mirroring to occur. Mirroring is done on end-pin violations only if they are valid left/right or top/bottom pairs.

To disable the mirroring of devices and chains:

1. From the layout window menu bar, choose *Options – Layout XL* and click the *Generation* tab.
2. Switch off the *Mirror transistors* option and click *OK*.

This switches off both device mirroring and chain mirroring during abutment.

Resolving shorts by mirroring rather than permuting pins is desirable because pin permutation

- Assumes that the device is symmetric and therefore the source and drain parameters are not swapped
- Changes the connectivity of the design.

Resolving the short violation through mirroring removes these drawbacks.

Note: No pin permutation is performed on chained devices.

Requirements for Abutment in Layout XL

Abutment in Layout XL is triggered when two pins of different instances overlap. If they meet the following criteria, the abutment engine executes.

- The abutment engine is enabled.
- Both instances must be pcells set up for abutment or cells with the appropriate abutment properties set.

For more information, see [“Setting Up Cells for Abutment”](#) on page 184.

- Both instances must share the same master or must have the abutClass property set.

If two instances have the same master, then they can be abutted, even if the `abutClass` is set to different values on each instances

To abut two cells with different masters, you must add the `abutClass` property to the pin of each cell and enter the same abutment class name as the value for each property.

For more information, see [About Pcell Super and Submaster Cells](#) in the *Virtuoso Layout Suite SKILL Reference*.

- The instance must overlap.
- Both instance pins must be connected to the same net
- Both instance pins must be defined on shapes of the same layer or on layers that are defined as equivalent layers in the technology file
- Both instance pins can have the same abutment direction if the rotation of one (but not both) of the pins is either `R180` or `MY`, or any other rotation that transforms the left-hand side to the right-hand side
- During placement, the *Connectivity extractor* and *Abut transistors* options in the Layout XL Options form must be turned on.

When the *Abut transistors* option is turned on, devices that have not been abutted but are overlapping are abutted. If the devices are already abutted, `autoAbutment` does not re-abut the devices.

- If one of the instances is in any type of group (including a synchronous clone), the other instance must be in the same group.

Abutted devices can share diffusion, contacts, metal tabs, or any shape combined in an instance pin.

Setting Up Cells for Abutment

You can set up both regular cells and pcells for abutment. To set up a regular cell so that it can be abutted, add the following properties on the pins of the cellview:

- `abutFunction`
- `abutOffset`
- `abutAccessDir`
- `abutClass`

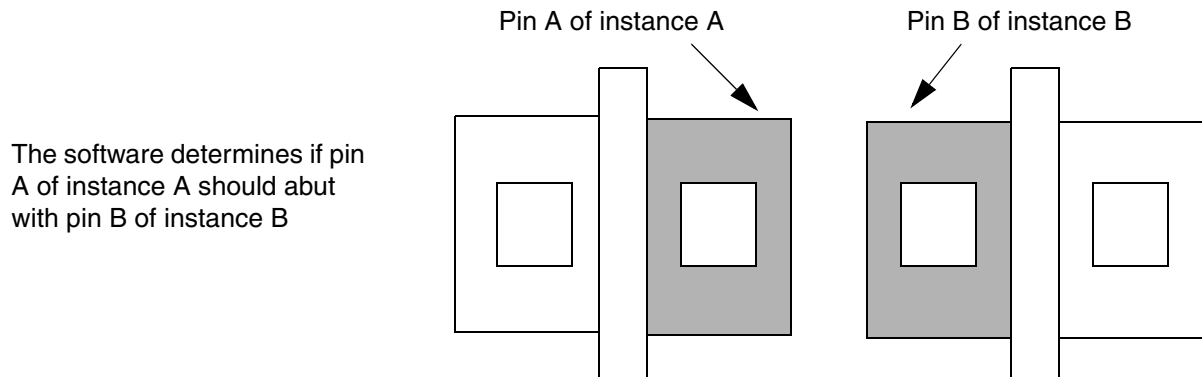
You can also add these properties to a cellview in order to set up abutment for all the pins in the cellview. If any pin in the cellview has properties that differ from the properties set for the cellview, the properties on the pin override the properties on the cellview.

For examples of how to set abutment properties, see the sample pcell libraries at

- `your_install_dir/tools/dfII/samples/ROD/rodPcells/components/mos/mos.il`
- `your_install_dir/tools/dfII/samples/ROD/rodPcells/components/mos/sample_mos.il`

Steps in Automatic Abutment

1. Layout XL recognizes that two pcell instances can be abutted together, triggered either by overlapping two instance pins that share the same net or when you create parameterized cell or Quick Cell instances during chaining and folding.

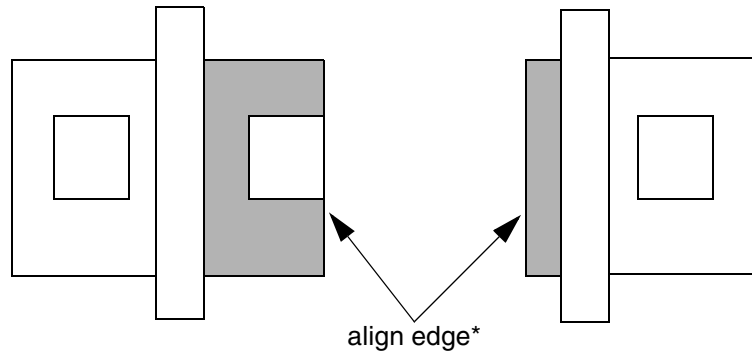


2. Automatic abutment identifies cells for abutment by master name or class.
3. Auto-abutment calls `abutFunction`, a user-defined SKILL function that performs the necessary changes to the parameterized cell instances themselves.

You must provide the SKILL function because Layout XL has no knowledge of the user-defined parameterized cell.

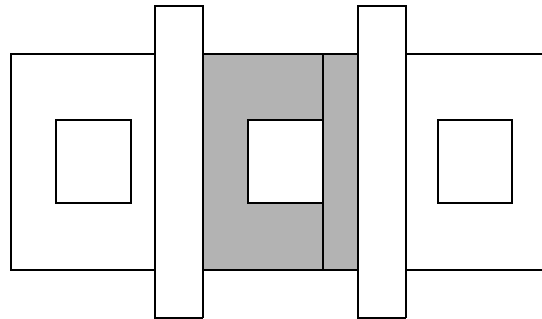
`abutFunction` adjusts the parameters of pcells and calculates reference edge offsets of conventional cells. It stores the old values, which can be called again if you choose to `unabut` the instances.

If yes, the software calls the user-defined SKILL function to change the topology of instance A and instance B and return the offset to align pin A of instance A and pin B of instance B.

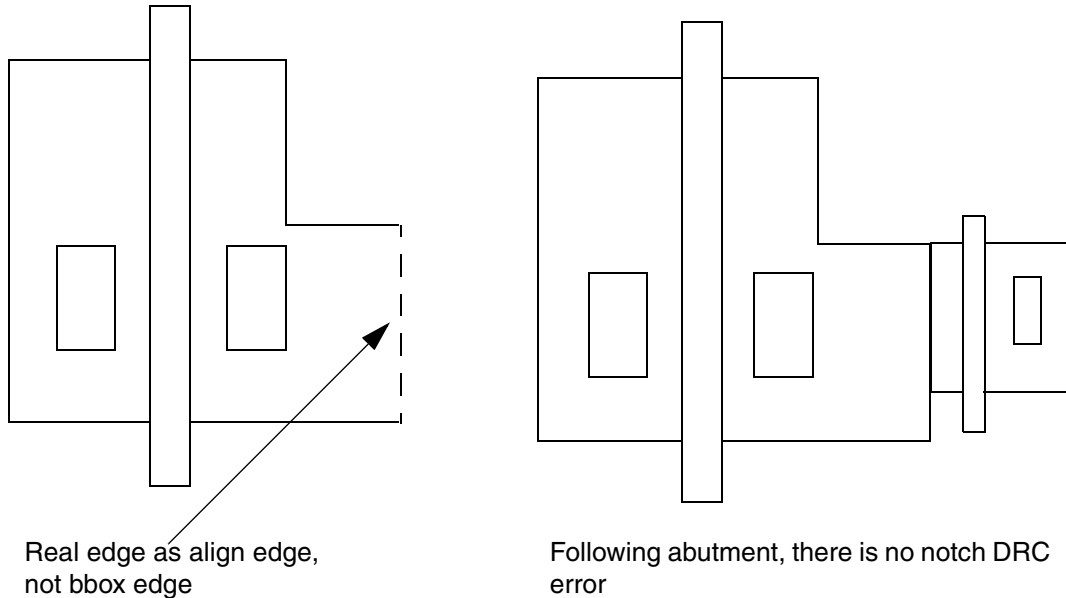


4. If the cells can be abutted (the abutment connection condition is 1 or 2), the cells are abutted to the reference edges and the pins are aligned perpendicular to the direction of abutment.

The software correctly aligns pin A of instance A and pin B of instance B



Note that the align edge is always the real outermost edge is used, as shown in the below. This lets you maintain an L-shaped pin when applying abutment.



If the cells cannot be abutted, the abutment connection condition is 3 and the cells remain in their original configuration.

For information on abutment connection conditions, see [abutFunction](#).

Sample Parameterized Cells Set Up for Abutment

Example 1

This example shows what each of the different parts does.

```
-----  
; Function: abutFunction(iA iB pA pB pASide connection event  
; @optional (group nil))  
;  
; Inputs: group - the abutment group that these two cells belong to.  
;  
;  
; iA          = Instance Id of cell that will move during abutment  
;  
;  
; iB          = Instance Id of cell being abutted to  
;  
-----
```

Virtuoso Layout Suite XL User Guide

Device Abutment

```

;
;   pA           = Overlapping Pin Fig of iA
;
;
;   pB           = Overlapping Pin Fig of iB
;
;
;   pASide       = Abutting pin access direction
;
;
;   connection = an integer value of 1 or 2  that indicates:
;
;   1. pins are connected to the same net and do not
;      connect to any other pin.
;
;   2. pins are connected to the same net and the net
;      connects to other pins
;
;
;   event        = integer that represents abutment event:
;
;   1. compute abutment offset
;   2. pcell parameter adjustment for abutment
;   3. pcell parameter adjustment for unabutment
;
;
;   group        = abutment group pointer available to events 2 and 3
;
;
; Outputs: depends
;
; Side effects:
;
;
procedure( abutFunction(iA iB pA pB pASide connection event @optional (group
nil))
  prog((result)
    case(event
      (1 ; Compute offset
        result = getAbutmentOffset(iA iB pA pB pASide connection)
      )
      (2 ; Adjust pcell parameters
        result = setAbutmentParams(iA iB pA pB pASide connection group)
      )
      (3 ; Adjust pcell parameters back to default resetAbutmentParams(group iA iB)
        result = t
      )
      (t ; Anything else return a nil
        result = nil
      )
    )
  )
  return(result)
)

```

Virtuoso Layout Suite XL User Guide

Device Abutment

Example 2

This example shows how to add abutment properties to pins in an inverter pcell.

```
; *** the following 4 pins are on metall and are on ***
; *** the power and ground rails. This will allow ***
; *** abutment to other standard cells ***

obj = leftVddPin~>dbId
dbReplaceProp(obj "abutAccessDir" "list" list("left"))
dbReplaceProp(obj "abutClass" "string" "stdcell")
dbReplaceProp(obj "abutFunction" "string" "stdCellFunc")

; *** the following property is a technique for passing ***
; *** information to the user defined function. In this ***
; *** case the minimum cell height is dependent on many ***
; *** factors including design rules and is originally ***
; *** calculated in the pcell code. This avoids ***
; *** duplicating calculations in the user-defined ***
; *** function ***

dbReplaceProp(obj "minCellHeight" "float" minH)

obj = rightVddPin~>dbId
dbReplaceProp(obj "abutAccessDir" "list" list("right"))
dbReplaceProp(obj "abutClass" "string" "stdcell")
dbReplaceProp(obj "abutFunction" "string" "stdCellFunc")
dbReplaceProp(obj "minCellHeight" "float" minH)

obj = leftGndPin~>dbId
dbReplaceProp(obj "abutAccessDir" "list" list("left"))
dbReplaceProp(obj "abutClass" "string" "stdcell")
dbReplaceProp(obj "abutFunction" "string" "stdCellFunc")
dbReplaceProp(obj "minCellHeight" "float" minH)

obj = rightGndPin~>dbId
dbReplaceProp(obj "abutAccessDir" "list" list("right"))
dbReplaceProp(obj "abutClass" "string" "stdcell")
dbReplaceProp(obj "abutFunction" "string" "stdCellFunc")
dbReplaceProp(obj "minCellHeight" "float" minH)

; *** The following 4 pins are the diffusion pins ***
; *** of the individual MOS devices in the inverter ***
; *** this will allow abutment of other individual ***
; *** MOS devices to the inverter ***

obj = POutPin~>dbId
dbReplaceProp(obj "abutAccessDir" "list" list("right"))
dbReplaceProp(obj "abutClass" "string" "pttran")
dbReplaceProp(obj "abutFunction" "string" "mosAbutFunc")
dbReplaceProp(obj "contactParam" "string" "POutCnts")
dbReplaceProp(obj "w" "float" pw)
; *** the automatic spacing properties are used if abutment fails ***
dbReplaceProp(obj "vxlInstSpacingDir" "list" list("right"))
dbReplaceProp(obj "vxlInstSpacingRule" "float" .35 )

obj = NOutPin~>dbId
dbReplaceProp(obj "abutAccessDir" "list" list("right"))
dbReplaceProp(obj "abutClass" "string" "ntran")
```

Virtuoso Layout Suite XL User Guide

Device Abutment

```
dbReplaceProp(obj "abutFunction" "string" "mosAbutFunc")
dbReplaceProp(obj "contactParam" "string" "NOutCnts")
dbReplaceProp(obj "w" "float" nw)
dbReplaceProp(obj "vxlInstSpacingDir" "list" list("right"))
dbReplaceProp(obj "vxlInstSpacingRule" "float" .35 )

obj = PPwrPin~>dbId
dbReplaceProp(obj "abutAccessDir" "list" list("left"))
dbReplaceProp(obj "abutClass" "string" "ptran")
dbReplaceProp(obj "abutFunction" "string" "mosAbutFunc")
dbReplaceProp(obj "contactParam" "string" "PPwrCnts")
dbReplaceProp(obj "w" "float" pw)
dbReplaceProp(obj "vxlInstSpacingDir" "list" list("left"))
dbReplaceProp(obj "vxlInstSpacingRule" "float" .35 )

obj = NPwrPin~>dbId
dbReplaceProp(obj "abutAccessDir" "list" list("left"))
dbReplaceProp(obj "abutClass" "string" "ntran")
dbReplaceProp(obj "abutFunction" "string" "mosAbutFunc")
dbReplaceProp(obj "contactParam" "string" "NPwrCnts")
dbReplaceProp(obj "w" "float" nw)
dbReplaceProp(obj "vxlInstSpacingDir" "list" list("left"))
dbReplaceProp(obj "vxlInstSpacingRule" "float" .35 )
```

Calls to dbReplaceProp for Abutment Groups

When using the function `dbReplaceProp`, you must call `dbReplaceProp` for abutment groups before calls to `dbReplaceProp` for pcell properties because the IDs of the objects before and after abutment occurs cannot be guaranteed to be the same. This is because the instance master is purged as soon as the instance reference count associated with the instance master becomes zero. Therefore you cannot rely on the same data or IDs after pcell re-evaluation.

Pin Figure Names must be Unique within a Terminal

When developing transistor pcells, wherever there is more than one figure attached to any `oaPin` (as is the case when two figures are strongly connected), each figure attached to the pin must have a name that is unique within the same terminal and its corresponding must-connect terminals. If this is not the case, use the `dbSetPinFigName` SKILL function to assign a unique name to each figure attached to the pin.

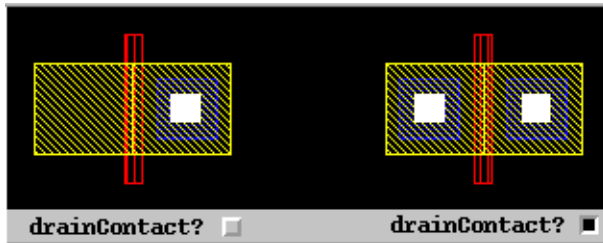
Creating CMOS Pcells to Use with Abutment

To create pcells that have a built-in abutment capability,

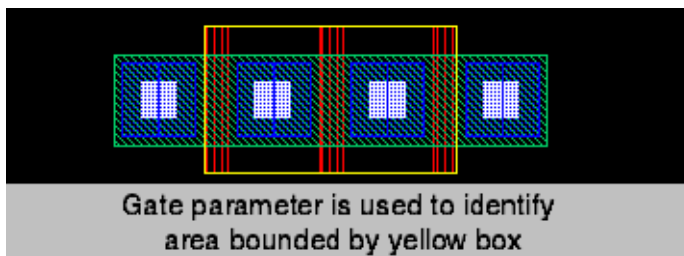
1. Create a pcell specifically for abutment.

Editing an existing pcell for abutment is much more difficult than creating a new one for the purpose. For information on creating pcells, see the [Virtuoso Parameterized Cell Reference](#).

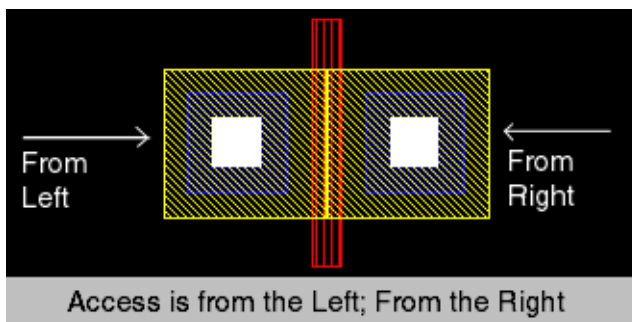
2. Add the following abutment parameters to the pcell.
 - a. Conditional inclusion or exclusion of contacts. For more information, see [abutCondInclusion](#).



- b. Gate pin identification. For more information, see [abutGateNet](#).

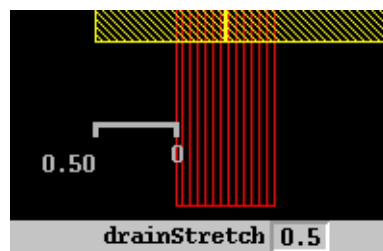
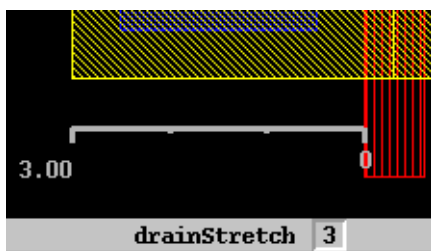


- c. Access direction identification. For information, see [abutAccessDir](#).

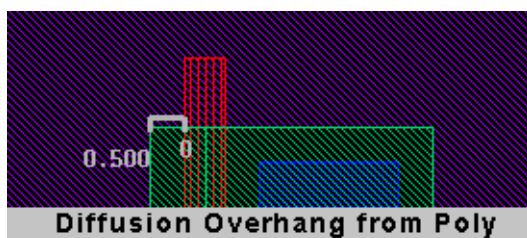


- d. Specify stretchable material. For information, see [abutStretchMat](#).

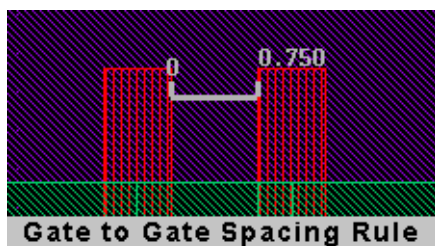
The first element in the `abutStretchMat` list is `drainStretch`.



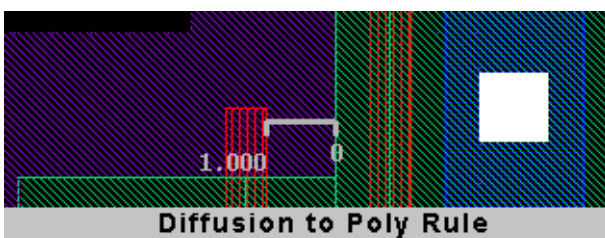
The second element in the `abutStretchMat` list is `abutMinExt`.



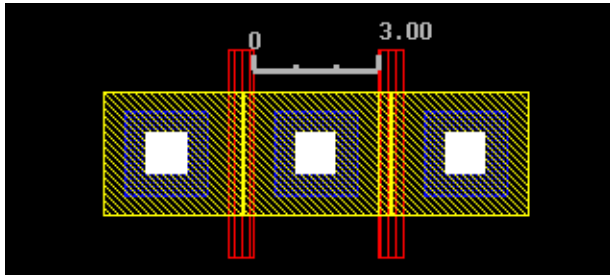
The third element in the `abutStretchMat` list is `abutRule1Ext`.



The fourth element in the `abutStretchMat` list is `abutRule2Ext`.



The fifth element in the `abutStretchMat` list is `abutContactExt`.



Abutting MOS Devices without a User-Defined Function

Another way to process abutment and unabutment without creating an `abutFunction` callback, is to use the `abutMosStretchMat` property. The `abutMosStretchMat` property is used with the automatic abutment properties to define abutment; for more information, see [abutMosStretchMat](#).

This section describes the automatic abutment properties that you will have to add to pcells to use auto-abutment on standard MOS pcells without having to define your own `abutFunction`.

Adding Automatic Abutment Properties

As well as creating specific parameters in the pcells and named diffusion shape pins, you also have to add the following properties on the shapes the pins are created on. The auto-abutment properties specify whether or not the pcells can abut.

- [abutClass](#) – Pins with the same `abutClass` are able to abut
- [abutAccessDir](#) – Defines the directions in which the pins are able to abut.
- [abutGateNet](#) – Defines the name of the net the gate shapes are on.
- [abutCondInclusion](#) – Defines the name of the boolean property that adds or removes contact shapes over the diffusion pin.

Abutting Multiple Pins

Layout XL supports the abutment of more than one pin to a wider single pin.

Multiple pins on the edge of a cell can also be abutted. Any pair of pins with the correct properties can trigger automatic abutment, but once automatic abutment has been triggered,

other pins on that edge that touch pins on the cell it is abutted to will not trigger automatic abutment again. If those pins do not connect, auto-permute tries to resolve the conflict. If the conflict cannot be resolved, a connection violation is flagged.

Note: Make sure that you define abutment in such a way that no Design Rule Checker (DRC) or connectivity violations are introduced by abutment between any selected pair of pins.

Snapping Instances in the Direction Perpendicular to the Abutment

The layout environment variable `abutPerpSnapOn` controls whether Layout XL snaps an instance in the direction perpendicular to the abutment. When `abutPerpSnapOn` is set to `t`, Layout XL automatically snaps the moving instance vertically such that the lower-left y coordinates of the two abutting pins are the same.

Additional Pins on Ignored Instances

The abutment engine ignores instance terminals belonging to instances with the schematic properties `lvsIgnore` or `ignore`. This means that additional pins on ignored instances are also ignored, allowing abutment to take place where it otherwise would not and enabling a tighter placement of abutted components.

For example, a schematic has two MOS devices connected in series (one's source to the other's drain). The schematic also places a parasitic capacitor between these MOS devices. The capacitor has the `ignore` property attached to it. When generating the layout, Layout XL correctly ignores the capacitor and its pin, and abuts the two MOS devices.

Manual Device Abutment

Device abutment is the process by which a number of MOS transistors (or the individual folds of folded transistors) are chained in a specific order. Chaining devices helps reduce layout area and capacitance.

Prerequisites

The transistors to be chained

- Need not be bound to a schematic instance.
- Must be properly set up for abutment. For more information, see [Setting Up Cells for Abutment](#).

- Must have the `lxComponentType` property set on the cell or library to which they belong. You set this property using the *Edit Component Types* command. For more information, see the *Configure Physical Hierarchy* command's Component Types Mode.

All parameter values are retained after chaining. The chaining engine always creates pairs of P and N chains with the P chain on top (because the power rail is typically at the top of a cell) and the N chain at the bottom (because the ground rail is typically at the bottom).

You can use the following commands to chain devices manually.

- *Generate Chained Devices* lets you chain devices interactively while you are editing a layout. See Chaining Devices Interactively.
- *Generate Selected From Source* can automatically chain devices while generating a layout. See Chaining Devices when Generating Selected From Source.
- *Move* lets you control chaining while moving devices. See Moving Chained Transistors.

Important

The results generated by the chaining engine are affected by the setting for the lxStackPartitionParameters environment variable. Note that the *Generate Chained Devices* command considers the first parameter of this environment variable only if there are 100 instances or more in the selected set. *Generate Selected From Source* always considers both parameters.

Chaining Devices Interactively

To chain transistors interactively while you are editing a layout:

1. From the layout window menu bar, choose *Connectivity – Generate – Chained Devices*.

The Generate Chained Devices form appears.



2. Set the options you require.

- ☐ To prevent existing chains from being broken during interactive chaining, select *Preserve Existing Chains*.
- ☐ To maintain the relative starting positions of the selected devices in the generated chain, select *Use Device Order*.
- ☐ To select entire chains with a single click, choose *Extend Selection*. Otherwise, each click selects an individual device, manufactured device, or folded leg.

3. In the layout canvas, select the devices you want to chain.

Alternatively, click the *Show Device List* button to see the list of devices available for chaining. The list box appears in the Generate Chained Devices form.

You can also select devices in the layout canvas before you start the command. In this case, the selected devices are listed in the form when it opens.

The selected devices are highlighted in the canvas.

4. Click *Apply* and move the cursor into the layout canvas.

The selected transistors appear as an image of a transistor chain that follows the cursor.

If the majority of the selected devices are oriented vertically, the generated chain is also oriented vertically.

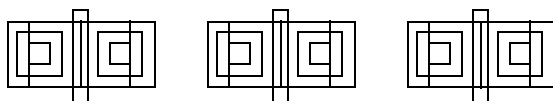
- a.** Click the middle mouse button to rotate a chain by R90.
- b.** Shift-click the middle mouse button to flip a chain alternately about its X and Y axes.

Note: You must have `lxBindKeys.il` loaded in order to use this functionality.

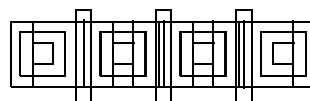
5. Click on the layout canvas where you want to place the chained transistors.

The transistors are placed where you click.

Before chaining



After chaining



Important

You can also chain dummy instances interactively by using the procedure described above. If you select the *Use Device Order* check box, the dummies are treated as regular devices during the chaining operation and can support abutment on one side or on both sides depending on the state of the `chainDummyFlexBothEndNets` environment variable. If *Use Device Order* is not selected, the dummies are ignored during chaining. As a result, irrespective of their order, the dummies are placed at the end of the chain and their initial connectivity is retained.

Additional Information

- Chains that comprise only NMOS or only PMOS transistors are aligned with the bottom edge. If you use the *Generate All From Source* command to create a cluster comprising both an NMOS and a PMOS transistor, the PMOS is placed on top aligned to the lower edge, and the NMOS is placed below it aligned to the upper edge.
- Folded transistors are chained with the number of folds specified in the Generate Folded Devices form. For more information, see Folding a Transistor.

Removing a device from a Chain

To remove a device from a chain:

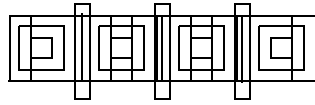
1. In the layout canvas, select the device you want to remove from a chain.

Note: The `chainExtendSelection` environment variable must be set to `nil` to let you select an individual device in a chain.

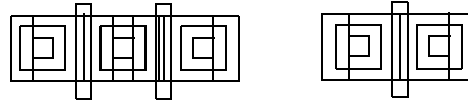
2. Choose *Edit – Move*.

3. Click on the layout canvas where you want to place the selected device.

Before removing device from chain



After removing device from chain



Chaining Devices when Generating Selected From Source

To chain devices automatically during the *Generate Selected From Source* command,

- In the Generate Selected Components form, select the *Group As In Schematic* and *Chaining* options.



Note: The results generated by the chaining engine during this command are affected by the setting for the lxStackPartitionParameters environment variable.

Virtuoso Layout Suite XL User Guide

Device Abutment

Layout XL cannot automatically chain devices in Place Individually mode. To do this,

Virtuoso Layout Suite XL User Guide

Device Abutment

1. Select the **Extraction Tab** of the Layout XL Options form and choose *Update connectivity information when design is modified*.

The screenshot shows the 'Layout XL Options' dialog box with the 'Extraction' tab selected. The 'Connectivity Extraction' section is highlighted with a red box around the 'Update connectivity information when design is modified' checkbox, which is checked. Below this, the 'Extract connectivity to level' is set to 0. The 'Derive extractable layers from constraint group' dropdown is set to 'virtuosoDefaultExtractorSetup'. A 'Diagnostics Report' button is visible. A list of checkboxes for various extraction options is shown, with several checked. At the bottom, there are input fields for the maximum number of open, short, and illegal connections, with values 100 and 1000. The 'Space-based Router' section is also visible, with 'Extract' set to 'pcells' and 'Extracted Pin Style' set to 'Labeled Shapes Only'.

Layout XL Options

General Display **Extraction** Generation Parameters

Connectivity Extraction

☒ Update connectivity information when design is modified

Extract connectivity to level: 0

Derive extractable layers from constraint group:
virtuosoDefaultExtractorSetup

Diagnostics Report

☐ Keep shape assignment

☒ Connect unassigned instance terminals

☐ Assign shapes from attached labels

☐ Assign shapes from overlapping labels as defined by 'stampLabelLayers' rule

☐ Verify well and substrate connections

☒ Verify weak-connect violations

☒ Verify must-connect violations

☒ Verify open violations

☒ Verify unimplemented instance terminals

☒ Verify blockage violations

☒ Verify pins against routing blockages

	By net	By cellview
Maximum number of open violations:	100	1000
Maximum number of short violations:	100	1000
Maximum number of illegal connections:		1000

Space-based Router

Extract: pcells

Extracted Pin Style

☒ Labeled Shapes Only

☐ Connected Shapes on Same Layer

☐ Whole Net on Routing Layers

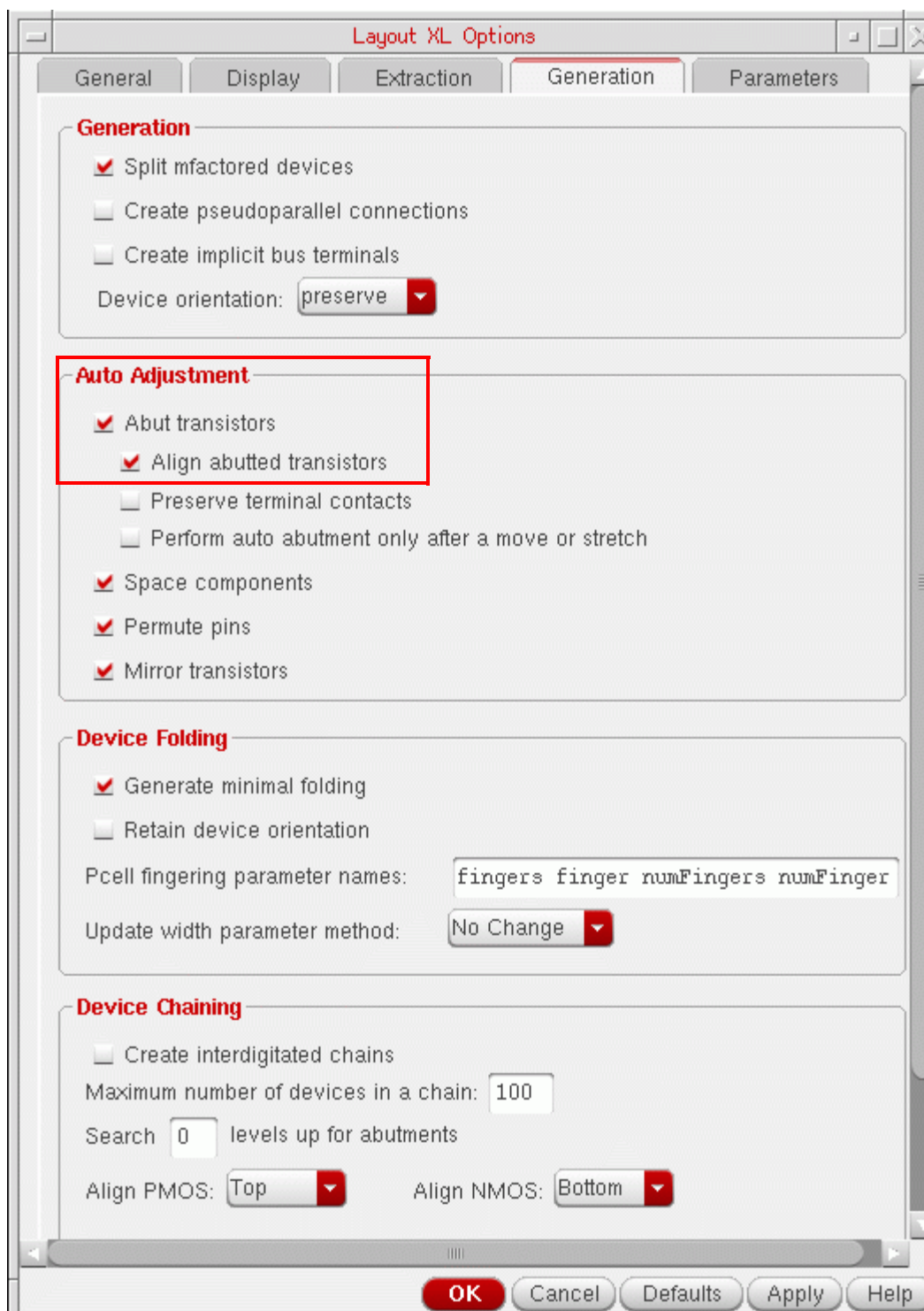
☒ Extract Connectivity through Poly

OK Cancel Defaults Apply Help

Virtuoso Layout Suite XL User Guide

Device Abutment

2. Click the Generation Tab and select *Abut transistors*.



3. From the layout window menu bar, choose *Generate Selected From Source*.

The Generate Selected Components form appears.

4. In the schematic, select the instances you want to place, then click in the layout where you want to place the first and second instances.

The software places the instances where you click in the same order that you selected them.

5. Where the flight lines indicate that abutment is possible, place the devices so that the pins overlap.

The devices are chained automatically.

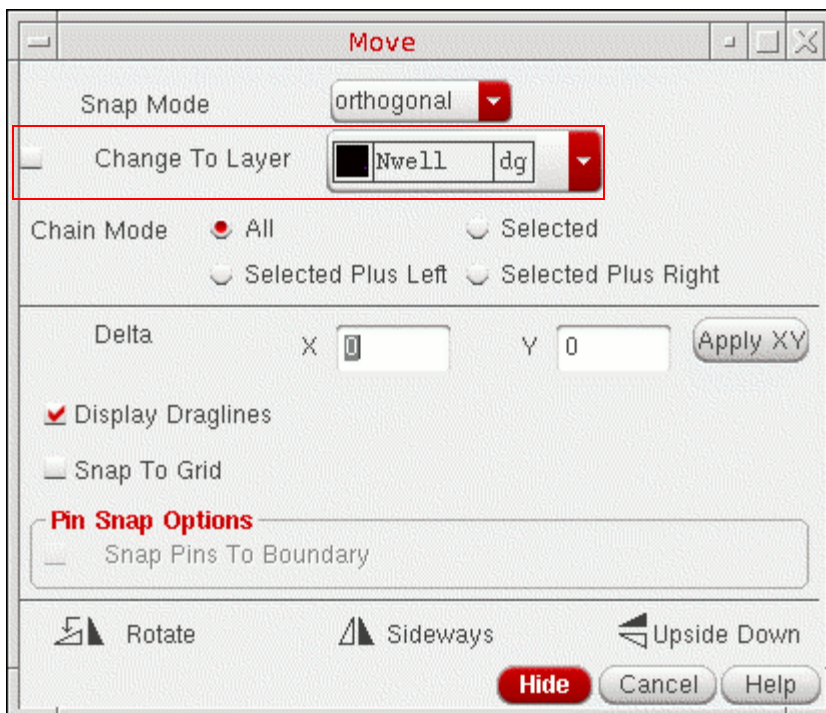
Moving Chained Transistors

To move chained devices while maintaining the chains,

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

The Move form is displayed.

2. Select a chained device and change the *Chain Mode* to *All*.



3. Click on the reference point for the move.

All the devices in the chain are highlighted.

4. Click in the layout where you want to place the devices.
5. Change the *Chain Mode* option to *Selected* and click on the selected device.

The selected device is moved and the device is no longer a part of the chain.

Note: To move the device back into the chain, choose *Edit – Undo*.

6. Select a device in the center of the chain and change the *Chain Mode* to *Selected Plus Right*.
7. Click on the reference point for the move.

You see that all the devices to the right of the selected device remained chained as you move the devices.

Note: *Selected Plus Left* option selects all chained devices to the left of the selected device.

8. Click in the layout where you want to place the devices.

Automatic Device Abutment

Use the following commands to automatically abut devices.

- Layout generation automatically abuts transistors when the *Chaining* option is selected. For more information, see [Chaining Transistors Automatically during Layout Generation](#).
- The *Update Components And Nets* command automatically abuts when the *Chaining* option is selected. For more information, see [Chaining Transistors Automatically when Updating Components and Nets](#).

Chaining Transistors Automatically during Layout Generation

To chain transistors automatically during layout generation,

1. From the layout window menu bar, choose *Connectivity – Generate – All From Source*.

The [Generate Layout](#) form is displayed.

2. On the *Generate* tab, select either of the Chaining options:

- ☐ To automatically chain the transistors, select the *Chaining* check box and click *OK*.

The software chains the transistors automatically during layout generation.

- ☐ To automatically chain the individual folds of a transistor, select the *Chain Folds* check box and click OK.

The software automatically chains the individual folds of a transistor. Note that the *Chain Folds* check box is active only when *Chaining* is deselected and *Folding* is selected.

Note: The results of the Chaining operation are affected by the state of the [IxStackPartitionParameters](#) environment variable.

For more information on generating a layout, see [Generating All Components from Source](#).

Chaining Transistors Automatically when Updating Components and Nets

To chain transistors automatically when updating components and nets,

1. From the layout window menu bar, choose *Connectivity – Update – Components And Nets*.

Note: If the schematic has any devices that are missing from the layout, the Generate Layout form is displayed.

2. On the *Generate* tab, select either of the chaining options:

- ☐ To automatically chain the transistors, select the *Chaining* check box and click *OK*.

The software chains the transistors automatically during layout generation.

- ☐ To automatically chain the individual folds of a transistor, select the *Chain Folds* check box and click OK.

The software automatically chains the individual folds of a transistor. Note that the *Chain Folds* check box is active only when *Chaining* is deselected and *Folding* is selected.

Important

Update Components And Nets chains only new devices created by the command itself. The new chains are separate and do not attach to old chains. Devices that had been chained prior to running *Update Components And Nets* are not moved from their existing positions.

For more information on this command, see [Updating Components and Nets](#).

Abutting Parameterized Cells and Quick Cells

Abutment does not require that two instances share the same supermaster. A parameterized cell (pcell) instance can abut with any other pcell instance as long as the respective instance pins are on the same layer or equivalent layers and the `abutClass` properties have the same value. The same is true for Quick Cells (QCells).

Note: Pcells that have been liquid edited try to preserve their edits during the abutment. For more information on liquid editing of pcells, see [Liquid Editing](#) in the *Virtuoso Parameterized Cell Reference User Guide*.

This section describes how to change the abutment code in your pcells so that they will successfully abut with QCells in Layout XL. It covers the following topics.

- [Recognizing a QCell](#)
- [Understanding QCell Parameters](#)
- [Specifying Whether to Use QCell or Pcell Rules](#)
- [Sample `userAbutWithQCell` Function](#)

Recognizing a QCell

When abutment is triggered between two instances, the abutment engine normally calls the `abutFunction` of the moving instance. However, if one of the instances is a QCell, the abutment engine instead calls the `abutFunction` of the user-defined pcell. By modifying the existing user `abutFunction` for the pcell in question, you can set up your user-defined pcells to abut with QCells.

The `abutFunction` must first recognize that one of the instances is a QCell. You do this using the `techIsInstQcell` SKILL function in a function similar to the one illustrated below.

```
procedure(userAbutFunc(instA instB pinA pinB pinAside conn
                      event@optional group)
  let((
    when(techIsInstQcell(instA)
      return(userAbutWithQCell(instA instB pinA pinB
                              conn event group))
    )
    when(techIsInstQcell(instB)
      return(userAbutWithQCell(instB instA pinB pinA
                              conn event group))
    )
  )
  ;;;;;;;;; The following code remains the same
  .....
```

To simplify the code, the first instance passed to `userAbutWithQCell` is the QCell instance; the second one is a user-defined pcell. The `pinAside` parameter is not passed to the

`userAbutWithQCell` function; if you want to pass this parameter, you must pass more information to tell `userAbutWithQCell` which instance is really `instA`.

Understanding QCell Parameters

The main task performed by the user-defined `abutFunction` is to change the topology of instances by modifying their parameters. Therefore, it is crucial that you understand the parameters used to define QCells. The section describes the most important QCell parameters used in `abutment`.

■ `leftAbutmentState` and `rightAbutmentState`

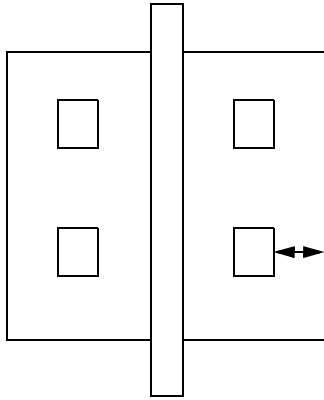
These parameters control the left and right pin shapes. You can change them to get different pin shapes in order to abut the user-defined pcells with QCells.

There are five valid integer values: 0, 1, 2, 3, 4. The default value is 1, which indicates that the QCell is in default status. All other values imply different enclosure values of diffusion and poly, and they also mean there is no contact.

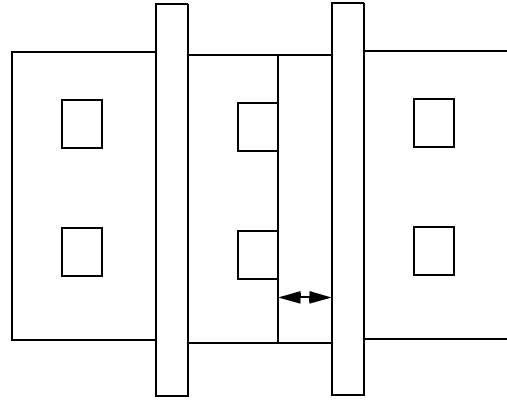
Virtuoso Layout Suite XL User Guide

Device Abutment

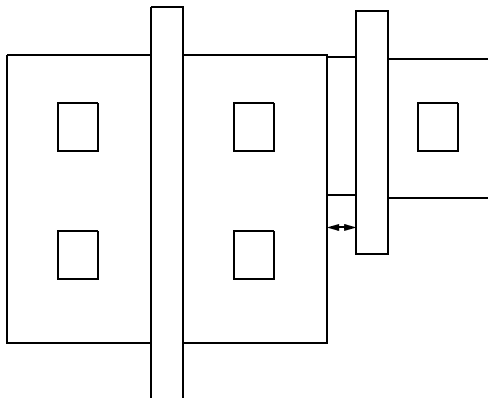
The following illustrations show the different values and how they apply in abutment; left means `leftAbutmentState`, right means `rightAbutmentState`.



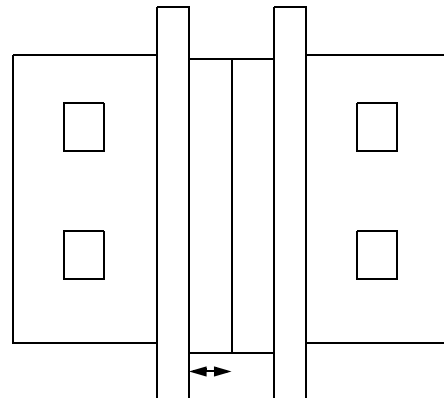
left = 1, right = 1
This is the default status



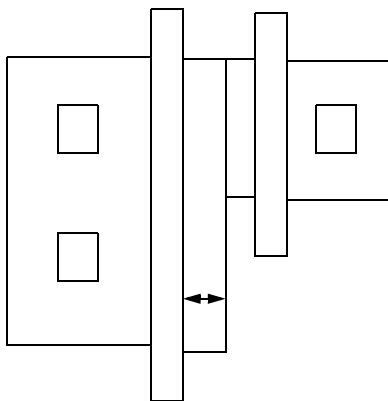
left = 0
enc = polyToCont



left = 2
enc = polyToDiff



left = 3
enc = polyToPoly/2

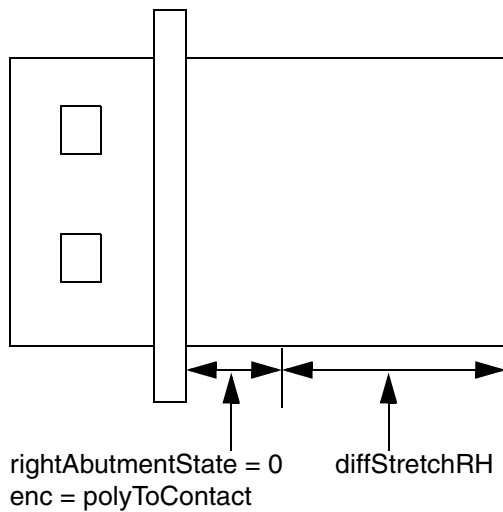


left = 4
enc = diffPolyEnc

■ `diffStretchLH` and `diffStretchRH`

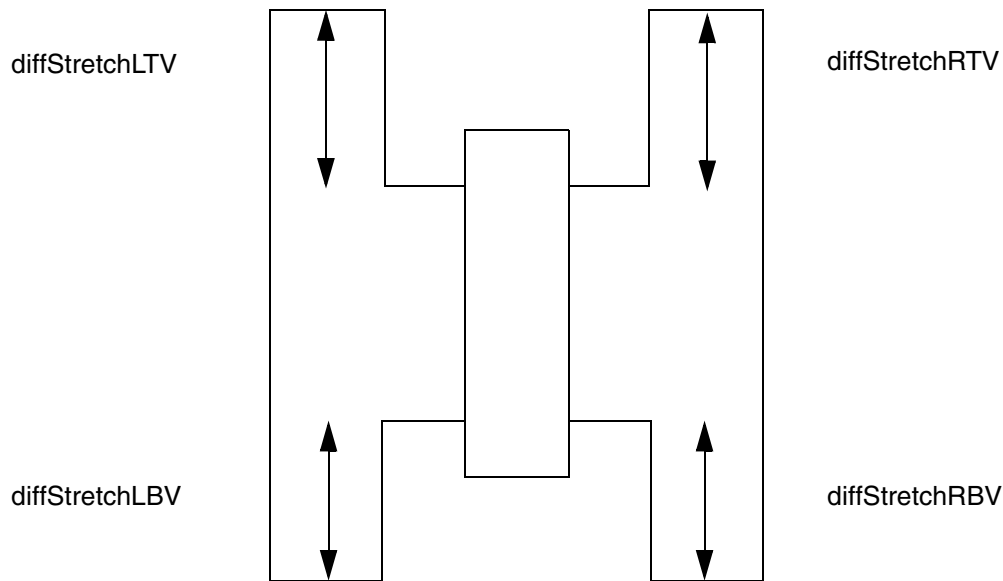
These parameters control the diffusion layer stretch. The value you specify here adds to the enclosure defined by `leftAbutmentState` and `rightAbutmentState` and is the real enclosure of diffusion and gate.

The following illustration shows `diffStretchRH` as an example.



■ `diffStretchLTV`, `diffStretchLBV`, `diffStretchRTV`, and `diffStretchRBV`

These parameters control the “shoulder” stretch of QCells, as shown in the illustration below. They let you maximize the area efficiency of the final abutment result by, for example, making the QCell L-shaped after abutment.



Caution

The sample code in this section simply sets these four parameters to 0 to set “traditional” abutment parameters. It does not describe all the abutment possibilities.

Specifying Whether to Use QCell or Pcell Rules

You can overwrite technology rules when defining a QCell, while user-defined pcells can also have their own rules that are different from the QCell rules. Different rules usually mean a different use model for the QCells and pcells. For example, a QCell with a certain rule might be used for low voltage, while a user-defined pcell with a different rule might be used for high voltage.

For example, if the QCell `polyToPoly` rule is set to 1.0, and the same rule of a user-defined pcell is set to 2.0, you may not want to abut these cells. To prevent this from happening, you must assign a different abutment class to each cell.

If you are abutting only QCells, you do this by specifying the `abutClass` property when you are defining your QCell using the graphical user interface. Simply assign different values for the `abutClass` property of the different QCells and they will never be allowed to abut with each other.

If you do want to abut cells with different rules, you can set the abutment code to get the rules you require using the `spcGetQCellDesignRules` function, which specifies which instance rules take precedence when the instances are abutted. You can set this function to reference either the QCell or the user-defined pcell.

Sample `userAbutWithQCell` Function

You can find abutment code for a sample pcell updated to abut with QCells at the following location in the Cadence installation.

```
your_install_dir/tools/dfII/samples/ROD/rodPcells/component/mos/mos.il
```

Note: The code shown in this section differs slightly from the `mos.il` specified above because only the QCell and sample pcell abutment-related functions are included here. For all other functions referenced but not included in this example, see to the `mos.il` file.

```
defstruct(spcQCellMosParams
  fingerWidth
  widthLT
  widthLB
  widthRT
  widthRB
)

defstruct(spcQCellDesignRules
  gateSpacing
  gateDiffSpacing
  diffGateEnc
  gateContSpacing
  diffContEnc
)

procedure(spcMosAbutWithQCell(instA instB pinA pinB conn event group)
  let((oldValues sContP sStrP sContV sStrV qCellParams qContP
    (delta 1e-6) instAName instBName qCellRules diff qContV
    result)

    qCellParams = spcGetQCellAbutParams(instA)
    qCellRules = spcGetQCellDesignRules(instA)
    instAName = instA~>name
    instBName = instB~>name
    unless(event == 3 diff = pinB~>w - qCellParams->fingerWidth)

    case(event
      (1
        result = 0.0
      )
      (2
        ;; Restore old values for sample pcell.
```

Virtuoso Layout Suite XL User Guide

Device Abutment

```
sContP = pinB->contactParam
sStrP = pinB->stretchParam
sContV = get(instB->master->superMaster->parameters
             sContP)
sContV = sContV && sContV != "FALSE"
sStrV = get(instB->master->superMaster->parameters
             sStrP)

oldValues = list(
    list(instBName sContP "boolean" sContV)
    list(instBName sStrP "float" sStrV)
)

;;; When abutting a QCell, set diffStretchLTV and the other
;;; three parameters to 0 if they are not.
qContP = pinA->contactParam
qContV = get(instA qContP) || 1
if(qContP == "leftAbutmentState" then
    when(abs(qCellParams->widthLT) > delta
        dbReplaceProp(instA "diffStretchLTV" "float" 0.0)
        oldValues = cons(list(instAName "diffStretchLTV"
                                "float" qCellParams->widthLT)
                           oldValues)
    )
    when(abs(qCellParams->widthLB) > delta
        dbReplaceProp(instA "diffStretchLBV" "float" 0.0)
        oldValues = cons(list(instAName "diffStretchLBV"
                                "float" qCellParams->widthLB)
                           oldValues)
    )
else
    when(abs(qCellParams->widthRT) > delta
        dbReplaceProp(instA "diffStretchRTV" "float" 0.0)
        oldValues = cons(list(instAName "diffStretchRTV"
                                "float" qCellParams->widthRT)
                           oldValues)
    )
    when(abs(qCellParams->widthRB) > delta
        dbReplaceProp(instA "diffStretchRBV" "float" 0.0)
        oldValues = cons(list(instAName "diffStretchRBV"
                                "float" qCellParams->widthRB)
                           oldValues)
    )
)

oldValues = cons(list(instAName qContP "int" qContV)
                  oldValues)
case(conn
(1    ;;; if no third connection, then remove both contacts
    dbReplaceProp(instB sContP "boolean" nil)
    cond(
        (diff > delta    ;;; sample Pcell width > QCell width
         dbReplaceProp(instA qContP "int" 2)
         dbReplaceProp(instB sStrP "float"
                        qCellRules->diffGateEnc)
        )
    (diff < -delta    ;;; sample Pcell width < QCell width
     dbReplaceProp(instA qContP "int" 4)
     dbReplaceProp(instB sStrP "float"
                    qCellRules->gateDiffSpacing)
    )
)
```

Virtuoso Layout Suite XL User Guide

Device Abutment

```

(t      ;; Same width
  dbReplaceProp(instA qContP "int" 3)
  dbReplaceProp(instB sStrP "float"
                  qCellRules->gateSpacing/2.0)
)
)
result = t
)
(2      ;; Keep contact
  cond(
    (diff > delta      ;; sample pcell width > QCell width
      dbReplaceProp(instB sContP "boolean" t)
      dbReplaceProp(instA qContP "int" 2)
    )
    (diff < -delta     ;; sample pcell width < QCell width
      dbReplaceProp(instA qContP "int" 1)
      dbReplaceProp(instB sContP "boolean" nil)
      dbReplaceProp(instB sStrP "float"
                      qCellRules->gateDiffSpacing)
    )
  )
  (t      ;; same width
    dbReplaceProp(instA qContP "int" 0)
    dbReplaceProp(instB sContP "boolean" t)
    dbReplaceProp(instB sStrP "float"
                    qCellRules->gateContSpacing
                    +qCellRules->contWidth)
  )
  )
  result = t
)
(t      printf("      Unknown conn %L\n" conn)
  result = nil
)
)
when(result
  dbReplaceProp(group "oldValues" "list" oldValues))
)
(3
  spcRestoreInstsProp(instA instB group)
  result = t
)
)
result
)
)
procedure(spcGetQCellAbutParams(inst)
  let((qCellParams superMaster params values)

    superMaster = inst->master->superMaster
    params = setof(prop superMaster->prop prop->name ==
                    "parameters")
    params = car(params)
    values = params->value

    qCellParams = make_spcQCellMosParams()

    qCellParams->fingerWidth = inst->fingerWidth ||
                              spcFindValue(values "fingerWidth")
    qCellParams->widthLT = inst->diffStretchLTV ||

```


Virtuoso Layout Suite XL User Guide

Device Abutment

```

                                spcFindValue(values "diffStretchLTV")
qCellParams->widthLB = inst~>diffStretchLBV ||
                                spcFindValue(values "diffStretchLBV")
qCellParams->widthRT = inst~>diffStretchRTV ||
                                spcFindValue(values "diffStretchRTV")
qCellParams->widthRB = inst~>diffStretchRBV ||
                                spcFindValue(values "diffStretchRBV")

    qCellParams
  )
)

procedure(spcFindValue(values name)
  prog(()
    foreach(value values
      when(value~>name == name return(value~>value))
    )
    return(nil)
  )
)

procedure(spcGetQCellDesignRules(inst)
  let((superMaster values tfId gateLayer diffLayer rules contLayer)

    rules = make_spcQCellDesignRules()

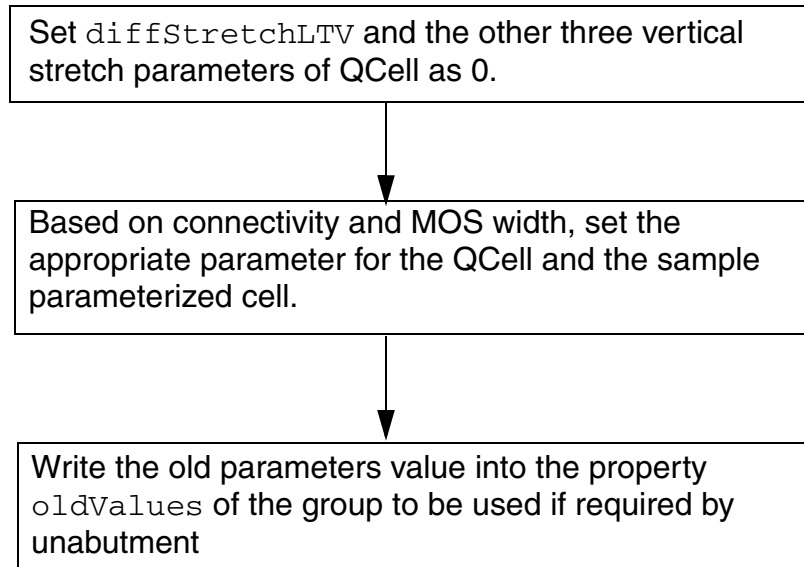
    superMaster = inst~>master~>superMaster
    values = superMaster~>classParameters~>value
    tfId = techGetTechFile(inst~>master)

    gateLayer = spcFindValue(values "gatePolyLayer")
    diffLayer = spcFindValue(values "diffusionLayer")
    contLayer = spcFindValue(values "contactLayer")

    rules->gateSpacing = spcFindValue(values "polyPolySpacing") ||
      techGetSpacingRule(tfId "minSpacing" gateLayer)
    rules->gateDiffSpacing =
      spcFindValue(values "polyDiffusionSpacing") ||
      techGetSpacingRule(tfId "minSpacing" gateLayer
        diffLayer)
    rules->diffGateEnc =
      spcFindValue(values "diffusionPolyEnclosure") ||
      techGetOrderedSpacingRule(tfId "minEnclosure"
        diffLayer gateLayer)
    rules->gateContSpacing =
      spcFindValue(values "polyContactSpacing") ||
      techGetSpacingRule(tfId "minSpacing" gateLayer
        contLayer)
    rules->diffContEnc =
      spcFindValue(values "diffusionContactEnclosure") ||
      techGetOrderedSpacingRule(tfId "minEnclosure"
        diffLayer contLayer)
    rules->contWidth = spcFindValue(values "xContactLayerW") ||
      techGetSpacingRule(tfId "minWidth" contLayer)

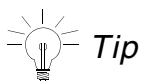
    rules
  ))
)
```

In case of event 1, the code calculates the offset. In event 2, it updates the parameters of the sample pcell and QCell and stores the old parameters. Finally, in event 3, it restores the old parameter values stored in event 2 to apply unabutment. In event 2, the basic algorithm is:



Abutment and the Virtuoso Chip Assembly Router

If you are using the Virtuoso Chip Assembly Router to route your design and you remove a device from a chain, the missing contact is not restored to the device.



Tip

Perform interactive placement operations that involve abutment changes in Layout XL before you route the design using the Virtuoso Chip Assembly Router.

Generating a Layout

This chapter shows you how to use the Virtuoso® Layout Suite XL layout editor (Layout XL) to generate a layout and make an initial placement of components in it.

This chapter covers the following topics:

- [Naming Conventions](#) on page 215
- [Place and Route Boundary](#) on page 220
- [Generating All Components from Source](#) on page 222
- [Generating Selected Components from Source](#) on page 239
- [Generating Clones](#) on page 252
- [Generating Synchronous Clones](#) on page 272
- [Generating Synchronous Copy](#) on page 286
- [Loading Physical Information from Another Cellview](#) on page 297

Naming Conventions

When generating and manipulating components in the layout view, Layout XL uses the naming conventions outlined in this section.



Manually changing a system-generated name, or renaming an instance such that its new name conflicts with a system-generated name, might lead to unpredictable behavior.

Virtuoso Layout Suite XL User Guide

Generating a Layout

General Convention

By default the OR bar (|) is used to delimit hierarchy and to prefix the names of all layout instances generated by Layout XL.

Schematic	Layout
P0	P0
NAND0 / P0	NAND0 P0

You can generate instance names without the leading OR bar by setting the `prefixLayoutInstNamesWithPipe` environment variable to `nil`.

Schematic	Layout
P0	P0
NAND0 / P0	NAND0 P0

Mfactored Devices and Folded Devices

Devices specified using the `mfactor` property in the schematic and those that are folded in the layout are named `instName.integer` in the layout.

For example, if you have a schematic instance named P0 with

```
mfactor = 3
```

the layout instances generated are named as follows.

Schematic	Layout (with OR bar)	Layout (without OR bar)
P0	P0.1	P0.1
	P0.2	P0.2
	P0.3	P0.3

Series-Connected Devices

Devices specified using the `sfactor` property in the schematic are named `instName.sinteger` in the layout.

Virtuoso Layout Suite XL User Guide

Generating a Layout

For example, if you have a schematic instance named `R0` with

```
sfactor = 3
```

the layout instances generated are named as follows.

Schematic	Layout (with OR bar)	Layout (without OR bar)
R0	R0.s1	R0.s1
	R0.s2	R0.s2
	R0.s3	R0.s3

lxCombination Devices

Devices defined using the `lxCombination` property in the schematic are named *instName.msinteger* in the layout.

For example, if you have a schematic instance named `R0` with

```
lxCombination = (2k*2)/3+1500/4
```

the ten layout instances generated are named as follows:

Schematic	Layout (with OR bar)	Layout (without OR bar)
R0	R0.ms1	R0.ms1
	R0.ms2	R0.ms2
	R0.ms3	R0.ms3
	.	.
	.	.
	.	.
	R0.ms10	R0.ms10

Nets in Series-Connected and lxCombination Devices

Nets associated with series-connected and `lxCombination` devices are named *instName.sinteger.expandedNetinteger* in the layout.

For example,

```
R0.s1.expandedNet1
```

where “expandedNet” signifies that the layout net has no real match in the schematic.

Constraint Transfer

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

The constraint transfer is triggered:

- Automatically when you start Layout XL.
 - Constraints in the schematic are transferred to the top-level layout view. Constraints that have been created or changed in the schematic but not yet saved are also transferred.
- Automatically when you use the *Generate All From Source* command. Previously-transferred constraints are deleted and updated constraints are created in the layout.
- Manually when you use the *Update Layout Constraints* command. For more information, see [Updating Layout Constraints](#).
- Optionally, when you load information from another physical cellview. For more information, see [Loading Physical Information from Another Cellview](#).

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

Constraint Manager Assistant

You can manipulate these constraints on both schematic and layout views using the Constraint Manager assistant.

This lets you

- Enter custom constraints in the schematic to drive layout. These are transferred to layout when Layout XL is initialized.
- Enter additional constraints in the layout, which are supported by the various physical design applications.
- Select a constraint in the Constraint Manager and see the constraint in question highlighted in the canvas.

Constraints transferred from schematic to layout are read-only and cannot be modified in the layout view. However, you can create new constraints in the layout to override the schematic constraints.

Virtuoso Layout Suite XL User Guide

Generating a Layout

For more information on how to use the Constraint Manager, see [The Constraint Manager Assistant](#) in the *Virtuoso Unified Custom Constraints User Guide*.

Constraint Mapping in One-to-Many Relationships

The following table shows how constraints are mapped when there is a one-to-many schematic to layout relationship.

Constraint type	Schematic	Layout
Symmetry pair	<code>symmetry(A, B)</code>	<code>symmetry(A.1, B.1)</code> <code>symmetry(A.2, B.2)</code> <code>symmetry(A.<N>, B.<N>)</code>
Matched orientation	<code>relOrient(A, B, "orient")</code>	<code>relOrient(A.1, B.1, "orient")</code> <code>relOrient(A.1, ..., A.<N>, "R0")</code> <code>relOrient(B.1, ..., B.<M>, "R0")</code>
Layout structure	<code>layoutStructure(A, B)</code> <code>layoutStructure(A, B, B, A)</code>	<code>layoutStructure</code> <code>(A.1, ..., A.<N>, B.1, ..., B.<M>)</code> <code>SlayoutStructure</code> <code>(A.1, B.1, B.2, A.2)</code>
All other	<code>allOther(A, B)</code>	<code>allOther</code> <code>(A.1, ..., A.<N>, B.1, ..., B.<M>)</code>

Constraint Mapping in Many-to-One Relationships

Many-to-one relationships are handled by not adding duplicate members to layout constraints.

The following situations cause a warning to be issued and no layout constraint generated.

- A symmetry constraint containing a pair of instances with have different multiplication factors.
- A layout structure constraint with schematic instances specified multiple times but which do not have matching layout instances; for example, the schematic instance is referenced three times in the constraint but has an mfactor of only 2 in the layout.

Creation and Synchronization of Clusters

To promote interoperability with designs that originate in Encounter and allow layout placement tools to use `oaCluster` constraints, the constraint transfer mechanism converts

a `ciCon(cluster)` constraint in the schematic to an `oaCluster` constraint in the layout and keeps the two synchronized during subsequent transfer operations.

Important

`ciCon(cluster)` constraints and `oaCluster` constraints are synchronized only in a forward direction (from `ciCon(cluster)` to `oaCluster`). A change to an `oaCluster` does not affect its corresponding `ciCon(cluster)`.

Note: `oaCluster` constraints with no corresponding `ciCon(cluster)` constraints are never deleted by constraint transfer operations. If you delete from the layout an `oaCluster` constraint with a corresponding `ciCon(cluster)` constraint, the `oaCluster` is recreated the next time constraints are transferred from the schematic.

Place and Route Boundary

When you initialize a layout, Layout XL automatically generates a place and route boundary for the design.

Setting the Default Size for the Place and Route Boundary

To set the default size for the place and route boundary,

1. From the layout window menu bar, choose *Connectivity – Generate – All From Source*.

The Generate Layout form is displayed.

2. On the *Generate* tab, turn on the *PR Boundary* option.
3. On the *PR Boundary* tab, set the options to generate the shape and size of boundary you need.

For more information, see PR Boundary Tab.

4. When you click *OK* to generate the layout, the system draws the specified boundary in the layout canvas.

Creating a Place and Route Boundary Manually

To create a design boundary manually,

- Use *Create – P&R Objects – P&R Boundary*.

For more information, see [Creating Boundaries](#) in the *Virtuoso Layout Suite L User Guide*.

Stretching, Moving and Deleting the Place and Route Boundary

To stretch, move, or delete the design boundary,

- Use the layout editor *Edit* commands.

For more information on these commands, see [Editing Objects](#) in the *Virtuoso Layout Suite L User Guide*.

Moving Components Inside the Place and Route Boundary

To move all generated components and pins inside the boundary,

- From the layout window menu bar, choose *Connectivity – Generate – Place As In Schematic*.

For more information, see [Moving Generated Components into the Design Boundary](#).

Updating the Placement Status Automatically

To automatically update the placement status of instances and pins when they are moved into or out of the PR boundary,

- Type the following in the CIW.

```
envSetVal("layoutXL" "updatePlacementStatus" 'boolean t)
```

For details, see [updatePlacementStatus](#).

This environment variable is honored by all interactive commands and automatic tools which cause instances and pins to be moved into or outside the PR boundary. It also includes situations where the PR boundary is moved or stretched to enclose or exclude an instance or pin.

When switched on, instances and pins with status `unknown` that are wholly inside the PR boundary after the edit are updated to status `placed`. Instances and pins with status `placed` that are wholly outside the PR boundary after the edit are updated to `unknown`.

To update the placement status of instances or pins in designs that are loaded in Layout XL for the first time, or which have been edited outside the Layout XL or GXL environment, do one of the following.

- Choose the *Place – Update Placement Status* menu command.
- Use the lxUpdatePlacementStatus SKILL function.

Generating All Components from Source

Use the *Generate All From Source* command to generate layout representations of the schematic design components. *Generate All From Source* deletes any existing components in the layout view and generates everything from scratch.

To generate all components from source,

1. From the layout window menu bar, choose *Connectivity – Generate – All From Source*.

Virtuoso Layout Suite XL User Guide

Generating a Layout

The Generate Layout form is displayed.

The screenshot shows the 'Generate Layout' dialog box. It has a title bar with the text 'Generate Layout' and standard window controls. Below the title bar are four tabs: 'Generate', 'I/O Pins', 'PR Boundary', and 'Floorplan'. The 'Generate' tab is selected. Inside the 'Generate' tab, there are several sections:

- Generate** (red text):
 - ☒ Instances
 - ☐ Chaining
 - ☐ Folding
 - ☐ Chain Folds
 - ☒ I/O Pins
 - ☐ Except Global Pins
 - ☐ Except Pad Pins
 - ☒ PR Boundary
 - ☐ Snap Boundary
- Device Correspondence** (red text):
 - ☐ Preserve User-Defined Mappings
- Connectivity Extraction** (red text):
 - ☒ Extract Connectivity After Generation

At the bottom of the dialog are four buttons: 'OK' (red), 'Cancel', 'Defaults', and 'Help'.

The form remembers values set previously in the current Virtuoso session. When you open it for the first time, it shows the default values set in your `.cdsenv` file.

You can also load predefined cellview information from an existing layout cellview. See Loading Physical Information from Another Cellview.

2. Set the options on the form as needed. For more information, see
 - a. Specifying the Components to be Generated
 - b. Specifying the I/O Pins to be Generated
 - c. Specifying the Place and Route Boundary

d. Preserving Floorplanning Objects

3. Click *OK* in the Generate Layout form.

If you already have design elements in the layout canvas, a message box warns you that this command deletes all layout instances and wiring in the design.

Click *Yes* to dismiss the message and generate the layout view you specified.

Layout XL generates instances and pins equivalent to the symbols and terminals in the schematic view. Instances and pins are placed below the place and route boundary in the same relative positions as in the schematic. However, you can choose to place the instances and pins within the place and route boundary (design boundary) by setting the lxGenerateInBoundary environment variable.

Iterated pins are sorted and placed together in the layout. Constraints defined in the schematic are transferred to the layout view, as are the signal types assigned to individual nets.

For the bus terminal and bus bit order (ascending or descending) information to be transferred from the schematic to the layout, you need to set the environment variable, `createImplicitBusTerminals`, before running the *Generate All From Source* command. This allows for the bus information to be implicitly annotated on the layout side, removing the need for running *VerilogAnnotate*. For more information on the environment variable, see `createImplicitBusTerminals`.

Alternatively, you can choose to implicitly create bus terminals by selecting the **Create Implicit bus terminals** option on the *Generation* tab. For more information, see *Generation Tab*.

Troubleshooting Generate All From Source

Any problems encountered during layout generation are reported in the CIW and in a Layout XL Info text window. Use *Check Against Source* to get more information on the problems encountered and how to resolve them.

Generate All From Source and Schematic Parameters

- *Generate All From Source* copies all the schematic parameters to the new layout instance with the following exceptions.

- ☐ `mfactor` and `sfactor`
- ☐ Parameters that are always ignored.

- `instancesLastChanged`
 - `instNamePrefix`

Virtuoso Layout Suite XL User Guide

Generating a Layout

```
lxIgnoreParamForCAS  
lxIgnoredParams  
lxMFactorNum  
lxParamsToIgnore  
lxParamsToIgnoreForCheck  
lxPlacementStatus  
lxRounding  
lxStopList  
lxTimeStamp  
lxUseCell  
pin#  
posi
```

- ☐ Parameters set using the `lxParamsToIgnore` property or in the **Parameters Tab** form.
- *Generate All From Source* applies all CDF callbacks before any instance is created if `lxEvalCDFCallbacks` is set to `t`. The complete list of parameters found on the schematic instance (including the adjusted folding, `sFactor`, or `lxCombination` parameters) is used to call the CDF; the resulting list is passed to the layout instance.
- *Generate All From Source* does not maintain the value of the `lxMaxWidth` parameter unless all device widths and all `lxDeviceWidth` values are specified consistently; for example, all in meters (10 μm is 1-e-5) or all in user units (10 μm is 10).

Generate All From Source and Inherited Connections

- If you use inherited connections to assign more than one value to a global net, the `netSet` property (which specifies the new value for a global signal) is not copied from the schematic instance to the layout instance.
- Any `netExpression` pin properties that have not been set are copied to the layout terminals. If multiple `netExpression` properties exist for the same terminal name, then you are prompted to select one property to be preserved on the terminal.
- If a layout instance master terminal has an inherited net expression (an implicit inherited connection) for a global layout net, *Generate All From Source* creates a physical-only layout terminal with no schematic equivalent. The terminal has the `physOnly` property set to `t` so that it can be correctly ignored by Layout XL functions such as *Update Components And Nets* and *Check Against Source* and by the digital applications further on in the design flow.

Specifying the Components to be Generated

To specify the components to be generated,

Virtuoso Layout Suite XL User Guide

Generating a Layout

1. Select the Generate Tab of the Generate Layout form.



2. In the *Generate* group box, select the *Instances*, *I/O Pins*, and *PR Boundary* options, as required.
 - a. To chain transistors, select the *Chaining* check box.
 - b. To divide transistors in the design into two or more individual folds, select the *Folding* check box.
 - c. To chain the individual folds of a transistor, select the *Chain Folds* check box.

Note: For the *Chain Folds* option to be available, only the *Folding* check box must be selected. If you select the *Chaining* check box as well, *Chain Folds* is deactivated.

Note: Any devices to be chained or folded must belong to a component type which has *Component class* set to *PMOS* or *NMOS* and values set for the *Active layer*, *Width parameter*, and *Folding threshold* parameters. For more information, see [Component Types Mode](#).

- d. To stop Layout XL generating layout pins for global nets in the schematic, select *Except Global Pins*.
- e. To stop Layout XL generating layout pins for schematic pins that are connected to I/O pads, select *Except Pad Pins*.
- f. To generate a snap boundary, select *Snap Boundary*.

You can generate a snap boundary only if the *PR Boundary* option is switched on.

- 3. To preserve any user-defined mapping of devices between the schematic and the layout, turn on *Preserve User-Defined Mappings*.

This option preserves only user-defined one-to-one, many-to-many, many-to-one, and one-to-many device correspondence defined in the [Define Device Correspondence](#) form. It does not report missing devices or shapes within a mapped group. For more information, see [Updating Device Correspondence](#).

- 4. To run connectivity extraction after the layout has been generated, turn on *Extract Connectivity After Generation*.

For more information, see [Connectivity Extraction](#).

- 5. To create a label specifying the library, cell, and view of your design just inside the center top edge of the boundary, set the [createBoundaryLabel](#) environment variable.

Specifying the I/O Pins to be Generated

You specify the pins to be generated on the [I/O Pins Tab](#).

For each pin listed, the form shows the parameters that will be used to generate its equivalent in the layout. You can remove or change the specification of any of the listed pins, or add new pins to be generated.

When generating pins, Layout XL uses the same naming convention as the Virtuoso Schematic Editor, allowing you to assign different names to terminals and nets. Where terminal and net names are different in the schematic, Layout XL creates pins with the same terminal name as in the layout, even though the net name associated with the pin might be different.

By default this section lists all the top-level pins detected in the schematic design, including schematic pins that are connected to I/O pads (cells of type `pad`, `padSpacer`, or `padAreaIO`). To generate only pads but no pins, turn on the *Except Pad Pins* option on the Generate Tab (or on the Update Tab if you are using the Update Components and Nets form).

Power and ground pins defined at a lower level of the design hierarchy are not listed on the *I/O Pins* tab but are nevertheless generated in the layout view. The system issues an information message to tell you what has happened.

The system generates a pin for a stub net (a non-global `netSet` automatically created by the schematic extractor) only if the stub net is connected to an instance terminal. If it is unconnected, no pin is generated.

Layout XL also correctly propagates connectivity to nets that are connected via patchcords.

Specifying Default Values for All Pins

To specify default values for all pins,

Virtuoso Layout Suite XL User Guide

Generating a Layout

1. Bring up the I/O Pins Tab of the Generate Layout form.

Generate Layout

Generate I/O Pins PR Boundary Floorplan

Specify Default Values for All Pins

Layer: Width: Height: Num: Create:

☐ Metal1 dg 0.12 0.12 1 ☒ Apply

Specify Pins to be Generated

Select: IN Number Of Matches: 2 Add New Pin...

Term Name	Net Name	Layer	Width	Height	Num	Create
"INN"	"INN"	("Metal1" "drawing")	0.12	0.12	1	t
"INP"	"INP"	("Metal1" "drawing")	0.12	0.12	1	t
"IREF"	"IREF"	("Metal1" "drawing")	0.12	0.12	1	t
"OUT"	"OUT"	("Metal1" "drawing")	0.12	0.12	1	t
"SIDDQ"	"SIDDQ"	("Metal1" "drawing")	0.12	0.12	1	t
"VDD"	"VDD"	("Metal1" "drawing")	0.12	0.12	1	t
"VSS"	"VSS"	("Metal1" "drawing")	0.12	0.12	1	t

Name: Layer: Width: Height: Num: Create:

AS IS ☐ Metal1 dg 0.12 0.12 1 ☒ Update

Pin Label

☒ Create Label As: ☒ Label Options...
☐ Text Display

OK Cancel Defaults Help

Specify default values for all pins and click *Apply*.

2. Choose the default routing layer from the *Layer* cyclic field.

The list of pin layers is obtained from the technology information applied to the design.

3. Specify the default *Height* and *Width* of the pins and the *Number* of pins you want to create.
4. Check the *Create* box to specify that the pins are to be created in the layout view.
5. Click *Apply*.

The specified values are applied to all the pins displayed in the list box and the corresponding environment variables (`initIOPinLayer`, `initPinHeight`, `initPinWidth`, and `initPinMultiplicity`) are set appropriately.

Virtuoso Layout Suite XL User Guide

Generating a Layout

Note: initPinHeight and initPinWidth are set only if the value you specify is greater than the default minimum width set for the layer in the technology file.

Specifying Values for Selected Pins

To update the values of selected pins in the list,

1. Bring up the I/O Pins Tab of the Generate Layout form.

Generate Layout

Generate I/O Pins PR Boundary Floorplan

Specify Default Values for All Pins

Layer: dg Width: 0.12 Height: 0.12 Num: 1 Create: ☒ Apply

Specify Pins to be Generated

Select: IN Number Of Matches: 2 Add New Pin...

Term Name	Net Name	Layer	Width	Height	Num	Create
"INN"	"INN"	("Metal1" "drawing")	0.12	0.12	1	t
"INP"	"INP"	("Metal1" "drawing")	0.12	0.12	1	t
"IREF"	"IREF"	("Metal1" "drawing")	0.12	0.12	1	t
"OUT"	"OUT"	("Metal1" "drawing")	0.12	0.12	1	t
"SIDDQ"	"SIDDQ"	("Metal1" "drawing")	0.12	0.12	1	t
"VDD"	"VDD"	("Metal1" "drawing")	0.12	0.12	1	t
"VSS"	"VSS"	("Metal1" "drawing")	0.12	0.12	1	t

Name: Layer: dg Width: 0.12 Height: 0.12 Num: 1 Create: ☒ Update

Pin Label

☒ Create Label As: ☐ Label ☐ Text Display Options...

OK Cancel Defaults Help

Select the pins to update.

Specify new values and click *Update*.

1. In the list box, select the pins you want to update.
2. Choose the routing layer for the selected pins from the *Layer* cyclic field.

The list of pin layers is obtained from the technology information that applies for the design.

3. Specify the default *Height* and *Width* of the selected pins and the *Number* of each pin you want to create.
4. Check the *Create* box to specify that the selected pins are to be created in the layout view; clear it to prevent a pin from being generated in the layout.
5. Click *Update*.

The selected pins are updated with the new values.

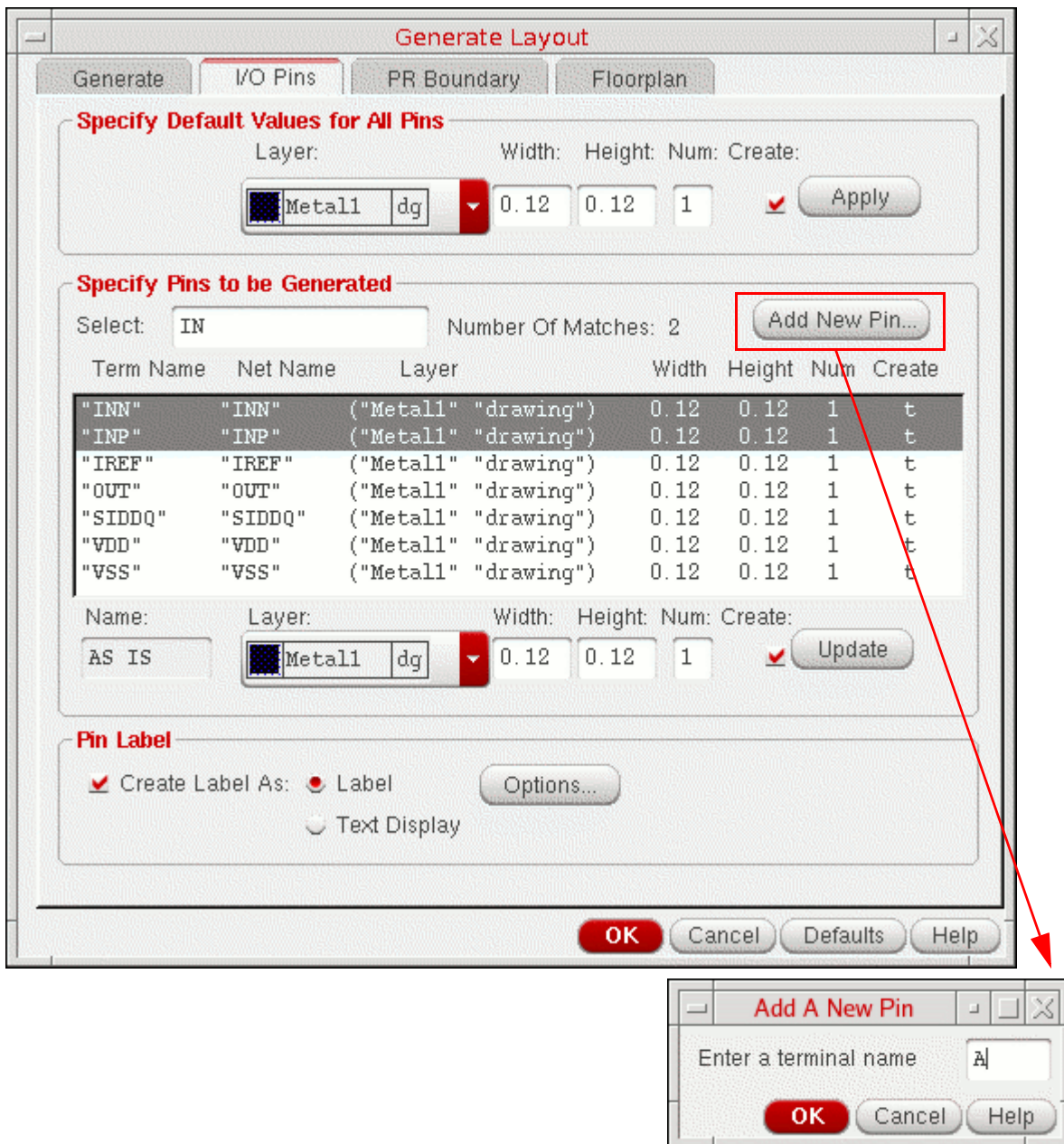
Adding a New Pin

To add a new pin,

Virtuoso Layout Suite XL User Guide

Generating a Layout

1. Bring up the I/O Pins Tab of the Generate Layout form, click *Add New Pin*.



2. Type in a pin name and click *OK*.

If the pin to be added has the same attributes as any of the existing pins in the pin list (pin name, pin layer, and/or purpose), then the added pin is highlighted. You can change the number of the highlighted pin in order to create the number of pins you require.

Creating Pin Labels

To generate name labels for each pin,

1. Bring up the I/O Pins Tab of the Generate Layout form.

Generate Layout

Generate | **I/O Pins** | PR Boundary | Floorplan

Specify Default Values for All Pins

Layer: Metall dg Width: 0.12 Height: 0.12 Num: 1 Create: ☒ Apply

Specify Pins to be Generated

Select: IN Number Of Matches: 2 Add New Pin...

Term Name	Net Name	Layer	Width	Height	Num	Create
"INN"	"INN"	("Metall" "drawing")	0.12	0.12	1	t
"INP"	"INP"	("Metall" "drawing")	0.12	0.12	1	t
"IREF"	"IREF"	("Metall" "drawing")	0.12	0.12	1	t
"OUT"	"OUT"	("Metall" "drawing")	0.12	0.12	1	t
"SIDDQ"	"SIDDQ"	("Metall" "drawing")	0.12	0.12	1	t
"VDD"	"VDD"	("Metall" "drawing")	0.12	0.12	1	t
"VSS"	"VSS"	("Metall" "drawing")	0.12	0.12	1	t

Name: AS IS Layer: Metall dg Width: 0.12 Height: 0.12 Num: 1 Create: ☒ Update

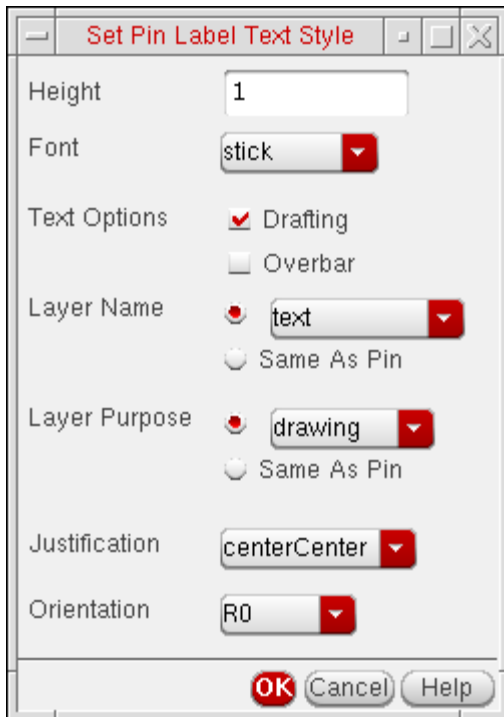
Pin Label

☒ Create Label As: ☒ Label Options...
☐ Text Display

OK Cancel Defaults Help

2. Turn on the *Create Label As* option and choose whether you want to create a *Label* object or a *Text Display* object.
3. If you choose to create *Label* objects, you can set the label style by clicking *Options*.

The Set Pin Label Text Style form is displayed.



4. Choose the options you want and click *OK*.

The corresponding environment variables are listed below.

labelDrafting
labelFontStyle
labelHeight
labelJustify
labelOverbar
orientation
pinTextLayer
pinTextPurpose
pinTextSameLayer
pinTextSamePurpose

Ignoring Mismatched Pins

If a symbol in a schematic has more pins than the corresponding device on the layout, Layout XL generates a warning about mismatched pins. To ignore mismatched pins,

1. In the schematic window menu bar, choose *Edit – Hierarchy – Descend Edit* to descend into the symbol of the instance.

2. In the schematic window, choose *Edit – Properties – Cellview*.

The Edit Cellview Properties form is displayed.

3. Click *Add* to display the Add Property form and add a new Boolean property on the pin, for example, `unusedPin`, and set it to `t`.
4. Add the property name (`unusedPin`) as a value in the `propsUsedToIgnoreObjs` environment variable in your `.cdsenv` file.

```
layoutXL propsUsedToIgnoreObjs string "unusedPin"
```

Or type the following in the CIW.

```
envSetVal ("layoutXL" "propsUsedToIgnoreObjs" 'string "unusedPin")
```

The next time you generate or update a layout or run the *Check Against Source* command, the software does not issue warnings about mismatched or missing pins.

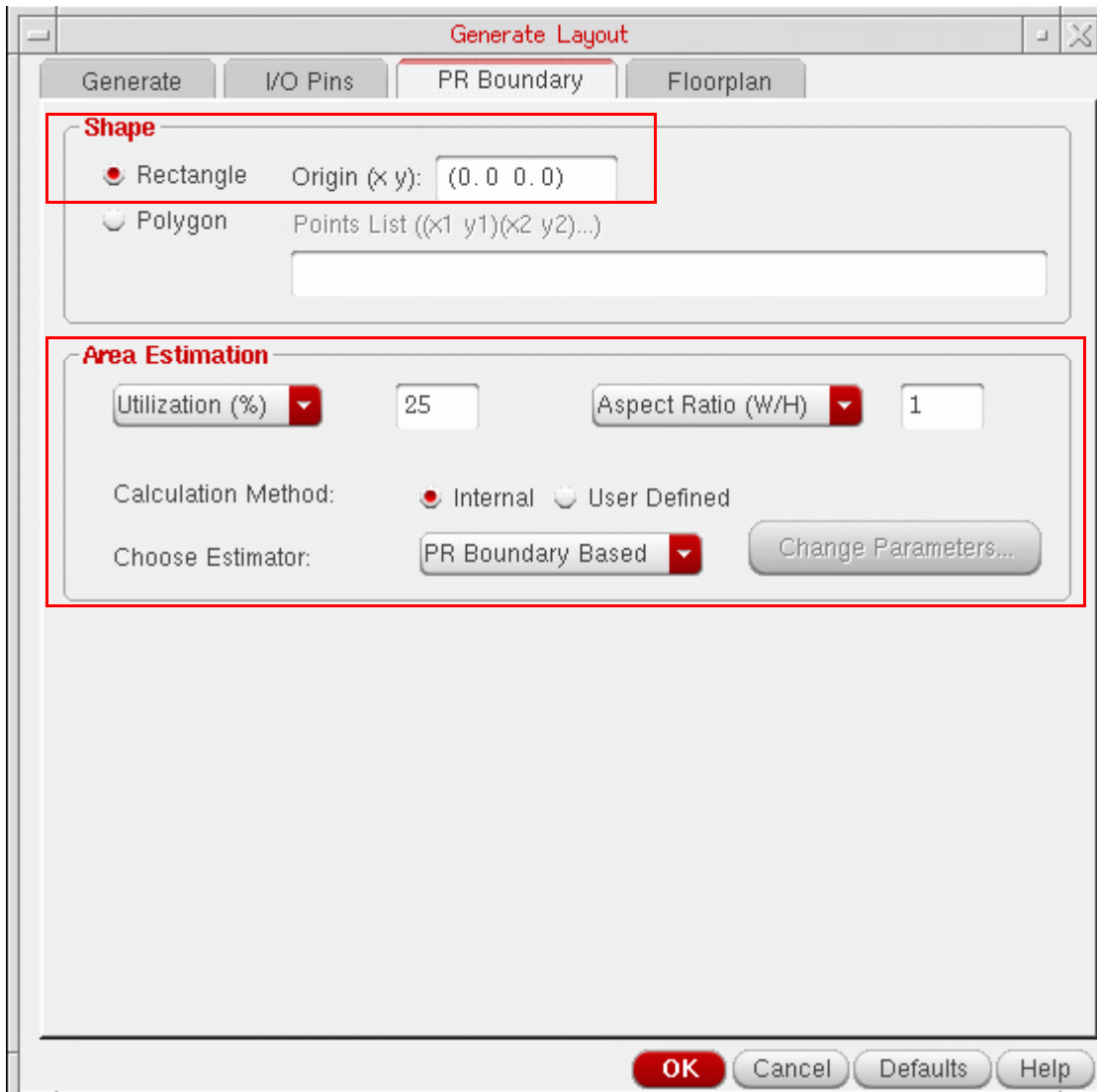
Specifying the Place and Route Boundary

You specify the place and route boundary on the PR Boundary Tab.

Specifying a Rectangular Boundary

To specify a rectangular place and route boundary,

1. On the PR Boundary Tab, choose *Rectangle*.



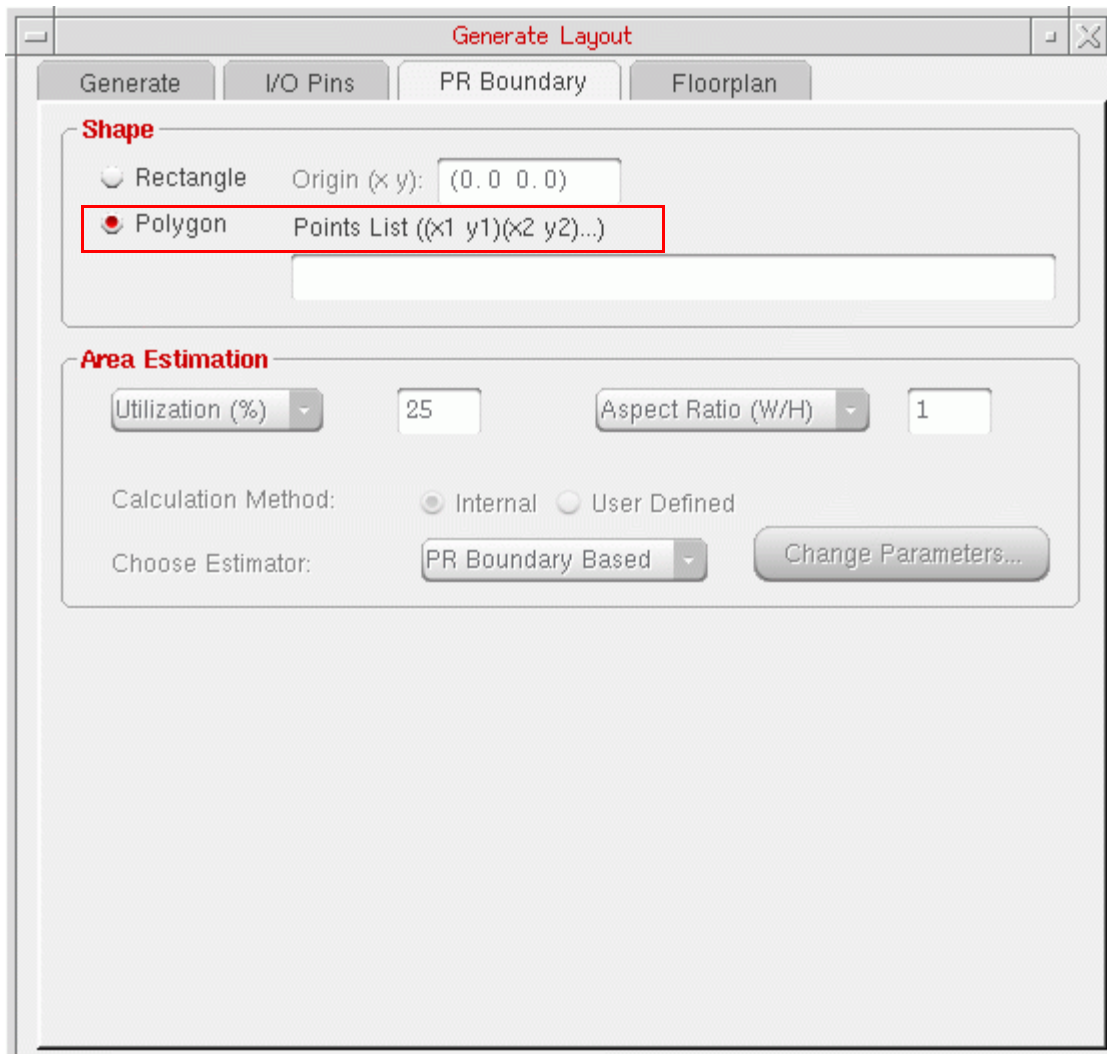
2. Specify the coordinates for the *Origin* for the boundary shape.
3. Use the controls in the *Area Estimation* group box to calculate the dimensions of the boundary.

For more information on how to use these controls, see [PR Boundary Tab](#).

Specifying a Polygonal Boundary

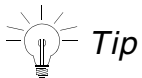
To specify a polygonal place and route boundary,

1. On the PR Boundary Tab, choose *Polygon*.



2. Type the X and Y coordinates of each vertex of the polygon in the *Points List* field. For example, for a rectangle, type the following.

(0 0) (0 10) (10 8) (8 0) (0 0)



Tip

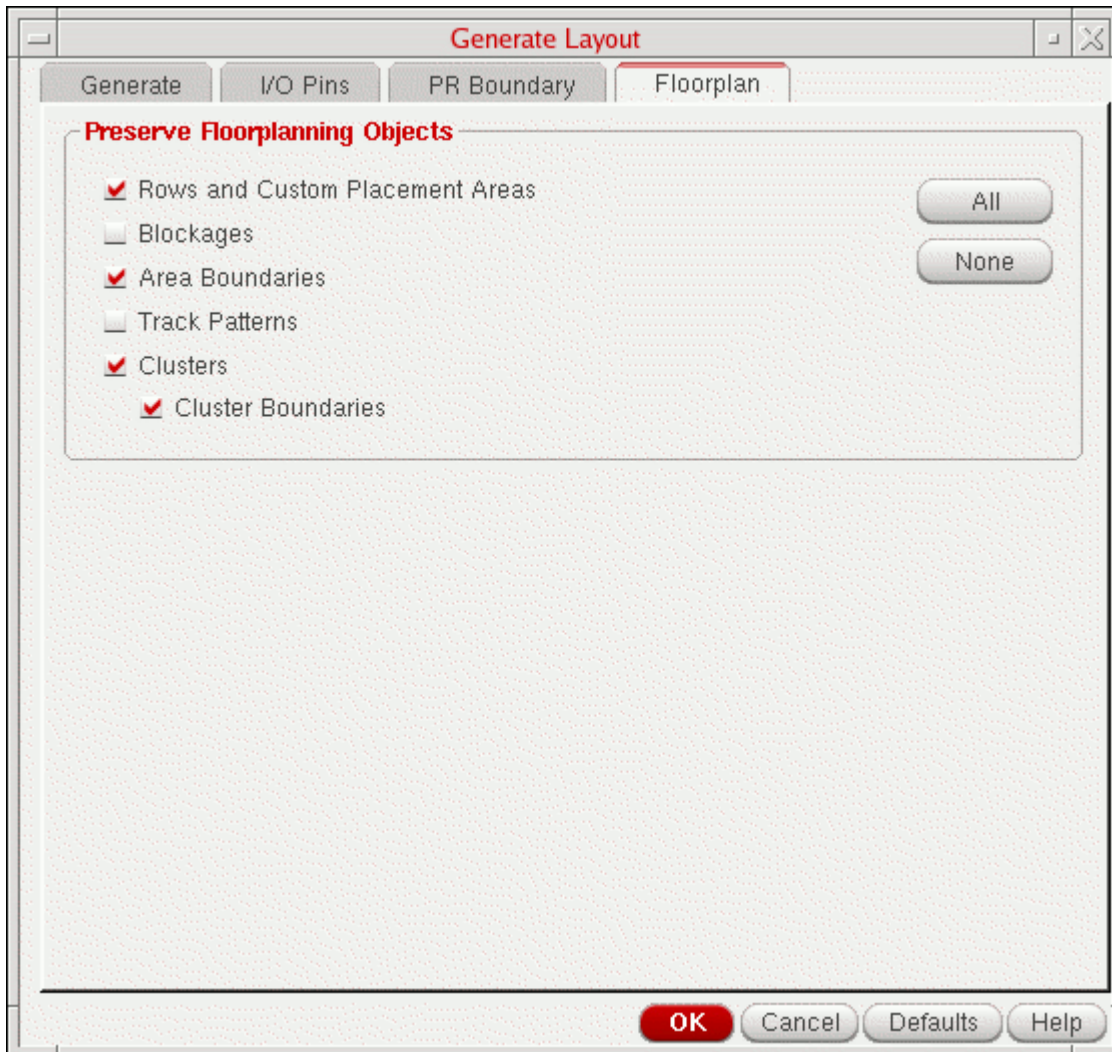
When you move the cursor into the canvas, the X and Y coordinates of the location of the cursor are displayed in the status banner at the bottom of the layout canvas.

Preserving Floorplanning Objects

Use the controls on the Floorplan Tab to preserve existing floorplanning objects during layout generation.

- On the *Floorplan* tab, check the objects you want to preserve.

Click *All* to select all the items in the list; *None* to deselect them all.



Generating a Layout from Source Using SKILL

You can also use the public SKILL API to run *Generate All From Source* using either of the following methods.

- The [IxGenFromSource](#) SKILL function.
- A combination of the following functions, which let you specify what is generated directly in SKILL.

[IxGenerateStart](#)

[IxGetAvailablePinLPPs](#)

[IxGetPinNets](#)

[IxSetBoundaryOptions](#)

[IxSetAreaEstimationOptions](#)

[IxSetGenerateOptions](#)

[IxSetNetPinSpecs](#)

[IxSetPreserveFloorplanningOptions](#)

[IxGenerateFinish](#)

For more information, see the sections on the above functions in the *Virtuoso Layout Suite SKILL Reference*.

Generating Selected Components from Source

Use the *Generate Selected From Source* command to manually generate schematic instances and pins in the layout view. You can select the components to be generated either before or after you launch the command and then generate the components either all together or one at a time in the layout view.

- When generating components together, you can select a combination of instances and pins and change the alignment, spacing, and orientation of the selected components before you place them in the layout. You can also enable automatic folding for instances.
- When generating components individually, you can change the values of any properties and attributes associated with each object before you generate it. You can also enable automatic chaining and folding for instances.

For more information, see

- [Generating Components Together](#)
- [Generating Individual Instances](#)
- [Generating Individual Pins](#)

■ Listing Unplaced Components

Additional Information

- The command honors any constraints entered using the Virtuoso Constraint Manager. It also applies all callbacks before any instance is created. The complete list of parameters found on the connectivity source instance (including the adjusted folding, sfactor, or lxCombination parameters) are used to call the CDF; the resulting list is passed to the layout instance.
- When *Generate Selected From Source* is active, clicking on a component in either the design canvas or the Navigator assistant only highlights the object, it does not select it. Consequently, any information displayed in the Property Editor assistant is not updated.
- If a layout instance master terminal has an inherited net expression (an implicit inherited connection) for a global layout net, *Generate Selected From Source* creates a physical-only layout terminal with no schematic equivalent. This terminal has the `physOnly` property set to `t` so that it can be correctly ignored by Layout XL functions such as *Update Components And Nets* and *Check Against Source* and by the digital applications further on in the design flow.

Generating Components Together

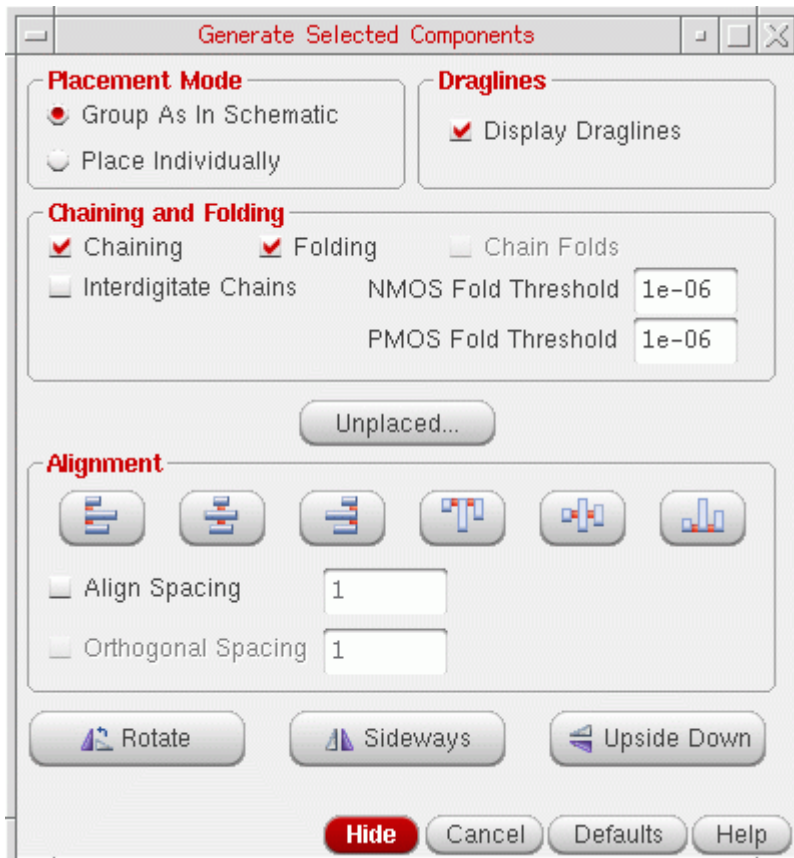
To generate multiple schematic instances and pins in a single operation,

1. From the layout window menu bar, choose *Connectivity – Generate – Selected From Source*.

Virtuoso Layout Suite XL User Guide

Generating a Layout

The Generate Selected Components form is displayed (press F3 if it does not display).



Group As In Schematic mode is enabled by default. In this mode, you cannot change the properties and attributes associated with the selected components. You can set pseudoparallel nets, however. For more information, see Generating Selected Components with Pseudoparallel Nets.

2. In the schematic window, select the instances and pins you want to generate using one of the following methods.
 - ❑ Click the components in the schematic canvas or Navigator assistant. Use **Shift-click** or click-and-drag area selection to select multiple components; deselect components using **Ctrl-click**.
 - ❑ Click *Unplaced* to view the Instance/Pin List where you can choose the components you want and click *Apply* to select them in the schematic window. See Listing Unplaced Components for more information.

The selected schematic components remain highlighted until they are either deselected or generated in the layout view. You cannot select an object in the schematic if an object with the same name already exists in the layout.

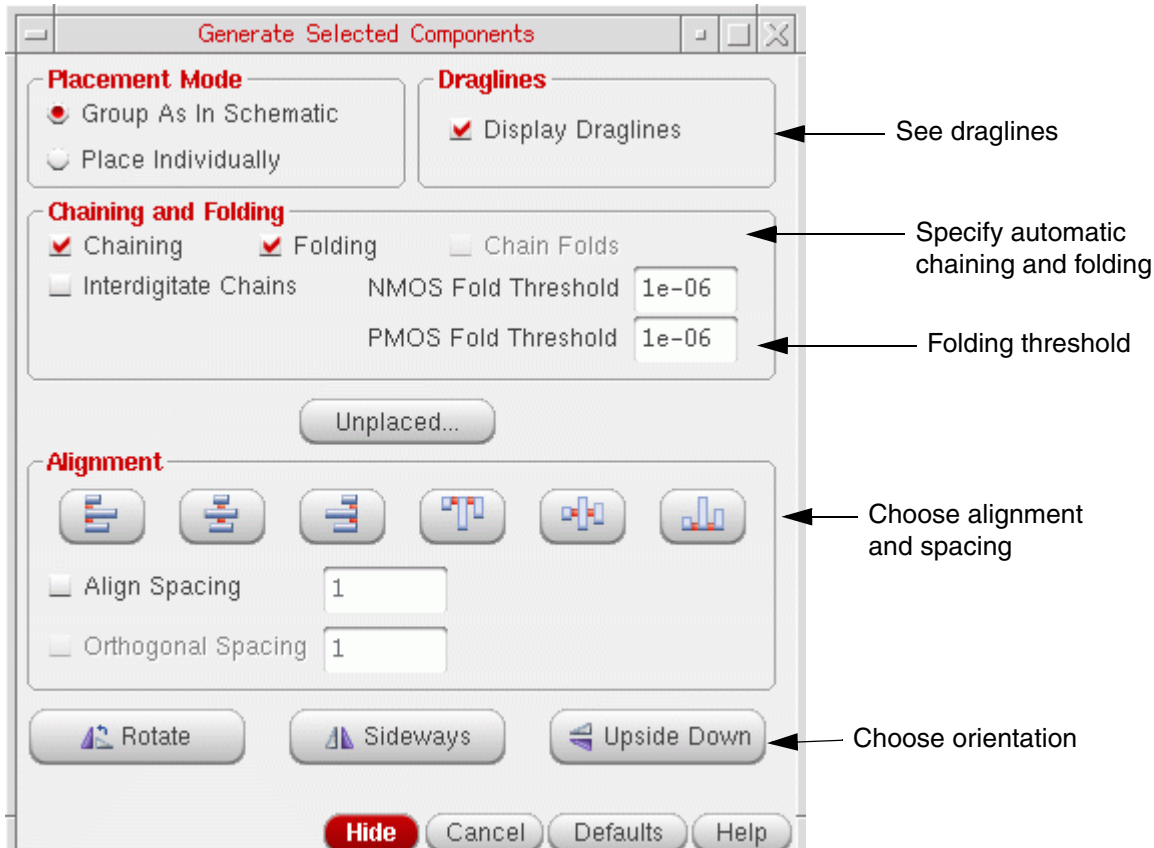
Virtuoso Layout Suite XL User Guide

Generating a Layout

3. Move your cursor into the layout canvas.

The outlines of the selected components follow the cursor in the canvas.

4. Set the options you want for the selected components.



- ❑ Switch on *Display Draglines* to see flight lines showing the connectivity between the selected components and the components that already in the layout. For information on changing these draglines, see [Changing the Appearance of Draglines](#).
- ❑ Select *Chaining* and *Folding* to automatically chain and fold the instances you are generating. To chain the individual folds of a transistor, select the *Chain Folds* check box.

Note: The *Chain Folds* option is active when *Folding* is selected and *Chaining* is deselected.

- ❑ Select *Interdigitate Chains* to automatically identify the nodes that qualify as pseudoparallel connections and define them as such during chaining and abutment.

Note: A pseudoparallel net connects nodes that are always the same voltage, so

current does not pass through the net.

- ❑ Use the *Alignment* options to automatically align and space the components as they are placed. For example, click *Align Bottom* to align the bottom edges and set *Align Spacing* to 2 to automatically space the selected instances by 2 user units when they are generated. See [Using the Align Toolbar](#) for more information.
- ❑ Choose the orientation you need using the *Rotate*, *Sideways*, and *Upside Down* buttons.

For details on all these options, see [Generate Selected Components](#).

5. Click in the layout canvas to place the selected components at the required location.

Note: Iterated pins placed in the schematic are placed in order (i.e. Q1, Q2, Q3) in the layout.

Generating Individual Instances

To generate schematic instances one at a time in the layout view,

1. From the layout window menu bar, choose *Connectivity – Generate – Selected From Source*.

Virtuoso Layout Suite XL User Guide

Generating a Layout

The Generate Selected Components form is displayed (press **F3** if it does not display).

The screenshot shows the 'Generate Selected Components' dialog box. It features a title bar with standard window controls. The main area is divided into several sections: 'Placement Mode' with radio buttons for 'Group As In Schematic' (selected) and 'Place Individually'; 'Draglines' with a checked checkbox 'Display Draglines'; 'Chaining and Folding' with checkboxes for 'Chaining' (checked), 'Folding' (checked), and 'Chain Folds' (unchecked), plus input fields for 'NMOS Fold Threshold' and 'PMOS Fold Threshold' both set to '1e-06'; an 'Unplaced...' button; 'Alignment' with six icons for alignment and two checkboxes for 'Align Spacing' and 'Orthogonal Spacing' both set to '1'; and three buttons for 'Rotate', 'Sideways', and 'Upside Down'. At the bottom are 'Hide', 'Cancel', 'Defaults', and 'Help' buttons.

2. Click *Place Individually* to generate instances one at a time.
3. In the schematic window, select the instances you want to generate using one of the following methods.
 - ❑ Click on the instances in the schematic canvas or Navigator assistant. Use **Shift-click** or click-and-drag area selection to select multiple instances; deselect instances using **Ctrl-click**.
 - ❑ Click *Unplaced* to view the Instance/Pin List where you can choose the instances you want and click *Apply* to select them in the schematic window. See Listing Unplaced Components for more information.

The selected instances remain highlighted until they are either deselected or generated in the layout view. You cannot select an instance in the schematic if an instance with the same name exists already in the layout.

The form expands to show the name and master of the first instance in the selected set (the current instance), along with the properties set on that instance.

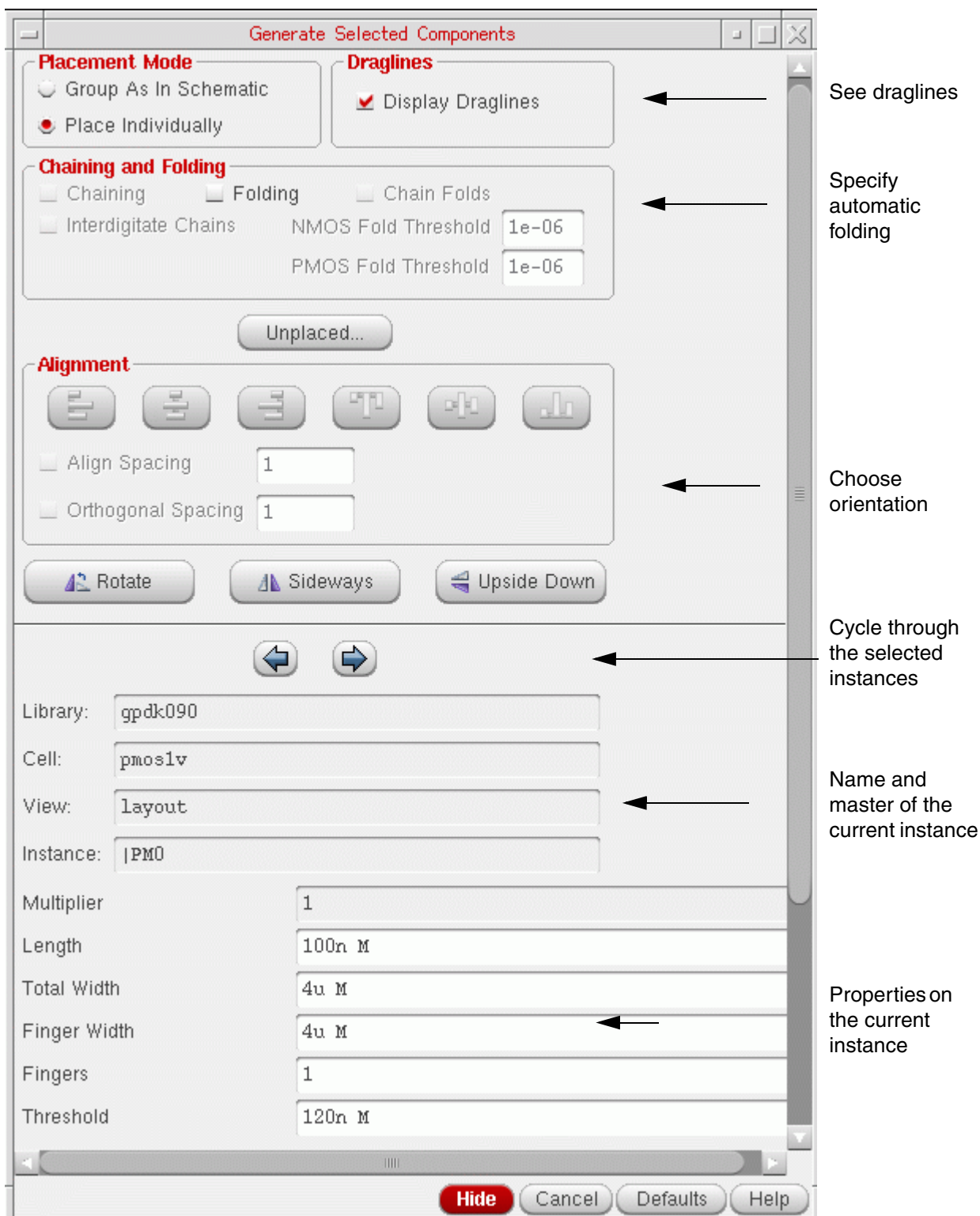
Virtuoso Layout Suite XL User Guide

Generating a Layout

The current instance is highlighted in the schematic in the color assigned to the `hilite` `drawing2` layer. Selected instances that have not yet been generated are highlighted in the color assigned to the `hilite` `drawing` layer.

Virtuoso Layout Suite XL User Guide

Generating a Layout



4. Move your cursor into the layout canvas.

The outline of the current instance follows the cursor in the layout window.

5. Set the options and property values you want.

- ☐ Switch on *Display Draglines* to see flight lines showing the connectivity between the current instance and the other components already in the layout view. For information on changing these draglines, see [Changing the Appearance of Draglines](#).
- ☐ Select *Folding* to automatically fold the current instance.
- ☐ Select *Chain Folds* to automatically chain the folded instance.
- ☐ Choose the orientation for the instance using the *Rotate*, *Sideways*, and *Upside Down* buttons.
- ☐ Change the instance property values if required. The instance properties are not shown when *Folding* is selected. To update a property, you must first turn off *Folding*.

The *Chaining* and *Alignment* options are disabled because you are placing only one instance at a time. For details on all these options, see [Generate Selected Components](#).

6. Click in the layout canvas to place the current instance at the required location.

If the layout library and cellview are different from those of the schematic, the command copies all the parameters (including default parameters) into the layout view. If any of the default parameter values in the schematic is an empty string, this can cause pcell evaluation to fail.

If you selected more than one instance, the next instance in the list becomes the current instance in the form. You can use the *Previous* and *Next* arrow buttons at the top of the properties section to quickly move back or forward through the set of selected instances without generating them.

7. Repeat steps 4 through 6 to generate the next instance.

Generating Individual Pins

To generate schematic pins one at a time in the layout view,

- 1.** From the layout window menu bar, choose *Connectivity – Generate – Selected From Source*.

Virtuoso Layout Suite XL User Guide

Generating a Layout

The Generate Selected Components form is displayed (press F3 if it does not display).



2. Click *Place Individually* to generate pins one at a time.
3. In the schematic window, select the pins you want to generate using one of the following methods.
 - ❑ Click on the pins in the schematic canvas or Navigator assistant. Use **Shift-click** or click-and-drag area selection to select multiple pins; deselect pins using **Ctrl-click**.
 - ❑ Click *Unplaced* to view the Instance/Pin List where you can choose the pins you want and click *Apply* to select them in the schematic window. See Listing Unplaced Components for more information.

The selected pins remain highlighted until they are either deselected or generated in the layout view. You cannot select a pin in the schematic if an instance with the same name exists already in the layout.

Virtuoso Layout Suite XL User Guide

Generating a Layout

The form expands to show the name of the first pin in the list (the current pin), along with the attributes set on that pin.

Generate Selected Components

Placement Mode

- ☐ Group As In Schematic
- ☒ Place Individually

Draglines

- ☒ Display Draglines

Chaining and Folding

- ☐ Chaining
- ☒ Folding
- ☒ Chain Folds
- ☐ Interdigitate Chains
- NMOS Fold Threshold:
- PMOS Fold Threshold:

Unplaced...

Alignment

☐ Align Spacing:

☐ Orthogonal Spacing:

Terminal:

Layer: Width: Height:

☐ Create Label As: ☐ Label ☐ Text Display

I/O Type: ☒ input ☐ output ☐ inputOutput ☐ switch ☐ jumper ☐ unused ☐ tristate

Access Direction: ☐ Top ☐ Bottom ☐ Left ☐ Right ☐ Any ☒ None

The current pin is highlighted in the schematic in the color assigned to the `hilite drawing2` layer. Selected pins that have not yet been generated are highlighted in the color assigned to the `hilite drawing` layer.

4. Move your cursor into the layout canvas.

The outline of the current pin follows the cursor in the layout window.

5. Set the options and pin attributes you want.

- ☐ Switch on *Display Draglines* to see flight lines showing the connectivity between the current instance and the other components already in the layout view. For information on changing these draglines, see [Changing the Appearance of Draglines](#).
- ☐ Change the pin attributes if required. Any changes you make are applied to the current pin and also become the defaults for all subsequent pins in the selected set. See [Generate Selected Components - Pin Options](#) for details.

The *Chaining and Folding* and *Alignment* options and the orientation buttons are disabled because you are placing pins.

6. Click in the layout canvas to place the current pin at the required location.

The pin has the connectivity indicated in the schematic and is automatically snapped to the placement grid.

If you selected more than one pin, the next pin in the list becomes the current pin in the form. You can use the *Previous* and *Next* arrow buttons at the top of the pin attributes section to quickly move back or forward through the set of selected pins.

7. Repeat steps 4 through 6 to generate the next pin in the selected set.

Listing Unplaced Components

To see a list of all schematic components not yet generated in the layout view,

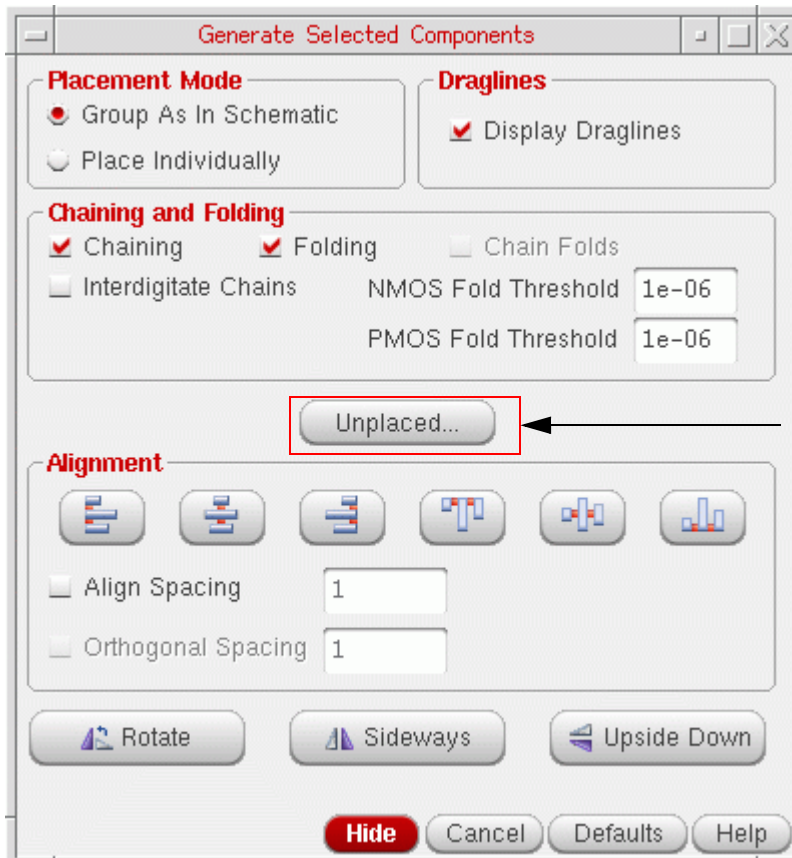
- 1.** From the layout window menu bar, choose *Connectivity – Generate – Selected From Source*.

The [Generate Selected Components](#) form is displayed (or press `F3`).

Virtuoso Layout Suite XL User Guide

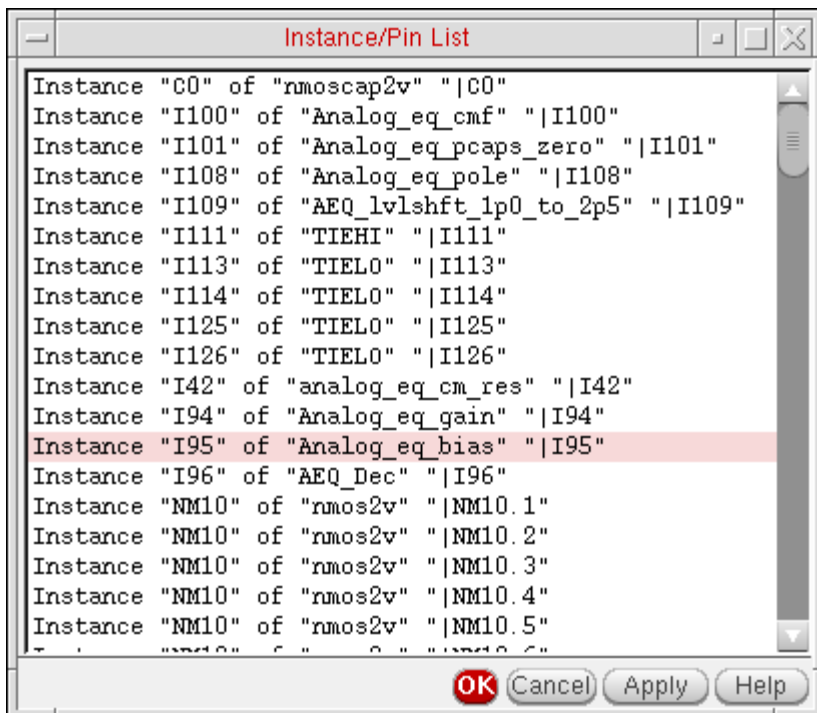
Generating a Layout

2. Click *Unplaced*.



Click *Unplaced* to show schematic components not yet generated.

The Instance/Pin List is displayed.



Wires, labels, text, instances with an ignore property, and instances that have already been placed in the layout are not considered available components and are not shown in the list.

3. Choose one or more instances and pins in the list and click *Apply*.

The Generate Selected Components form changes to show the available information on the first component in the selected set.

Generating Clones

Cloning is the ability to replicate a section of the layout, associated with a section of the schematic, in such a way that the new piece of layout material can be placed at more than one location, with each part preserving the hierarchical structure of the design. Cloning differs from copying in that the cloned structure maintains its schematic correspondence and inherits connectivity information whereas a synchronous copy does not maintain any schematic correspondence. For more information on creating a synchronous copy, see Generating Synchronous Copy.

Cloning can help you lay out

- Datapath bit slices, datapath and memory control blocks with identical logic cones, programmable logic structures, and other arrayed and iterated structures in device and gate-level custom digital design.
- Groups of analog/mixed signal components with identical constraints (for example, current mirrors, cascode stages, differential pairs, and other isomorphic structures).
- Instances of the same symbol master – or instances with the same logic but with differing parameters and hence a different symbol master – that have a flat layout representation.
- A chained set of devices or a part of the chain.

The section of the schematic or layout that is used as a template for the clone is called the *clone source*. The section of the schematic that is to be implemented by replicating the source is the *clone target*. The result of replicating the clone source to implement the clone target is the clone.

You can use multiple connectivity sources and multiple layout cellviews to generate clones using devices, pins, and (if selected from the layout canvas) interconnect structures such as wires and path shapes. You can also clone components that are implemented as one-to-many relationships defined using the `mfactor` property on a schematic instance. For more information, see [Cloning Mfactored Components](#).

Invalid Source Structures

A source structure is considered invalid for cloning if it includes any device that is part of a user-defined many-to-one binding (specified using the [Define Device Correspondence](#) form). In this case, a warning appears in the CIW explaining the error and the structure is rejected.

Target Search Space

You can search for matching target structures among all the devices in the schematic or among only the devices in the currently selected set. During the search, Layout XL ignores

- Any device that is already implemented in the layout.
- Any device that is part of a modgen constraint (because it is possible that it will be implemented as part of a modgen in the layout).

Exact and Non-Exact Matches

To clone components, you select the structure to be replicated (the clone source) and the software searches the design for target structures which match that source. You create a clone when you generate a target structure in your design.

To be considered a valid target, the instance master referenced by the components in the target must exactly match the instance master referenced by the components in the source. A target that also matches the source in terms of connectivity is considered an *exact match*. A target that tolerates slight differences in connectivity is a *non-exact match*, also known as a mutant clone.

You can use the options in the [Generate Clones](#) form to specify which differences are tolerated.

- *Allowed Permutation* considers [pin permutability](#) when searching for matching target structures. If an appropriate [permuteRule](#) is defined for the device in question, and if permuting the pins on a device results in a match, then that match is reported (but only if no matching targets were found without permutation).

Example: Two NMOS devices connected by their D terminals will match two NMOS devices connected by their S terminals, provided there is a permute rule in place that allows the S and D pins to be permuted.

- *Exact Parameters* requires that the parameter values on the components in a target structure match exactly the parameter values on the components in the clone source. When switched off, Layout XL also reports target structures that have different parameters and values.

Example When switched off, an NMOS device in the source with $w = 200$ will match an NMOS device with $w = 100$ in the target.

If required, you can update the parameters and values to match when you generate a clone by checking the *Update Layout Parameters to Match Schematic* option near the bottom of the [Generate Clones](#) form.

- *Exact Connectivity* requires that the connectivity of the components in a target structure matches exactly the connectivity of the components in the clone source. When switched off, Layout XL also reports target structures where the set of instances is the same, but they are connected differently.

Example When switched off, two NMOS devices connected by their D and S terminals in the source will match two NMOS devices connected only by their S terminals in the target.

Note that searching for non-exact connectivity matches takes significantly longer than searching for exact matches. You can use the options on the [Non-Exact Connectivity Matches](#) form to limit the search.

For information about

How the cloning engine finds matching targets, see the [Generate Clones](#) form.

- Solving problems with cloning, see [Troubleshooting Cloning](#).

Types of Clones

The [Generate Clones](#) form lets you generate clones as

- *Synchronized Family*

The clone source and individual clones are each generated as a group in the layout view and linked to the other members in the same synchronized family. Each group contains all the physical shapes, vias, instances, and groups in the layout implementation of the respective clone. If you make a change to one member of a synchronized family, all the other members are automatically changed in the same way. See [Generating Synchronous Clones](#).

- *Grouped Objects*

The clone source and individual clones are each generated as a group in the layout view but are not linked to each other in a synchronized family. Each group contains all the physical shapes, vias, instances, and groups in the layout implementation of the respective clone. Clones that are generated as groups are constrained in the ways they can be moved and rotated. See [Cloning a Group of Components](#).

- *Free Objects*

The components in the clone are generated as ungrouped components in the layout view. See [Generating a Clone using the Generate Clones Form](#).

Cloning Instances with no Schematic Counterpart

The *Generate Clones* command correctly handles layout instances with no schematic counterpart (dummy devices), including those that are part of a chain.

Layout XL considers a layout instance to be a dummy device if either the instance or its master has at least one property included in the list of *Properties used to ignore objects during check* on the [Parameters Tab](#) of the Layout XL Options form. For more information, see [Ignoring Components with no Schematic Counterpart](#).

When Layout XL detects a dummy device among the objects in the clone source, it generates a similar instance with the same properties in the target structure. This new instance is automatically named I_x , where x is an appropriate integer value assigned by the database.



Tip

For information on handling dummy devices in synchronous clones, see [Synchronous Clones and Dummy Devices](#).

Cloning Iterated Instances

Because they all have the same master and same connectivity, cloning considers the individual bits of an iterated instance to be equivalent. Consequently, cloning cannot guarantee the same relative placements of devices in the clone source and the matching targets. If required, you can change the relative placement of instances in the target structure manually by changing the correspondence of the individual instances in the source and the target. See [Editing a Target Structure Before Generation](#) for more information.

Cloning Instances with Inherited Connections

If a layout instance master terminal has an inherited net expression (an implicit inherited connection) for a global layout net, *Generate Clones* creates a physical-only layout terminal with no schematic equivalent. The physical-only terminal has the `physOnly` property set to `true` so that it can be correctly ignored by Layout XL functions such as *Update Components And Nets* and *Check Against Source* and by the digital applications further on in the design flow.

Synchronous Clones and Modgens

You cannot create a synchronous clone that contains modgens or synchronous clones. If the clone source includes groups of type `modgen` or `syncClone`, those groups are created as regular groups in the generated clone.

The only exception to this is if you are generating a synchronized family and the clone source comprises a single synchronous clone. In this case, matching target structures are generated as synchronous clones belonging to the same synchronized family.

Generating a Clone using the Generate Clones Form

To clone a section of your design,

1. In the schematic or layout canvas, select the structure you want to clone using single or area selection.

Note: If the structure you want to clone is a partially-implemented mfactored component,

Virtuoso Layout Suite XL User Guide

Generating a Layout

you must select it in the layout canvas.

2. In the layout window menu bar, choose *Connectivity – Generate – Clones*.

The Generate Clones form is displayed. If the selected instances and shapes form a valid clone source, the clone source fields are already filled in when the form opens.

Generate Clones <ether adc_cascode_opamp layout>

Clone Source

Schematic Instances /M11.1 /M11.2

Layout Instances |M11.1 |M11.2

Update From Selected

Target Designs

Schematic Cellview ether adc_cascode_opamp schematic

Layout Cellview ether adc_cascode_opamp layout

Search

In Entire Schematic

With ☒ Allowed Permutation ☐ Exact Parameters ☒ Exact Connectivity More Options...

Search

Create Options

Create Clones as Synchronized Family

☒ Update Layout Parameters to Match Schematic

☒ Display Draglines

Clones Found

Edit...

Rotate Sideways Upside Down

INFO (LX-1366): Click the "Search" button to find clones of the selected source.

Close Defaults Help

You can also open the form with nothing selected, select the clone source in the layout or schematic canvas, and click *Update From Selected*. If the selected instances and shapes form a valid clone source, the instance names are entered in the clone source fields. You can use the same method to change the clone source without closing the form.

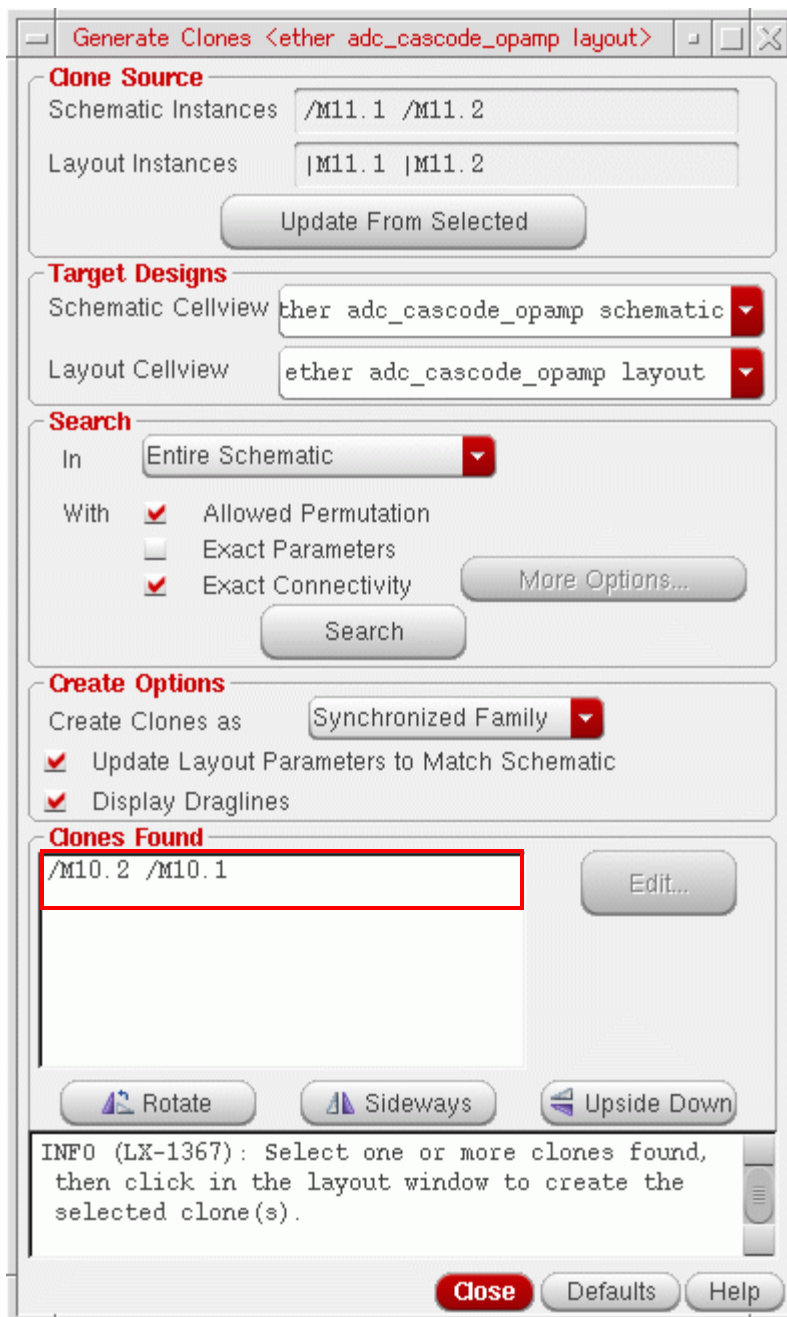
3. If required, set the *Schematic Cellview* and the *Layout Cellview*. By default, these are the cellview pair from which you launched the command.

See [Cloning between Multiple Cellviews](#) for information on using these fields.

4. Choose whether to search the *Entire Schematic* or only the *Selected Set in Schematic*.
5. Choose one or more of the search criteria *Allowed Permutation*, *Exact Parameters*, and *Exact Connectivity*.

See [Generate Clones](#) and [Non-Exact Connectivity Matches](#) for more information on these options.

6. Click *Search* to find matching target structures in the design.



The search ignores devices that are already implemented in the layout or that are part of a modgen constraint in the schematic.

The matching targets are added to the *Clones Found* list at the bottom of the form.

If the search is likely to take a long time, you will see a pop-up inviting you to do one of the following.

- ☐ Click *OK* to continue the search.
- ☐ Click *Cancel* to return to the Generate Clones form where you can tighten the criteria either by reducing the search area or by requesting exact matches only.
- ☐ Press `Ctrl-c` to stop the search. The system may take a few minutes to respond. The matches that were already found are not removed from the list of *Clones Found*.

7. Choose *Free Objects* from the *Create Clones as* pull-down and select the *Update Layout Parameters to Match Schematic* and *Display Draglines* check boxes, as required.

See Changing the Appearance of Draglines for more information.

8. Select one or more devices from the list of *Clones Found*, and move your cursor into the layout canvas.

An image of the clones to be generated follows your cursor.

9. Select the orientation options you want (*Rotate*, *Flip Horizontal*, *Flip Vertical*) and click in the layout canvas to place the first clone at the required location.

If you selected more than one target structure from the list, click again to place the second clone, and so on.

Editing a Target Structure Before Generation

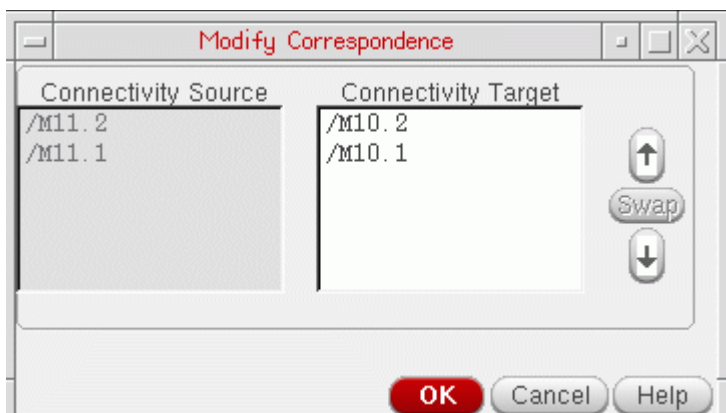
When searching for matching target structures, Layout XL does not consider the relative placement of instances in the source structure as long as swapping these instances does not change the source topology. Additionally, cloning considers the individual bits of an iterated instance to be equivalent because they all have the same master and same connectivity. For these reasons, the relative placement of instances in a target structure can differ significantly from that in the clone source.

The Modify Correspondence form lets you change the relative placement of instances in the target structure manually by changing the correspondence of the individual instances in the source and the target. When you modify the correspondence manually, Layout XL checks that the connectivity of the modified target structure remains valid before allowing you to generate the clone.

To modify the correspondence of source and target instances:

1. Create a list of matching target structures using steps 1-6 in [Generating a Clone using the Generate Clones Form](#).
2. Select a structure from the list and click *Edit*.

The [Modify Correspondence](#) form is displayed.



The instances in the source structure are listed under *Connectivity Source*. The list is read-only. The corresponding instances in the target structure are listed under *Connectivity Target*. When you select an instance in the target structure, the corresponding instance in the source structure is highlighted.

3. To change the correspondence of a single instance, select the instance in the *Connectivity Target* list and use the arrow buttons to move it up or down in the list.
4. To swap the correspondence of two instances, select the instances and click *Swap*.
5. When you have finished modifying the correspondence, click *OK*.

Before accepting the new correspondence, Layout XL checks that the connectivity of the target structure remains valid.

6. Generate the clones using steps 8-10 in [Generating a Clone using the Generate Clones Form](#).

Cloning a Group of Components

To clone a group of components,

- Create a cell containing all the devices and then clone the cell using the procedure in [Generating a Clone using the Generate Clones Form](#).

This ensures that all the cells or like circuit sections have the same layout parasitics associated with them.

Cloning between Multiple Cellviews

You can generate a clone in a different layout from the one in which you selected the clone source as long as:

- The second layout has the same connectivity source as the first layout.
- You select all the clone source objects from the layout from which you started the *Generate Clones* command (or its corresponding schematic).

To clone an object from one layout cellview into another layout cellview with the same connectivity source,

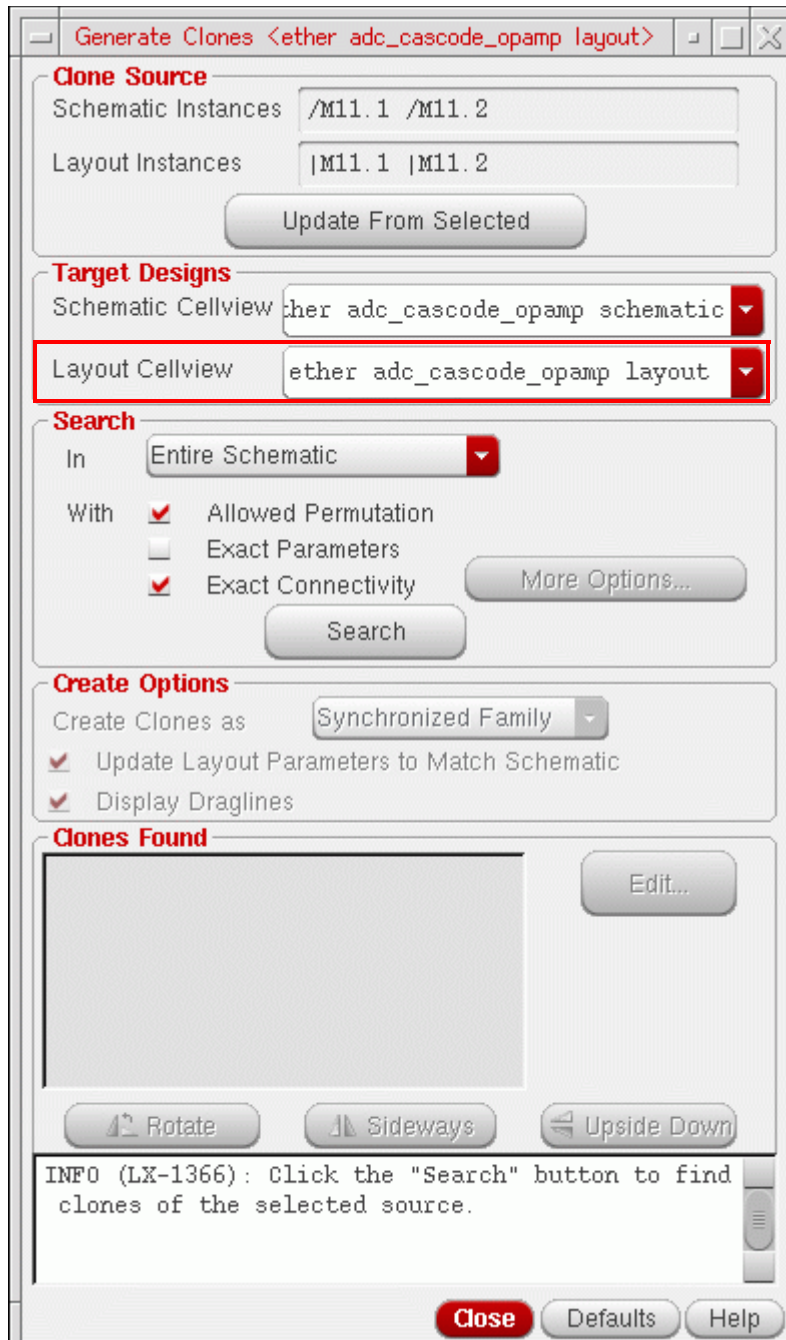
1. Open both layout cellviews in Layout XL.
2. Choose *Connectivity – Generate – Clones* in the layout window containing the clone source.

The Generate Clones form is displayed.

3. Select the clone source structure in the first layout window or in the schematic window and click *Update From Selected*.

If the selected instances and shapes form a valid clone source, the instance names are entered in the clone source fields.

4. In the Generate Clones form, change the *Layout Cellview* to the second layout window.



5. Set the search options as required and click *Search* to find matching target structures in the design.
6. Generate the clones using steps 8-10 in Generating a Clone using the Generate Clones Form.

Cloning Mfactored Components

Layout XL handles one-to-many relationships defined using the mfactor property on a schematic instance.

For example, if you have a component with an mfactor of 100, you can choose ten of these components, wire them together in the layout canvas, and then clone the resulting structure ten times to generate all the mfactored instances.

You can clone mfactored components both partially and in their entirety across multiple hierarchy levels, including source structures that include folded and abutted devices. The following limitations apply.

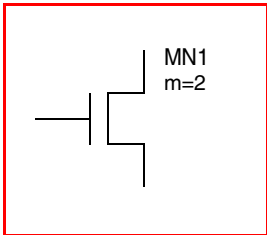
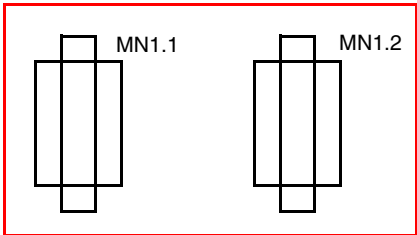
- The selected set must not include folded mfactored devices generated in a previous release; these must be regenerated in the current release.
- When cloning a folded instance, all the folds of the regular device or mfactored device part must be included in the clone source selection.
- When cloning mfactored devices that are partially implemented in the layout, you must select the clone source from the layout canvas.

The examples that follow illustrate what the cloning functionality considers to be valid targets for a number of different mfactored source structures.

Example 1: Mfactored component instance

If the clone source is an mfactored component instance (for example, an NMOS instance with $m=2$), then the following are considered valid clone targets.

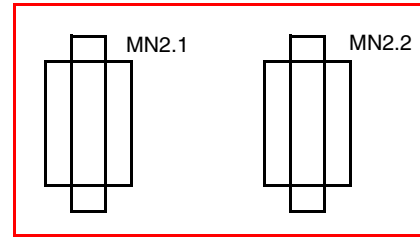
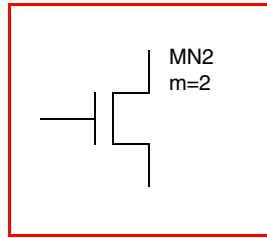
- Another mfactored component instance; for example, another nmos instance also with $m=2$ (see example below).
- Two non-mfactored NMOS instances connected in parallel.

	Schematic	Layout
Clone Source		

Virtuoso Layout Suite XL User Guide

Generating a Layout

Clone Target



Example 2: Mfactored component instance partially generated in the layout

If the clone source is one or more parts of an mfactored component instance that is partially generated in the layout, then the following are considered valid clone targets.

- The remaining components in the mfactored instance (see example below).
- The appropriate number of non-mfactored NMOS instances connected in parallel.

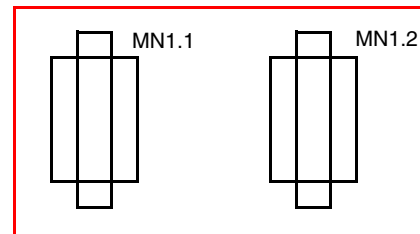
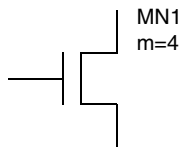
Important

You must select the clone source in the layout canvas because of the one-to-many relationship.

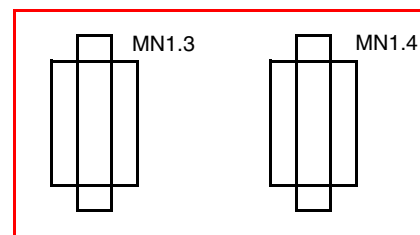
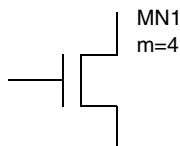
Schematic

Layout

Clone Source



Clone Target



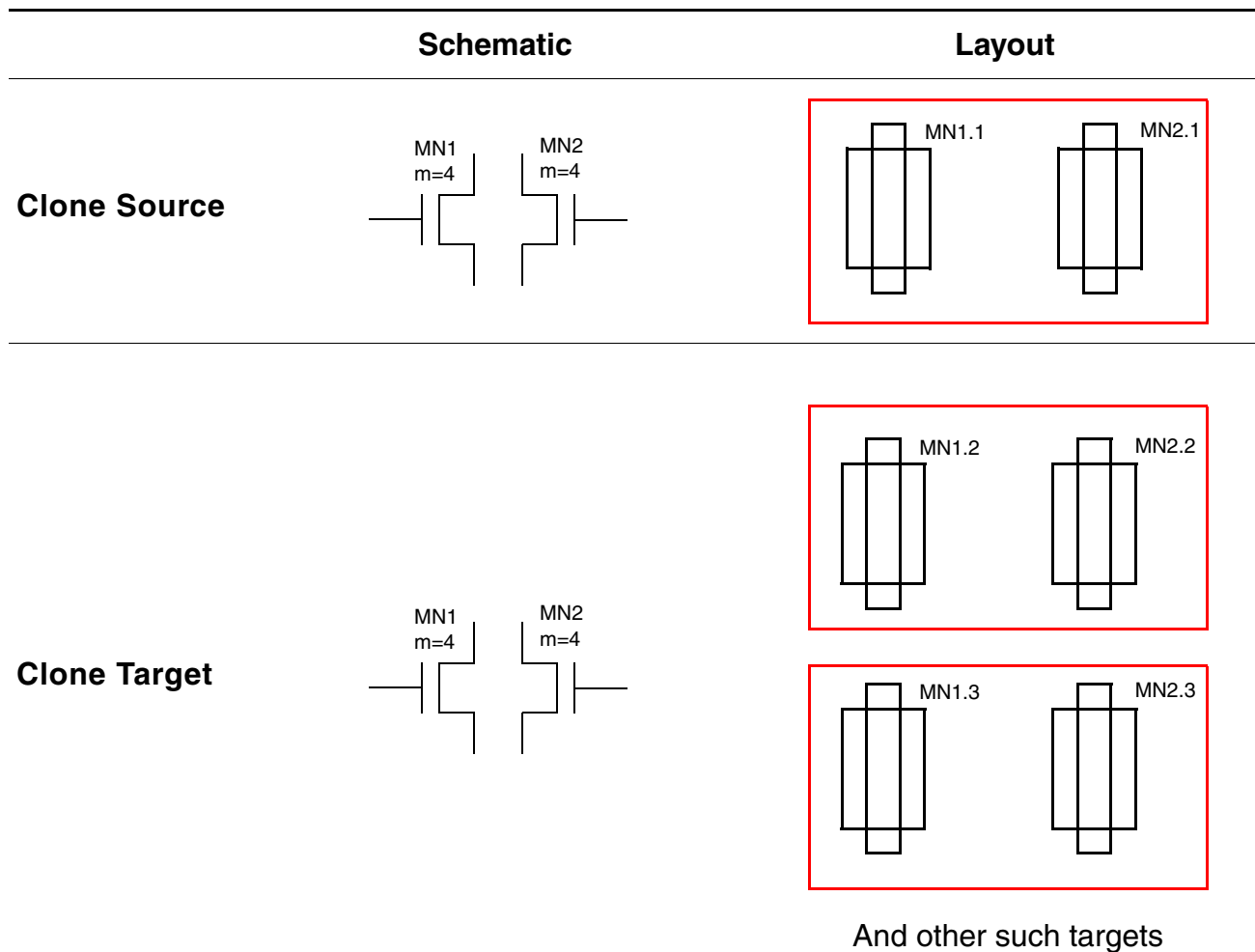
Example 3: One or more of multiple mfactored instances partially generated in the layout

If the clone source is one or more of multiple mfactored component instances that are partially generated in the layout, then the following are considered valid clone targets.

- The remaining mfactored component instances that follow the same pattern (see example below).
- The appropriate number of non-mfactored NMOS instances connected in parallel.

Important

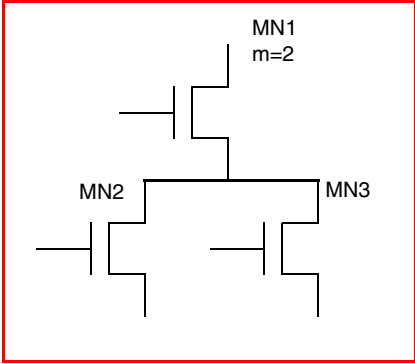
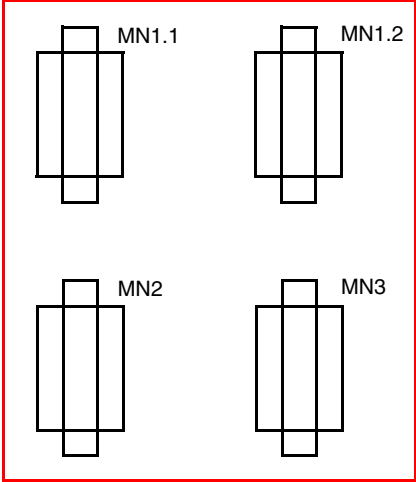
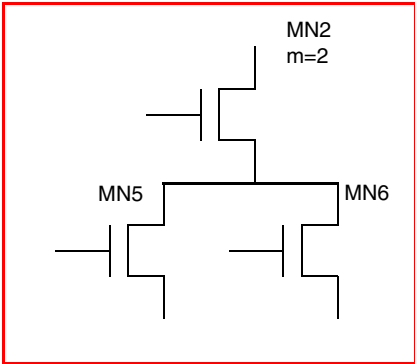
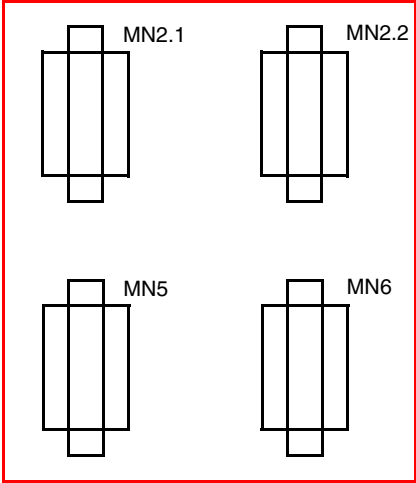
You must select the clone source in the layout canvas because of the one-to-many relationship.



Example 4: Mfactored component instance with other component instances

If the clone source is an mfactored component instance along with other component instances, then the following are considered valid clone sources.

- Any other identical configuration with mfactored component instances which have the same mfactor value.
- The appropriate number of non-mfactored NMOS instances connected in parallel.

	Schematic	Layout
Clone Source		
Clone Target		

Virtuoso Layout Suite XL User Guide

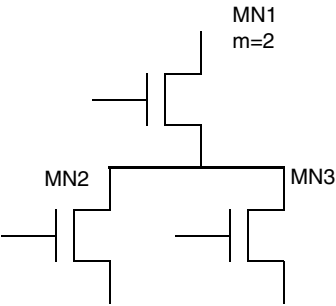
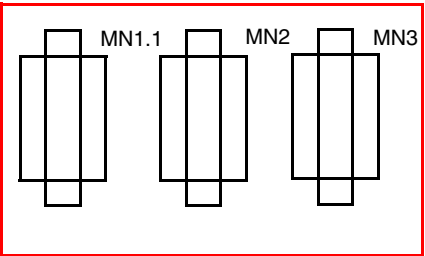
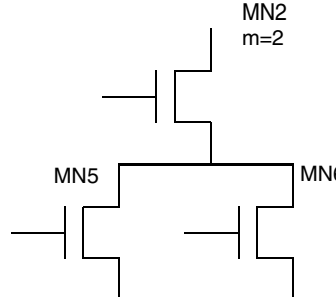
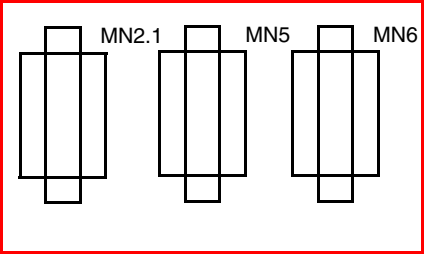
Generating a Layout

Example 5: One or more of an mfactored component instance partially generated in the layout along with other component instances

If the clone source is one or more of an mfactored component instance that is partially generated in the layout along with other component instances, then any configuration with the same topology but which only partially implements the mfactored components in the same way as the clone source is considered a valid clone target.

Important

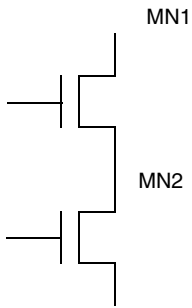
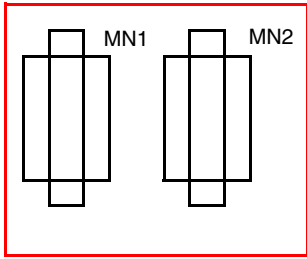
You must select the clone source in the layout canvas because of the one-to-many relationship.

	Schematic	Layout
Clone Source		
Clone Target		

Example 6: Although clone source contains only non-mfactored devices, clone target can use mfactored devices

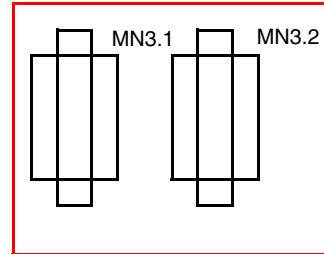
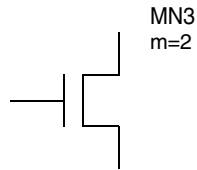
If the clone source is two non-mfactored NMOS instances connected in parallel, then the following can be considered valid clone targets.

- An mfactored NMOS instance with $m=2$
- Parts of an mfactored NMOS mfactored instance with $m>2$

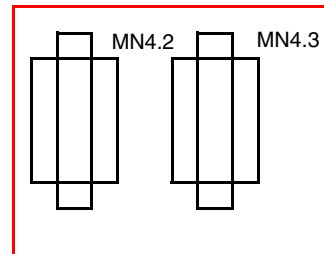
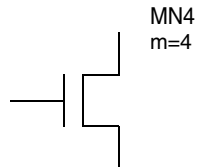
	Schematic	Layout
Clone Source		

Virtuoso Layout Suite XL User Guide

Generating a Layout



Clone Targets



Example 7: Clone target uses parts of a device with an mfactor value greater than that on the clone source

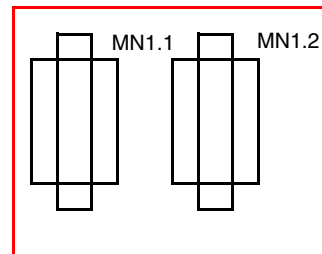
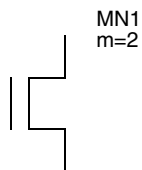
If the clone source is an mfactored component ($m=m_1$), which is fully realized in the layout, then the appropriate number of parts of an mfactored component ($m>m_1$), is considered a valid clone target.

Schematic	Layout

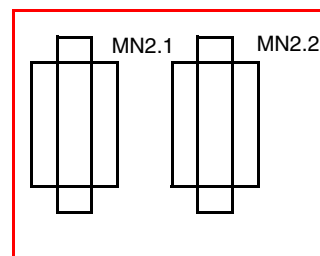
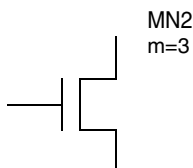
Virtuoso Layout Suite XL User Guide

Generating a Layout

Clone Source



Clone Targets

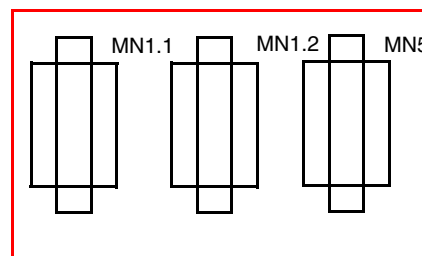
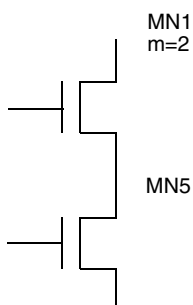


The same principle applies when devices with different mfactor values are combined with other devices.

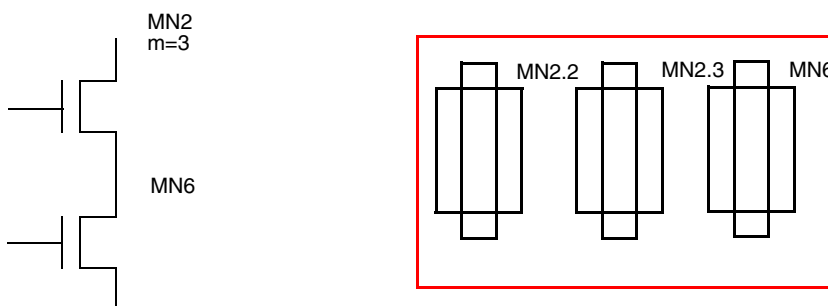
Schematic

Layout

Clone Source



Clone Targets



Example 8: No valid clone targets

The clone source is a fully-implemented mfactored component ($m=m_1$) and the schematic has the following other components that are not yet realized in the layout:

1. Another mfactored component with the same master (where $m < m_1$).
2. Another non-mfactored component with the same master.

For example,

The clone source is = MN1.1, MN1.2, and MN1.3 ($m=3$ in schematic)

The schematic also contains MN2.1 and MN2.2 ($m=2$ in schematic) and a non-mfactored instance MN3.

In this case there are no matching targets for the clone source.

Generating Synchronous Clones

Synchronous clones are the clones that stay synchronized during editing— changes made to one member of a family are automatically reflected in all the other members of the clone family.

You can use the *Synchronized Family* option on the Generate Clones form to create multiple clones as members of a single synchronized family. The clone source and the individual clones are each generated as a group in the layout view and linked to the other members in the same synchronized family, with each group containing all the physical shapes, vias, instances, and groups in the layout implementation of the target structure.

Note: Labels and text displays are not synchronized.

If you add a new instance to an existing synchronous clone using either the *Create Instance* or *Generated Selected From Source* command, or by removing an ignore property from an existing instance inside the clone, the clone in question is not desynchronized from its family. Note, however, that it may no longer form a valid clone source, in which case you will be unable to use the *Generate Clones*, *Create Family*, or *Add to Family* commands on that structure.

Synchronous Clones and Instance Creation

If you add a new instance to an existing synchronous clone using the *Create Instance*, *Copy*, or *Generate Selected From Source* commands, the system automatically adds a similar instance to the other clones in the family, ensuring that the family remains synchronized. If you add or remove an ignore property from an existing instance inside the clone, the clone is not desynchronized from its family, but the change to the property is not replicated in the other clones in the family.

In both cases, the clone in question may no longer form a valid clone source, which means you will be unable to use the *Generate Clones*, *Create Family*, or *Add to Family* commands on that structure.



Tip

For information on adding dummy devices to a synchronous clone, see [Synchronous Clones and Dummy Devices](#).

Synchronous Clones and Dummy Devices

Layout XL considers a layout instance to be a dummy device if either the instance or its master has at least one property included in the list of *Properties used to ignore objects during check* on the [Parameters Tab](#) of the Layout XL Options form.

When Layout XL detects a dummy device among the objects in the clone source, it generates a similar instance with the same properties in the target structure. This new instance is automatically named I_x , where x is an appropriate integer value assigned by the database.

Layout XL considers a dummy device as a simple shape in the clone structure. Therefore, dummy devices are synchronized in the same way as any other shape in a family of synchronous clones. Any physical editing, such as move, rotate, or delete performed on a dummy device (including those used in chains of devices) is reflected in every other member of the synchronized family.

To add a previously defined dummy device to a synchronous clone:

1. Create the instance for the dummy device outside the clone family.

2. Add the appropriate ignore-for-check property to the instance.

See [Ignoring Components with no Schematic Counterpart](#).

3. Use the *Edit – Group – Add To Group* command to push the dummy device inside the clone family.

All the other occurrences of the synchronous clone are populated with a similar dummy device. These new devices are named automatically and are connected as required by the Layout XL connectivity extractor.

Note: When using the *Add To Family* and *Create New Family* commands on the [Update Clone Families](#) form, any dummy devices present in the reference group are copied in other occurrences of the clone. For more information, see [Creating a New Clone Family](#) and [Adding a Group to an Existing Family](#).

Synchronous Clones and Modgens

You cannot create a synchronous clone that contains modgens or synchronous clones. If the clone source includes groups of type `modgen` or `syncClone`, those groups are created as regular groups in the generated clone.

The only exception to this is if you are generating a synchronized family and the clone source comprises a single synchronous clone. In this case, matching target structures are generated as synchronous clones belonging to the same synchronized family.

Synchronous Mutants

Mutants are clones that may not exactly match the source in connectivity. So, when searching for clone targets on the **Generate Clones** form with the option **Exact Connectivity** set to false, you may find mutant targets in your search result. These mutants will be displayed as *mutant (number) instanceName1 instanceName2*, and so on.

As the resulting mutants do not exactly match the source in connectivity, they have missing or extra nets called partial nets. However, these mutants can be considered as copies of the layout clone source with the connectivity of the schematic targets. Therefore, all the routing shapes present in the clone source are copied to the generated mutant, even if the shape net is a partial net, which is missing in the mutant target. As a consequence, the generated target clone is physically identical to the clone source, even if the clone is a mutant. This allows for synchronous mutants to be created as the cloning synchronization mechanism does not require any logical aspects to be met as long as the clones are physically identical.

Note: As the logical connectivity of the mutant instances (inherited from the clone schematic target) may be different from the physical connectivity of the routing shapes (inherited from

the clone layout source), the XL connectivity extractor may generate short markers when processing the mutants.

Synchronous Clones and Device Abutment

You cannot abut two devices if one of the devices is in a synchronous clone and the other is not in the same synchronous clone.

Synchronous Clones and Pin Permutation

You cannot permute two pins if one of the pins is on a device in a synchronous clone and the other pin is on a device that is not in the same synchronous clone.

If you permute two pins on an instance in a synchronous clone, the change is not propagated to the other members of the family, nor is the clone desynchronized from the other members of the family, because pin permutation is a logical rather than a physical update.

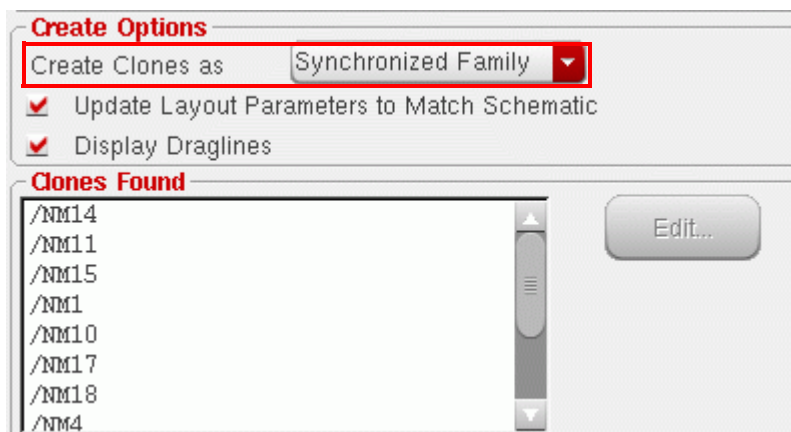
Generating a Family of Synchronous Clones

To generate a family of synchronous clones:

1. Create a list of matching targets using steps 1-6 in [Generating a Clone using the Generate Clones Form](#).

The matching targets are displayed at the bottom of the [Generate Clones](#) form.

2. Choose *Synchronized Family* from the *Create Clones as* pull-down.



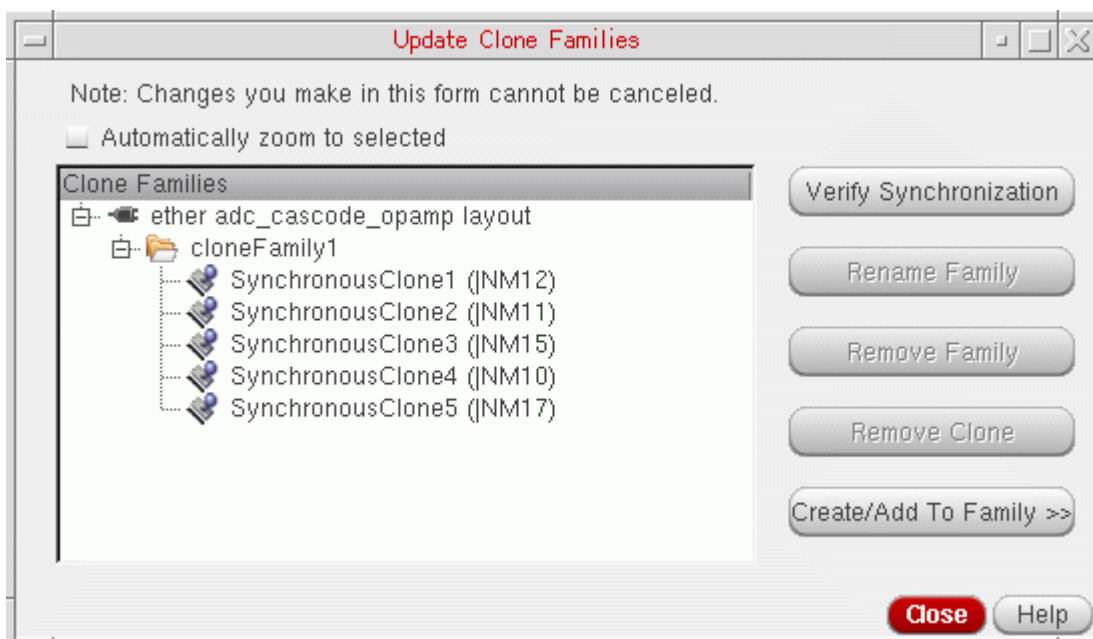
3. From the list of clones found, select all the target structures you want to include in the synchronized family.

4. Move your cursor into the layout canvas and click once to generate each of the clones you selected.

Note: If the synchronous family you are creating contains only one group, Layout XL uses this existing group to implement the synchronous clone. This prevents creation of an unnecessary level of hierarchy.

5. To confirm that the clone family was created, choose *Connectivity – Update – Clone Families*.

The Update Clone Families form is displayed, showing the family you created.



For more information on the tasks you can perform in this form, see Updating Clone Families.

Editing a Clone in a Synchronized Family

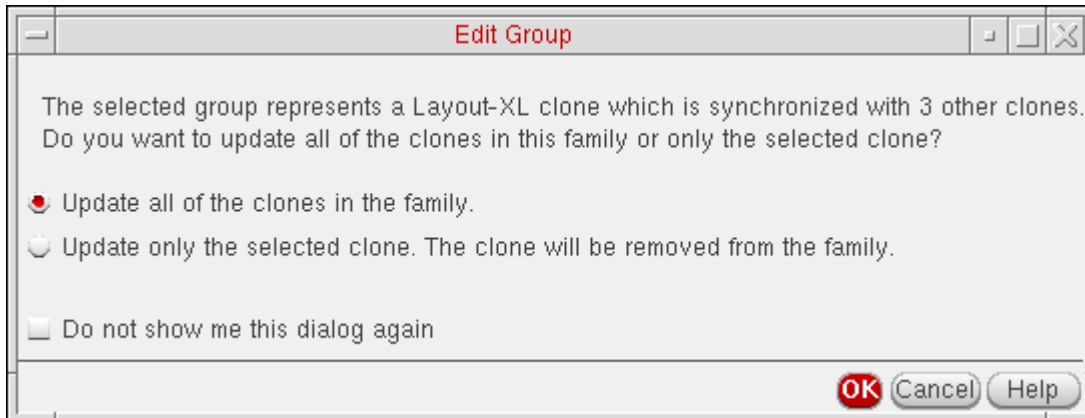
Because synchronous clones are implemented using groups, you cannot edit the components in the clone from the top level of your design. To edit a synchronous clone, you must use the *Edit In Place* command.

If you make a change to the physical topology of one member of a synchronized family, that change is propagated to all the other clones in the family.

To edit a member of a synchronized family,

1. From the layout window menu bar, choose *Edit – Hierarchy – Edit In Place* and select the clone you want to edit.

Layout XL warns you that the clone you are about to edit is a member of a synchronized family and asks you to confirm whether you want the changes propagated to all the clones in the family.



If you choose to update only the selected clone, it will be removed from the family.

You can also choose not to see this dialog again.

2. Click *OK* to dismiss the Edit Group dialog, select one of the instances in the clone you are editing, and choose *Edit – Move* to move the instance to a different location.

The corresponding instance is moved in each of the other clones in the synchronized family.

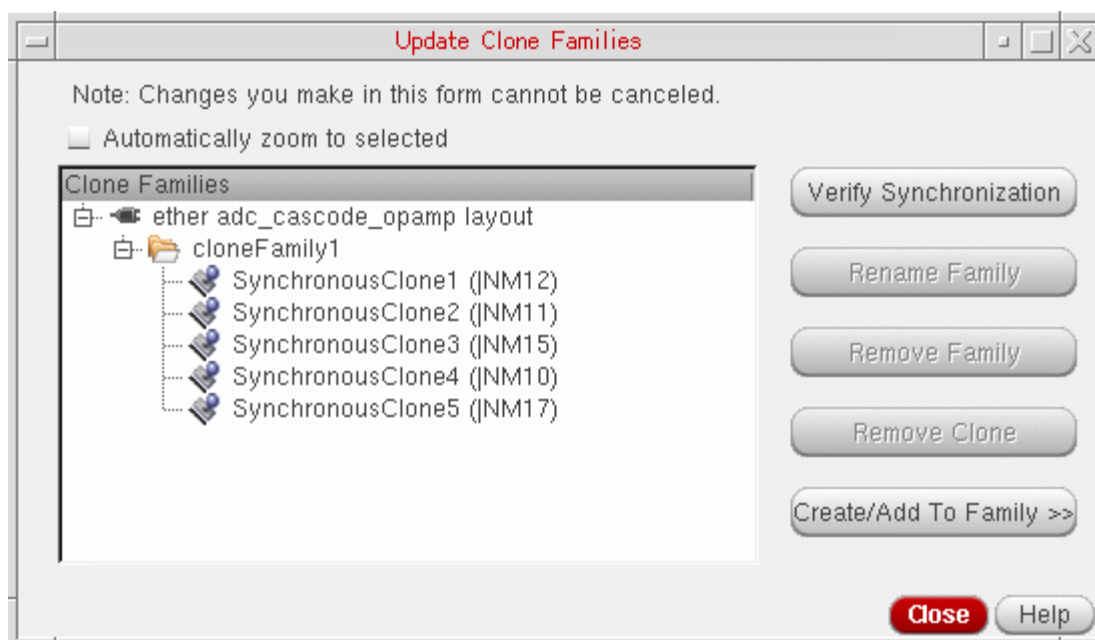
3. Now choose *Create – Wire* to create a wire in the clone.

When you complete the command, a similar wire is created in each of the other clones in the synchronized family.

4. From the layout window menu bar, choose *Edit – Hierarchy – Return* to exit the Edit In Place mode.

Updating Clone Families

The Update Clone Families form lets you manage the families of synchronized clones in your design. To see this form, choose *Connectivity – Update – Clone Families*.



The entries in the form are highlighted depending on what is selected in the layout window. The updates you make in the layout window are reflected immediately in the Clone Families pane.

The *Automatically zoom to selected* option zooms in on the clone or clone family that is currently selected in the *Clone Families* pane, or on the layout group selected in the *Layout groups* pane (see Creating a New Clone Family and Adding a Group to an Existing Family).

Clicking on the top level fits the whole cellview in the layout window. If you select more than one layout group, the software zooms in on the bounding box that encloses the selected groups.

Note: This option is considered only when you make the selection in the Update Clone Families form; selecting a synchronous clone in the canvas does not zoom the display to the selected clone or group.

For more information on the tasks you can perform in this form, see

- Verifying a Clone Family
- Removing a Clone from a Family

- Removing a Clone Family
- Creating a New Clone Family
- Adding a Group to an Existing Family

Verifying a Clone Family

If you edit one or more members of a clone family in an application other than Layout XL (for example, in Layout L), you might need to verify the family to ensure its members are consistent with each other.

You can verify one or more clone families with a single button push. Layout XL checks the consistency of the clones in each family and takes the following actions.

- Any clone that is out of synch with the other clones in the same family is removed from the family.
- If two or more clones are out of synch with the family but in synch with each other, they are moved to a new clone family.
- If all the clones in the family are different, the family is removed.

Note: You cannot undo the changes made by *Verify Synchronization*.

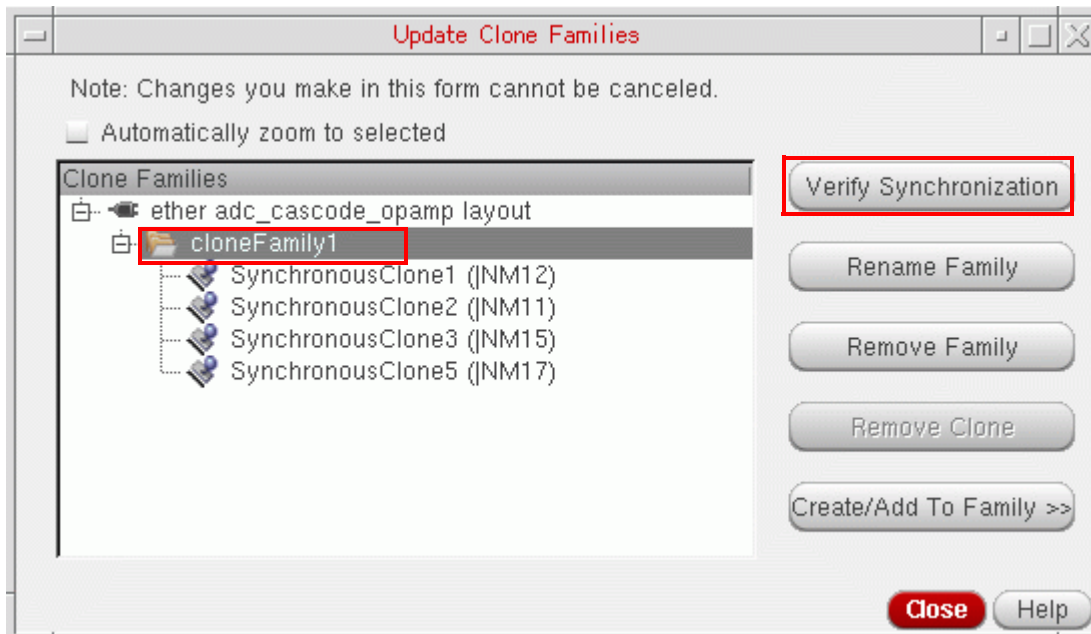
To verify that a clone family is synchronized,

1. From the layout window menu bar, choose *Connectivity – Update – Clone Families*.

Virtuoso Layout Suite XL User Guide

Generating a Layout

The Update Clone Families form is displayed.



2. Select the clone family you want to verify and click *Verify Synchronization*.

To verify that all the clone families are synchronized, click *Verify Synchronization* with none of the families selected.

Layout XL modifies the clone families as outlined above and prints the results to the CIW.

3. Click *Close* to dismiss the Update Clone Families form.

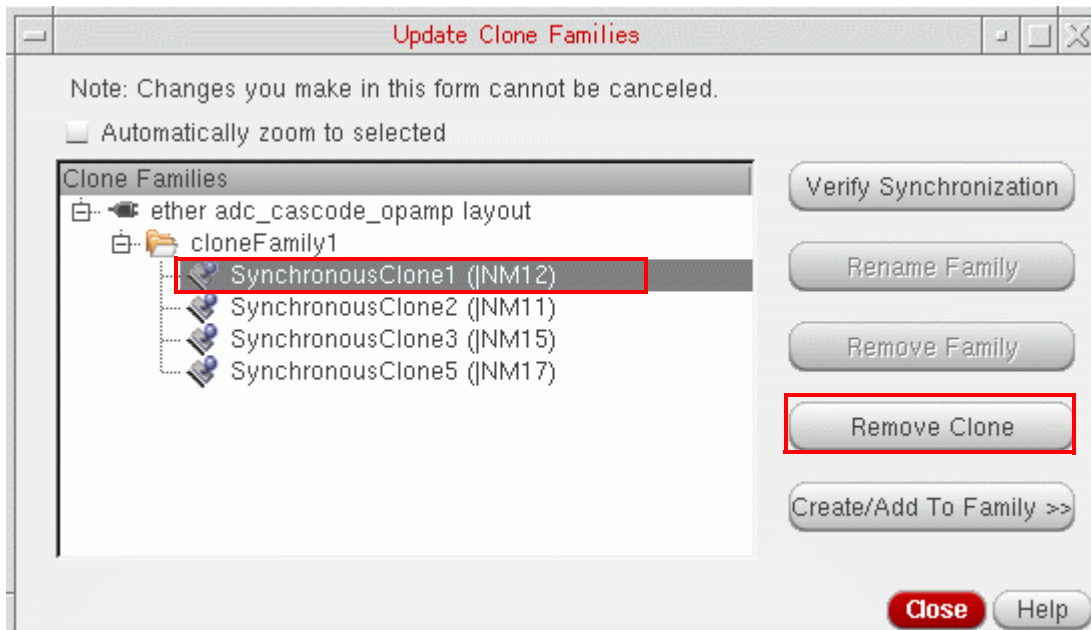
Removing a Clone from a Family

To remove a clone from a family,

1. From the layout window menu bar, choose *Connectivity – Update – Clone Families*.

The Update Clone Families form is displayed.

2. Select the clone to be removed and click *Remove Clone*.



The selected clone is removed from the family. It is still present in the layout but is no longer synchronized with the other members of the family.

If you remove all except a single clone from a family, the family is automatically removed as well.

Note: You cannot remove a clone if one of its members is currently being edited in place. You need to exit the *Edit In Place* command first.

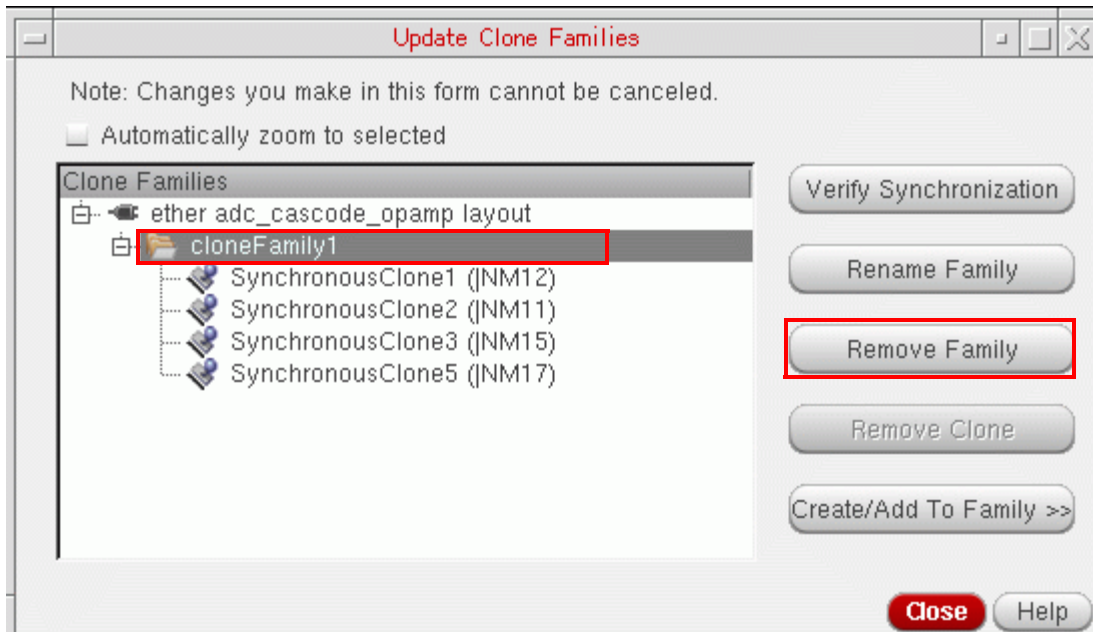
3. Click *Close* to dismiss the Update Clone Families form.

Removing a Clone Family

To remove a clone family from your design,

1. From the layout window menu bar, choose *Connectivity – Update – Clone Families*.
The Update Clone Families form is displayed.

2. Select the family you want to remove and click *Remove Family*.



The clone family is removed. The clones it contained are still present in the layout but they are no longer synchronized.

Note: You cannot remove a family if one of its members is currently being edited in place. You need to exit the *Edit In Place* command first.

3. Click *Close* to dismiss the Update Clone Families form.

Creating a New Clone Family

To create a new synchronized clone family containing two or more of the layout groups currently in your design,

1. From the layout window menu bar, choose *Connectivity – Update – Clone Families*.
The Update Clone Families form is displayed.
2. Click *Create/Add To Family*.

Virtuoso Layout Suite XL User Guide

Generating a Layout

The form is expanded to include the advanced options.



3. Ensure there is no existing family selected in the *Clone Families* pane, and then select at least two groups from the list of *Layout groups*.

The *Synchronization reference* drop-down and the *Create Family* button are enabled.

 Important

The *Layout groups* pane lists all the regular layout groups not currently assigned to a clone family. It does not list any other type of group. When you remove a clone from an existing group, its type is change to `none` and it is added to the list of layout groups. However, its name is left unchanged, which means that the list might contain names like `SynchronousClone123`. Such groups are **not** synchronous clones.

4. From the *Synchronization reference* drop-down, choose the reference group with which the others are to be synchronized.
5. Click *Create Family*.

As long as the selected groups are valid clones of each other, the software creates the new family, synchronizes it against the reference group, and adds it to the *Clone Families* pane.

Note: Labels and text displays are not synchronized. Any dummy devices present in the reference group are copied in other occurrences of the clone.

6. Click *Close* to dismiss the Update Clone Families form.

Adding a Group to an Existing Family

To add a layout group to an existing family,

1. From the layout window menu bar, choose *Connectivity – Update – Clone Families*.
The Update Clone Families form is displayed.
2. Click *Create/Add To Family*.

The form is expanded to include the advanced options.

3. In the *Clone Families* pane, select the family you want to update,
4. In the *Layout groups* pane, select the group to be added to the family.

The *Synchronization reference* drop-down and the *Add To Family* button are both enabled.

Virtuoso Layout Suite XL User Guide

Generating a Layout



Important

The *Layout groups* pane lists all the regular layout groups not currently assigned to a clone family. It does not list any other type of group. When you remove a clone from an existing group, its type is change to `none` and it is added to the list of layout groups. However, its name is left unchanged, which means that the list might contain names like `SynchronousClone123`. Such groups are **not** synchronous clones.

5. From the *Synchronization reference* drop-down, choose one of the following.

- ☐ The reference group with which all other members of the family are to be synchronized.
- ☐ The existing family, if the new group is to be changed to match existing family members.

6. Click *Add to Family*.

As long as the selected group can be considered a clone of the other members in the family you selected, it is added to the family.

The family is updated in accordance with the synchronization reference, and the *Clone Families* pane is updated.

Note: Labels and text displays are not synchronized. Any dummy devices present in the reference group are copied in other occurrences of the clone.

7. Click *Close* to dismiss the Update Clone Families form.

Generating Synchronous Copy

Synchronous Copy is generated when an object or a set of objects that exist in the layout is copied and the synchronization between the copy source and the copy target is maintained. The copy source and the copy target are each generated as a group in the layout view. Each group is called a synchronous copy and a set of such groups is called a synchronized family. Any changes made to the copy source are automatically reflected in the copy target and vice versa. For example, if you edit the routing in the copy source or the copy target, the change is replicated in the other group.

Note: Labels and text displays are not synchronized because these are *clone non-checked objects*. For more information, see Clone Non-Checked Objects.

You use the *Create Synchronous Copy* check box available in the Copy or Repeat Copy form to create synchronous copies.

Note: *Create Synchronous Copy* feature is available in Layout XL and above.



The synchronous copy generated by using the Copy or Repeat Copy form cannot be differentiated from the synchronous clones generated using the Generate Clones form. These clones differ only in the way they are generated.

Unlike the mechanism used when generating clones by using the Generate Clones form, *Create Synchronous Copy* provides a synchronous grouping feature that does not involve

searching the schematic for an appropriate match. Therefore, you can create a synchronous group of objects that exist only in the layout. For example, you can synchronously copy some routing objects, which are objects that exist only in the layout, to create a synchronous routing group. You can also use a single device, a set of devices, or a chained set of devices that exist in the layout, as the copy source. The copy source can contain the entire chain or a part of it.

Copying from One Cellview to Another

If the target cellview is different from the source cellview, synchronous copy cannot be created.

If you run the *Copy* or *Repeat Copy* command with *Create Synchronous Copy* selected to create a copy target in a cellview other than the cellview in which the source is selected, synchronous copies are not created. However, a regular copy of the source is created in the source cellview and the following warning message is generated:

“Cannot create synchronous clones because the source and the target clones are not in the same cellview. The clone is created as a free object instead.”

Generating a Synchronous Copy

To generate a synchronous copy:

1. In the layout canvas or Navigator, select the copy source for which you want to create a synchronous copy.
2. Choose *Edit – Copy*, and then press F3.

The *Copy* form displays.

Note: Alternatively, you can right-click the copy source in the layout canvas and choose *Copy*. Press F3 to display the *Copy* form.

3. Select the *Create Synchronous Copy* check box to create a copy that is synchronized with the source.

Note:

- By default, the *Create Synchronous Copy* check box is deselected every time you open the *Copy* form.
- If you do not select the *Create Synchronous Copy* check box, a copy of the selected source is created, but it is not synchronized with the source. The only exception to this is observed when you choose a synchronous clone or a synchronous copy as the copy source. In this case, the copy target is a

synchronous copy even if the *Create Synchronous Copy* check box is not selected. For more information, see [Synchronous Copy with syncClone Source Group](#).

4. Select other copy options, as appropriate.
5. Click *Hide* to close the form.
6. Select the area on the layout canvas where you want to place the copy target.

The copy source and the copy target are added to a synchronized family, which can be accessed by using the Navigator.

Important

You can also generate a synchronous copy by using the *Create Synchronous Copy* check box on the *Repeat Copy* form. To open the form, choose *Edit – Repeat Copy*, and then press F3. For more information about the options available in this form, see [Repeat Copy](#).

Video

For a video demonstration of this function, see [Generating Synchronous Copy](#).

Selecting a Clone Family

To select a clone family:

- ➔ In the layout canvas or Navigator, right-click a clone and choose *Select Clone Family*.

All the clones in the family of the chosen clone are selected in the layout canvas and also in the Navigator.

Alternatively, you can use the [lxSelectSynchronousFamily](#) SKILL function to select a clone family.

Updating a Clone Family

To update a clone family:

- ➔ 1. In the layout canvas or Navigator, right-click a clone and choose *Update Clone Families*.

Note: Alternatively, in the layout window, choose *Connectivity – Update – Clone Families*.

The Update Clone Families form is displayed. You can use this form to rename the family of the selected clone, verify the synchronization of the family members, or to make other similar updates to the family of the selected clone.

 Important

The *Create Family* and *Add To Family* commands are not available when updating a synchronous copy.

 Caution

Any changes that you make by using the Update Clone Families form are irreversible.

Removing a Clone from a Family

To remove a clone from its family:

- ➔ In the layout canvas or Navigator, right-click the clone that you want to remove from its family and choose *Remove Clone From Family*.

The selected clone is removed from its family, which means that the clone is no longer synchronized with the other family members.

Note: Alternatively, you can use the lxRemoveSynchronousCloneFromFamily SKILL function to remove the selected clone from its family, or use the *Remove Clone* button available in the Update Clone Families form.

Removing a Clone Family

To remove a clone family:

1. In the layout canvas or Navigator, right-click a clone that belongs to the clone family you want to remove and choose *Update Clone Families*.

The Update Clone Families form is displayed.

2. Select the clone family from the list of families and click the *Remove Family* button.

The selected clone family is removed, which means that the family members are no longer synchronized.

Note: Alternatively, you can use the lxDeleteSynchronousCloneFamily SKILL function to remove the selected clone family.

Synchronous Copy with syncClone Source Group

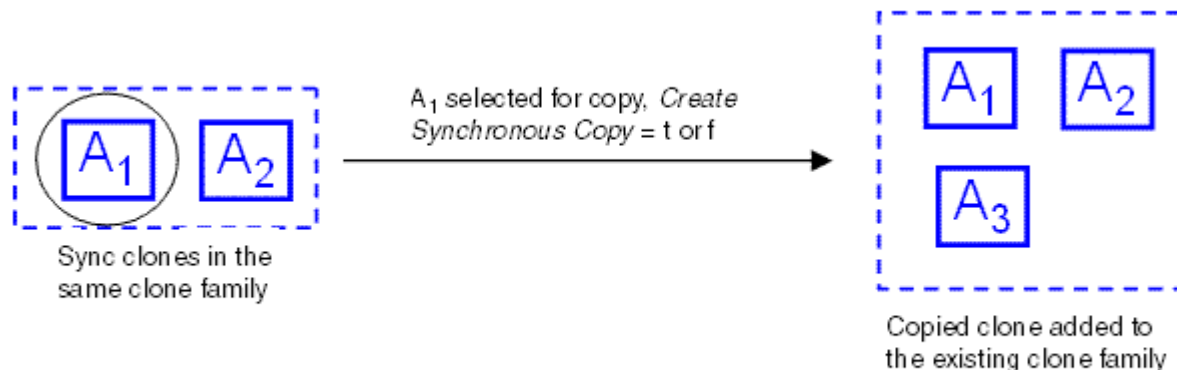
Before IC6.1.5, if you selected a synchronous clone—a group of type “syncClone”—as the clone source, the result would be a regular group instead of a synchronous clone.

Now, you can use a synchronous clone as the copy source to create a new clone. The copy target is a group of type “syncClone” and is added to the same synchronized family as the copy source. However, depending on the number of synchronous clones you choose to copy, the result of the *Copy* command varies.

Copying a Single Synchronous Clone

If you select a single synchronous clone as the copy source, the resulting copy is a synchronous copy, irrespective of the state of the *Create Synchronous Copy* check box. The copy target is added to the synchronized family to which the copy source belongs.

For example, you have two synchronous clones, A_1 and A_2 , in a clone family. If you copy either of the synchronous clones, a new synchronous clone, A_3 , is added to the synchronized family even if the *Create Synchronous Copy* check box is not selected. The result is the same if the *Create Synchronous Copy* check box is selected.



Converting a Synchronous Clone Group to a Basic Group

To convert a synchronous clone group to a basic group—a group of type “none”:

- ➔ Right-click the synchronous clone and choose *Remove Clone From Family* from the shortcut menu. For more information, see [Removing a Clone from a Family](#).

Note: Alternatively, you can remove a clone from its family by using the *Remove Clone* button on the Update Clone Families form. For more information, see Updating Clone Families.

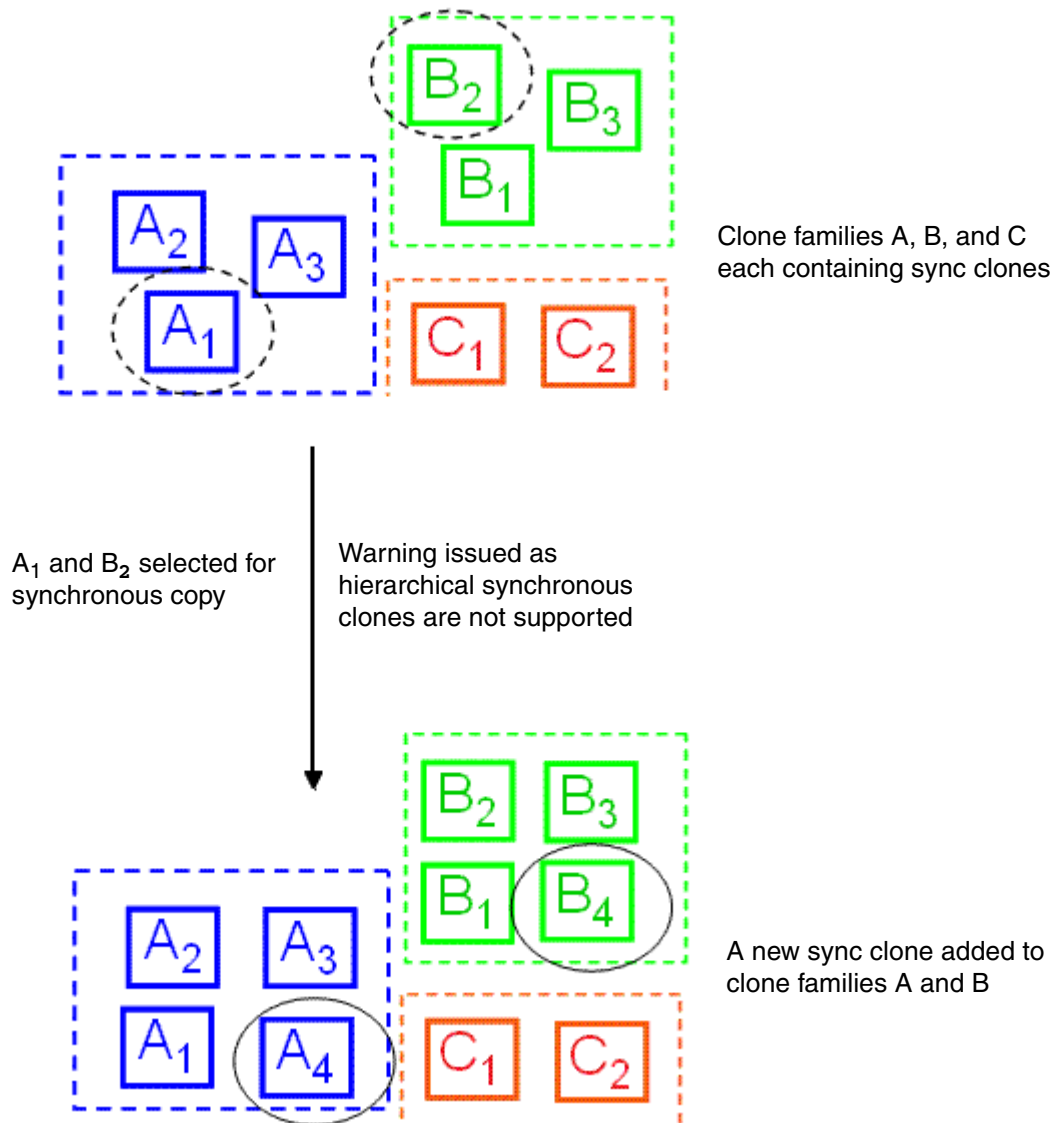
The selected synchronous clone is removed from its family. As a result, the group type changes from “syncClone” to “none” and the group is no longer synchronized with the other members of the family.

Copying Multiple Synchronous Clones

If you select multiple synchronous clones from a single synchronized family or from multiple families, Layout XL processes the request based on the state of the *Create Synchronous Copy* check box.

Case 1: Copying Multiple Synchronous Clones with Create Synchronous Copy Selected

Consider three clone families: A, B, and C. If you copy the synchronous clones A_1 and B_2 , as shown in the figure below, with the *Create Synchronous Copy* check box selected, a warning message is generated indicating that the hierarchical cloning request could not be completed. However, a copy of each selected clone is created in its own family.



Case 2: Copying Multiple Synchronous Clones with Create Synchronous Copy Deselected

If you do not select the *Create Synchronous Copy* check box while copying multiple synchronous clone sources, no warning message is generated and a copy of each selected source is added in its own family.

Synchronous Copy with Other Source Groups

Depending on the group type that is selected for generating a synchronous copy, the generated copy may or may not be synchronous with the source. However, a basic copy of the copy source is always created.

Basic Groups

If you select a basic group—a group of type “none”—as the copy source and *Create Synchronous Copy* is not selected, a regular copy of the object is created and the resultant object is also of group type “none”. However, if *Create Synchronous Copy* is selected, both the copy source group and the copy target group become synchronous copies and their group type changes from “none” to “syncClone”. The copy source and the copy target are gathered in a new synchronized *family*.

Modgen Groups

If the copy source contains a modgen group, synchronous copy fails because Layout XL does not support synchronization of modgens. If the *Create Synchronous Copy* check box is selected, the source modgen is copied as a basic group and the following warning message is generated:

“The copy source contains a group with an unsupported type; the copy cannot be synchronized.”

Non-Basic Groups

Non-basic groups include groups such as custom row and placement area. If the copy source contains a non-basic group and the *Create Synchronous Copy* check box is selected, synchronous copy fails and a regular copy of the group is created instead. The following warning message is generated indicating that the selected copy source is not supported for synchronous copying:

“The copy source contains a group with an unsupported type; the copy cannot be synchronized.”

Synchronous Copy with Different Source Objects

Microwave Objects

If the copy source is a microwave object and the *Create Synchronous Copy* check box is selected, the copy source and the copy target are generated as synchronous copies. However, synchronization of microwave objects is currently not supported. Therefore, edits made on one microwave object may not be reflected in the other copies.

PinFig Objects

If the copy source is a pinFig and the *Create Synchronous Copy* check box is selected, the copy source and the copy target are generated as synchronous copies. Any edits made to one of the pinFigs is replicated in all the other copies.

Mosaics

If the copy source is a mosaic and the *Create Synchronous Copy* check box is selected, the copy source and the copy target are generated as synchronous copies. Any edits made to one of the mosaics is replicated in all the other copies.

Rod Objects

If the copy source is a rod object and the *Create Synchronous Copy* check box is selected, the copy source and the copy target are generated as synchronous copies. Any edits made to one of the objects is replicated in all the other copies.

For more information about rod objects, see the [Virtuoso Relative Object Design User Guide](#).

P&R Objects

P&R objects include blockages, rows, placement area, clusters, prBoundary, snap boundary, and area boundary. Most of these objects, except some, such as prBoundary and snap boundary, can be synchronously copied using the *Copy* or *Repeat Copy* commands. If the copy source contains an unsupported object and the *Create Synchronous Copy* check box is selected, a warning message is generated and the object is not copied. Only those objects that are effectively copied are added to the source synchronous copy.

Bound Instances

If the copy source is a bound instance and you run the *Copy* command with the *Create Synchronous Copy* check box selected, a synchronous copy is created. If you run the *Copy* command with the *Create Synchronous Copy* option deselected, a regular copy of the source is created.

Note: When generating a synchronous copy using the *Create Synchronous Copy* option, no binding is created between the copy target instances created in the layout and the existing devices in the schematic. To create synchronous copies in the layout that are bound to the existing schematic devices, you must use the Generate Clones form instead.

Steiners and Guides

Steiners and guides are Clone Non-Checked Objects. These objects are not supported by the *Copy* and *Repeat Copy* commands.

If the copy source contains a steiner or a guide, the copy operation fails and an appropriate warning message is generated, indicating that the selected source type is unsupported.

Clone Non-Checked Objects

Clone non-checked objects are non-synchronized objects such as labels and text displays. When only clone non-checked objects are selected as the copy source, synchronous copies are not created. However, these objects, except some, such as prBoundary, are copied by the *Copy* and *Repeat Copy* commands.

For example, if you select an instance and a label as the copy source, both the objects are synchronously copied when you run a *Copy* or *Repeat Copy* command with the *Create Synchronous Copy* check box selected. However, if you edit, move, or delete the label in one copy, the changes are not reflected in the corresponding label in the other copy because the labels in the two copies are not synchronized.

SKILL Functions

The following is a list of SKILL functions that are supported by the Synchronous Cloning engine.

SKILL Function Name	Description
<code>IxDeleteSynchronousCloneFamily</code>	Deletes the specified family of synchronous clones. As a result, the member clones are desynchronized and they behave as basic groups. For more information, see <u>IxDeleteSynchronousCloneFamily</u>.
<code>IxGetCloneFamilyName</code>	Returns the family name of the specified clone. For more information, see <u>IxGetCloneFamilyName</u>.
<code>IxGetOtherClonesInFamily</code>	Returns a list of clones in a family other than the clone specified. For more information, see <u>IxGetOtherClonesInFamily</u>.
<code>IxRemoveSynchronousCloneFromFamily</code>	Removes a specified synchronous clone from its family. As a result, the clone is desynchronized and it behaves as a basic group. For more information, see <u>IxRemoveSynchronousCloneFromFamily</u>.
<code>IxSelectSynchronousFamily</code>	Selects all the clones in the layout canvas and the Navigator that are members of the specified clone family. If all the family members are already selected, no more clones are selected on the canvas or in the Navigator. If any new clones are identified, they are selected on the canvas and in the Navigator. For more information, see <u>IxSelectSynchronousFamily</u>.

SKILL Function Name	Description
<code>lxSetCloneFamilyName</code>	Sets a new name for the family to which the specified clone belongs. For more information, see <u>lxSetCloneFamilyName</u> .
<code>lxVerifyCloneFamily</code>	Runs a consistency check on the specified synchronous clone family to verify if the family members are identical to each other. For more information, see <u>lxVerifyCloneFamily</u> .

Loading Physical Information from Another Cellview

The *File – Load Physical View* command lets you import into the current cellview predefined information from another layout cellview, known as a source physical cellview. This approach lets you

- Use the full range of functionality available through the Virtuoso Layout Suite graphical user interface or the public SKILL API to create and edit the source cellview. You no longer need to manually edit an ASCII template file.
- Extend the functionality to accommodate new database objects as they are developed.
- Use any existing layout cellview as the source cellview.

Important

There is no mechanism to resolve inconsistencies in the technology data referenced by the source cellview and the target cellview – both must use the same technology database.



Video

For a video demonstration of this function, see [Using the Load Physical View Command](#).

Creating a Source Cellview

To create a source cellview,

1. Use the Virtuoso Layout Suite graphical user interface or the public SKILL API to create a cellview in the normal way.
2. Save the cellview by doing one of the following.
 - ☐ Choose *File – Save* to make the current working cellview the source cellview.
 - ☐ Choose *File – Save A Copy* to save the current cellview as a source cellview with a different name.

Loading a Source Cellview

To load information from a source cellview into a target cellview,

1. From the layout window menu bar, choose *File – Load Physical View*.

Virtuoso Layout Suite XL User Guide

Generating a Layout

The Load Physical View form is displayed.

Virtuoso Layout Suite XL User Guide

Generating a Layout

Virtuoso Layout Suite XL User Guide

Generating a Layout

Load Physical View

Choose Source Physical View

Library Name:

Cell Name:

View Name:

☒ Select All ☐ None

Update Instances

☒ Using Instance Name ☐ Using Master Name

☒ Standard Cells ☒ Custom Cells

☒ Macros ☒ IO Pads

☒ Add physical only instances

☒ Re-master parameterized instances

Load Pins

☒ Pins Mode: ☒ Replace ☐ Update

Nets

☐ Update Sig Type

Add Geometries

☐ Delete Shapes and Wires in Destination

☒ Shapes ☒ Wires

Update Boundaries

☒ PR Boundary ☒ Snap Boundary

Replace Rows

☒ Standard Rows ☒ Custom Rows

Replace Obstructions

☒ Blockages ☒ Halos

Type

☒ Placement ☒ Routing

☒ Slot ☒ Pin

☒ Fill ☒ Feedthru

☒ Screen

Transfer Constraints

☒ All Constraints and Process Overrides

Mode: ☒ Replace ☐ Update

2. Type in the *Library*, *Cell*, and *View* names for the source cellview in the fields provided.
3. Choose the object information you want to copy into the target cellview by checking the relevant boxes in each section.

To select everything, check the *Select All* box. To deselect everything, check the *None* box.

Choosing *Add Geometries* or *Transfer Constraints* requires a Layout GXL license. All other controls run in Layout XL.



Tip

If you choose to load a particular category, **all** the objects in that category are updated. For example, if you choose to update *Standard Cells*, then all the standard cells in the source cellview are updated to the target cellview. To copy only some cells, open both cellviews and use the *Copy* and *Paste* commands instead.

4. Click *OK* to copy the selected object information from the source cellview into the target cellview.

The target cellview is updated accordingly.

For information on how each type of information is handled, see [How Cellview Information is Handled](#) on page 303.

Loading Soft Block Parameters from a Source Cellview

When using the Configure Physical Hierarchy window to define soft block parameters, you can use the *Initialize Soft Block Parameters Using Physical View* command, to import import soft block parameters from an existing layout cellview.

This command uses the same functionality as the *Load Physical View* command described in the previous section. For more information, see [Loading Soft Block Parameters from Another Cellview](#) on page 176.

Modifying a Source Cellview

You modify a source cellview in the same way as you would any other cellview.

- Load the cellview into the appropriate Virtuoso Layout Suite application and use the graphical user interface to edit the cellview as required and save, or
- Use the publicly available SKILL commands to edit the cellview as required.

How Cellview Information is Handled

This section explains how information is loaded from the source into the target cellview for each of the categories listed on the Load Physical View form. It lists the conditions under which certain information cannot be loaded and explains the dependencies between objects.

Update Instances

To preserve the connectivity of the current design, only those instance attributes which do not affect either the connectivity or the constraints are loaded from the source into the target.

Source Cellview	Target Cellview	Action
Instance I1 present	Instance I1 not present	Instance is not updated.
Instance I1 present	Instance I1 present	Update the location, orientation, and placement status of the target instance, unless its placement status is either <code>locked</code> or <code>firm</code> . Attributes such as instance master, block type, cell type and number of bits are not changed because these can affect connectivity and constraints.

Add Physical Only Instances

Physical-only instances are instances that exist in the physical domain and do not have corresponding instances in the logical domain, for example, filler cells.

The *Add physical only instances* check box lets you add physical-only instances from the source cellview to the target cellview.



Tip

The `oaBlockDomainVisibility` attribute determines whether a block is physical-only or not. You can also use the `ins -> physOnly (t/nil)` SKILL function to identify if the instance is physical or not.

Virtuoso Layout Suite XL User Guide

Generating a Layout

The *Add physical only instances* check box adds physical-only instances in the target cellview if they do not already exist. If the physical-only instances exist already in the target cellview, the existing instances are updated.

	Source Cellview	Target Cellview
<i>Before</i>	I1 (LIB:AND) I2 (LIB:FILLER) (physOnly) I3 (LIB:FILLER1) (physOnly)	I1 (LIB:AND) I2 (LIB:FILLER) (physOnly)
<i>After</i>	I1 (LIB:AND) I2 (LIB:FILLER) (physOnly) I3 (LIB:FILLER1) (physOnly)	I1 (LIB:AND) I2 (LIB:FILLER) (physOnly) Updated I3 (LIB:FILLER1) (physOnly) Added

In the above example, I1 and I2 instances are updated in the target cellview because they existed in the source cellview. The I3 instance is added in the target cellview because it did not exist in the target cellview.

Remaster Parameterized Instances

The *Re-master parameterized instances* option lets you remaster those instances in the target cellview whose corresponding instance in the source cellview is a parameterized cell (pcell) instance.

If the instance already exists in the target cellview, the instance is updated in the target cellview. However if the pcell instance does not exist in the target cellview, the *Re-master parameterized instances* option searches for the instance in the module hierarchy. If the instance is found, the software updates the module instances in the target cellview to the design instances corresponding to the submasters in the source cellview.

For example, consider a source cellview and a target cellview that has the following instances:

Source Cellview	Target Cellview
I1 (INV_PCELL) subMaster:PCELL0 I2 (INV_PCELL) subMaster:PCELL1	Neither design instance I1 or I2 exists. Modules PCELL0 and PCELL1, and the I1 and I2 instances exist in the module hierarchy.

In this case, the **module instances** I1 and I2 in the target cellview are re-mastered to the **design instances** corresponding to the submasters in the source cellview.

This is done to ensure that the source cellview and the target cellview are in sync when data is imported using `oa2verilog` and `verilog2oa`. For example, a source cellview can have the following instances:

I1 (INV_PCELL) -subMaster:PCELL0 (different set of parameters)

I2 (INV_PCELL) -subMaster:PCELL1 (different set of parameters)

When data is imported using `oa2verilog` (the verilog format does not support pcells), it creates two modules and two instances corresponding to the modules as shown below.

```
module inv_pcell0(IN,OUT);
    inout IN;
    inout OUT;
endmodule // inv
```

```
module inv_pcell1(IN,OUT);
    inout IN;
    inout OUT;
endmodule // inv
```

```
module TOP (...);
    ...
    inv_pcell1 I2();
    inv_pcell0 I1();
    ...
endmodule // TOP
```

The target cellview when created from this verilog file, which is now changed during ECO, by using `verilog2oa` will have the I1 and I2 instances, which will be module instances and not design instances. Therefore these will exist in the module domain only.

Therefore, the **module instances** I1 and I2 in the target cellview are remastered to the **design instances** corresponding to the sub-masters in the source cellview.

Load Pins

The *Load Pins* option provides the following two modes for updating pins.

Replace Mode

If a terminal corresponding to the source terminal exists in the target cellview, the existing terminal pins in the target cellview are deleted and the terminal pins from the source cellview

Virtuoso Layout Suite XL User Guide

Generating a Layout

are copied to the target cellview. If a terminal corresponding to the source terminal does not exist in the target cellview, the extra terminals are not created in the target cellview. The table summarizes the behavior.

Source Cellview	Target Cellview	Action
T1 terminal P1 pin F1 pin figure exist	T1 terminal does not exist	The extra terminal (T1) is not created in the target cellview to keep the schematic and logical views in sync
T1 terminal P1 pin F1 pin figure exist	T1 terminal P2 pin F3 and F4 pin figures exist	The P2 pin, F3, and F4 pin figures are deleted from the target cellview. The P1 pin and the F1 pin figure are copied from the source cellview to the target cellview.

Note: The placement status of the target pins is not considered during replacement.

Update Mode

If the names of the terminal, pin, and pin figure in the `terminal:pin:pin figure` structure of the target cellview matches the corresponding names in the source cellview, the layer-purpose pair and the bounding box of the target pin figure is updated (as long as the placement status is not `locked` or `firm`).

If only terminal and pin names match, the extra pin figures are created in the target cellview (see example below). If only terminal names match, the extra pins and pin figures are created in the target cellview. The existing pins and pin figures are not deleted in the target cellview.

Virtuoso Layout Suite XL User Guide

Generating a Layout

For example, consider the pin structure in the source and target cellview in the following table.

	Pin Structure in Source Cellview	Pin Structure in Target Cellview
<i>Before</i>	T1 P1 F1, F2, F3 P2 F1 Tx Px Fx	T1 P1 F1, F2, P2 F1
<i>After</i>		T1 P1 F1, F2, F3 P2 F1

The F3 pin figure in pin P1, the P2 pin, and the F1 pin figure in P2 pin (highlighted above) are added in the target cellview. In addition, the F1 and F2 pin figures in P1 pin are updated and no new terminal corresponding to Tx is created.

Add Geometries



This functionality checks out a Layout GXL license.

Geometries are considered to be shapes (rectangles, polygons, paths, circles, ellipses, and donuts) and wires (pathSegs and vias). Because of the difficulty in establishing that shapes are at exactly the same location in the source and target views, this control simply copies geometries from the source to the target. This can result in the creation of overlapping shapes in the target layout.

Source Cellview	Target Cellview	Action
Shape or wire present	A shape or wire is present at the same location	Shape or wire is copied, which might lead to the creation of overlapping shapes.
Shape or wire present	No shape or wire present at the same location	Shape or wire is copied

Virtuoso Layout Suite XL User Guide

Generating a Layout

The table below summarizes how net assignments are handled when a shape is copied from the source to the target.

Source Cellview	Target Cellview	Action
Shape is assigned to <code>netA</code> and has <code>lxStickyNet=t</code>	<code>netA</code> present	Shape is copied and assigned to <code>netA</code> . <code>lxStickyNet</code> property is added to the shape and set to <code>t</code> .
Shape is assigned to <code>netA</code> and has <code>lxStickyNet=t</code>	<code>netA</code> not present	Only the shape is copied. <code>lxStickyNet</code> property and net assignment are both ignored.
Shape has no <code>lxStickyNet</code> property		Only the shape is copied.
Shape has no net assignment but has <code>lxStickyNet=t</code>		Only the shape is copied. <code>lxStickyNet</code> property is ignored.

Update Boundaries

This information applies to both the place and route boundary and the snap boundary.

Source Cellview	Target Cellview	Action
Boundary present	Boundary not present	Copy the boundary from the source to the target cellview.
Boundary present	Boundary present	Update the points of the target boundary, along with other attributes such as <code>IOBox</code> and <code>CoreBox</code> .

Replace Rows

The standard and custom rows in the target cellview are deleted and replaced by the rows defined in the source cellview.

Replace Obstructions

An obstruction is implemented either as a blockage or a halo of one of the following types.

- Placement (area blockage or halo)
- Routing (layer blockage or halo)
- Slot (layer blockage or halo)
- Pin (layer blockage or halo)
- Fill (layer blockage or halo)
- Feedthru (layer blockage or halo)
- Screen (layer blockage or halo)

All the obstructions in the target cellview are deleted and replaced by the obstructions defined in the source cellview with the following conditions.

- A blockage can be owned by an instance or by a cluster. If the owner is present in the target cellview, it is updated on copying the blockage from source to target.
- A halo can be owned by an instance or by the place and route boundary. Before copying a halo, the software checks that the owner is present in the target cellview. If it is not, the halo is not copied.

Important

Wiring and *Via* obstructions are to be removed from DFII, therefore they are not handled by this functionality.

Transfer Constraints

Important

This functionality checks out a Layout GXL license.

Constraints are the relationship constraints specified in the schematic cellview by the circuit designer and process overrides driven by the process technology. Constraints are transferred from the source cellview to the target cellview based on the default mapping rules specific to each constraint. The *Transfer Constraints* option provides the following modes.

Replace mode deletes the existing constraints in the target cellview and then transfers the constraints from the source cellview to the target. In this mode, all the existing constraints in the target cellview are lost.

Update mode updates the constraints in the target cellview with those from the source cellview, including any constraints that exist in the source but not in the target. Similarly, if the target cellview contains constraints that do not exist in the source cellview, these constraints are retained in the target cellview.

Editing the Layout

This chapter shows you how to edit the placement of components in the layout and how to add additional components. This chapter discusses the following topics.

- [Constraint-Aware Editing](#) on page 312
- [Moving Objects](#) on page 315
- [Adding Components](#) on page 323
- [Adding a Pin](#) on page 326
- [Transistor Folding](#) on page 328
- [Pin Permutation](#) on page 341
- [Working with Incomplete Nets](#) on page 347
- [Editing In Place](#) on page 368
- [Editing Soft Blocks](#) on page 369
- [Soft Block Pins](#) on page 410
- [Setting the Pin Connectivity Model](#) on page 423

For information on defining pin groups for external connections (must-connect groups), see [Using Connectivity](#) in the *Virtuoso Layout Suite L User Guide*.

For information on using constraints to restrict the placement of objects, see the [Virtuoso Unified Custom Constraints User Guide](#).

Constraint-Aware Editing

Constraint-Aware Editing mode ensures that the Layout XL manual editing commands respect physical constraints that may be set on your design components.

Constraint-Aware Editing also checks constraints and displays violation markers at the appropriate location in the design window, if necessary.

As design edits are made, Constraint-Aware Editing continuously checks and readjusts the markers. Constraint-Aware Editing mode is ON by default.

This feature allows layouts to be generated interactively while constraints are respected and the overall topology of the design is maintained, ensuring the designer's original intent is honored throughout the development of the physical implementation.

Enabling and Disabling Constraint-Aware Editing Mode

Constraint-aware editing is a session-wide setting. It applies to all existing and new windows opened in the current Virtuoso session. To toggle constraint-aware editing, do one of the following.

- With your cursor in the layout window, press F10.

Note: This works only if you have not reassigned the F10 key to a different function. For a complete list of default bindkeys for Layout XL, see [Default Bindkeys Defined for Layout XL](#).

- Press the CAE button on the *Options* toolbar.



- Click the right mouse button and choose *Constraint-aware Editing* while using one of the supported commands: *Create Wire*, *Create Point to Point*, *Copy*, *Move*, *Stretch*, and *Rotate*.
- Change the setting on the [General Tab](#) of the Layout XL Options form,
 - a. From the layout window menu bar, choose *Options – Layout XL*.
 - b. On the *General* tab, check or uncheck the box labeled *Constraint-aware editing*.

Click *OK* to accept the changes.

Objects, Commands, and Constraints Supported in Constraint-Aware Editing Mode

Constraint-Aware Editing supports the following design object constraint members:

- Instances, pins, and groups
- Wires and vias, but only for the `locked` constraint
- Place and route boundary, but only for pin placement constraints when using the *Stretch* command.

The interactive commands that are supported by Constraint-Aware Editing are *Edit – Copy*, *Edit – Move*, *Edit – Stretch*, and *Edit – Rotate*. These honor the following constraints:

<i>Alignment</i>	Defines how the constrained objects are aligned, either relative to each other or to some other reference object.
<i>Fixed</i>	Prevents the constrained objects from being edited by an automatic tool, but allows them to be edited manually.
<i>Locked</i>	Prevents the constrained objects from being edited in any way.
<i>Matched Parameters</i>	Defines that the specified parameters for the constrained objects must have values in ratios specified by the ratio parameter. An empty Match Subset implies that all device parameters must be in the correct ratio. If a layout XL environment variable named <code>matchedToIgnore</code> exists then the parameter names specified by that space-separated string will not be checked.
<i>Orientation</i>	Defines allowed orientations for the constrained objects.
<i>Matched Orientation</i>	Defines that the constrained objects must have the same orientation.
<i>Symmetry</i>	Defines symmetries for the constrained objects about some axis.
<i>Distance</i>	Defines a range of allowed distances between constrained objects with respect to a specified reference point (i.e. origin, left side, right side).

Virtuoso Layout Suite XL User Guide

Editing the Layout

<i>Cluster</i>	Prevents non-cluster objects from encroaching cluster space of constrained objects.
<i>Cluster Boundary</i>	Defines a space for a cluster. Depending upon whether Cluster Boundary is set as exclusive or not and whether flexibleFlag parameter is true or not the constraint will prevent the movement of the members outside the cluster boundary or will adjust the cluster boundary as per the movement. For the interactive CAE move to happen, the flexibleFlag is ignored.

For detailed information on each of these constraints, see [Default Constraint Types](#) in the *Virtuoso Unified Custom Constraints User Guide*.

Notes on Constraint-Aware Editing

- If you use design-rule-driven editing in the Enforce mode in conjunction with constraint-aware editing, design rules are enforced not only for the object you are editing, but for all its constraint partners as well.

Note: For information on design-rule-driven editing, see the [Virtuoso Design Rule Driven Editing User Guide](#).

- Constraint-aware editing does not edit or ghost constraint partners in pseudo-hierarchy levels. For example, if you have a symmetry constraint with one member inside a group and another member outside the group, then an edit to one member will not cause an additional edit to the other, even if the constraint becomes violated. In this situation, the following warning message is displayed:

“Device will not be edited because it is on a different hierarchy level than its constraint partner. Ungroup this device to prevent constraints from being broken.”

- When rotating a quad-symmetric device configuration using the axes of symmetry, you must select both the axes—vertical and horizontal—and then rotate them. If you rotate only one of the symmetry axes, the quad symmetry of the device is destroyed, resulting in constraint violations.
- The Layout GXL analog auto placer creates markers for constraint violations even when constraint-aware editing is not enabled. This lets you see the violations created by the placer regardless of the editing mode.

Note: For information on the analog auto placer, see the [Virtuoso Analog Placement User Guide](#).

Moving Objects

There are three ways to move devices, pins, or shapes in the layout canvas.

- Use the *Move* command to manually drag the object to its new location.
- Use the Move options form to specify more precisely the conditions of the move operation.
- Use the *Place As In Schematic* command to move all the generated components inside the design boundary. If your design carries any unbound instances or pins, these will be placed below the design boundary.

Moving Objects Manually

To move an object manually in the layout canvas,

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

The layout window prompts you to select an object to move.

2. Click on an object in either the layout or the schematic.

The object is highlighted in both the schematic and layout windows. An outline of the object appears in the layout canvas and follows the cursor until you click in the layout canvas to place the object.

Draglines show the connections from pins of the selected object to pins of the nearest instance or the nearest I/O pin. For more information, see [Changing the Appearance of Draglines](#).

The layout window prompts you to click on the new location for the object.

3. Click on the location where you want to place the object.

The object is moved to the new location.

Note: If you select a component in the layout or the schematic before you choose the *Move* command, Layout XL highlights the corresponding component in the schematic or layout.

4. Click on another object to move or press `ESC` to cancel the command.

Note: Using the *Move* command cancels any existing commands in the current or in other windows.

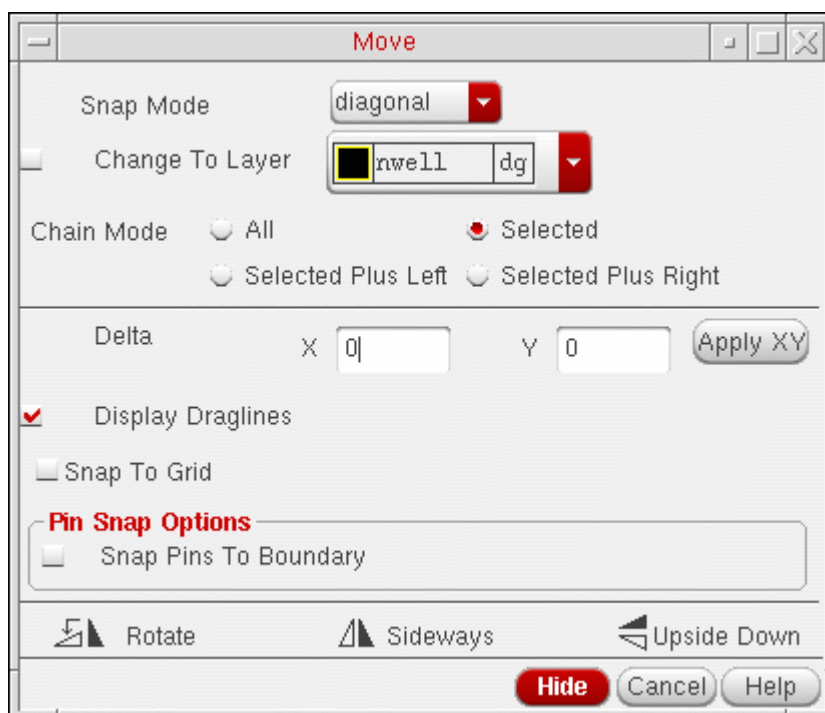
Moving Objects Using the Move Form

You can use the Move form to change

- The layer of the object you select.
- The angles at which you can move the object.
- The presence of draglines on the object as you move it.
- The orientation of the object you select.

To move an object using the Move options form,

1. From the layout window menu bar, choose *Edit – Move*.
2. Press F3 to display the Move options form.



3. To change the angle at which you can move an object, choose one of *anyAngle*, *diagonal*, *orthogonal*, *horizontal*, *vertical* from the *Snap Mode* cyclic field.
4. To change the layer on which a shape is drawn, turn on *Change To Layer* and choose the new layer from the cyclic field.

The layer is updated when you move the cursor into the layout canvas.

5. To keep chained devices together as you move them, select a *Chain Mode*. Choose from *All*, *Selected*, *Selected Plus Left*, or *Selected Plus Right*.

These options are available only in Layout XL.

6. To show draglines for an object you move, turn on *Display Draglines*.

For more information on draglines, see [Changing the Appearance of Draglines](#).

7. To specify a distance to move the selected component, use the *Delta X* and *Y* fields.

- a. Type the distance to move in the horizontal direction into the *X* field.
- b. Type the distance to move in the vertical direction into the *Y* field.
- c. Click *Apply X/Y* to complete the move.

8. To change the orientation of the object, click on the button describing the transformation you want: *Rotate*, *Sideways*, or *Upside Down*.

The orientation of the object changes when you click in the layout canvas.

Changing the Appearance of Draglines

When using the commands listed below, Layout XL displays draglines which connect the pins on the object to the nearest pins on another object on the net. When you move the object closer to a different object on the same net, the lines reconnect to the new object.

Note: This capability is available only in Layout XL.

- *Generate Selected From Source*
- *Generate Clones*
- *Move*
- *Stretch*

If you need to know more about the potential connections for the instance you are working on, you can change both the number and the appearance of the draglines using the options under *Draglines* in the [General Tab](#) of the Layout XL Options form.

You can display draglines for all the connections to the instance you are working on, hide the draglines for global nets and for power and ground nets, and have each dragline shown in a different color.



Tip

Displaying many draglines during interactive commands can impair the performance of Layout XL. To mitigate these effects, use the `maxDragFig` environment variable to limit the number of figures that can be in a dragset.

To set `maxDragFig` interactively so that it takes effect in the current session, type the following in the CIW, where `win` is the window to which the limit applies.

```
win~>maxDragFig = 1000
```

To set a default value for `maxDragFig`, which will be used in all future Layout XL sessions, put the following line in your `.cdsenv` file.

```
graphic maxDragFig int 1000 nil
```

Moving Generated Components into the Design Boundary

To move all the generated components in the layout and place them inside the design boundary at a position relative to their position in the schematic:

- From the layout window menu bar, choose *Connectivity – Generate – Place As In Schematic*.

Notes

- The Place As In Schematic (PAS) command moves all the instances and pins inside the design boundary, maintaining their position relative to the schematic. However, any unbound instances and pins are positioned below the design boundary.

The PAS command places physical-only power and ground pins at the top and bottom center of the design boundary, respectively.

To move the I/O pins so that they align to the edges of the design boundary, use the *Place – Pin Placement* command. For more information, see [Pin Placement](#) in the *Virtuoso Custom Digital Placer User Guide*.

- The PAS command keeps the hierarchical instances together and tries to fit all the instances and pins within the design boundary. If there is no design boundary, Layout XL places the devices within a square the size of the default boundary (25% utilization).
- Factored and iterated devices are kept together and positioned in one or more rows depending on the size and number of the devices.
- When placing iterated pins, the default shape for these pins is determined based on their closest boundary edge. If the pins are too long for placement along the closest boundary

edge, they are wrapped. Else, the pins are placed at the exact center of the design boundary.

- If some objects in the design, such as pins, are much smaller than the other components; the small objects might not initially appear in the layout, even though they have been moved. To view these small objects, you need to zoom into the layout.
- Although the PAS command aims at removing any component overlaps to retain the schematic topology, some overlaps might still exist, resulting in shorts. Markers for these shorts are displayed in the [Annotation Browser](#).
- Layout XL does not compact the layout, so the placement of components might not meet the Design Rule Checker (DRC) or other design requirements.
- Constraints entered using the Constraint Manager assistant are not honored by the PAS command. If a design has constraints, PAS will display the following message in the CIW:

“Design has placement constraints. The 'Analog Auto Placer' can be used to satisfy placement constraints.”

For more information about design constraints, see the [Virtuoso Unified Custom Constraints User Guide](#).

Swapping Components

The *Swap Components* command lets you swap the positions of two components in the layout window.

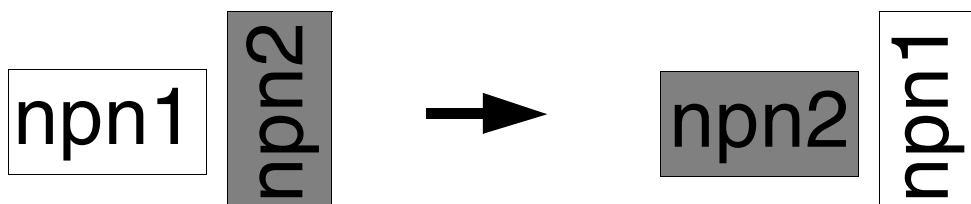
Layout XL does not move any connections along with the components. If swapping a pair of components causes a short, markers show you where the short is. If swapping components causes an open connection, flight lines show the open connection.

Note: The *Swap Components* command is not constraint aware.

To swap components,

1. From the layout window menu bar, choose *Edit – Advanced – Swap Components*.
2. Click on the first component in the layout canvas.
3. Click on the second component in the layout canvas.

Layout XL switches the locations of the components you selected, but retains the orientation of the component originally associated with the position.

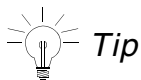


To undo the swap,

- Choose *Edit – Undo* from the layout window menu bar.

Locking and Unlocking Devices

The Layout XL device locking capability sets a temporary locked constraint on a device in the layout. A locked device cannot be moved during interactive or automatic placement.



You can view the locked constraint for the selected devices in the Property Editor assistant's *Placement Status* field and in Constraint Manager assistant. For more information on the latter, see the [*Virtuoso Unified Custom Constraints User Guide*](#).

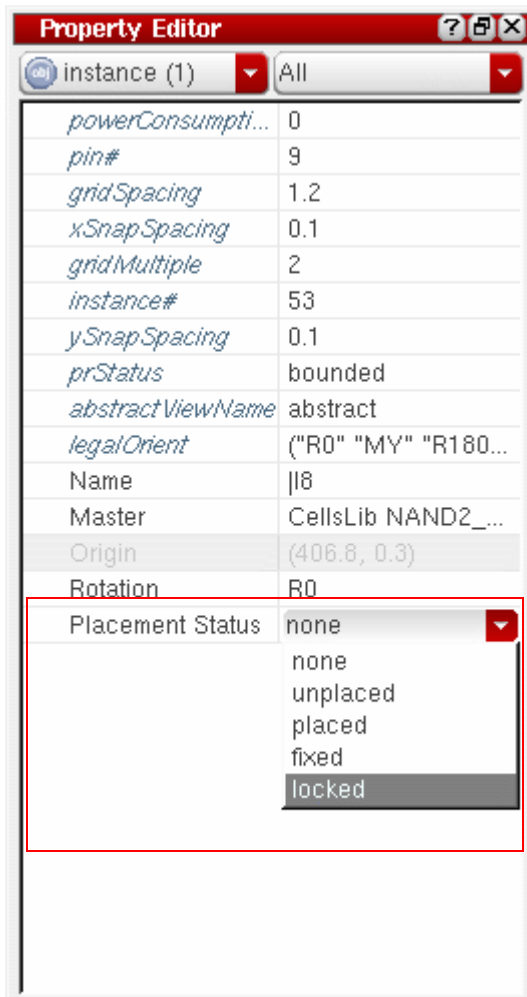
Locking a Device Using the Property Editor

To lock a device using the Property Editor assistant,

1. From the layout window menu bar, choose *Window – Assistants – Property Editor*.
The Property Editor assistant is displayed.
2. In the canvas, select the device you want to lock.

The property information for the selected device is displayed in the Property Editor.

3. In the Property Editor assistant, select the *Placement Status* field and choose *locked* from the drop-down list in the right hand column.



A locked constraint is placed on the selected device. The Constraint Manager assistant is updated accordingly.

Locking a Device Using the Constraint Manager

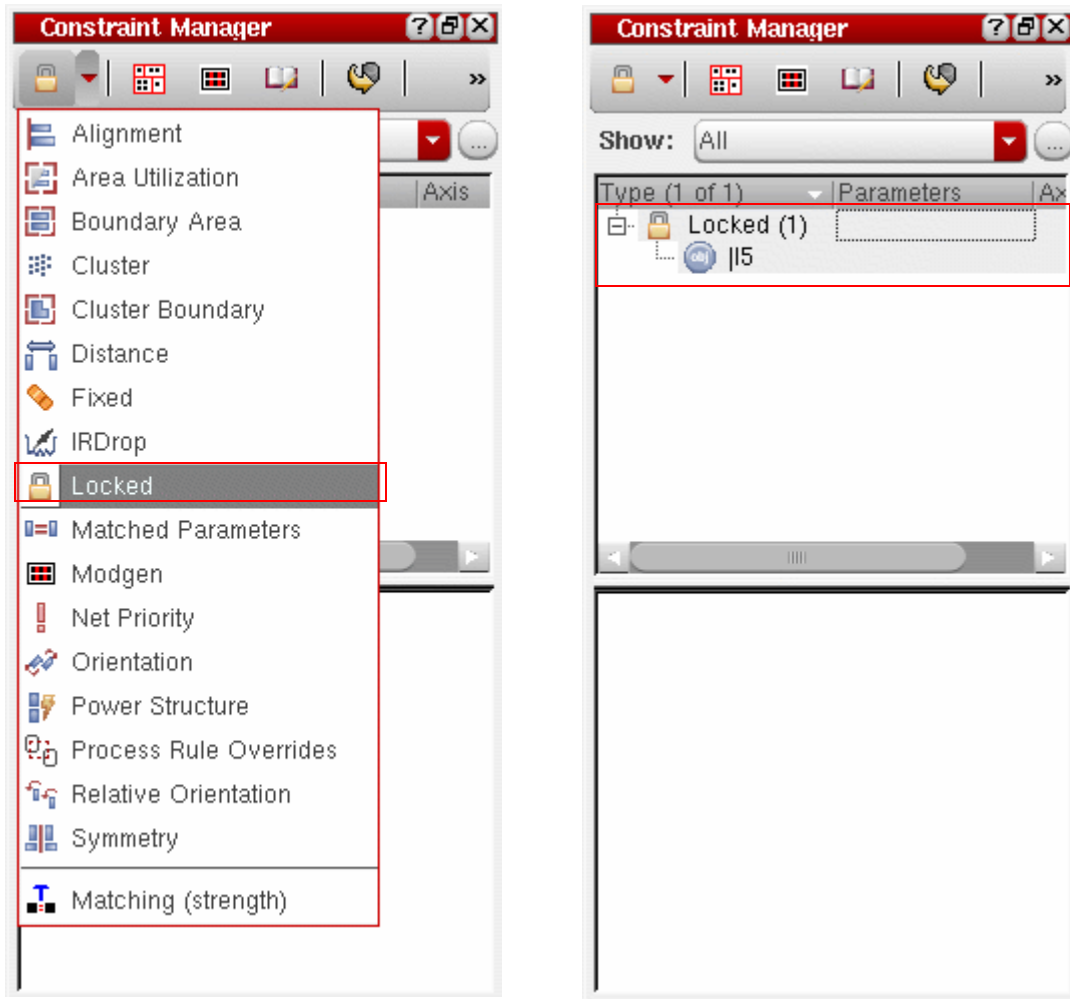
To lock a device using the Constraint Manager assistant,

1. From the layout window menu bar, choose *Window – Assistants – Constraint Manager*.

The Constraint Manager assistant is displayed.

2. In the canvas, select the device you want to lock.

3. Choose *locked* from the constraint pull-down in the Constraint Manager assistant toolbar.



A locked constraint is placed on the selected device and the Constraint Manager assistant is updated accordingly.

Unlocking a Device

To remove a lock on a device,

1. From the layout window menu bar, choose *Window – Assistants – Constraint Manager*.

The Constraint Manager assistant is displayed.

2. Choose the constraint you want to delete and click the *Delete* icon in the Constraint Manager toolbar.

The constraint is removed from the device.

You can also remove the locked constraint by changing the *Placement Status* in the Property Editor assistant. To do this,

To lock a device using the Property Editor assistant,

1. From the layout window menu bar, choose *Window – Assistants – Property Editor*.

The Property Editor assistant is displayed.

2. In the canvas, select the device you want to unlock.

The property information for the selected device is displayed in the Property Editor.

3. In the Property Editor assistant, select the *Placement Status* field and change the value from *locked* to *none*.

The locked constraint is removed from the selected device. The Constraint Manager assistant is updated accordingly.

Adding Components

Adding an Instance

There are two ways to add to the layout an instance that is not present in the schematic.

- Add the instance to the layout using the *Create Instance* command.

The incremental binder engine creates the appropriate bindings automatically, or you can use the *Define Device Correspondence* command to create user-defined bindings as before.

When adding physical-only instances, add the ignore property either at creation time or afterwards using the Property Editor assistant.

- Add the instance to the schematic, connect it to a net, and then use *Update Components And Nets* to generate the new instance in the layout view. You need write permission for the schematic to do this.

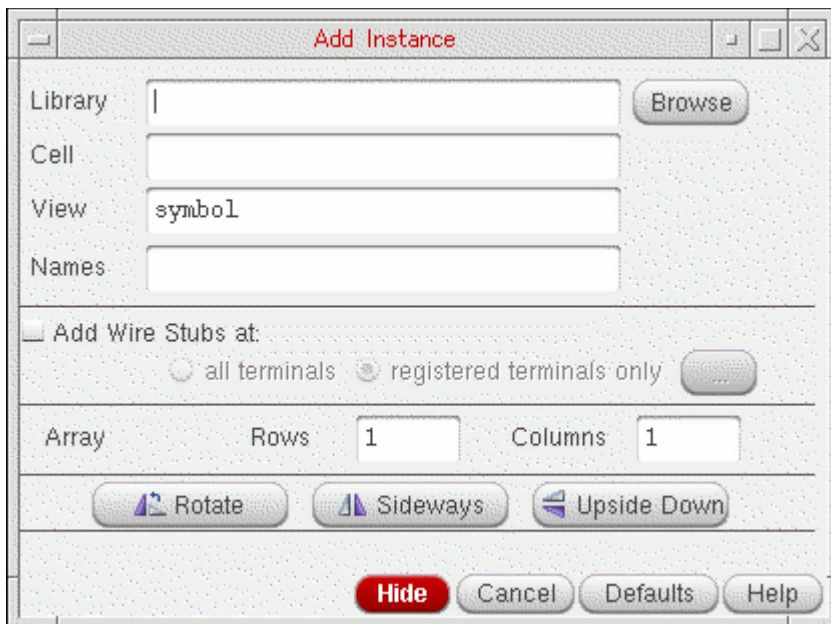
To add an instance to the layout using *Update Components And Nets*,

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

Virtuoso Layout Suite XL User Guide

Editing the Layout

The schematic editor's Add Instance form is displayed.



2. Type the *Library*, *Cell*, and *View* names for the new instance in the fields provided.
 3. Move the cursor to the schematic. The outline of the instance follows the cursor.
 - ❑ To rotate the instance, click the right mouse button.
 - ❑ To mirror the instance, press `Shift` and click the right mouse button.
- Note:** You must have the Layout XL bindkeys loaded in order to use the right mouse button to rotate and mirror.
4. Click in the schematic where you want to place the instance.
 5. To connect the instance to a net, use the schematic editor's *Create – Wire* command.
For more information, see Adding Wires in the *Virtuoso Schematic Editor User Guide*.
 6. To add the new instance to the layout view, choose *Connectivity – Update – Components And Nets* from the layout window menu bar.
 7. Click *Yes* to re-extract the schematic.

Layout XL extracts the schematic and adds the new instance to the layout.

The incremental binder engine automatically determines the binding for the newly added instance. To change the binding after creation, use the Define Device Correspondence form.

Note: If the instance does not need to be connected to a net, add it directly to the layout and add the ignore property to it.

Assigning Instance Pins to a Net

If you place an instance in a layout that does not have a connectivity source, the software does not connect the instances to any net unless you specifically assign the instance pins to a net using the *Assign Nets* command.

You can also use *Assign Nets* to connect a new instance to an existing net in a design that has a connectivity source.

To assign the pins of an instance to a net,

1. Place new instances in the layout using the *Create – Instance* command.
2. Use the Annotation Browser assistant to display the incomplete nets in the canvas.

For more information, see [Working with Incomplete Nets](#).

3. Choose *Connectivity – Nets – Assign*.

The layout window and the CIW prompt you to select a pin.

4. Click on a pin in one of the new instances you have placed.

Note: You can select multiple pins to be attached to the same net.

The pin you select is highlighted. The layout window and the CIW prompt you to select a net.

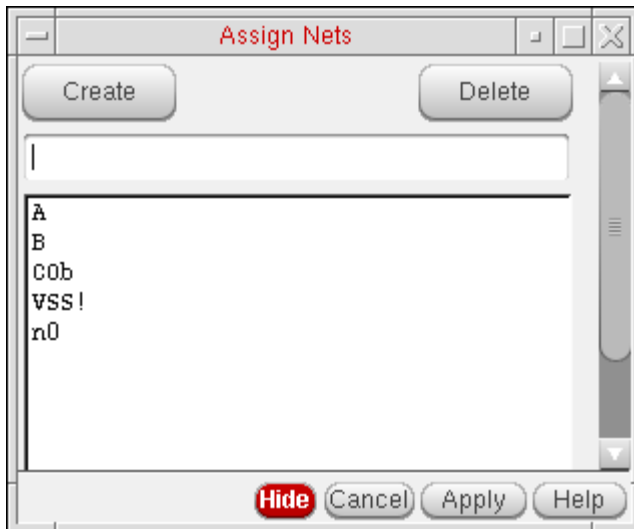
5. Click on a pin that is already connected to the net you want to add the new pin to.

The flight lines representing that net are extended to show that the new pin has been added to the net. The CIW lists the name of the pin and the name of the net it was connected to.

The layout window and the CIW prompt you to select another pin to add to a net.

6. Press **F3** to display a list of all the nets in the layout.

The Assign Nets form is displayed.



7. To add a pin you selected in the layout to a net listed in this form, click on the name of the net in the form and then click *Apply*.

The flight lines representing that net are extended to show that the pin is added to the net.

8. Press `ESC` to exit the command.

Adding a Pin

Pins that occur in both the schematic and the layout must have identical names. If you add pins that are not in the schematic to the layout, Layout XL maintains their connectivity in the layout.

To add a pin to the layout,

1. In the Layer Selection Window, choose the layer on which you want to create the pin.
2. From the layout window menu bar, choose *Create – Pin*.

Virtuoso Layout Suite XL User Guide

Editing the Layout

The Create Shape Pin form is displayed. For more information on this form, see [Creating Objects](#) in the *Virtuoso Layout Suite L User Guide*.

3. In the *Terminal Names* field, type the terminal name for the pin.

The pin name must match the name of an existing net in the schematic. You can create more than one pin with the same name.

To create a pin for a net that is not on the schematic (for example, a feedthrough), give the pin a new name.

4. In the *I/O Type* section, choose the appropriate I/O type.
5. In the layout canvas, click to place one corner of the pin; then click to place the opposite corner of the pin.

The pin appears in the layout canvas. If the Annotation Browser assistant is showing incomplete nets, flight lines are drawn to show the pin connected to the net.

Notes

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

You can also add pins defined with shapes. You can use any layer-purpose pair for pin shapes. Pins made as large as possible make routing easier.

For more information on the Create Shape Pin form, see [Creating Objects](#) in the *Virtuoso Layout Suite L User Guide*.

Transistor Folding

Transistor folding lets you interactively divide an individual transistor or chain of transistors into two or more layout instances with terminals all connected in parallel to the same nets. Folding lets you change the aspect ratio of the transistors in your design while retaining the original orientation of the device that was folded. The net on the left of the device (or on the bottom for vertically-oriented devices) is the same before and after folding.

The transistors to be folded need not be bound to a schematic instance but they must be assigned to a component type, which stores the width attributes set on the cell. You set this information using the *Edit Component Types* command. For more information, see [Component Types Mode](#).

The individual device folds are named `instName.integer` in the layout. For example, if you have an instance named `I0` and you divide it into three folds, the folds are named as follows.

Instance Name	Folded Instance Names
I0	I0.1 I0.2 I0.3

You must also have the [mfactorSplit](#) environment variable set to `t` in a setup file for the design, otherwise the *Generate Folded Devices* command does not work.

For more information, see

- [Folding a Transistor](#)
- [Ignoring Schematic Mfactor during Folding](#)

■ Unit Identifiers in Layout XL

Additional Information on Interactive Folding

- Interactive folding might overwrite the default `lxMaxWidth` parameter using the specifications you supply. For example, if you divide the transistor into too few folds, each fold might end up longer than the `lxMaxWidth` parameter you specified for the component type. Automatic folding always respects the `lxMaxWidth` parameter.
- If you enter an `lxMaxWidth` parameter that would yield a device with more than 999 folds (for example, a device width of 2 and `lxMaxWidth` of 1μ , which could result in a device folded into 2M folds), the CIW prints a warning and does not fold the device. You can still fold such a large device manually.
- Interactive folding rounds width values using `lxRounding` to provide the accurate Total Width and individual width values prior to creating devices in the layout. For example, assume you have a device IP0 with $w = 3u$ and `lxRounding = (w 0.5u round)`. If you IFold IP0 into 7 folds with equal widths, each width in the form will be displayed as 428.571429n. Now, if `lxRounding` is applied when the devices are placed in the layout, the width of each folded device is set to 500n, making the form widths inconsistent with the device widths in the layout.

Instead, if the interactive folding rounds the width values using `lxRounding` at the time of the initial calculation, the form widths achieved are consistent with the layout widths. This provides the users a more realistic and reliable fold width information before the actual creation.

- Interactive folding always considers the parameter that sets the device width in the source, even if you have specified that this parameter is to be ignored in the `paramsToIgnoreForCheck` environment variables.
- When *Generate Folded Devices* is active, you cannot select any pcell that has a fingering value set to greater than 1.
- To keep folded devices from becoming off-grid, set the *Rounding* option in the Configure Physical Hierarchy window. For more information, see [Rounding Parameter Values](#).

Folding a Transistor

To fold a transistor,

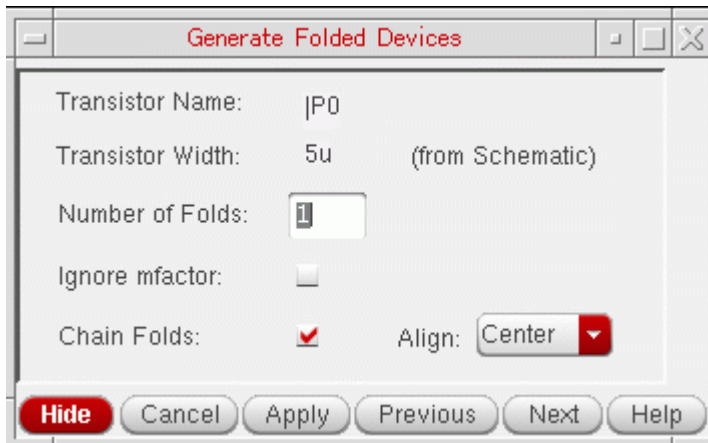
1. Select the transistor you want to fold in either the schematic or layout canvas.

Note: When selecting a device that has already been folded, make sure you select the whole device. To do this select it from the schematic window, or select all its folds in the

layout.

2. Choose *Connectivity – Generate – Folded Devices*.

The Generate Folded Devices form is displayed.



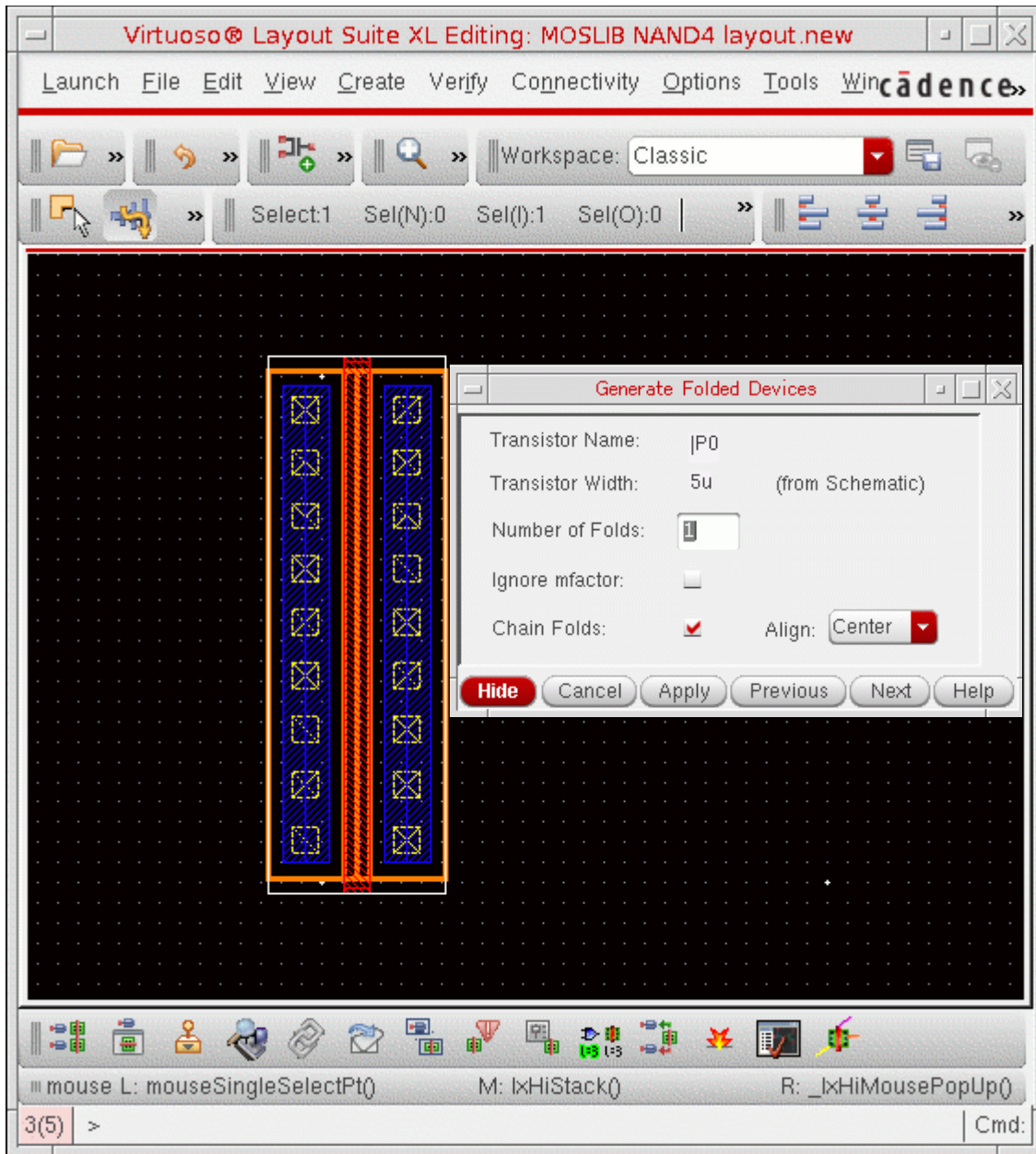
The name of the selected transistor is shown in the *Transistor Name* field. (If you did not select any transistors before choosing the command, no transistor-specific information is shown in the form until you select at least one transistor.)

The width of the selected transistor is shown in the *Transistor Width* field, which also indicates whether the value was retrieved from the schematic view or the layout view.

Virtuoso Layout Suite XL User Guide

Editing the Layout

3. Type the *Number of Folds* into which you want to divide the selected transistor.



4. Click *Apply* in the form and move the cursor into the layout canvas.

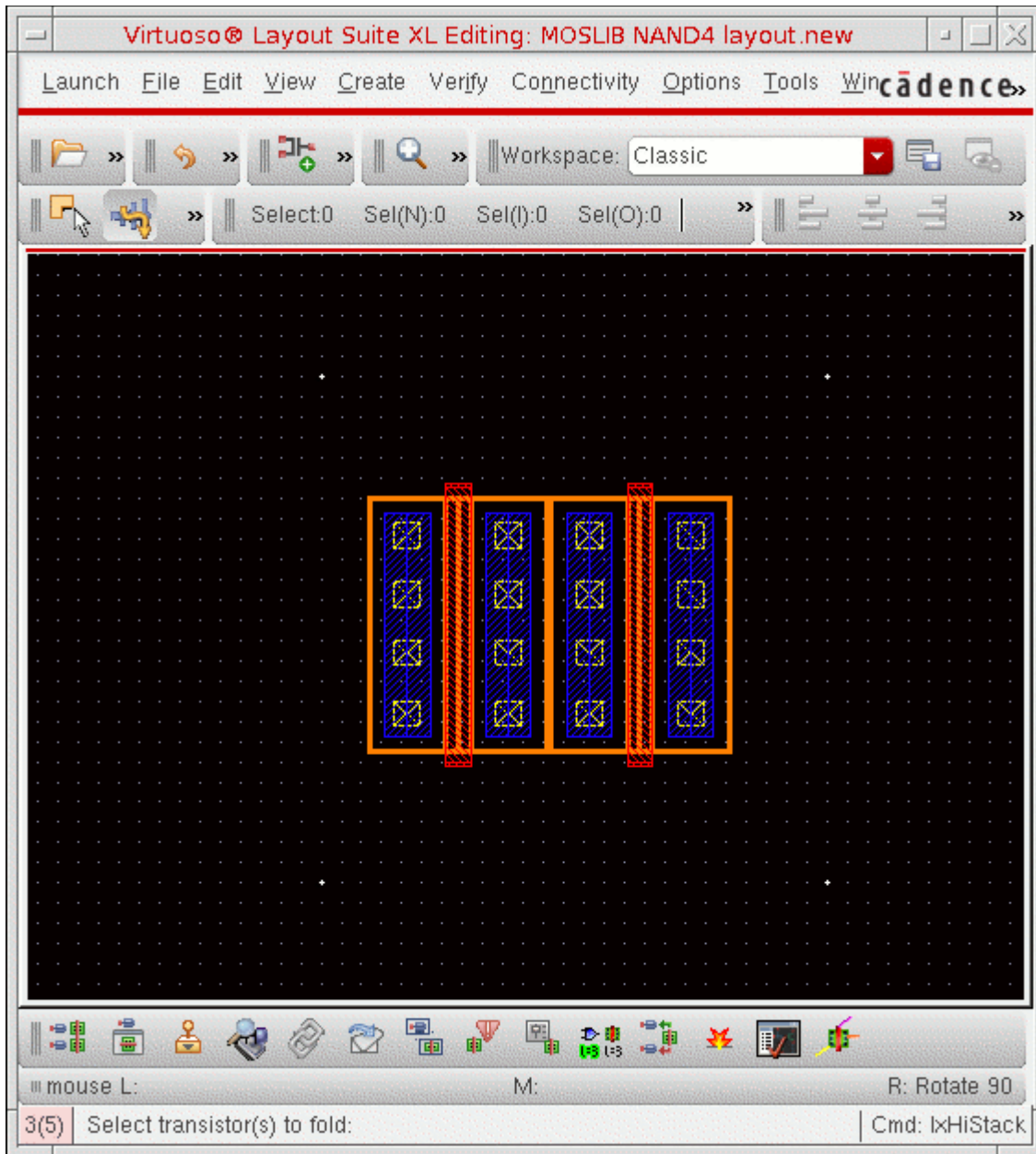
The outline of the folded device follows the cursor in the canvas.

You can click the middle mouse button to rotate the device by R90, or Shift-click the middle mouse button to flip the device alternately about its X and Y axes. (You must have `lxBindKeys.il` loaded in order to use this functionality.)

Virtuoso Layout Suite XL User Guide

Editing the Layout

5. Click in the layout where you want to place the folded transistor.



The folded transistor is placed in the canvas. The original device orientation is preserved. The net on the left of the device (or on the bottom for vertically-oriented devices) is the same before and after folding.

6. Select the transistor again in the canvas.

The form updates to show the individual folds and the widths specified for them.

Virtuoso Layout Suite XL User Guide

Editing the Layout

7. To automatically chain the newly created folds, select *Chain Folds*.

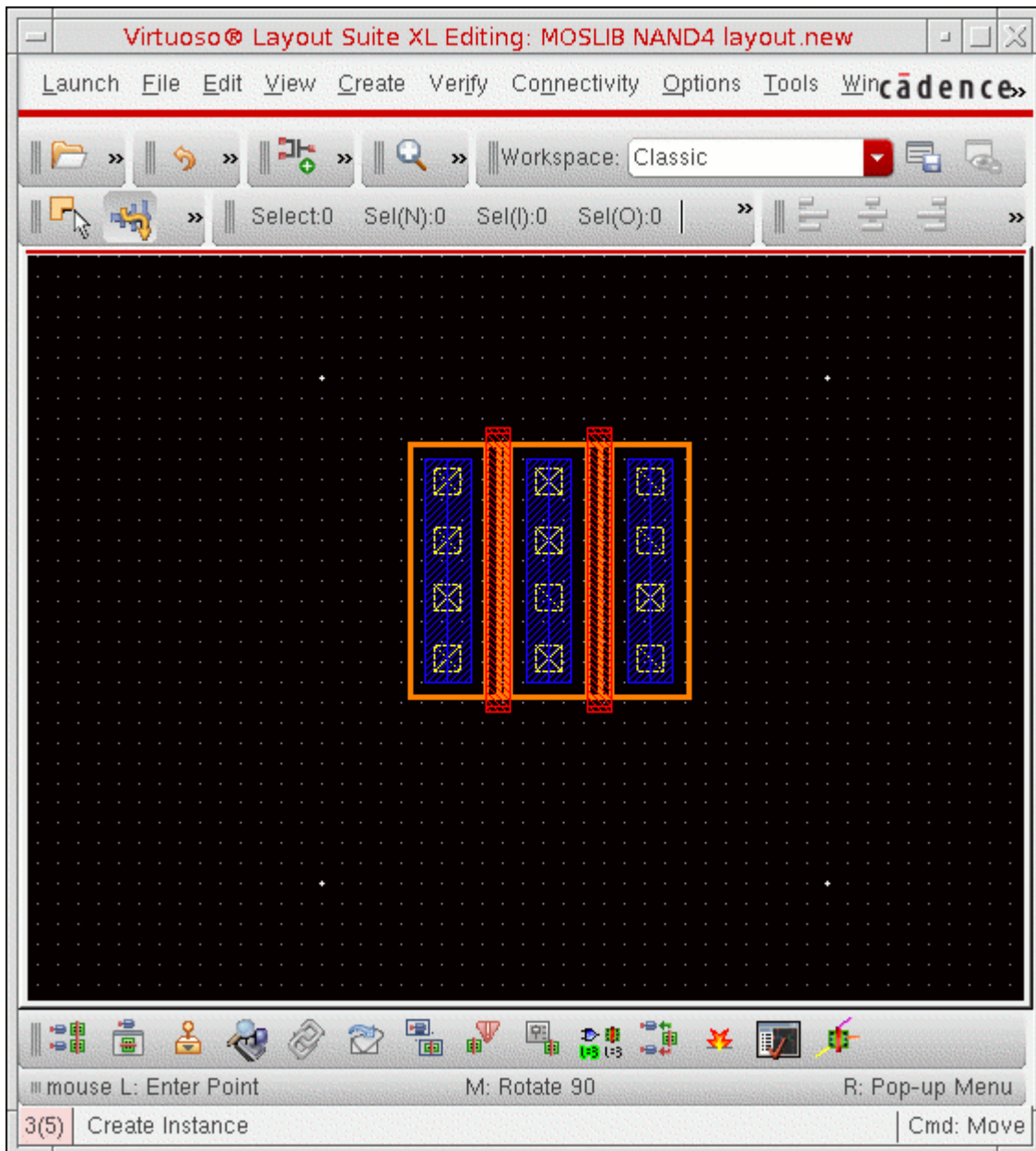
The screenshot shows the 'Generate Folded Devices' dialog box. The 'Chain Folds' checkbox is checked, and an arrow points to it with the text 'Automatically chain the new folds'. The dialog also includes fields for 'Transistor Name' (JP0), 'Transistor Width' (5u), 'Number of Folds' (2), and 'Ignore mfactor' (unchecked). There are buttons for 'Add Fold', 'Same Width', 'Delete', 'Hide', 'Cancel', 'Apply', 'Previous', 'Next', and 'Help'.

Automatically chain the new folds

Virtuoso Layout Suite XL User Guide

Editing the Layout

- Click *Apply* in the form, move the cursor into the layout canvas, and click where you want to place the folded device.

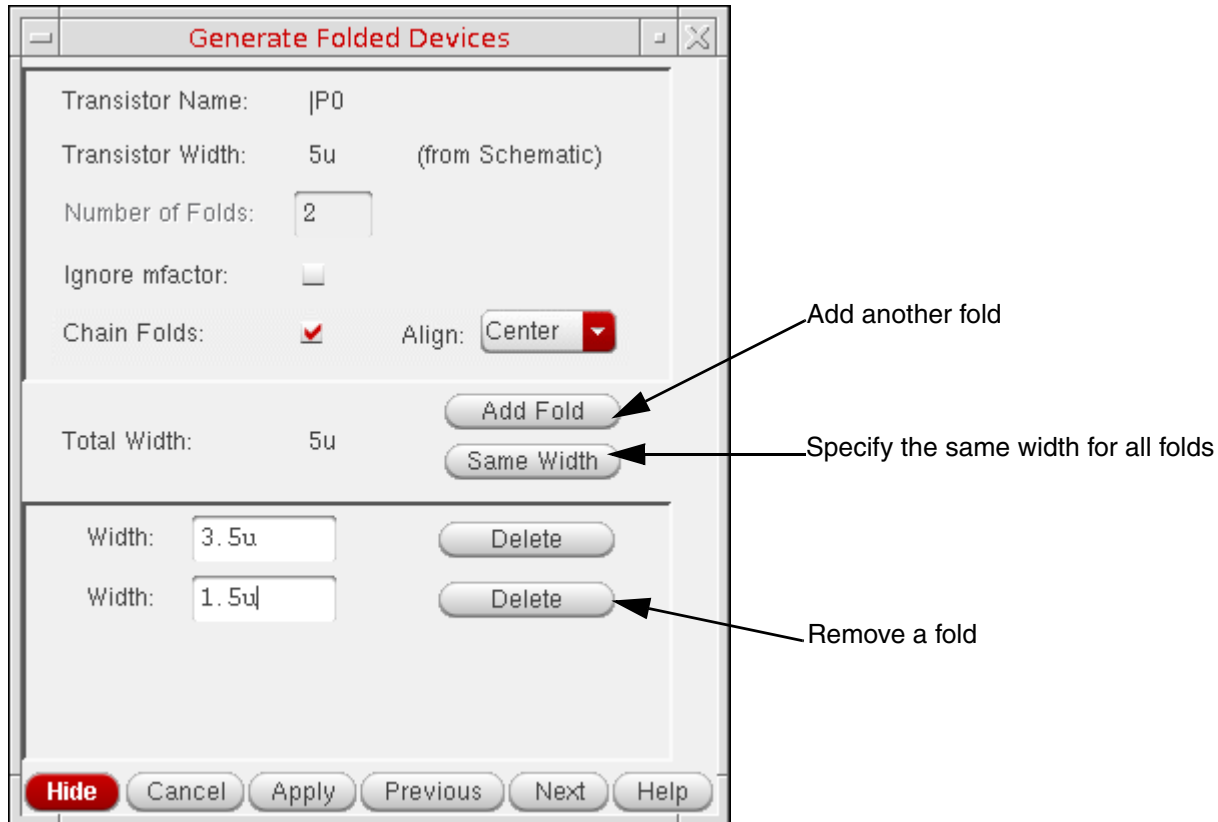


The new folds are now chained together.

Virtuoso Layout Suite XL User Guide

Editing the Layout

9. Select the transistor again in the canvas and change the widths of the individual folds in the form.



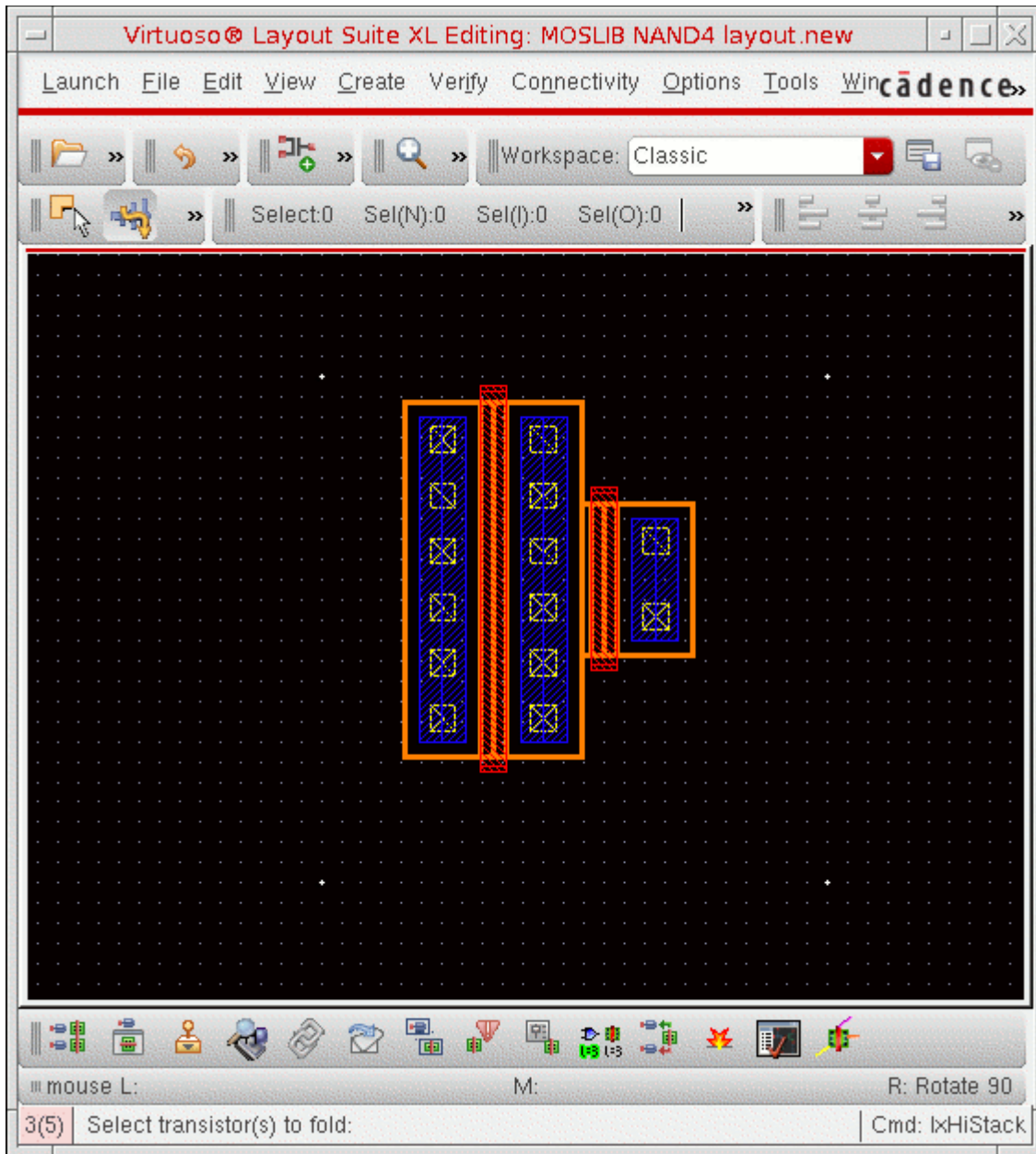
Click the *Same Width* button to specify the same width for each fold, or type individual values into each *Width* field.

The system issues a warning if the total width you specify exceeds the schematic width or the effective width. It also automatically adds the *Transistor Width* unit identifier to any width value with no unit. See [Unit Identifiers in Layout XL](#) for a complete list of these identifiers.

Virtuoso Layout Suite XL User Guide

Editing the Layout

10. Click *Apply* in the form, and then click in the canvas to place the new folded device.



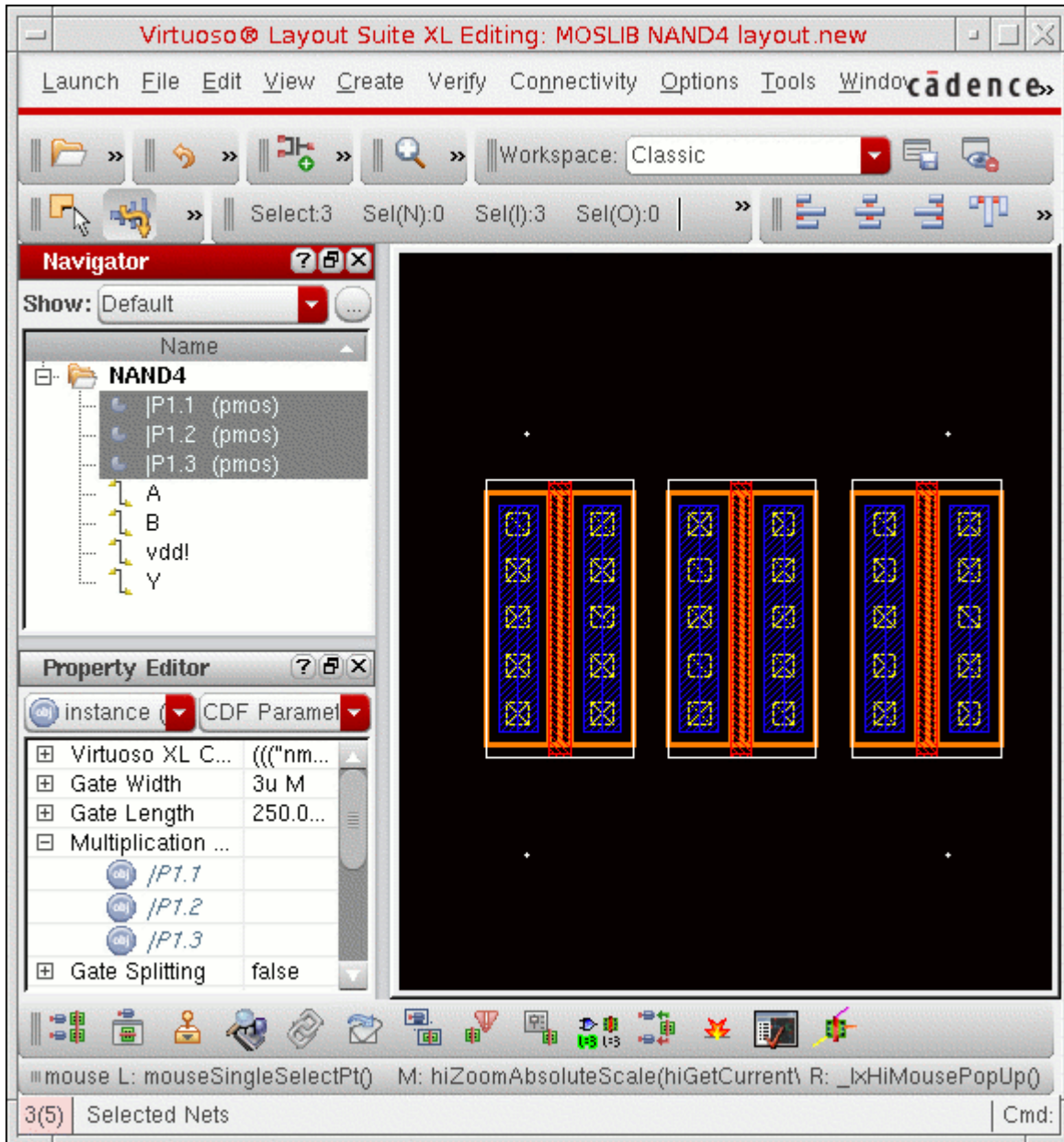
Ignoring Schematic Mfactor during Folding

By default, interactive folding is constrained by the multiplication factor property set for the instance in the schematic view. You can use the *Ignore mfactor* option in the Generate Folded Devices form to gain more flexibility when folding your device.

Virtuoso Layout Suite XL User Guide

Editing the Layout

Consider a schematic instance P1 with a multiplication factor of 3 and each part with a gate width of 3u. The device is generated in the layout view as shown below.



You want to regenerate this device as two folds each 4.5u wide. To do this,

1. Select any of the mfactor parts in the layout window.
2. Choose *Connectivity – Generate – Folded Devices*.

The Generate Folded Devices form is displayed.

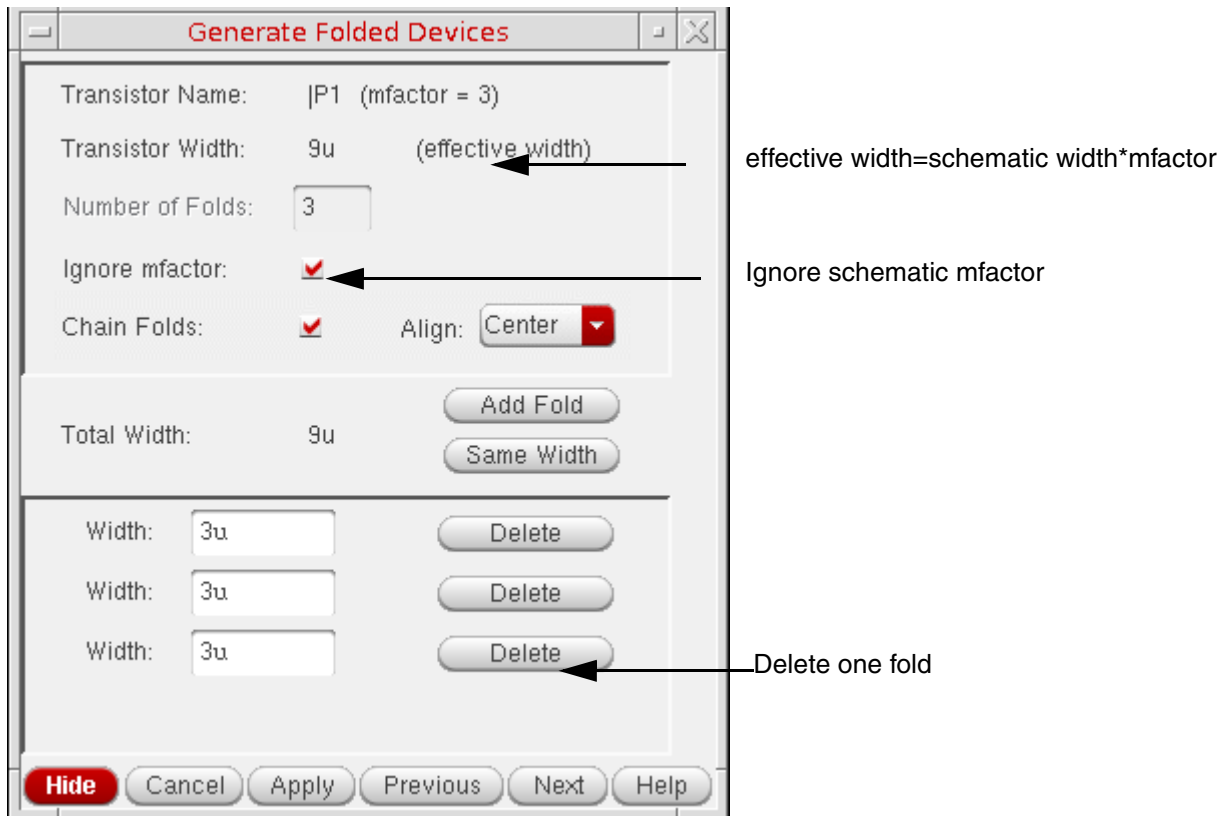
Virtuoso Layout Suite XL User Guide

Editing the Layout

3. Turn on the *Ignore mfactor* option.

The *Transistor Width* is set to the effective width in the schematic; i.e., the product of mfactor and schematic gate width.

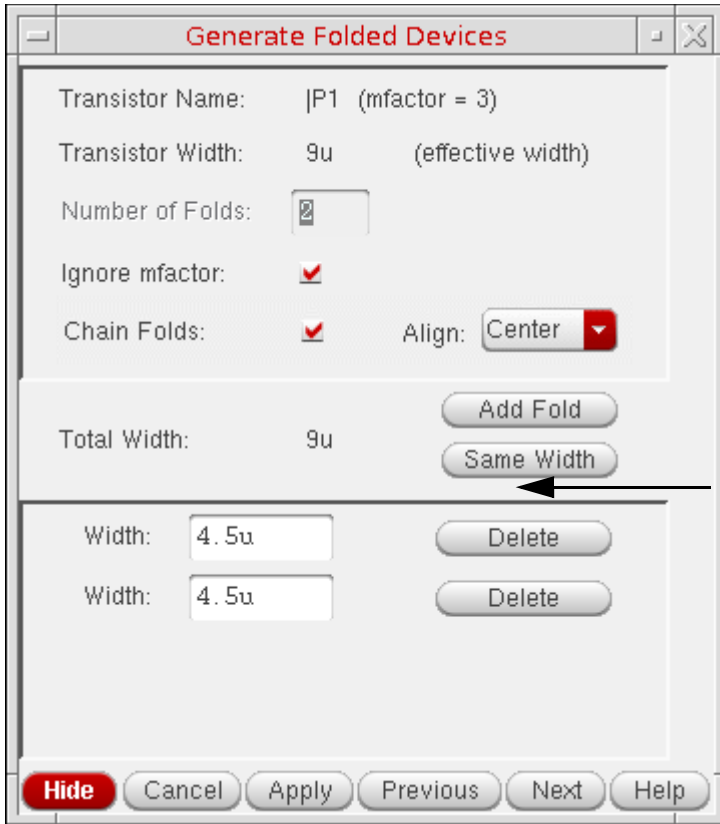
4. Delete one of the listed folds.



Virtuoso Layout Suite XL User Guide

Editing the Layout

5. Click the *Same Width* button to set the widths of the remaining folds to 4.5u each.



Generate Folded Devices

Transistor Name: |P1 (mfactor = 3)

Transistor Width: 9u (effective width)

Number of Folds: 2

Ignore mfactor: ☒

Chain Folds: ☒ Align: Center

Total Width: 9u

Add Fold

Same Width

Delete

Delete

Width: 4.5u

Width: 4.5u

Hide Cancel Apply Previous Next Help

Make each remaining the fold the same wic

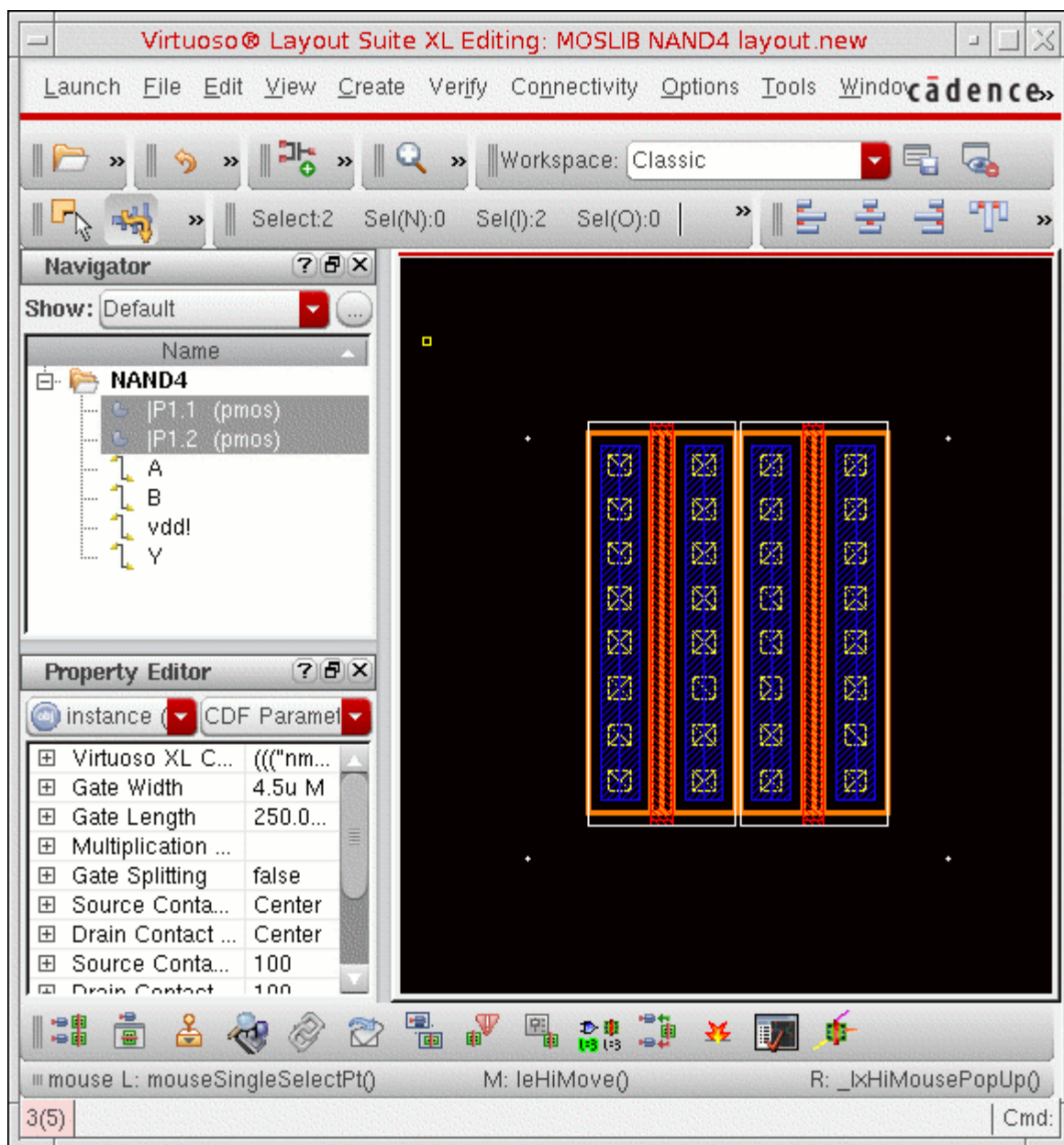
Note: You can also type values into each *Width* field. The system issues a warning if the total width you specify exceeds the schematic width or the effective width.

6. Click *Apply* in the form, and then click in the canvas to place the new folded device.

Virtuoso Layout Suite XL User Guide

Editing the Layout

The device is generated as two folds each 4.5u wide.



For information on automatically chaining the generated folds and on setting differing widths for individual folds, see [Folding a Transistor](#).

Note: *Check Against Source* will report that the device is implemented in the layout using fewer instances than are specified in the schematic.

Unit Identifiers in Layout XL

When specifying widths, use the following identifiers to specify different units.

Identifier	Denotes	Factor
y	yocto	10^{-24}
z	zepto	10^{-21}
a	atto	10^{-18}
f	femto	10^{-15}
p	pico	10^{-12}
n	nano	10^{-9}
u (the default)	micro	10^{-6}
m	milli	10^{-3}
k	kilo	10^3
K	kilo	10^3
M	mega	10^6
G	giga	10^9
T	tera	10^{12}
P	peta	10^{15}
E	exa	10^{18}
Z	zetta	10^{21}
Y	yotta	10^{24}

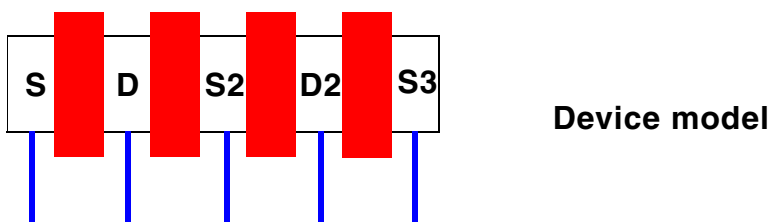
Pin Permutation

Layout XL lets you exchange the connectivity or net connections of the pins of a component. This operation is called pin permutation. The pins to be permuted must belong to different nets and must first be defined as permutable terminals. If one of the pins to be permuted is on an instance contained in any type of group (including a synchronous clone), the other pin must be on an instance in the same group.

Must Connect Handling in Pin Permutation

When a device has must-connected terminals connected to different nets at the current level, these nets, by definition, must be connected further up the hierarchy. For the purposes of pin permutation, such nets are considered equivalent.

In the device below, nets terminals *S*, *S2*, and *S3* are defined in a must-connect relationship. Terminals *D* and *D2* are in a separate must-connect relationship. All the terminals are connected to different nets at the current level.



The following permuteRule is specified for the device.

`p(S2 D)`

Any net connectivity changes made to *S2* will also be applied to all its other must-connected terminals, for example; *S*, *S3*. The same applies for *D* and *D2*. The permute model is as illustrated below.



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

Permuting Pins Manually



Any pin previously connected to a path, polygon, rectangle, or component pin cannot be permuted.

To manually permute pins within a component,

1. From the layout window menu bar, choose *Connectivity – Pins – Permute*.

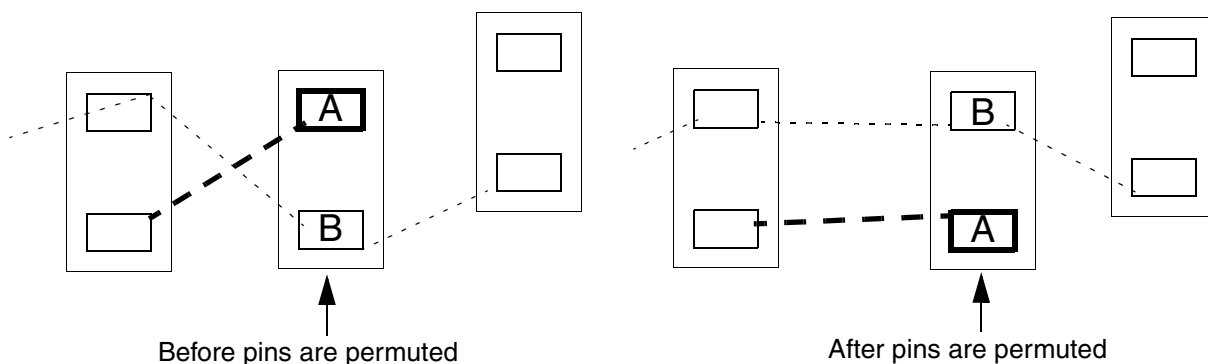
Layout XL prompts you to select a pin.

2. Click on the first pin in the layout canvas.

Layout XL prompts you to select a second pin.

3. Click on the second pin in the layout canvas.

Layout XL switches the net connections of the pins you selected. If flight lines are displayed, they change to show the new nets assigned to the pins.



Note: The *Permute Pins* command is modal, meaning that you can continue to select sets of pins to permute without re-choosing the command every time.

4. To undo the permutation, choose *Edit – Undo* from the layout window menu bar.

The pins return to their original net connection in the layout.

5. To cancel the command, press `ESC`.

Permuting Pins Automatically

When automatic pin permutation is switched on, Layout XL automatically permutes pins in a component if doing so corrects a short caused by

- Interactive routing (using automatic abutment).
- Moving or stretching a shape or using any operation that changes connectivity.

To switch on automatic pin permutation,

1. From the layout window menu bar, choose *Options – Layout XL*.

The Layout XL Forms form is displayed.

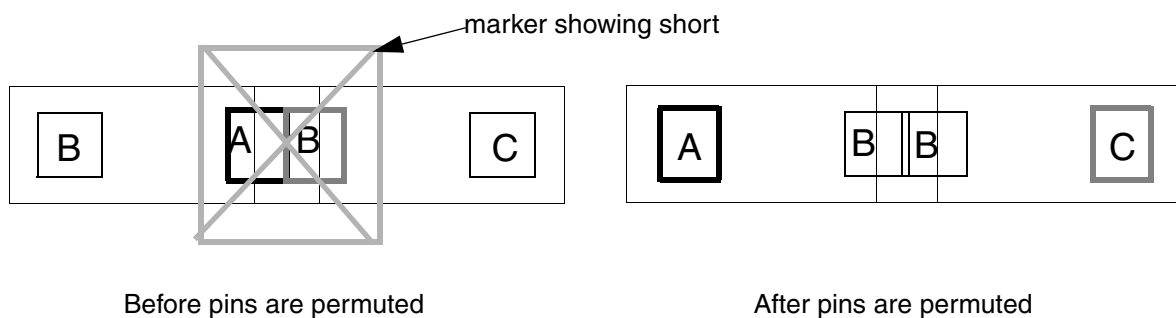
2. In the *Connectivity* section, turn on *Permute pins*.

3. Click *OK*.

Example 1

Layout XL now automatically permutes the pins of a single component when doing so removes a short.

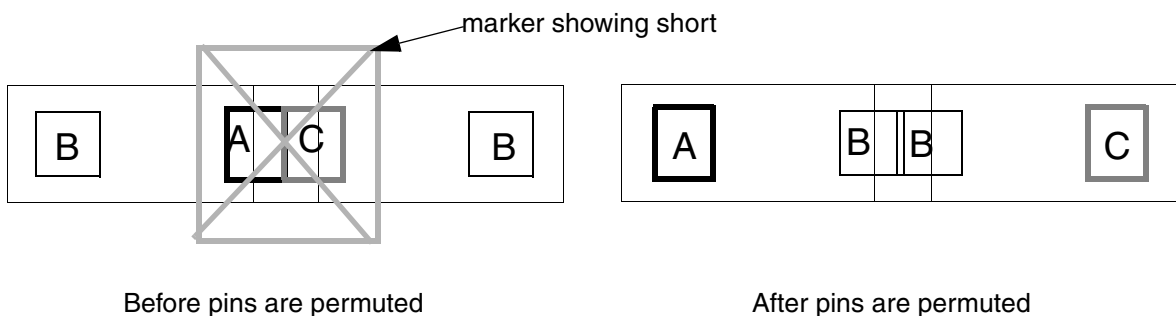
In the diagram below, two pins belonging to different nets (*A* and *B*) create a short when they are touching each other. If these pins are permutable and if reversing the pins of one component connects the two pins that are on the same net (*B*), Layout XL permutes them automatically.



Example 2

Layout XL also automatically permutes the pins of two components simultaneously when doing so removes a short.

In the diagram below, two pins belonging to different nets (*A* and *C*) create a short when they are touching each other. If these pins are permutable, and reversing the pins of both components connects the two pins that are on the same net (*B*), Layout XL permutes them.



Checking Permutation Information

To see information about permuted pins,

1. From the layout window menu bar, choose *Connectivity – Pins – Permute*.
2. Choose the pins you want to permute.
3. Press F3.

The Permutation Information window appears telling you what was done. If the pins you select do not permute, the info window tells you why.



Cadence does not recommend setting the XL Probe form's Object Filter option to "Pins" when probing pins for permuted instance. Instead set the option to "Nets" to view the correct information.

Updating Permuted Pins

Permuted pins are not preserved by the *Update Components And Nets* command. If any component in the design needs to be updated, the command resets the permutation status of all the cells to that originally stored in the schematic.

Disabling Permutation for an Instance

You can use the disablePermutation constraint to prevent permutation on a selected instance without having write access to the schematic design. The disablePermutation constraint overrides any permuteRule set for the instance in question and applies throughout the flow.

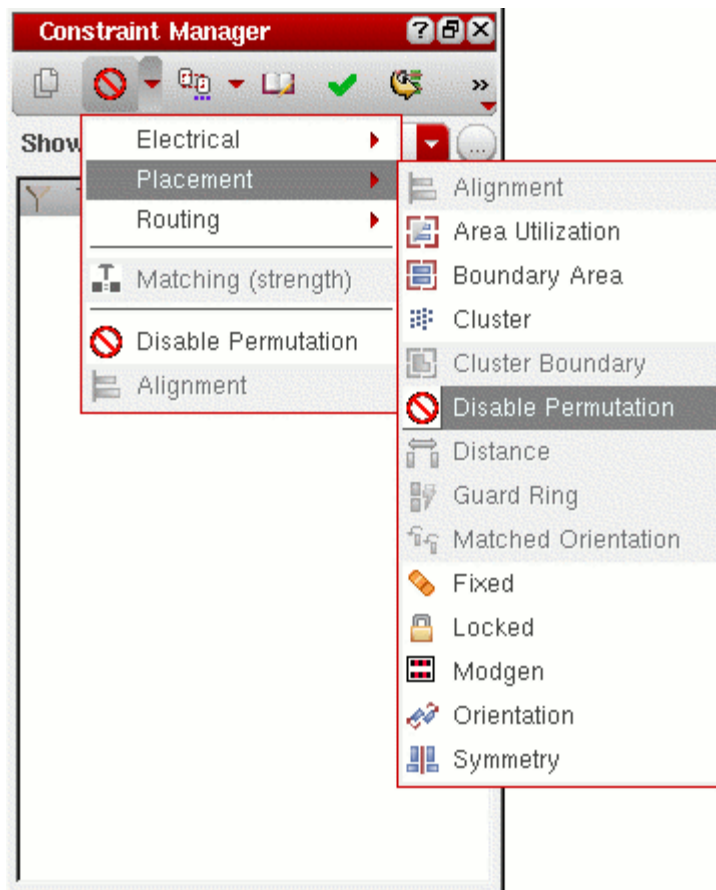
To create a disablePermutation constraint for an instance,

1. Select the instance in question in the layout view.

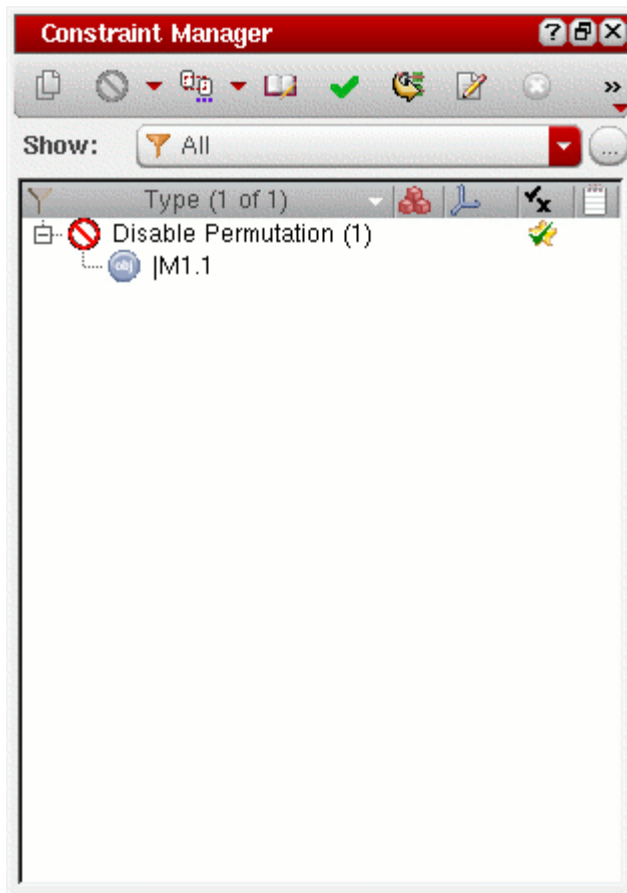
Virtuoso Layout Suite XL User Guide

Editing the Layout

2. In the Constraint Manager assistant, use the pull-down to choose *Disable Permutation*.



The Constraint Manager is updated to show the new constraint on the selected instance.



Working with Incomplete Nets

Layout XL can show you incomplete electrical connections between the component terminals, pins, and shapes associated with each net in your design. These incomplete connections include unrouted instance terminals either with no pins or with pins that are not extractable, and instance terminals with dangling must-join instance pins.

For display purposes the system defines an incomplete net as a net which has at least one *open marker* attached to it in the design database. Each open marker can be drawn as a flight line in the design canvas. Each incomplete net can therefore be represented by one or more flight lines in the design canvas.

To see incomplete net flight lines in the design canvas you must

- Extract the connectivity of your design to generate open markers

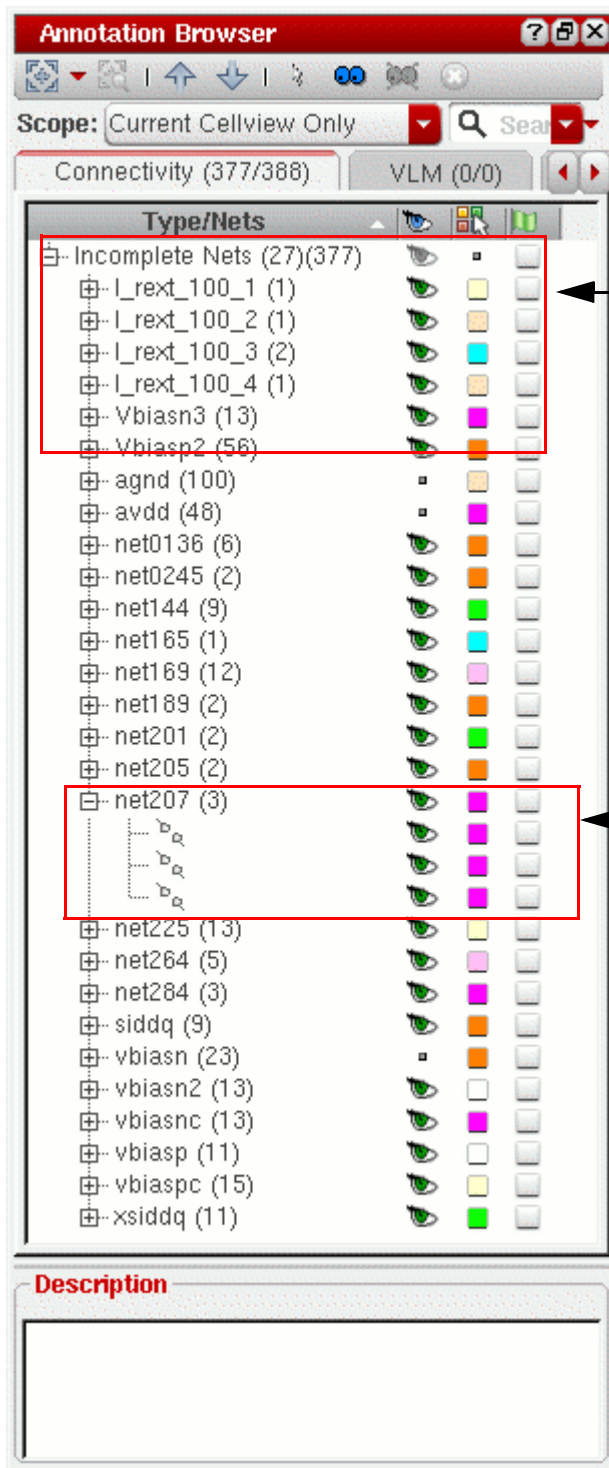
The connectivity extractor runs automatically during the *Generate All From Source*, *Generate Selected From Source*, and *Update Components And Nets* commands, and when you use the automatic placement and routing tools. You can also extract your design manually using the *Extract Layout* menu command or toolbar button. For more information, see [Connectivity Extraction](#).

- Initialize the Annotation Browser assistant

Virtuoso Layout Suite XL User Guide

Editing the Layout

To be able to show flight lines for incomplete net markers, the [Annotation Browser](#) must be initialized. You use the Annotation Browser to view the markers in your design and specify how they are displayed in the canvas.



Incomplete nets are listed in the *Incomplete Nets* category in the Annotation Browser's *Connectivity* tab. The first number in parentheses represents the number of incomplete nets; the second the total number of markers.

Expand an entry to see the open markers attached to the net. You can specify whether a marker is highlighted or not and which color is used to draw it in the canvas.

 Important

Although it must be initialized, the Annotation Browser assistant itself need not be displayed on the desktop in order to show and hide incomplete nets. For example, you can close it and show and hide incomplete nets in postselection mode directly from the Navigator assistant. To do this, though, you must switch off the *Remove highlighting when browser is closed* option in the Annotation Browser Options form, otherwise there can be no visible incomplete nets to show and hide. For more information, see Showing Incomplete Nets in Postselection Mode.

If you select one or more incomplete net markers in the Annotation Browser assistant, you can click the right mouse button and choose *Select Nets* to cross-select the nets in the layout canvas and Navigator assistant. You can then invoke the router on the selected incomplete nets.

When you edit an instance in place, any flight lines highlighted at the higher level of hierarchy remain highlighted while you edit in place. For more information on how to toggle the visibility of incomplete nets during interactive editing, how to show and hide all the incomplete nets associated with specific objects in your design, and how to filter the incomplete net markers shown in the browser, see

- Showing and Hiding All Incomplete Nets
- Showing and Hiding Selected Incomplete Nets
- Showing and Hiding Current Incomplete Nets
- Searching for Specific Incomplete Nets
- Automatically Showing Newly-Created Incomplete Nets
- Setting an Incomplete Net Filter

Showing and Hiding All Incomplete Nets

Use the *Show/Hide All Incomplete Nets* command to show and hide all the incomplete nets in your design. You can access this command using:

- The *Connectivity – Nets – Show/Hide All Incomplete Nets* command from the layout window menu.
- The `Shift+Ctrl+h` bindkey on your keyboard.
- The `lxToggleShowAllIncompleteNets()` SKILL function.

Virtuoso Layout Suite XL User Guide

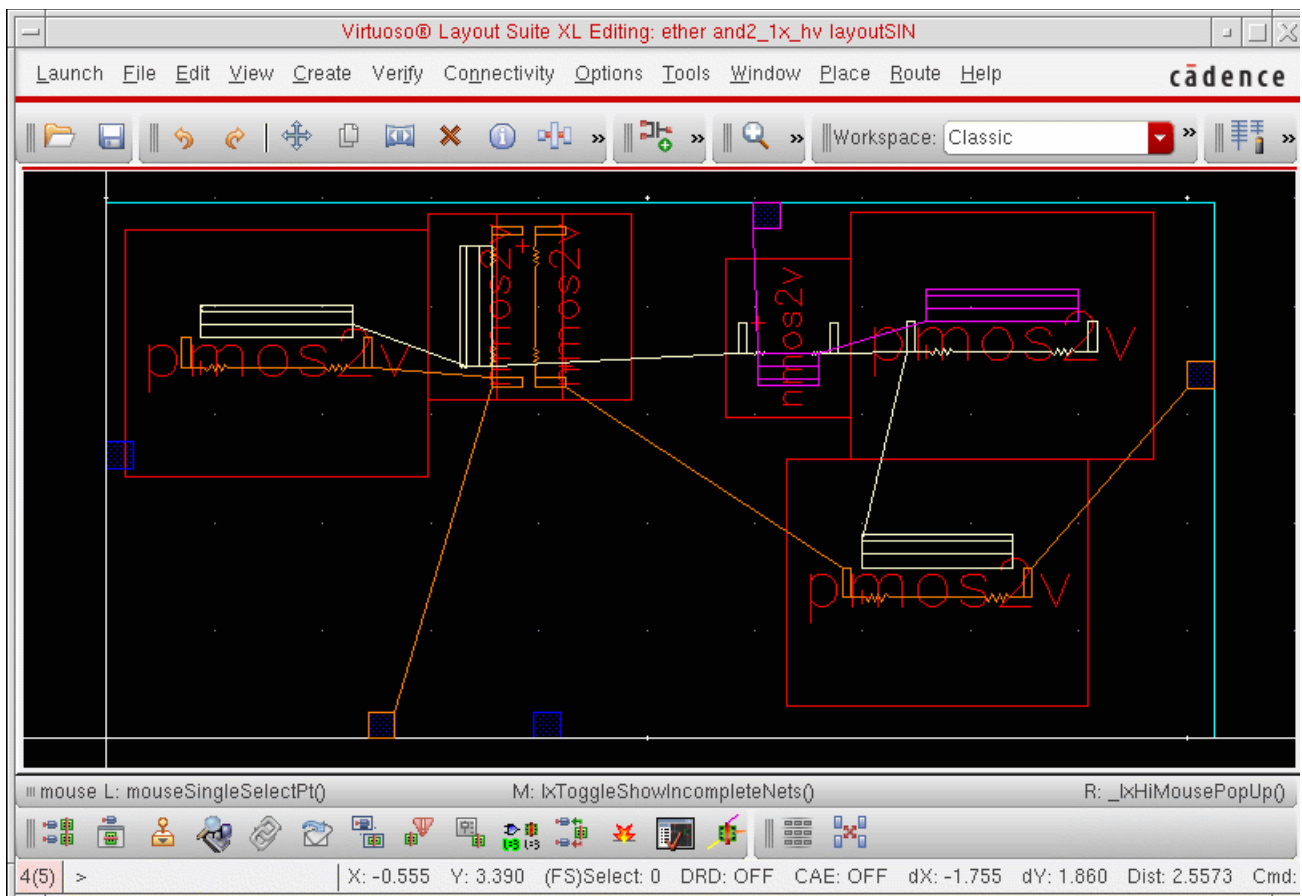
Editing the Layout

Note: If you run the *Show/Hide All Incomplete Nets* command with the Annotation Browser closed, the command fails. You must open the Annotation Browser before you run the command.

Showing and Hiding Current Incomplete Nets

Use the *Show/Hide Current Incomplete Nets* command to change the visibility of the currently visible incomplete nets in the design canvas. You can also access this command by pressing **h** on your keyboard, or by typing `lxToggleShowIncompleteNets()` in the CIW.

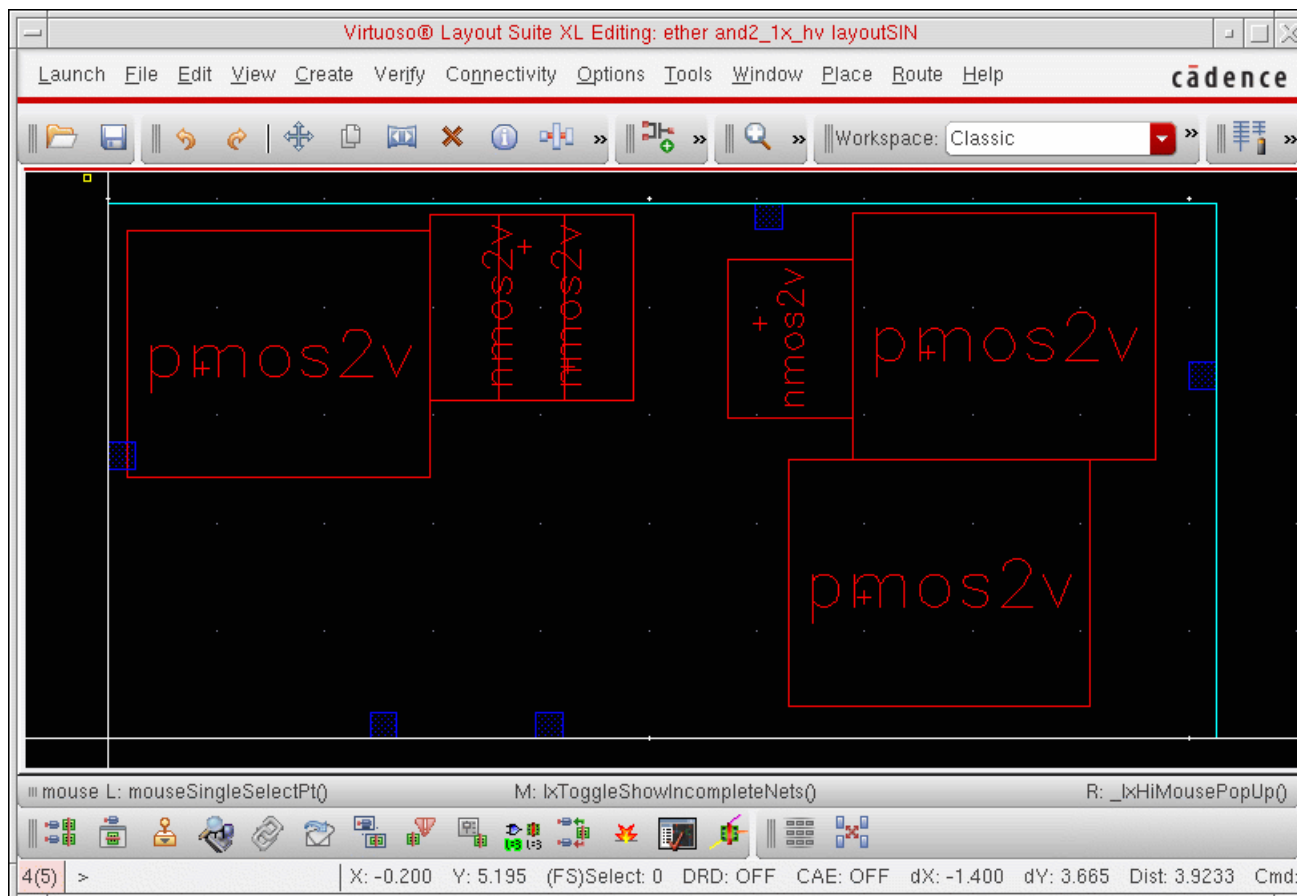
1. Place your cursor in a layout window showing **at least one incomplete net**.



Virtuoso Layout Suite XL User Guide

Editing the Layout

2. Choose *Connectivity – Nets – Show/Hide Current Incomplete Nets* or press **h** on your keyboard.



The function hides all the incomplete nets that were visible or partially visible in the canvas. The list of nets that were hidden is stored in the system.

3. To make the stored nets visible, choose *Connectivity – Nets – Show/Hide Current Incomplete Nets* (or press **h**) again.

If you run the function for the first time on a design with no incomplete nets on display, it does nothing because there is no stored list of nets to display.

The function does nothing unless the Annotation Browser has been initialized. The assistant itself need not be displayed on the desktop, however. If you do close the browser, make sure you switch off the *Remove highlighting when browser is closed* option in the Annotation Browser Options form, otherwise there can be no visible incomplete nets to hide.

When an incomplete net is only partially visible (some of the open markers on the net are visible, others are not) the function considers the incomplete net as fully visible. The first call

to the function hides the open markers that are currently visible; the second shows all the open markers associated with the incomplete net.

If you hide all the visible open markers in the canvas, then use the Annotation Browser controls to make an open marker visible, calling *Show/Hide Current Incomplete Nets* again does not restore the visibility of the previously hidden markers. The function **always** hides any visible open markers.

If you delete a net, any open markers associated with that net are removed from the stored list and cannot be restored when you run the command again.

Showing and Hiding Selected Incomplete Nets

Use the *Show/Hide Selected Incomplete Nets* command to show and hide the incomplete nets associated with selected objects in your design. You can access this command using

- The *Show/Hide Selected Incomplete Nets* button in the Layout XL toolbar. (This is the method used in this document.)
- The *Connectivity – Nets – Show/Hide Selected Incomplete Nets* command from the layout window menu.
- The `Shift+h` bindkey on your keyboard.
- The `lxShowHideIncompleteNets()` SKILL function.

When you start the command, the Annotation Browser is automatically initialized if required.

Note: The command does not support blockage, boundary, or row objects. These objects are silently ignored unless a preselected set contains only unsupported objects, in which case you see a message informing you of this fact.

The command behaves differently depending on whether you launch it with objects already selected or not. For details, see

- [Showing Incomplete Nets in Preselection Mode](#)
- [Showing Incomplete Nets in Postselection Mode](#)

Showing Incomplete Nets in Preselection Mode

To show and hide incomplete nets for preselected objects,

1. In the layout canvas, select the objects for which you want to show or hide incomplete nets.

Virtuoso Layout Suite XL User Guide

Editing the Layout

The system gets all the incomplete nets associated with the selected objects. The system considers a partially-selected object to be fully selected.

2. In the Layout XL toolbar, click the *Show/Hide Selected Incomplete Nets* button.



What happens next depends on the current visibility of the open markers associated with the selected incomplete nets.

- ☐ If all the open markers are visible in the design canvas, the function hides them all.
- ☐ If at least one open marker on one of the incomplete nets is hidden, the function makes them all visible.

For detailed information on how the function behaves for different object types, see [Behavior by Object Type in Preselection Mode](#).

3. Click the *Show/Hide Selected Incomplete Nets* button again to toggle the visibility of the incomplete nets.

Behavior by Object Type in Preselection Mode

The table below summarizes the behavior of the *Show/Hide Selected Incomplete Nets* command for different preselected objects.

Object Type	Behavior
<i>Net</i>	If the net is complete, the function does nothing; otherwise it toggles the visibility of the incomplete net as follows. ■ If all the open markers on the incomplete net are visible, the function hides them all. ■ If at least one open marker on the net is hidden, the function makes all the open markers on the net visible.

Virtuoso Layout Suite XL User Guide

Editing the Layout

Object Type	Behavior
<i>Shape or Via</i>	If the shape or via has no net associated with it or if the associated net is complete, the function does nothing; otherwise it toggles the visibility of the incomplete net as follows. ■ If all the open markers on the incomplete net are visible, the function hides them all. ■ If at least one open marker on the net is hidden, the function makes all the open markers on the net visible.
<i>Instance</i>	If the instance has no net connected to it, the function does nothing; otherwise it gets all the incomplete nets associated with the instance and toggles the visibility of these nets as follows. ■ If all the open markers on all the incomplete nets are visible, the function hides them all. ■ If at least one open marker on one net is hidden, the function makes all the open markers on all the nets visible.
<i>Group</i>	The function recursively gets all the incomplete nets associated with all the objects inside the group and toggles the visibility of these nets as follows. ■ If all the open markers on all the incomplete nets are visible, the function hides them all. ■ If at least one open marker on one net is hidden, the function makes all the open markers on all the nets visible. Note: An object inside a group is considered selectable even if it is set as not visible and not selectable in the LSW.
<i>Marker</i>	A marker object cannot have any nets associated with it. The function gets all the incomplete nets attached to all the objects associated with the marker and then toggles the visibility of these nets as follows. ■ If all the open markers on all the incomplete nets are visible, the function hides them all. ■ If at least one open marker on one net is hidden, the function makes all the open markers on all the nets visible.

Showing Incomplete Nets in Postselection Mode

You can also start the *Show/Hide Selected Incomplete Nets* command with nothing preselected in the canvas. When you then click on an object, the function shows the incomplete nets associated with that object. To do this,

1. With nothing selected in the layout canvas, click the *Show/Hide Selected Incomplete Nets* button in the Layout XL toolbar.



You are prompted to click in the canvas to show the incomplete nets for the selected object.

2. Click on an object in the canvas.

The function shows all the open markers for the incomplete nets associated with the object you selected.

In this mode, if you can see an object in the canvas, you can show the incomplete nets for that object regardless of whether it is set to be selectable in the LSW.

For detailed information on how the function behaves for different object types, see [Behavior by Object Type in Postselection Mode](#).

3. Use **Shift**-click to add another object to the selected set.

The function adds the incomplete nets associated with the object you just selected to the existing set of incomplete nets on display.

4. Use **Ctrl**-click to remove an object from the selected set.

The function hides the incomplete nets associated with the object you clicked on.

Note: You can achieve the same effect by clicking a second time immediately after you have clicked to show the nets in step 2 above.

5. Press **h** on your keyboard to hide all the incomplete nets currently displayed in the design canvas.

You can restore the visibility of those incomplete nets by pressing **h** again. For more information, see [Showing and Hiding Current Incomplete Nets](#).

6. To end the *Show/Hide Selected Incomplete Nets* function, do one of the following.

- ☐ Press the **Esc** key on your keyboard
- ☐ Press **Ctrl-c** on your keyboard

- ☐ Run another non-nestable function

When you leave the function, the incomplete nets are deselected but remain highlighted in the canvas.

For more information, see

- [Showing Incomplete Nets for Overlapping Objects](#)
- [Showing Incomplete Nets Using Area Selection](#)
- [Showing Incomplete Nets from the Navigator and Search Assistants](#)

Showing Incomplete Nets for Overlapping Objects

When you click on an area of the canvas where several objects overlap, the function gets the list of objects under the cursor that have incomplete nets associated with them, hides any visible incomplete nets that are directly under the cursor, and shows the incomplete nets for the first object in the list. When you click again, the function hides the incomplete nets for the first object and shows the incomplete nets for the second object, and so on. To see all the incomplete nets associated with a group of overlapping objects, press **Shift** and click the left mouse button repeatedly until all the objects under the cursor are selected.



The function hides only those incomplete nets that are directly under the cursor when you click. This lets you show incomplete nets in a different area of the design without hiding the incomplete nets currently displayed elsewhere.

Showing Incomplete Nets Using Area Selection

You can use area selection to select the objects for which you want to see incomplete nets.

Only objects that are fully enclosed by the selected area are considered. For example, if an area selection encloses an instance pin but not the instance itself, the function shows (or hides) only those incomplete nets associated with the corresponding instance terminal and excludes any other incomplete nets associated with the instance.

The same applies for objects inside a group. If the selected area fully encloses an object inside a group but not the group itself, the function shows (or hides) only the incomplete nets associated with the fully-enclosed object.

Showing Incomplete Nets from the Navigator and Search Assistants

The *Show/Hide Selected Incomplete Nets* function works from both the Navigator and Search assistants. When you select an object in either assistant, you see the incomplete nets associated with the selected object. When you deselect the object, the incomplete nets are hidden again. To hide all the incomplete nets, click on an empty area in the Navigator assistant.

Note: If you select one or more incomplete net markers in the Annotation Browser assistant, you can click the right mouse button and choose *Select Nets* to select the nets in the layout canvas and Navigator assistant. You can then invoke the router on the selected nets.

Behavior by Object Type in Postselection Mode

The table below summarizes the behavior of the *Show/Hide Selected Incomplete Nets* command for different postselected objects.

Object Type	Behavior
<i>Net</i>	If the net is complete, the function does nothing; otherwise it toggles the visibility of the incomplete net as follows. ■ If all the open markers on the incomplete net are visible, the function hides them all. ■ If at least one open marker on the net is hidden, the function makes all the open markers on the net visible.
<i>Shape or Via</i>	If the shape or via has no net associated with it or if the associated net is complete, the function does nothing; otherwise it toggles the visibility of the incomplete net as follows. ■ If all the open markers on the incomplete net are visible, the function hides them all. ■ If at least one open marker on the net is hidden, the function makes all the open markers on the net visible.

Virtuoso Layout Suite XL User Guide

Editing the Layout

Object Type

Behavior

Instance

If you click on a visible instance pin (with the *Display Stop Level* set to a value greater than 0 or the *Instance Pins* button turned on in the Display Options form), the function works on the top net connected to the corresponding instance terminal.

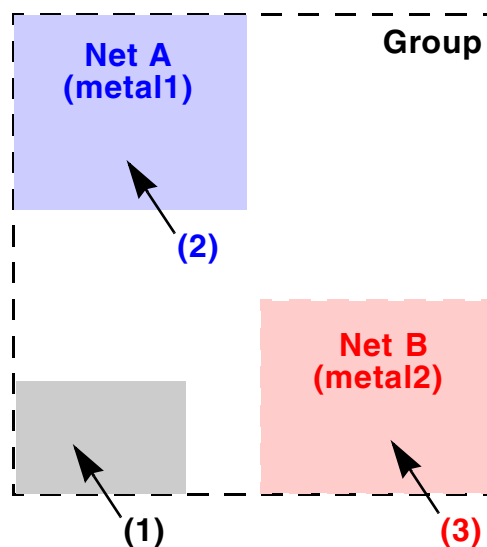
If there is no visible instance pin, the function works on all the top nets associated with the instance.

When the function grabs a net associated with an instance pin, if the net is complete, the function does nothing. It does not look at the other nets associated with the instance.

Group

When the group visibility is turned on in the LSW, if you click on a group at a location where there is no object, the function shows (or hides) all the incomplete nets attached to all the objects inside the group **(1)**.

When you click on a specific object inside a group, the function shows (or hides) only the nets that are attached to the selected object and **(2)** and **(3)**. This lets you work on a specific object contained in a group without descending into the group to select it.



Group visibility on in LSW

Click **(1)** shows net A and net B

Click **(2)** shows net A only

Click **(3)** shows net B only

Group visibility off in LSW

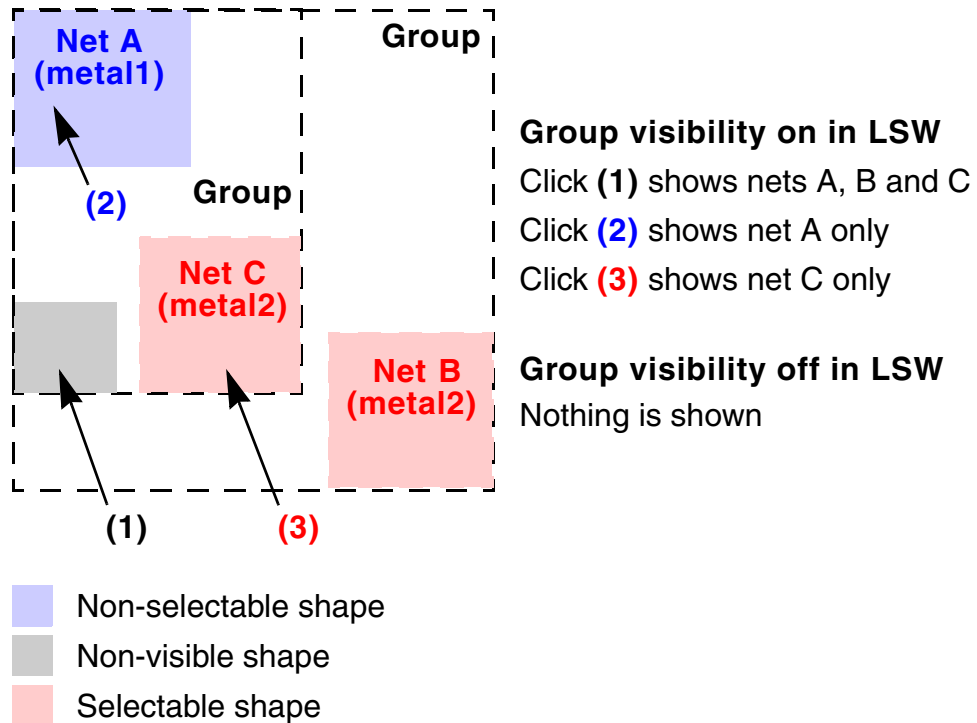
Nothing is shown

- Non-selectable shape
- Non-visible shape
- Selectable shape

Virtuoso Layout Suite XL User Guide

Editing the Layout

Object Type	Behavior
<i>Hierarchical Group</i>	The function grabs objects regardless of the level they are at in the group hierarchy. All the objects visible in a group are considered to be at the top level.



<i>Marker</i>	A marker object cannot have any nets associated with it. The function gets all the incomplete nets attached to all the objects associated with the marker and then toggles the visibility of these nets as follows. ■ If all the open markers on all the incomplete nets are visible, the function hides them all. ■ If at least one open marker on one net is hidden, the function makes all the open markers on all the nets visible.

Searching for Specific Incomplete Nets

You can use the Annotation Browser's *Search* function to locate all the incomplete net markers associated with a particular instance or net in the design. When you type in the name of a net, the browser updates automatically to show only those incomplete nets that feature

Virtuoso Layout Suite XL User Guide

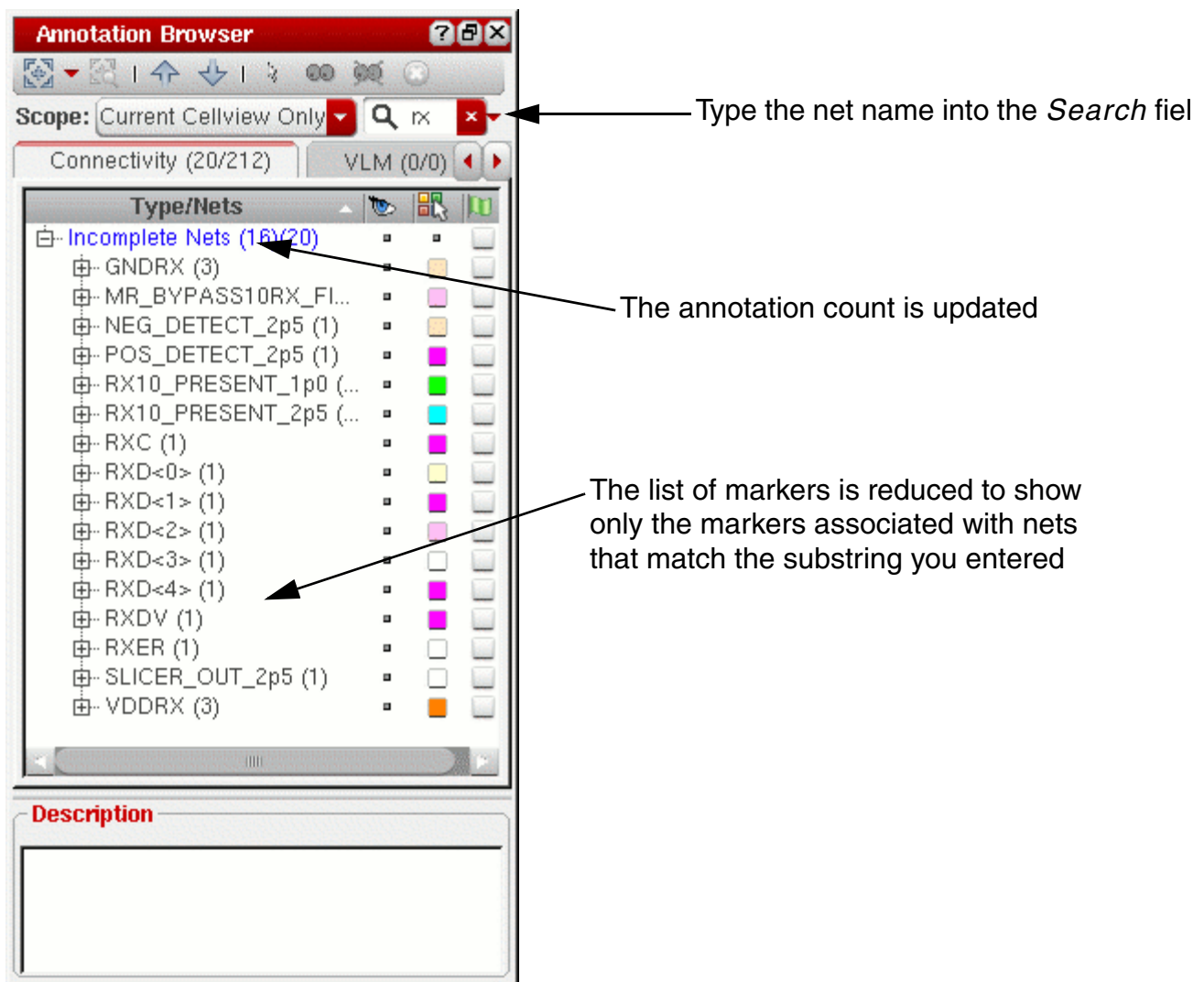
Editing the Layout

that name either in the labels in the browser pane or in the messages shown in the *Description* field.

To search for all the incomplete net markers associated with a particular net,

1. Type a string into the *Search* field; for example, `rx`.

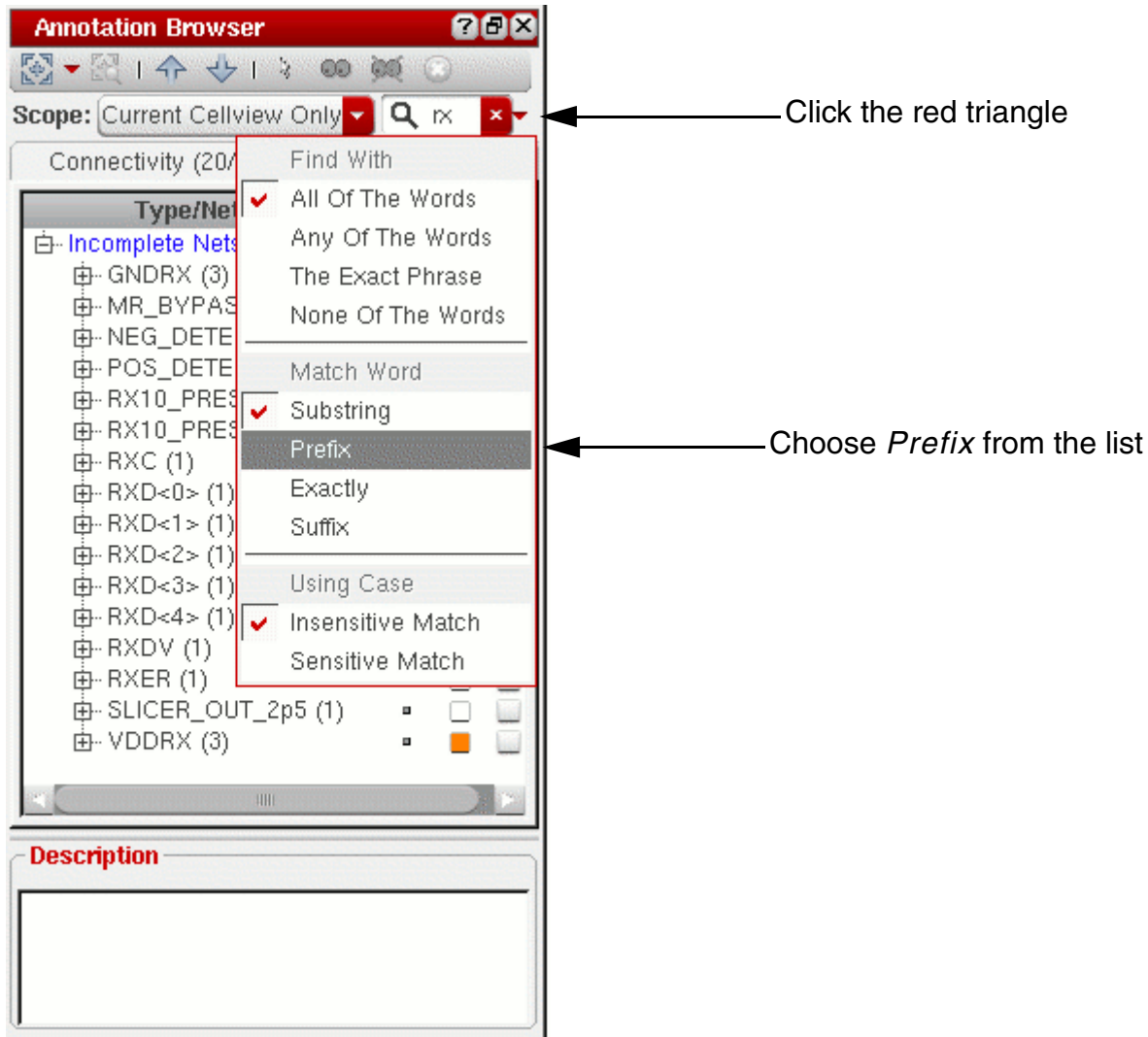
Note: By default the search function uses the text you enter as a case-insensitive substring. For details of the different search options, see the toolbar section in [Annotation Browser Graphical User Interface](#).



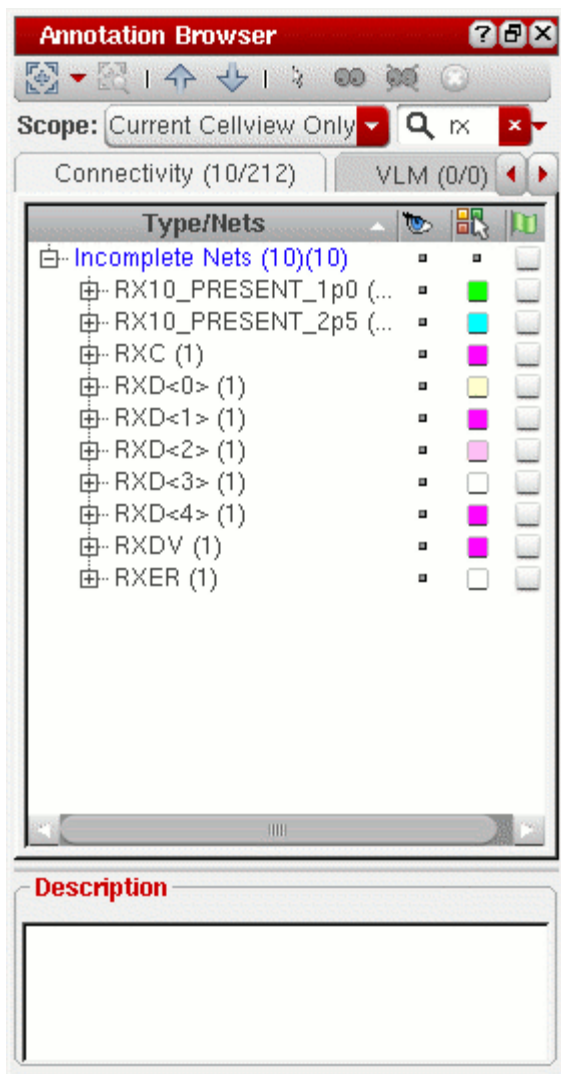
Virtuoso Layout Suite XL User Guide

Editing the Layout

2. To search for a prefix, click the small red triangle next to the *Search* field and choose *Prefix* from the drop-down list.



The list of markers and the annotation count are both updated accordingly.



3. To search a for suffix, click the small red triangle and choose *Suffix*.



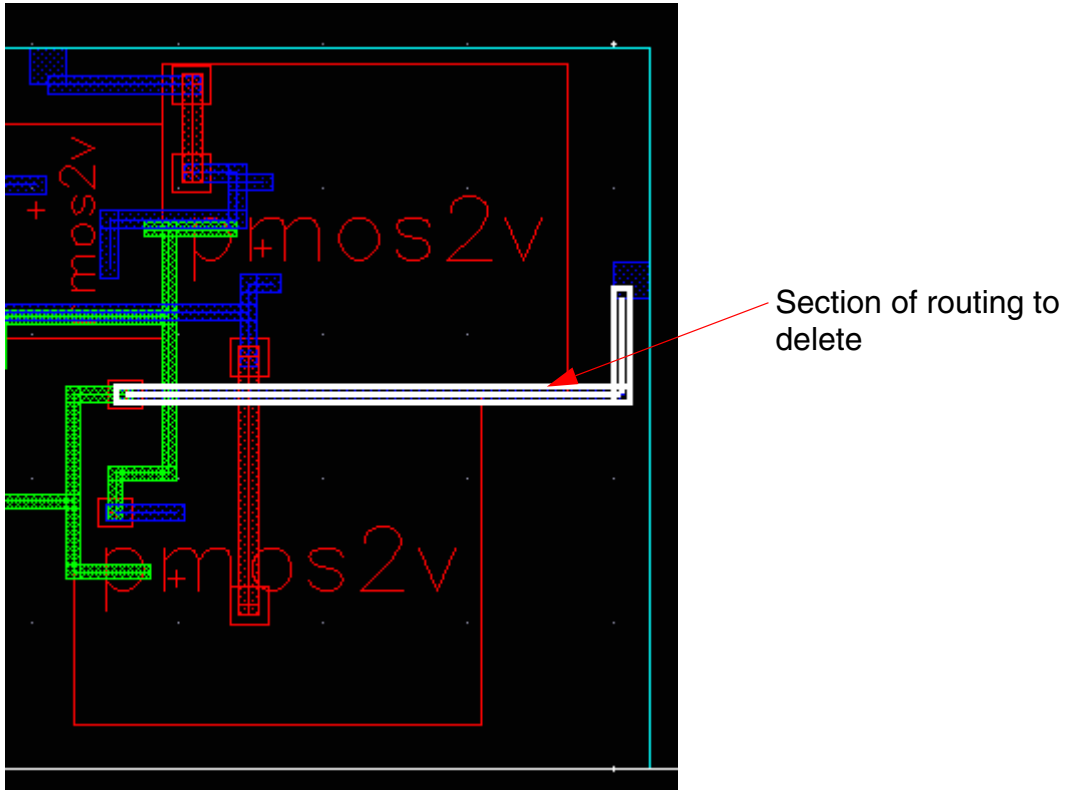
Tip

You can also filter the incomplete net markers shown in the browser based on the set of currently selected nets in the layout canvas or Navigator assistant. For more information, see [Setting an Incomplete Net Filter](#).

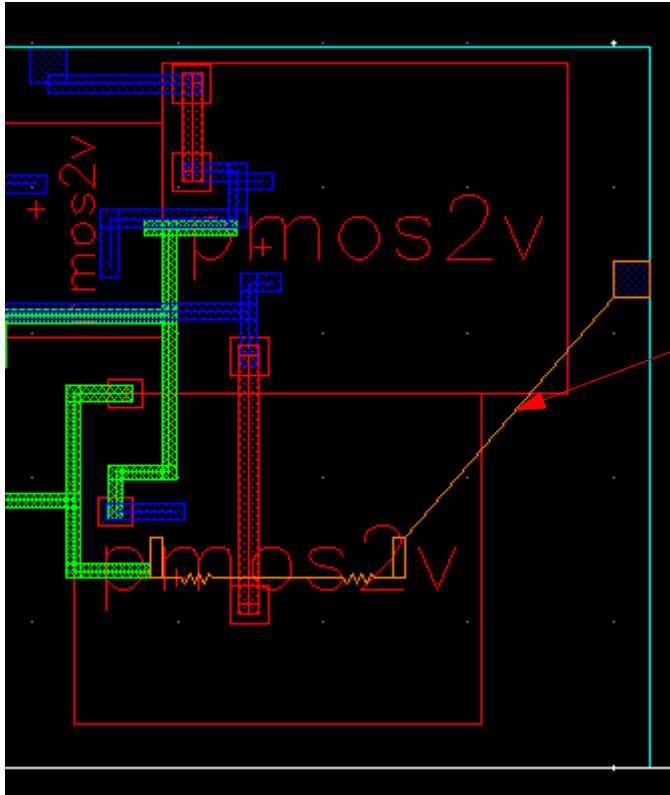
Automatically Showing Newly-Created Incomplete Nets

Layout XL can automatically show flight lines representing new incomplete nets you create during interactive editing. To do this,

1. In a layout window, delete a piece of routing material on a complete net.

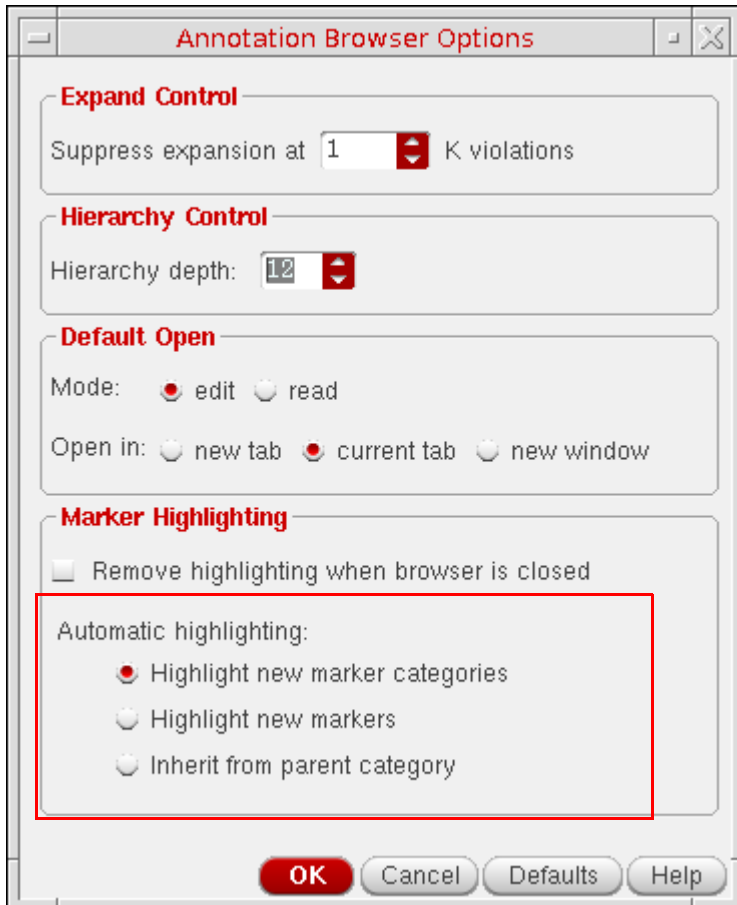


The system automatically shows a flight line representing the new incomplete net you created.



Flight line showing the
new incomplete net

You control this behavior using the *Automatic Highlighting* options in the Annotation Browser Options form.



By default, the system highlights markers only when a new marker category is created, for example a new incomplete net.

To automatically highlight a new open marker created on an existing incomplete net, set automatic highlighting to *Highlight new markers*.

To give a newly-created marker the same visibility as its parent category, set automatic highlighting to *Inherit from parent category*.

Hiding All Incomplete Nets

To hide all the incomplete net flight lines in the design canvas

- Press **h** on your keyboard.

The visibility of the markers is turned off in the Annotation Browser and the flight lines removed from the canvas.

Assigning Colors to Incomplete Nets

Layout XL uses the colors you assigned to the y_0 through y_9 entry layers when displaying flight lines in the design window. If there are more than ten incomplete nets in the design, the colors are recycled.

To assign a color other than the default to a flight line,

1. In the Annotation Browser assistant, click the *Set Highlight Color* column next to the entry for the net whose color you want to change.
2. Do one of the following.
 - ☐ Choose the color you want for that flight line from the predefined list.
 - ☐ Choose *cycle* to automatically select the next color in the predefined list.

The flight line changes to the color you specified.

To assign colors to all the flight lines under a particular node,

1. In the Annotation Browser assistant, click the *Set Highlight Color* column for the parent node of the markers you want to change.
2. Do one of the following.
 - ☐ To show all the markers contained in the node in the same color, choose the color you want from the drop-down list.
 - ☐ To show each of the markers contained in the node in a different color, choose *cycle*. The flight lines are assigned colors automatically based on the predefined list.

The flight lines change to the color you specified.

Zooming In on Incomplete Nets

To zoom in on the flight lines showing selected incomplete nets,

1. In the Annotation Browser assistant, select the nets you are interested in.

2. Click the *Zoom To Selected* button in the Annotation Browser toolbar.



The layout window display zooms to the flight lines for the selected nets. The level of detail you see depends on how many nets are selected and their positions in the design.

To automatically zoom in on the flight line showing the currently selected incomplete net,

- Click the *Auto Zoom* button in the Annotation Browser assistant toolbar.



The display automatically zooms and pans to the marker representing the currently selected entry in the Annotation Browser. When the selection changes, the display updates accordingly.

For more information on *Auto Zoom*, see the toolbar section in Annotation Browser Graphical User Interface.

Editing In Place

Use the *Edit In Place* command to edit a placed instance in your design and maintain the connectivity. You can also use the *Edit – Hierarchy – Descend* command to descend into an instance.

When you choose *Edit In Place* (or *Descend*), by default, Layout XL opens the connectivity reference for the cell into which you are descending. To prevent this, switch off the *Open connectivity reference* option on the General Tab of the Layout XL Options form. When switched off, the layout cellview is still opened in Layout XL mode, but with no connectivity reference.

Note: The *Edit In Place* command allows you to edit Pcells also. For more information on editing of these cells, see Liquid Editing in the *Virtuoso Parameterized Cell Reference User Guide*.

To edit a placed instance in your design,

1. Choose *Edit – Hierarchy – Edit In Place*.
2. Click on the instance whose cell you want to edit.

The banner at the top of the window changes to show that you are now editing the master cell for that instance. Notice the instance is highlighted along the border.

3. When you have finished editing, choose *Edit – Hierarchy – Return* to stop editing in place.

Layout XL returns you to the next level of the hierarchy.

To specify the level of hierarchy to return to, choose *Edit – Hierarchy – Return To Level*.

Editing Soft Blocks

The *Edit Soft Blocks* command lets you modify the soft blocks in your design. Use this command if you need to modify the attributes of individual soft blocks to take account of placement and connectivity requirements in your design.

Important

This command is recommended if you want to make changes that do not require the physical hierarchy to be regenerated. If you need to make more radical changes (by, for example, generating new soft blocks or flattening others that you no longer require), use the *Launch – Configure Physical Hierarchy* command in Soft Block mode. For more information, see [Configuring the Physical Hierarchy](#)

Using the *Edit Soft Blocks* command, you can modify

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

A number of Virtuoso Layout Suite commands have been enhanced to support Level-1 Editing mode, which is the ability to edit objects inside a soft block without using the *Descend* or *Edit In Place* commands. This section further describes how to enable Level-1 Editing mode and how to use the enhanced *Move*, *Stretch*, *Reshape*, and *Chop* commands.

Changing the Block Type

To change the type of a soft block,

1. From the layout window menu bar, choose *Edit – Soft Blocks*.

Virtuoso Layout Suite XL User Guide

Editing the Layout

The Edit Soft Blocks form is displayed.

Edit Soft Block(s)

Soft Block List

☐ Select All

Library Name	Cell Name	View Name
ether	top_1_port_pnr	layout
ether	Bias_port	layout
ether	anlg_cntl	layout

Update Selected Soft Block(s) Attributes

☐ Common

ether:top_1_port_pnr:layout

Library Name: ether

Cell Name: top_1_port_pnr

View Name: layout

Block Type: custom

Cell Type: softMacro

Boundary I/O Pins Obstructions

Boundary Type: ☒ Rectangle ☐ Polygon

Aspect Ratio (W/H): 2.00 Utilization(%): 100

Area Calculation

Area Calculation Method: ☒ Manual ☐ Internal Estimator ☐ User Defined Estimator

Area: 269999.69

OK Cancel Apply Help

2. In the *Soft Block List*, select the blocks whose type you want to change.

The selected blocks are made available in the *Update Selected Soft Block Attributes* list.

3. Select the blocks whose type you want to change.



Tip

If you want to change the type for all the listed blocks, check the *Common* box.

4. Choose the *Block Type* you need from the cyclic field.

5. Click *OK* or *Apply* to accept the change.

Changing the Boundary

You can change any fixed or non-fixed rectangular boundary or any polygonal boundary by adjusting the parameters that are defined in the Edit Soft Blocks form.

Changing a Rectangular Boundary in Fixed Mode

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

To change a rectangular boundary in fixed mode,

1. From the layout window menu bar, choose *Edit – Soft Blocks*.

The Edit Soft Blocks form is displayed.

2. Select the blocks whose boundary you want to change.

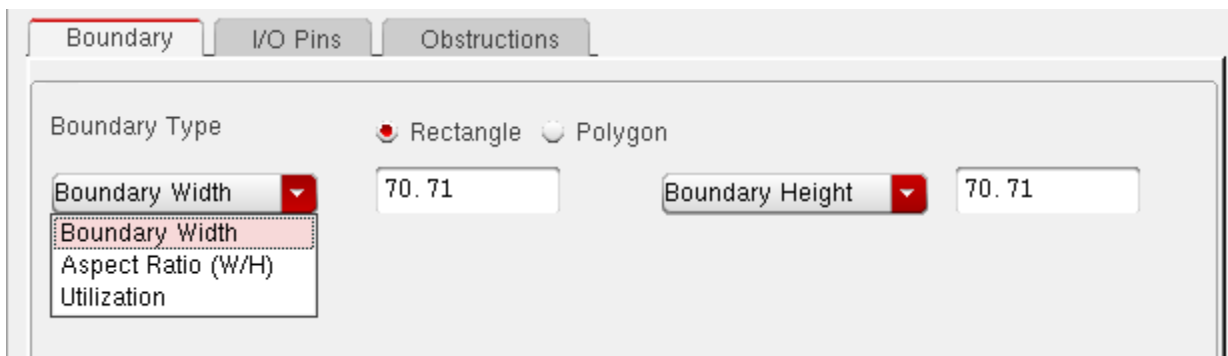
3. In the *Boundary* tab, check the *Rectangle* radio button.

4. Choose one of the following.

- ☐ *Aspect Ratio (W/H)* in the first cyclic field and *Height* in the second. In this case, the boundary width is calculated from the boundary height and aspect ratio.
- ☐ *Aspect Ratio (W/H)* in the first cyclic field and *Width* in the second. In this case, the boundary height is calculated from the boundary width and the aspect ratio.
- ☐ *Width* in the first cyclic field and *Height* in the second. In this case, these values you type in are used to create the boundary.

When you choose any of the above combinations, the form changes to hide the other fields.

The picture below show a rectangular boundary specified using the width and height options.



Changing a Rectangular Boundary in Non-Fixed Mode

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

To change a rectangular boundary in non-fixed mode,

1. From the layout window menu bar, choose *Edit – Soft Blocks*.

The Edit Soft Blocks form is displayed.

2. Select the blocks whose boundary you want to change.
3. In the *Boundary* tab, check the *Rectangle* radio button.

Virtuoso Layout Suite XL User Guide

Editing the Layout

4. Use the two cyclic lists to specify the utilization factor and one fixed parameter for the block. The table below lists the valid combinations in non-fixed mode.

<i>Utilization</i>	<i>Height</i>
<i>Utilization</i>	<i>Width</i>
<i>Utilization</i>	<i>Aspect ratio (W/H)</i>

5. Choose an *Area Calculation Method*.

The area calculation methods available depend on the cell type of the soft blocks.

- ☐ If the block is of type *softMacro*, you can use the *Manual*, *Internal Estimator*, or *User Defined Estimator* methods.
- ☐ If the block is of type *blockBlackBox*, you can use the *Manual*, *Area Per Gate*, or *User Defined Estimator* methods.

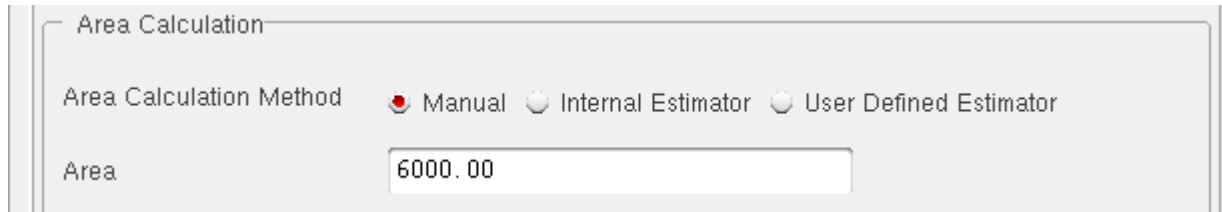
For more information on these methods, see

- ☐ [Manual Area Estimation](#)
- ☐ [Internal Estimators](#)
- ☐ [User Defined Estimators](#)
- ☐ [Area Per Gate Estimation](#)

6. When the area has been estimated, click *OK* or *Apply* to accept the changes to the boundary.

Manual Area Estimation

Use *Manual* area estimation if you know the block area. Simply type the area into the *Area* field.



Area Calculation

Area Calculation Method ☒ Manual ☐ Internal Estimator ☐ User Defined Estimator

Area

The boundary of the soft block modified based on the value specified in the second cyclic field.

Virtuoso Layout Suite XL User Guide

Editing the Layout

For example, consider a block where the *Utilization* is specified as 1 and the *Boundary Height* is 63 units. If you increase the area from 1000 to 2000 units, the width of the place and route boundary increases because the value of height is fixed at 63.

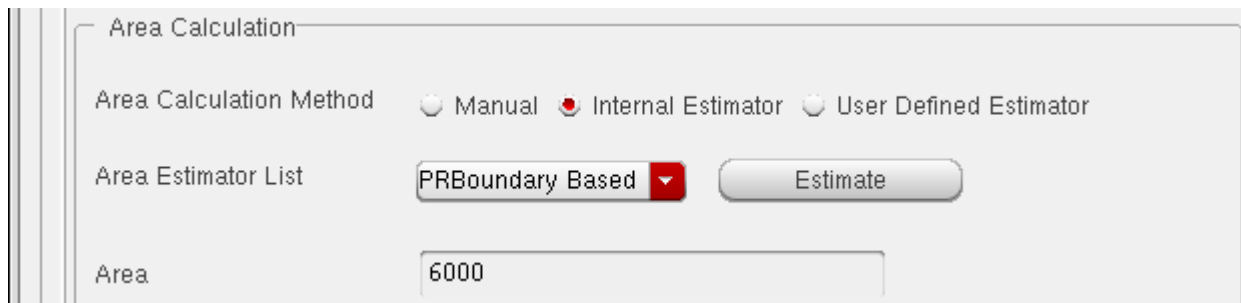
Similarly, if the value of the *Boundary Width* field is specified, the height of the block would be modified to arrive at the specified area.

Internal Estimators

Use *Internal Estimator* to have the system calculate the area of a soft block of type *softMacro* based on

- The sum of all the place and route boundaries of the objects inside the soft block
- The sum of all the bounding boxes of the components contained in the soft block.

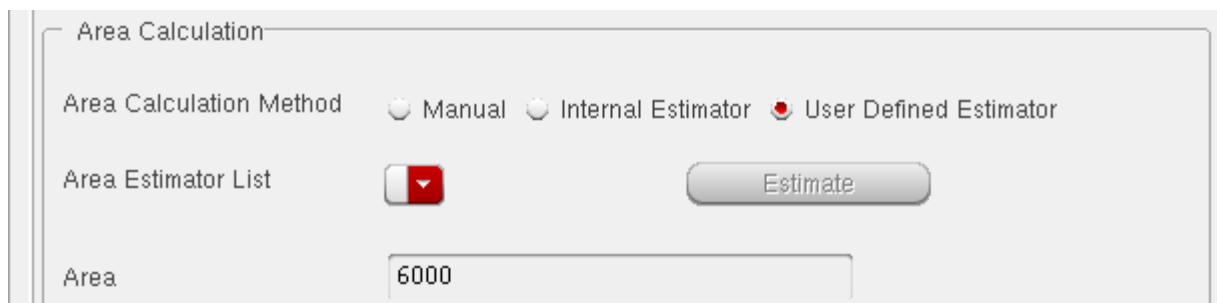
Choose either *PRBoundary Based* or *BBox Based* from the cyclic field and click the *Estimate* button to calculate the area.



The screenshot shows the 'Area Calculation' dialog box. Under 'Area Calculation Method', the 'Internal Estimator' radio button is selected. The 'Area Estimator List' dropdown menu is set to 'PRBoundary Based'. The 'Area' text field contains the value '6000'. The 'Estimate' button is visible to the right of the dropdown menu.

User Defined Estimators

User Defined Estimator lets you specify your own area calculation function using the framework described in [Creating and Registering a User Defined Area Estimation Function](#). (The picture below shows no user-defined function registered.)



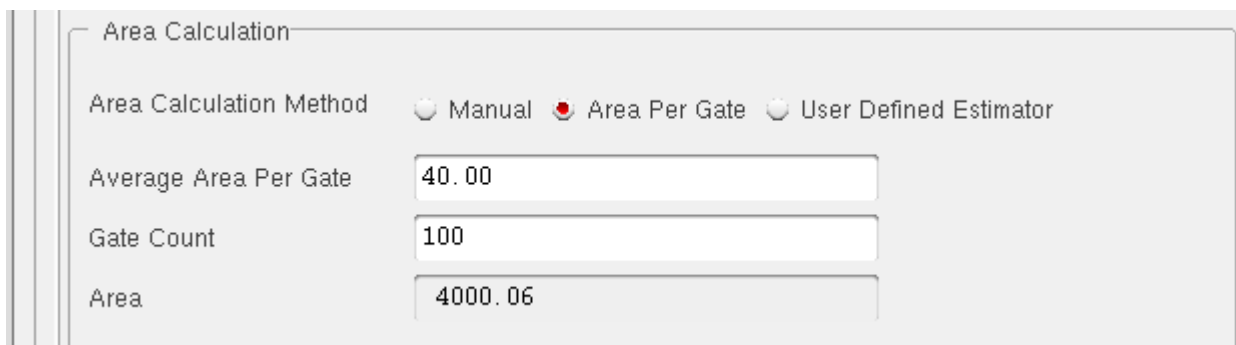
The screenshot shows the 'Area Calculation' dialog box. Under 'Area Calculation Method', the 'User Defined Estimator' radio button is selected. The 'Area Estimator List' dropdown menu is empty, showing a red arrow icon. The 'Area' text field contains the value '6000'. The 'Estimate' button is visible to the right of the dropdown menu.

You can view all the registered area estimators in the *Area Estimator List*. Select the estimators you want to use and click the *Estimate* button to calculate the area.

Area Per Gate Estimation

Use *Area Per Gate* to calculate the area for a soft block of type *blockBlackBox*. You enter the area per gate and the gate count and the system calculates the area based on the following formula.

$$\text{Area} = \text{Area Per Gate} * \text{Gate Count}$$



Area Calculation	
Area Calculation Method	☐ Manual ☒ Area Per Gate ☐ User Defined Estimator
Average Area Per Gate	40.00
Gate Count	100
Area	4000.06

The *Area* field is grayed out in this mode.

Changing a Polygonal Boundary

You can create or change a polygonal place and route boundary by specifying the points of each of the vertices of the polygon.

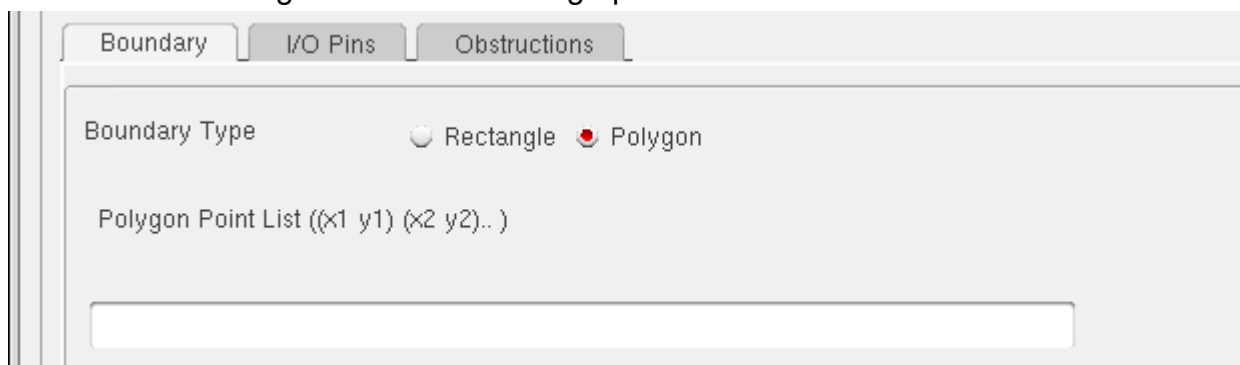
To change a polygonal boundary,

1. From the layout window menu bar, choose *Edit – Soft Blocks*.

The Edit Soft Blocks form is displayed.

2. Select the blocks whose boundary you want to change.
3. In the *Boundary* tab, check the *Rectangle* radio button.

The form changes as shown in the graphic.

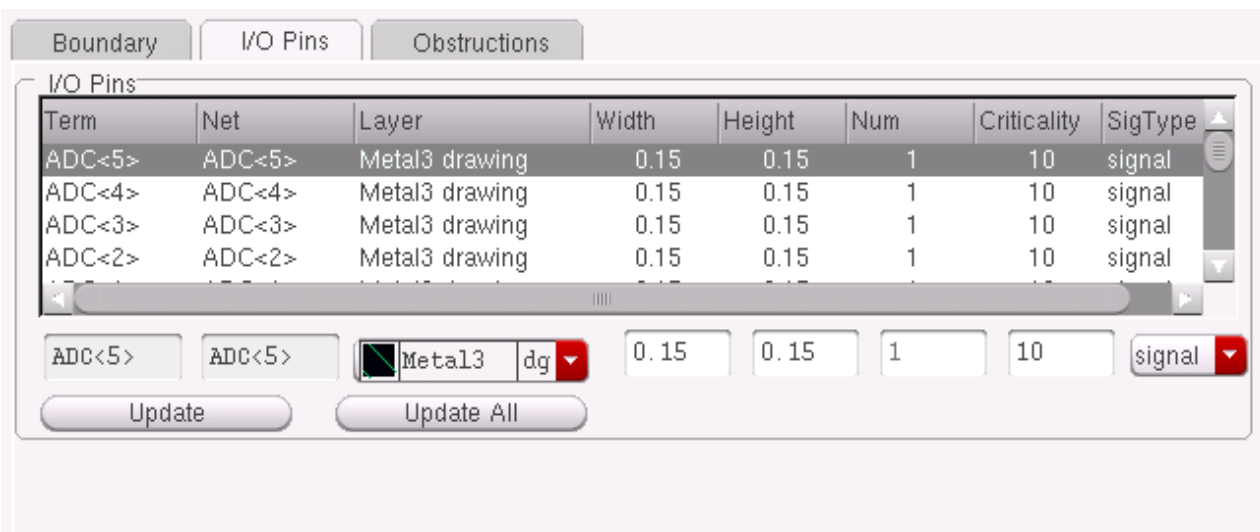


4. Specify the points for each of the vertices of the polygon in the *Polygonal Point List* text box.
5. Click *OK* or *Apply* to change the boundary.

Changing I/O Pin Attributes

The *I/O Pins* tab lets you update the layer, width, height, number, criticality, and signal type of existing pins. This table is initialized with *Term* and *Net Names* from the symbol view of the selected soft block.

When you click the *I/O Pins* tab, the form changes to display the *I/O Pins* table as shown below.



Term	Net	Layer	Width	Height	Num	Criticality	SigType
ADC<5>	ADC<5>	Metal3 drawing	0.15	0.15	1	10	signal
ADC<4>	ADC<4>	Metal3 drawing	0.15	0.15	1	10	signal
ADC<3>	ADC<3>	Metal3 drawing	0.15	0.15	1	10	signal
ADC<2>	ADC<2>	Metal3 drawing	0.15	0.15	1	10	signal

ADC<5>	ADC<5>	Metal3	dg	0.15	0.15	1	10	signal

Update Update All

You can select one or more rows to modify in the table. If you select multiple rows, the fields below the table list the respective values only if they are same for the selected rows. Otherwise, the fields display *AS IS*.

To add or change I/O pin parameters,

1. From the layout window menu bar, choose *Edit – Soft Blocks*.

The Edit Soft Blocks form is displayed.

2. Select the blocks whose pins you want to change.
3. Select the entry in the table that you want to update.
4. Specify the new value in the field below the column that you want to update.

For example, if you want to update the layer purpose for a net to *metal1 drawing*, click the cyclic list below the *Layer* column and select *metal1 dg*.

5. Do one of the following.

- ☐ Click *Update* to update the selected entries in the table.
- ☐ Click *Update All* to update all the entries in the table.

You can also update multiple soft blocks in the design simultaneously.

To update multiple soft blocks,

1. Select multiple rows in the *Soft Block List* or select the *Common* button in the *Update Selected Soft Block(s) Attributes* section.

Virtuoso Layout Suite XL User Guide

Editing the Layout

The *I/O Pins* tab changes as shown below.

The screenshot shows the 'Update Selected Soft Block(s) Attributes' dialog box. The 'Common' checkbox is checked. The list of selected blocks includes 'ether:top_1_port_pnr:layout', 'ether:Bias_port:layout', and 'ether:anlg_cntl:layout'. The 'Block Type' is set to 'custom' and the 'Cell Type' is 'softMacro'. The 'I/O Pins' tab is active, showing a table with the following data:

Term	Net	Layer	Width	Height	Num	Criticality	SigType
all	all	Metal3 drawing	1.00	1.00	1	10	signal

Below the table, there are input fields for each parameter: 'all' for Term, 'all' for Net, a color swatch and 'Metal3 dg' for Layer, '1.00' for Width, '1.00' for Height, '1' for Num, '10' for Criticality, and 'signal' for SigType. There are 'Update' and 'Update All' buttons at the bottom.

2. Change the parameters as required and click *Update*.

All the pins in all the selected blocks are updated with the new values.

Changing Obstructions

The *Obstructions* tab lets you add or change routing and placement obstructions implemented using layer and placement halos.

Soft Block	Type	Layer	Top	Bottom	Left	Right
design:block3:layout	routing	metal1	5.000	5.000	5.000	5.000

Blockage Type: placement, routing, fill, slot, pin, feedthru

Material: all

Layer: cont, metal1, metal2, metal3

Offset: Top 5, Bottom 5, Left 5, Right 5

Update Delete

- To define a *routing*, *fill*, *slot*, *pin*, *feedthru*, or *screen* obstruction, you need to set the *Blockage Type*, the *Layer* and *Offsets*. There is also a *Material* filter which you can use to select layers based on the function defined in the technology file.

Note: The list of extractable layers is retrieved from the constraint group set for the current layout cellview. If there is no constraint group set for the current layout cellview, the software retrieves the layers from the constraint group specified by the `setupConstraintGroup` environment variable. If there are no layers defined in either location, it issues a message and disables soft block definition.

- To define a *placement* obstruction, you need only set the *Offsets*. You can define only one placement obstruction per block.

Adding an Obstruction

To add an obstruction,

1. From the layout window menu bar, choose *Edit – Soft Blocks*.

The Edit Soft Blocks form is displayed.

Virtuoso Layout Suite XL User Guide

Editing the Layout

2. Select the block in which you want to create the halo.
3. Select the type of halo from the *Blockage Type* list.

The form changes as shown below.

The screenshot shows the 'Obstructions' tab in the Virtuoso Layout Suite XL. The 'Halos' section is active, displaying a table with columns: Soft Block, Type, Layer, Top, Bottom, Left, and Right. Below the table, the 'Blockage Type' list is open, showing options: placement, routing, fill, slot, pin, and feedthru. The 'Material' dropdown is set to 'all'. The 'Layer' dropdown is open, showing options: cont, metal1, metal2, and metal3. The 'Offset' section has input fields for Top, Bottom, Left, and Right, all set to 5. 'Update' and 'Delete' buttons are at the bottom.

4. Select the *Layer* on which the obstruction is to be created.

If required, you can use the *Material* cyclic list to filter the layers in the *Layer* field.

Virtuoso Layout Suite XL User Guide

Editing the Layout

5. Specify the offsets in the *Top*, *Left*, *Right*, and *Bottom* fields.

Boundary I/O Pins Obstructions

Halos

Soft Block	Type	Layer	Top	Bottom	Left	Right

Blockage Type: placement, routing, fill, slot, pin, feedthru

Material: all

Layer: cont, metal1, metal2, metal3

Offset: Top: 5, Bottom: 5, Left: 5, Right: 5

Update Delete

6. Click *Update*.

The halo is displayed in the *Halos* table.

Boundary I/O Pins Obstructions

Halos

Soft Block	Type	Layer	Top	Bottom	Left	Right
design:block3:layout	routing	metal1	5.000	5.000	5.000	5.000

Blockage Type: placement, routing, fill, slot, pin, feedthru

Material: all

Layer: cont, metal1, metal2, metal3

Offset: Top: 5, Bottom: 5, Left: 5, Right: 5

Update Delete



If you create a *placement* halo, you do not need to select the *Layer* field.

Updating an Existing Obstruction

To update an existing halo,

1. From the layout window menu bar, choose *Edit – Soft Blocks*.
The Edit Soft Blocks form is displayed.
2. Select the block which contains the halo to be updated.
3. Select the halo you want to update in the *Halos* table.
4. Modify any or all of the following.
 - ☐ Halo type in the *Blockage Type* field
 - ☐ Layer in the *Layer* field
 - ☐ The top, left, right, or bottom offsets
5. Click *Update*.
6. The updated halo is listed in the *Halos* table.

Deleting an Obstruction

To delete a halo,

1. From the layout window menu bar, choose *Edit – Soft Blocks*.
The Edit Soft Blocks form is displayed.
2. Select the block which contains the halo to be updated.
3. Select the halo you want to delete in the *Halos* table.
4. Click *Delete*.

The halo is deleted and removed from the *Halos* table.

Adding Obstructions on Multiple Blocks

You can select multiple blocks and add commonly defined halos to each of the blocks.

Virtuoso Layout Suite XL User Guide

Editing the Layout

To add a halo to multiple selected blocks,

1. Select multiple rows in the *Soft Block List* or select the *Common* button in the *Update Selected Soft Block(s) Attributes* section.

The form changes as shown below.

Update Selected Soft Block(s) Attributes

☒ Common

design:block:layout
design:block1:layout
design:block3:layout

Library Name

Cell Name

View Name

Block Type **AS IS**

Cell Type

Boundary I/O Pins Obstructions

Halos

Soft Block	Type	Layer	Top	Bottom	Left	Right

Blockage Type
routing
fill
slot
pin
feedthru

Material

Layer
metal1
metal2
metal3

Offset
Top
Bottom
Left
Right

2. Modify any or all of the following.
 - ☐ Halo type in the *Blockage Type* field
 - ☐ Layer in the *Layer* field
 - ☐ The top, left, right, or bottom offsets
3. Click *Update*.

Virtuoso Layout Suite XL User Guide

Editing the Layout

The specified halo is added to each of the selected blocks.

Update Selected Soft Block(s) Attributes

☒ Common

design:block:layout
design:block1:layout
design:block3:layout

Library Name

Cell Name

View Name

Block Type Cell Type

Boundary I/O Pins Obstructions

Halos

Soft Block	Type	Layer	Top	Bottom	Left	Right
design:block:layout	routing	metal1	5.000	5.000	5.000	5.000
design:block1:layout	routing	metal1	5.000	5.000	5.000	5.000
design:block3:layout	routing	metal1	5.000	5.000	5.000	5.000

Blockage Type

Material

Layer

Offset
Top
Bottom
Left
Right

Enabling Level-1 Editing Mode

Level-1 Editing mode is the ability to edit objects inside a soft block without using the *Descend* or *Edit In Place* commands.

Important

While in level-1 editing mode, you can edit the objects placed at level-1 only if the top-level is selectable.

To enable Level-1 Editing mode,

Virtuoso Layout Suite XL User Guide

Editing the Layout

1. From the layout window menu bar, choose *Options – Display*.

The Display Options form appears.

Display Options

Display Controls

- ☒ Open to Stop Level
- ☒ Axes
- ☒ Instance Origins
- ☒ EIP Surround
- ☐ Pin Names
- ☒ Dot Pins
- ☒ Use True BBox
- ☐ Cross Cursor
- ☐ Row Name
- ☐ Row Site
- ☐ Nets
- ☐ Access Edges
- ☐ Instance Pins
- ☐ Array Icons
- ☒ Label Origins
- ☒ Dynamic Highlight
- ☒ Net Expressions
- ☐ Stretch Handles
- ☐ Via Shapes

Path Display: **Borders and Centerlines**

Show Name Of: ☐ instance ☒ master

Array Display: **Display Levels**

- ☒ Full: Start 0, Stop 1
- ☐ Border
- ☐ Source

Grid Controls

Type: ☐ none ☒ dots ☐ lines

Minor Spacing: 1

Major Spacing: 5

X Snap Spacing: 0.1

Y Snap Spacing: 0.1

Filter

Size: 3 Style: empty

Snap Modes

Create: diagonal

Edit: diagonal

☒ Cellview ☐ Library ☐ Tech Library ☐ File

~/.cdsenv

Save To Load From Delete From

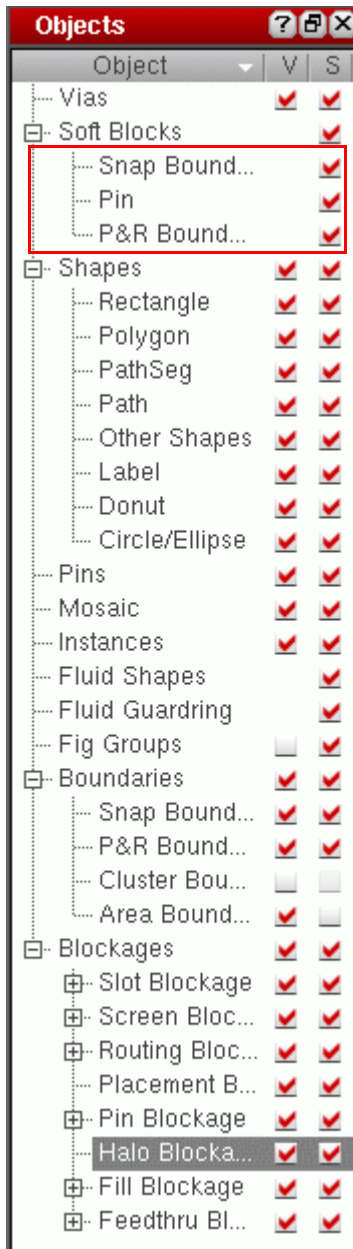
OK Cancel Defaults Apply Help

2. In the *Display Controls* group box, under *Display Levels*, set the *Stop* value to 1.

Virtuoso Layout Suite XL User Guide

Editing the Layout

3. In Objects assistant, check the *V* and *S* check box to make *Pins*, *P&R Boundary*, and *Snap Boundary* visible and selectable. Check the *S* check box to make *Soft Block Pins*, *P&R boundary*, and *Snap Boundary* selectable.



 Important

If an instance bounding box fully overlaps the place and route boundary, you need to click on the instance twice in order to select the place and route boundary. The first click selects the instance, and the second click selects the Level-1 place and route boundary.

Once in Level-1 Editing mode, you can do the following.

- [Moving an Object in Level-1 Editing Mode](#)
- [Stretching an Object in Level-1 Editing Mode](#)
- [Reshaping an Object in Level-1 Editing Mode](#)
- [Chopping an Object in Level-1 Editing Mode](#)
- [Finding and Replacing Nets](#)
- [Discarding Edits](#)
- [Saving Hierarchical Cellviews under a Different Name](#)

Moving an Object in Level-1 Editing Mode

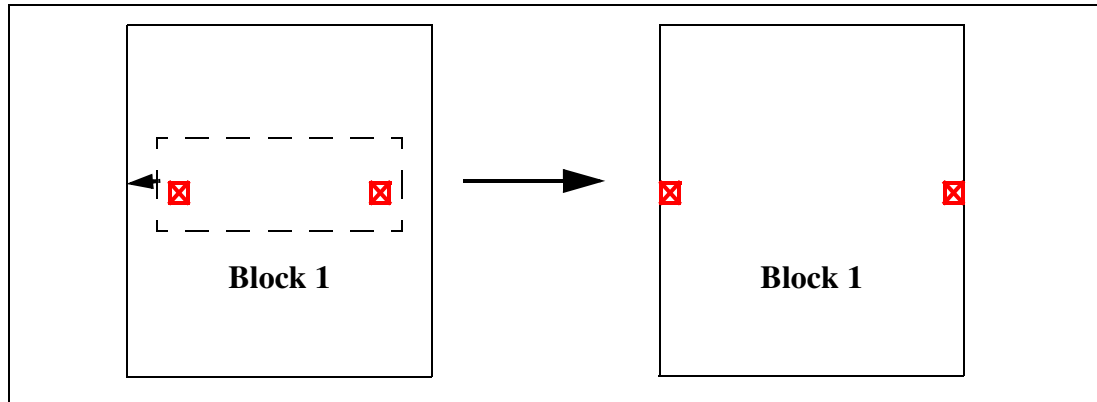
Use the *Move* command in Level-1 Editing mode to move place and route boundaries, snap boundaries, and shape pins without entering Edit In Place mode. You can select the objects to be moved either before or after you start the *Move* command.

 Tip

For information on the Layout XL Move form, see [Move](#) on page 1000. For general information on moving devices, pins and shapes in Layout XL, see

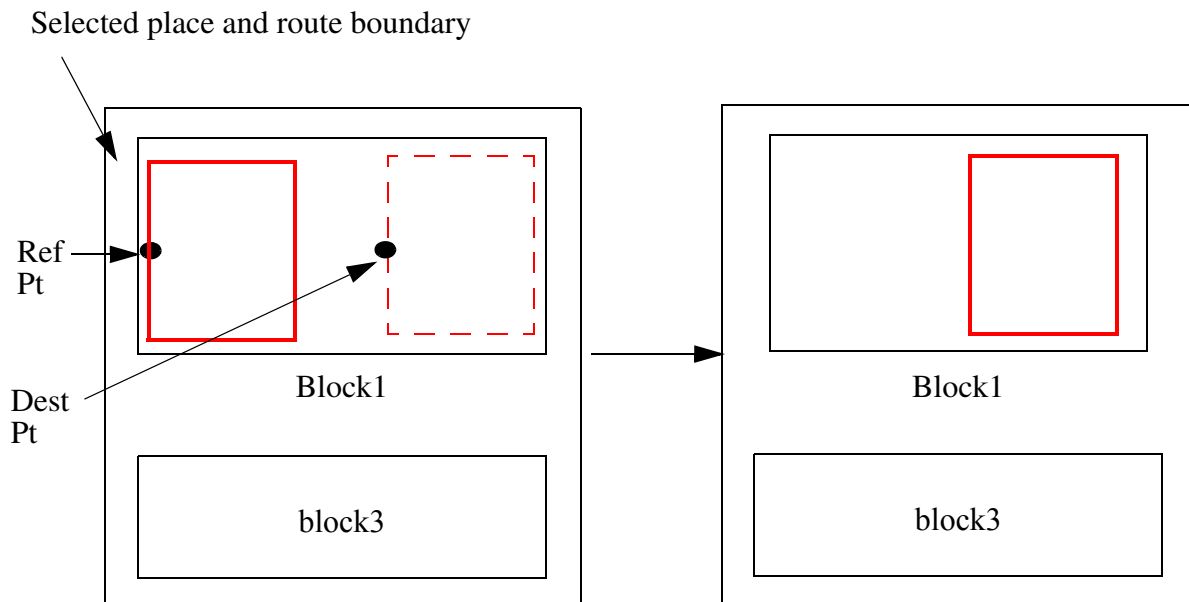
- ☐ [Moving Objects Manually](#)
- ☐ [Moving Objects Using the Move Form](#)
- ☐ [Moving Generated Components into the Design Boundary](#)

The picture below illustrates what happens when you move a shape pin.



In both cases, the pin is snapped to either the modified boundary edge or the closest boundary edge. A pin that is assigned a status of `firm` or `locked` is not moved. If a grid is not initialized, moved pins are not snapped.

The picture below illustrates what happens when you move a place and route boundary or a snap boundary.



Virtuoso Layout Suite XL User Guide

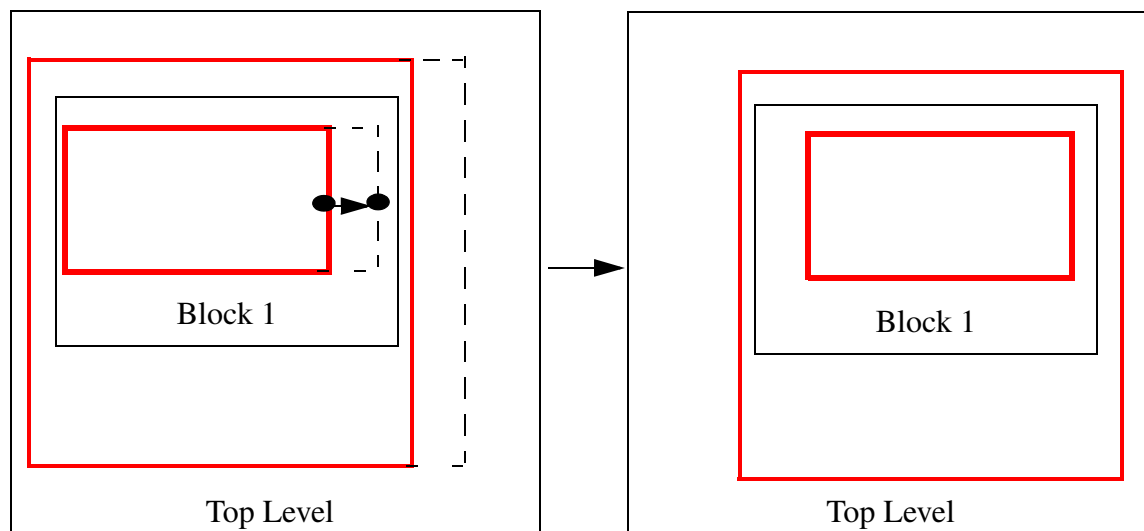
Editing the Layout

The place and route boundary is snapped according to the block type.

Block Type	Snap Behavior
<i>Analog</i>	The place and route boundary is snapped to the manufacturing grid.
<i>Digital</i>	The place and route boundary is snapped to the placement grid.

You can move multiple place and route boundaries or shape pins in a single operation. For example, you can select the corners or edges of place and route boundaries belonging to multiple blocks, as well as the top level, and apply the move operation to the entire selected set.

The picture below shows what happens when you move two place and route boundaries (Block 1 and Top Level) or snap boundaries at the same time.



Moving a Place and Route Boundary at Level 1

To move a place and route boundary at level 1,

1. With Level-1 Editing mode enabled, choose *Edit – Move* from the layout window menu bar.

Note: For more information on enabling Level-1 Editing mode, see [Enabling Level-1 Editing Mode](#).

A message in the status banner at the bottom of the layout window prompts you to select the figure to be moved.

2. Select the place and route boundary you want to move.

You are prompted to point to the reference point for the move.

3. Point to the reference point.

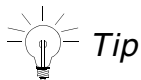
You are prompted to point to the new location for the move.

4. Point to the new location.

The place and route boundary is moved from the reference point to the new location.

Stretching an Object in Level-1 Editing Mode

Use the *Stretch* command in Level-1 Editing mode to stretch place and route boundaries, snap boundaries, and other pins without entering Edit In Place mode. You can stretch multiple pins, including the place and route boundary and snap boundary both at level 1 and at the top level, in a single operation.



Tip

For information on the Layout XL Stretch form, see [Stretch](#). For general information on stretching devices, pins, and other shapes, see [Editing Objects](#) in the *Virtuoso Layout Suite L User Guide*.

Stretched objects are snapped to a grid depending on the block type.

Block Type	Snap Behavior
<i>Analog</i>	The place and route boundary and pin edges are snapped to the manufacturing grid.
<i>Digital</i>	The place and route boundary is snapped to the placement grid and pin centers are snapped to the routing grid defined for the metal layer on which the pins are defined. Note: If the pins are not on a metal layer, they are snapped to the manufacturing grid.

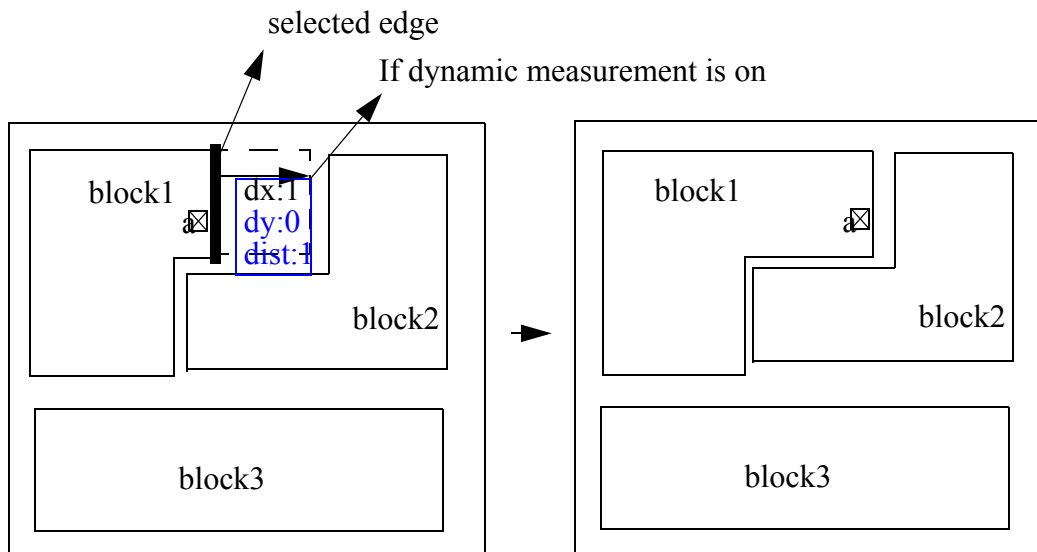
A pin that is assigned a status of `firm` or `locked` is not moved. If a grid is not initialized, moved pins are not snapped.

If you have Measurement Display mode switched on, the measurements (dx , dy and $dist$) of the object are shown to be changing dynamically as it is being stretched.

Important

You need to be in partial selection mode in order to select an edge or corner to stretch. Press **F4** to toggle between partial selection and full selection modes.

The picture below illustrates the effect of using the *Stretch* command on a level-1 place and route boundary or snap boundary. You can select the boundary to be stretched either before or after you have started the command.



Stretching a Single Edge

To stretch a single partial edge:

1. With Level-1 Editing mode enabled, choose *Edit – Stretch* from the layout window menu bar.

Note: For more information on enabling Level-1 Editing mode, see [“Enabling Level-1 Editing Mode”](#) on page 384.

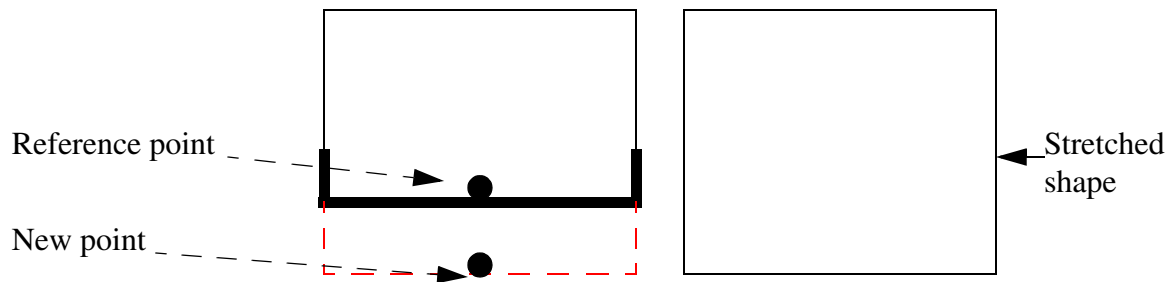
A message in the status banner at the bottom of the layout window prompts you to select the figure to be stretched.

2. Make sure that you are in partial selection mode and select the edges or corners you want to stretch.

The selected edge or corner becomes the reference point for the stretch.

3. Point to the new location for the edge.

The edge is stretched.



Stretching Multiple Edges

To select the edges or corners of multiple instances in partial selection mode,

1. With Level-1 Editing mode enabled, choose *Edit – Stretch* from the layout window menu bar.

Note: For more information on enabling Level-1 Editing mode, see [Enabling Level-1 Editing Mode](#).

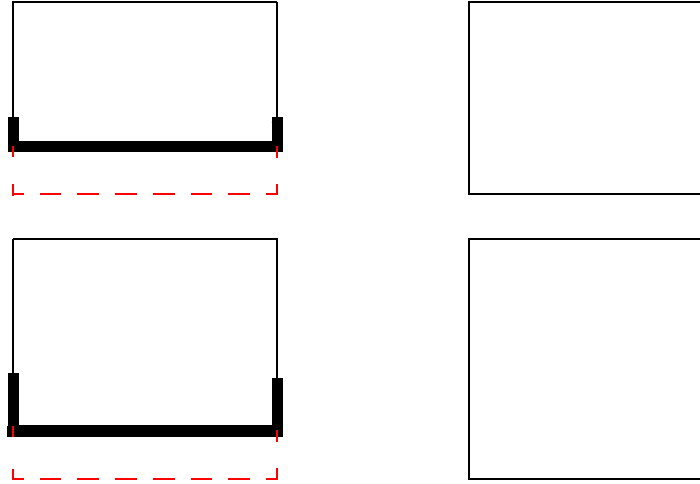
A message in the status banner at the bottom of the layout window prompts you to select the figure to be stretched.

2. Make sure that you are in partial selection mode and use `Shift`-click or area selection to select the edges or corners you want to stretch.

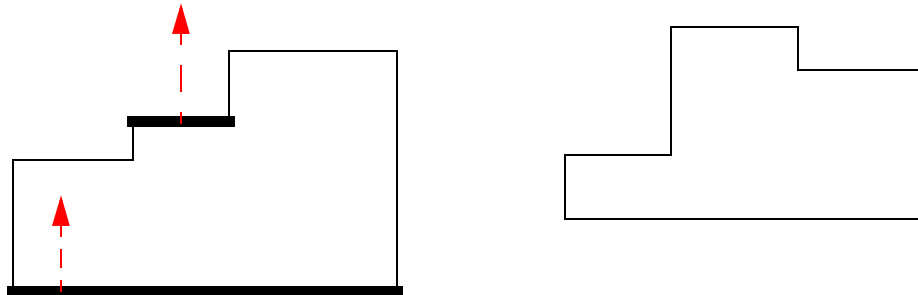
The selected edges become the reference point for the stretch.

3. Point to the new location for the edges.

The edges are stretched.



4. You can also stretch multiple partial edges of same soft block.



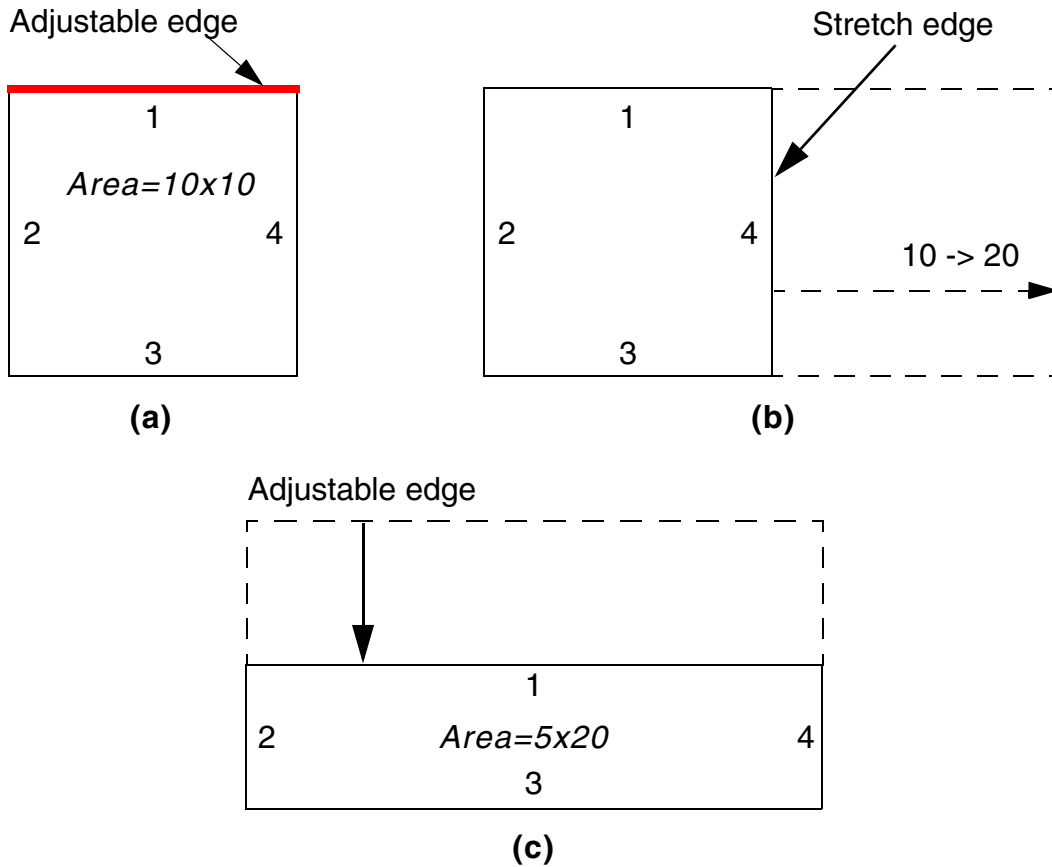
Maintaining the Area of a Soft Block during Stretch

The *Use Constant Area Stretch For Soft Block* option on the Stretch form keeps the area of a soft block constant while its place and route boundary is stretched.

Virtuoso Layout Suite XL User Guide

Editing the Layout

You specify the edges that can be adjusted in order to maintain a constant area during the stretch.



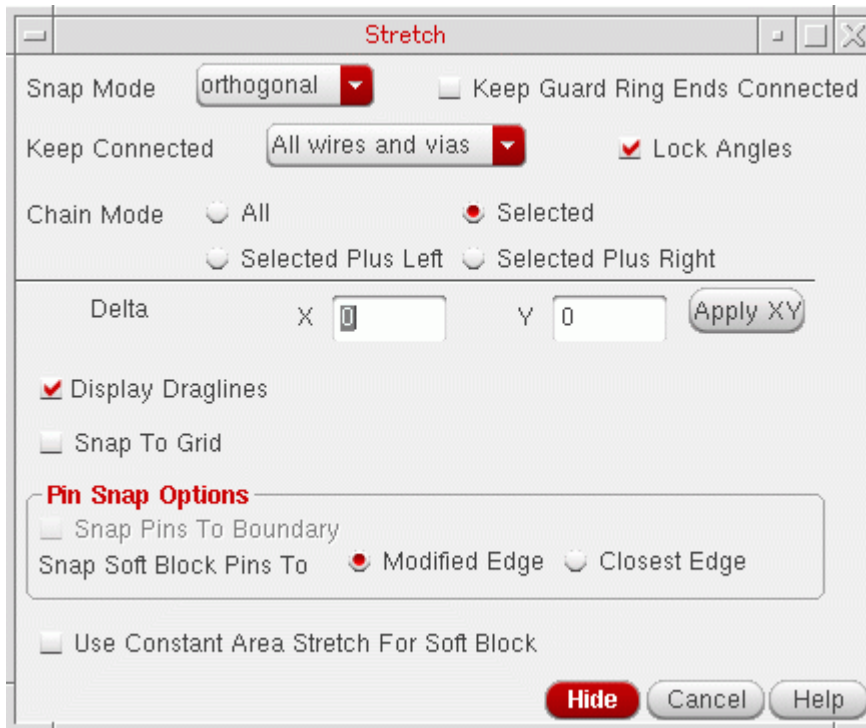
For example, the block in the picture above has an area of 100 square units. If you specify that edge 1 is an adjustable edge [figure (a)] and then increase the width of the block from 10 to 20 [figure (b)], then the Stretch command automatically adjusts the free edge from 10 down to 5 in order to maintain a constant area of 100 square units [figure (c)].

To do this,

Virtuoso Layout Suite XL User Guide

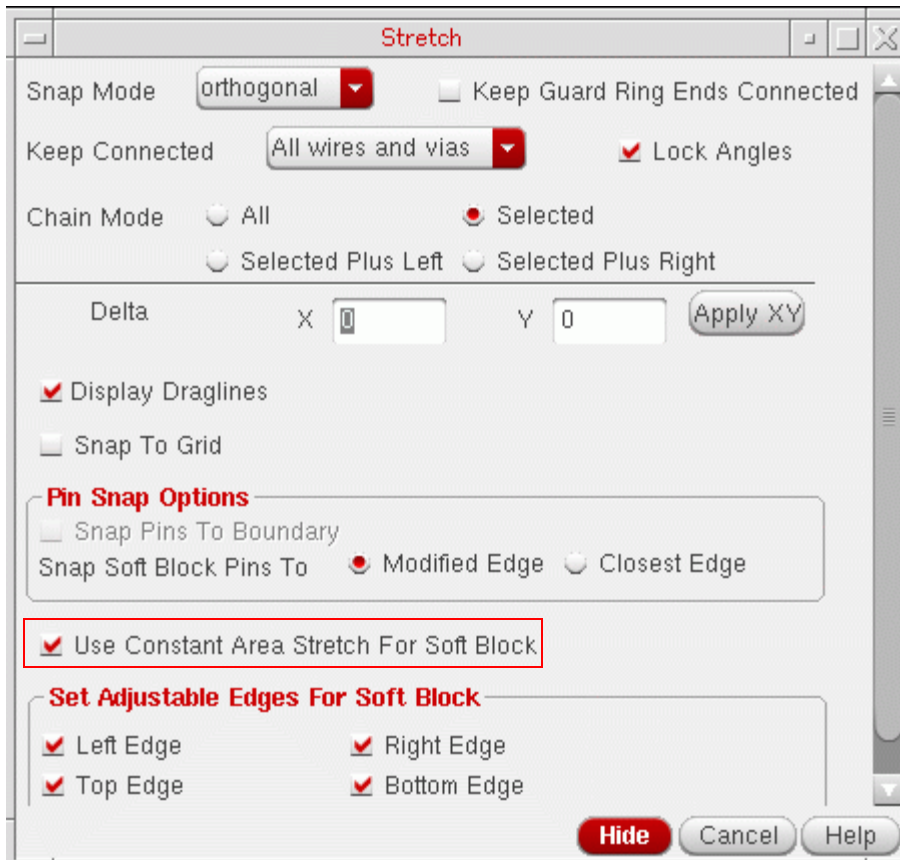
Editing the Layout

1. From the layout window menu bar, choose *Edit – Stretch* and press F3 to display the Stretch form.



2. Check the *Use Constant Area Stretch For Soft Block* option.

The form updates to show the *Set Adjustable Edges* options.



3. Make the *Top Edge* adjustable as illustrated in the picture above.
4. Select the block you want to change and perform the stretch as normal.

Changing the Angle of a Corner

To change the angle of a corner by stretching,

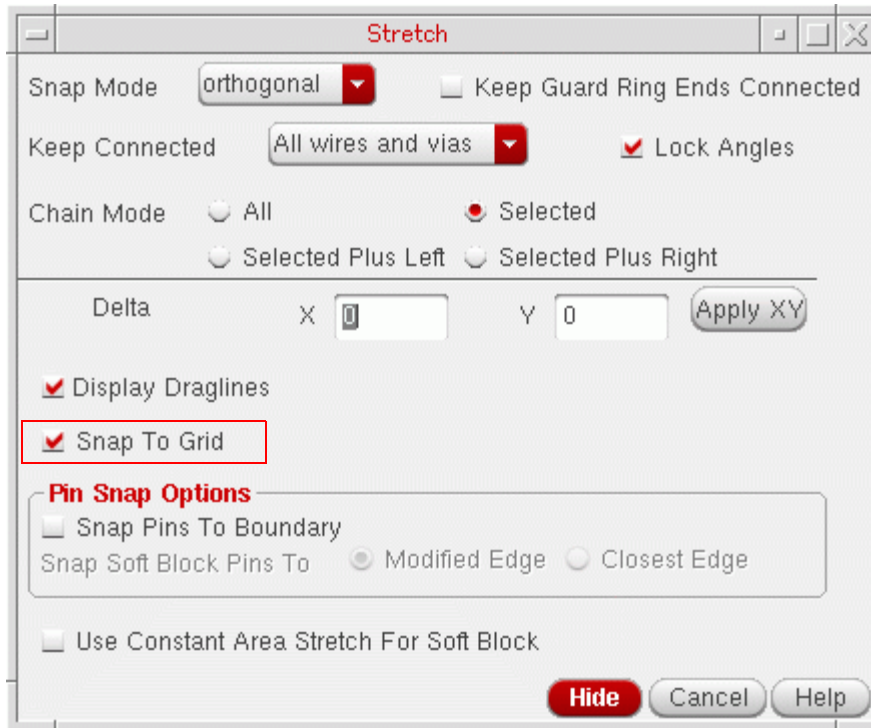
1. With Level-1 Editing mode enabled, choose *Edit – Stretch* from the layout window menu bar.

Note: For more information on enabling Level-1 Editing mode, see [Enabling Level-1 Editing Mode](#).

Virtuoso Layout Suite XL User Guide

Editing the Layout

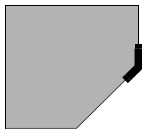
If the Stretch form does not appear automatically, press **F3** to display it.



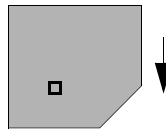
2. Turn off *Lock Angles* and turn on *Snap To Grid* to snap the place and route boundary or pin to a grid.

Note: For information on the *Snap To Grid* option, see the [Stretch](#) form.

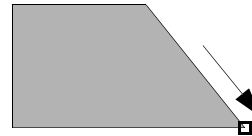
3. Click on the corner you want to stretch and then move your mouse to stretch the shape as required.



The selected corner



With *Lock Angles* on, you cannot change the angle of the corner.

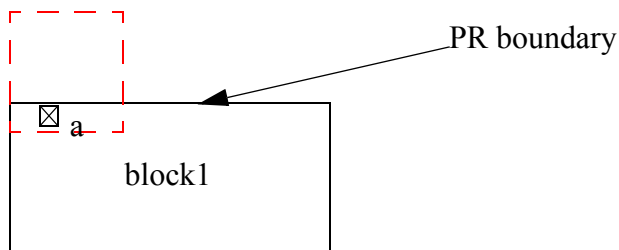


With *Lock Angles* off, you can stretch the corner in any direction.

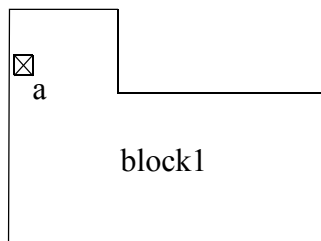
Reshaping an Object in Level-1 Editing Mode

Use the *Reshape* command in Level-1 Editing mode to reshape a place and route boundary without entering Edit In Place mode. To do this, you have to

1. Select the place and route boundary to be reshaped.
2. Draw a rectangle in such a way that it intersects at least one of the edges of the boundary or is contained within the place and route boundary.
3. Reshape the boundary with respect to this rectangle.



a) Initial block shape and pin placement



b) Block after *Reshape*

You can select the shapes to be reshaped either before or after you launch the *Reshape* command.



Tip

For information on the Layout XL Reshape form, see [Reshape](#). For general information on reshaping objects, see [Editing Objects](#) in the *Virtuoso Layout Suite L User Guide*.

Virtuoso Layout Suite XL User Guide

Editing the Layout

Stretched objects are snapped to a grid depending on the block type.

Block Type	Snap Behavior
<i>Analog</i>	The place and route boundary and pin edges are snapped to the manufacturing grid.
<i>Digital</i>	The place and route boundary is snapped to the placement grid and pin centers are snapped to the routing grid defined for the metal layer on which the pins are defined. Note: If the pins are not on a metal layer, they are snapped to the manufacturing grid.

A pin that is assigned a status of `firm` or `locked` is not moved. If a grid is not initialized, moved pins are not snapped.

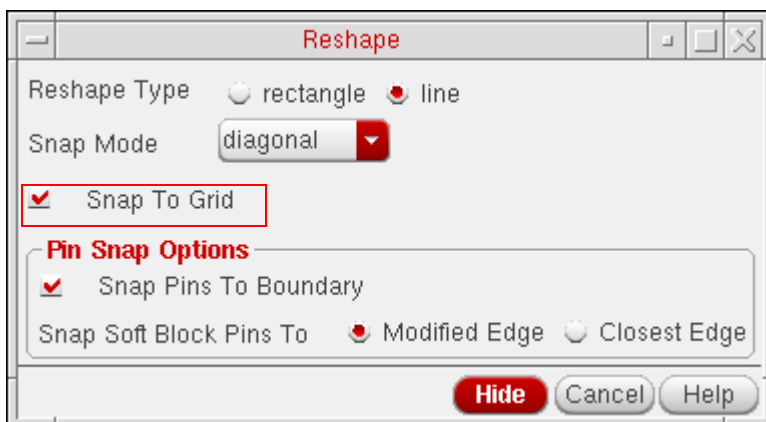
Reshaping a Place and Route Boundary at Level 1

To reshape a block place and route boundary,

1. With Level-1 Editing mode enabled, choose *Edit – Advanced – Reshape* from the layout window menu bar.

Note: For more information on enabling Level-1 Editing mode, see [Enabling Level-1 Editing Mode](#).

If the Reshape form does not appear automatically, press `F3` to display it.



2. Turn on *Snap To Grid* to snap the place and route boundary or pin to a grid.

Note: For information on the *Snap To Grid* option, see the [Reshape](#) form.

3. In the LSW, make sure that *Soft Blocks P&R* is selectable.

You are prompted to select the shape to be modified.

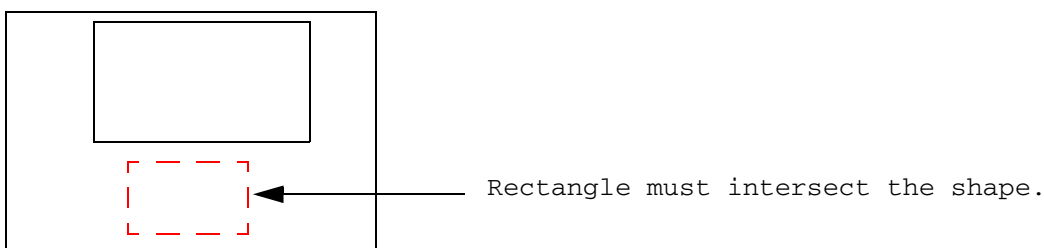
4. Select the place and route boundary to be reshaped.

You are prompted to point to the first corner of the reshape rectangle.

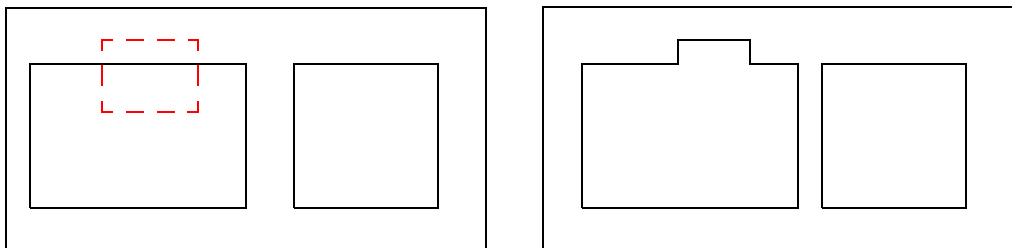
5. Point to the first corner

You are prompted to point to the diagonally opposite corner of the reshape rectangle.

If the boundary of the reshape rectangle does not intersect the boundary of the block to be reshaped, the system issues a message.



6. If you select multiple instances for reshaping, you need to reshape them one by one by using the procedure described above.

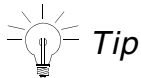
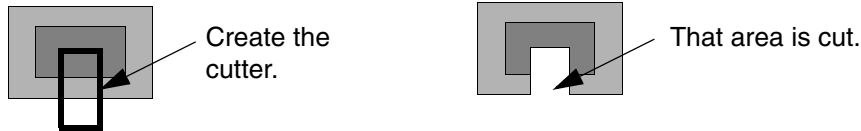


Chopping an Object in Level-1 Editing Mode

Use the *Chop* command in Level-1 Editing mode to cut away part of a place and route boundary without entering Edit In Place mode. To do this,

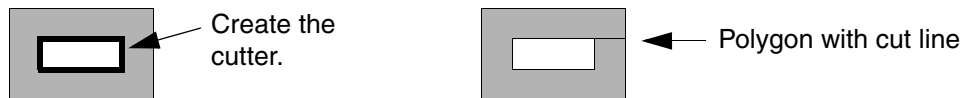
1. Select the boundary to be chopped.
2. Draw a rectangle in such a way that it intersects at least one of the edges of the boundary.

3. Chop the boundary with respect to this rectangle



Tip

If you create a polygonal cutter, you must double-click to complete the polygon cutter and the chop. If you cut a hole in an object, it is redrawn as a polygon with a cut line.



You can select the shapes to be chopped either before or after you launch the *Chop* command. You cannot chop a place and route boundary into multiple pieces. You cannot chop pins at all.

Chopped objects are snapped to a grid depending on the block type.

Block Type	Snap Behavior
<i>Analog</i>	The place and route boundary and pin edges are snapped to the manufacturing grid.
<i>Digital</i>	The place and route boundary is snapped to the placement grid and pin centers are snapped to the routing grid defined for the metal layer on which the pins are defined. Note: If the pins are not on a metal layer, they are snapped to the manufacturing grid.

During the snapping operation, if one pin (X) overlaps with another pin (Y) that is on the same layer and already snapped, the snap operation aborts with an error message:

WARNING Pin (Y) overlaps an already placed Pin (X) at location 2.4:294.6.

If there are many such pins overlapping with the already snapped pin, a list of these pins is provided in a similar error message.

Virtuoso Layout Suite XL User Guide

Editing the Layout

A pin that is assigned a status of `firm` or `locked` is not moved. If a grid is not initialized, moved pins are not snapped.



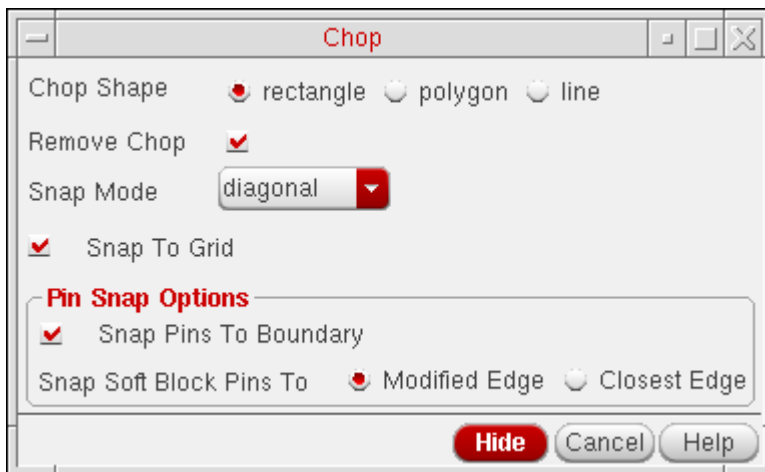
Tip

For information on the Layout XL Chop form, see [Chop](#). For general information on chopping objects, see [Editing Objects](#) in the *Virtuoso Layout Suite L User Guide*.

To cut away part of an object,

1. From the layout window menu bar, choose *Edit – Basic – Chop*.

If the Chop form does not appear automatically, press `F3` to display it.



2. Click to enter the first corner of the rectangle cutter.
3. Click to enter the opposite corner of the rectangle cutter.

The second click completes the chop.

Chopping a Place and Route Boundary at Level 1

If the *Snap To Grid* option is on, the chopping boundary is snapped depending on the block type.

Block Type	Snap Behavior
<i>Analog</i>	The chopping boundary is snapped to the manufacturing grid. Pin edges are snapped to the closest boundary edge.

Virtuoso Layout Suite XL User Guide

Editing the Layout

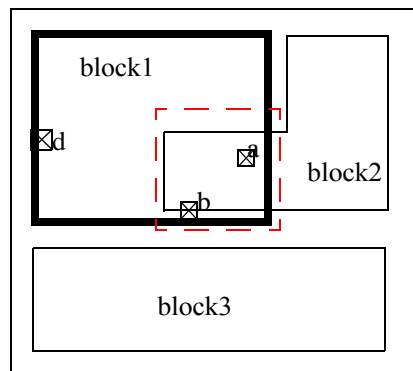
Block Type

Snap Behavior

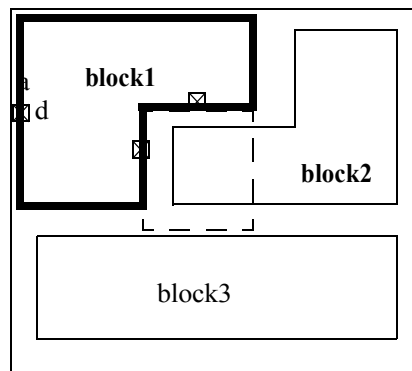
Digital

The chopping boundary is snapped to the placement grid.

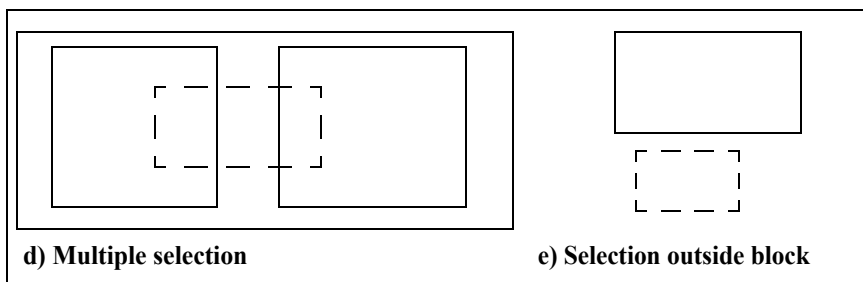
During boundary chopping, pins with placement status `placed` are automatically snapped to the closest edge. Pins with placement status `unplaced`, `fixed`, `locked`, or `unknown` are not moved, even if they were touching the boundary before it was chopped.



a) Initial block shape and pin placement



b) block after chop



d) Multiple selection

e) Selection outside block

To chop a block in post-selection mode,

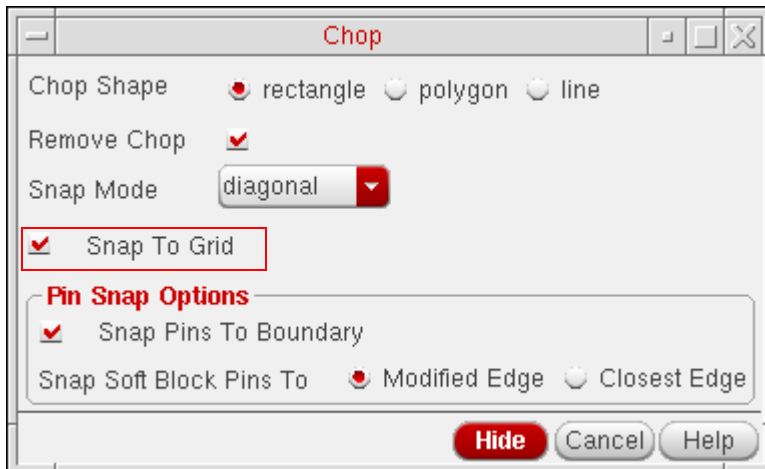
Virtuoso Layout Suite XL User Guide

Editing the Layout

1. With Level-1 Editing mode enabled, choose *Edit – Basic – Chop* from the layout window menu bar.

Note: For more information on enabling Level-1 Editing mode, see [“Enabling Level-1 Editing Mode”](#) on page 384.

If the Chop form does not appear automatically, press **F3** to display it.



2. In the Chop form, set *Chop Shape* to *rectangle* and turn on *Snap To Grid* to snap the object to its respective grid.

Note: For information on the *Snap To Grid* option, see the [Chop](#) form.

3. In the LSW, make sure *Soft Blocks P&R* is selectable.

You are prompted to point to the shape to be chopped.

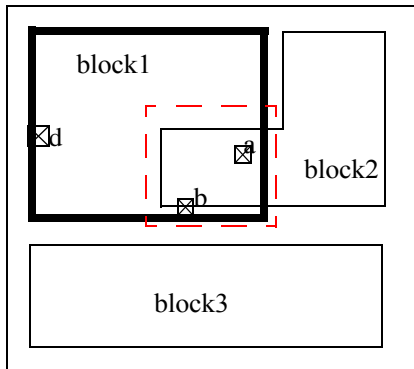
4. Select the Level-1 PR Boundary of the object to chop.

You are prompted to point to the first corner of the chop rectangle.

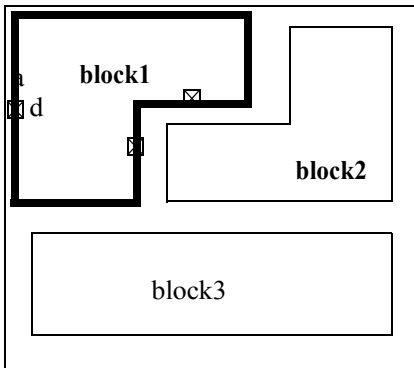
5. Point to the first corner.

You are prompted to point to the second corner of the chop rectangle.

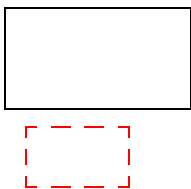
6. Point at the second corner.



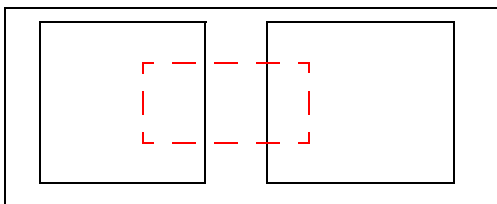
When the cutter is completed, the chop operation completes.



If the boundaries of the rectangle do not intersect the edges of the PR boundary to be modified, the place and route boundary is not chopped.

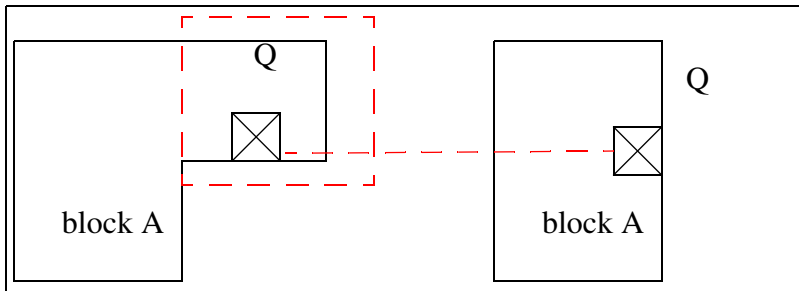


You can also chop multiple boundaries in a single operation.

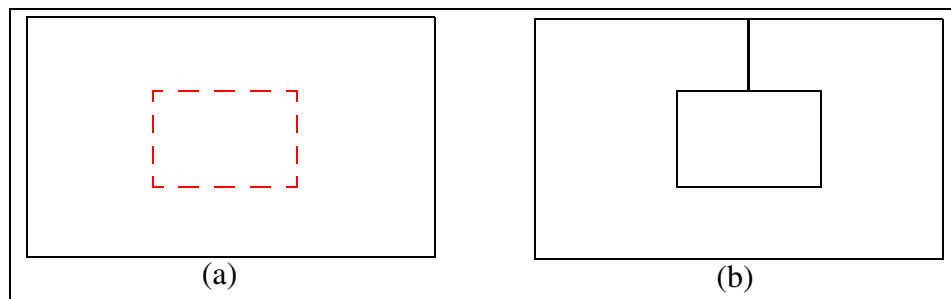


Examples

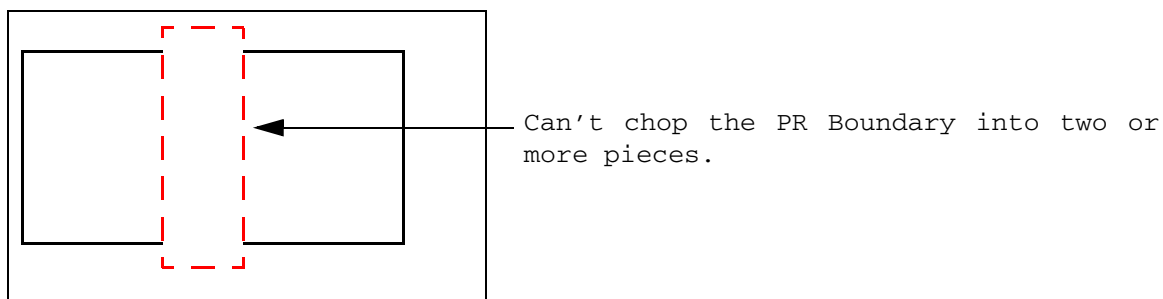
The picture below shows you what happens when you collapse three edges to form a single edge.



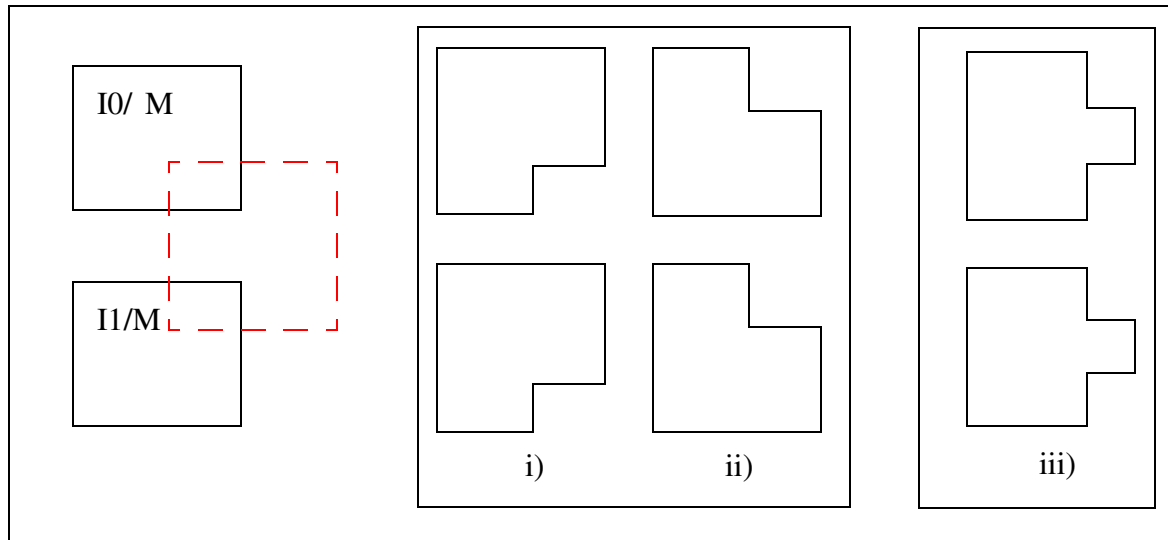
The picture below shows a donut-like shape created using the chop operation. You can select a place and route boundary and then draw a rectangle within it to chop it (a). This operation creates a polygon shape figure (b).



You cannot chop a place and route boundary into two or more pieces.



If you select multiple instances of the same master cellview, you need to chop only one of them and the changes are reflected on all the instances. The result, as shown in the picture below, can be *i* or *ii* but not *iii*.



Finding and Replacing Nets

Use the *Find/Replace* command to find and select a particular net in your design.



You can search for nets only in Layout XL or Layout GXL.

The table below shows the criteria you can use for the search.

Object	Properties	Condition/Logic	Values
<i>net</i>	<i>net name</i>	==	<i>string</i>
		!=	
	<i>signal type</i>	==	signal, ground, supply, clock, testLatch, analog, tieOff, tieHi tieLo, scan, reset
		!=	

Virtuoso Layout Suite XL User Guide

Editing the Layout



Tip

For general information on finding objects, see [Using Design Tools](#) in the *Virtuoso Layout Suite L User Guide*.

To find a net,

1. From the layout window menu bar, choose *Tools – Find/Replace*
The Find/Replace form is displayed.
2. Select *net* from *search for* field.
3. Click on *Add Criteria* to add a criteria for net selection.

4. Specify the criteria you want to narrow your search.
5. Click *Apply* to view the nets that satisfy the search criteria. The *Figure Count* tells you the number of nets that were found.
6. Use *Previous* and *Next* to cycle through the group of selected nets.
7. Click *Add Select* to select the current object in the search group.
8. Click *Select All* to select all the highlighted objects.

9. When you are finished selecting objects, click *Cancel*.

Discarding Edits

Use the *Discard Edits* command to discard all the edits made since the last time you saved the cellview. In Layout XL and Layout GXL, *Discard Edits* works hierarchically.



Tip

For general information on discarding edits, see [Layout Editor L Basics](#) in the *Virtuoso Layout Suite L User Guide*.

To discard edits,

1. From the layout window menu bar, choose *File – Discard Edits*.
You are prompted to confirm the discard operation.
2. Click *Yes* to discard all the edits made since the last time you saved the cellview.

Saving Hierarchical Cellviews under a Different Name

Use the *Save As* command to save your cellviews hierarchically under new names.

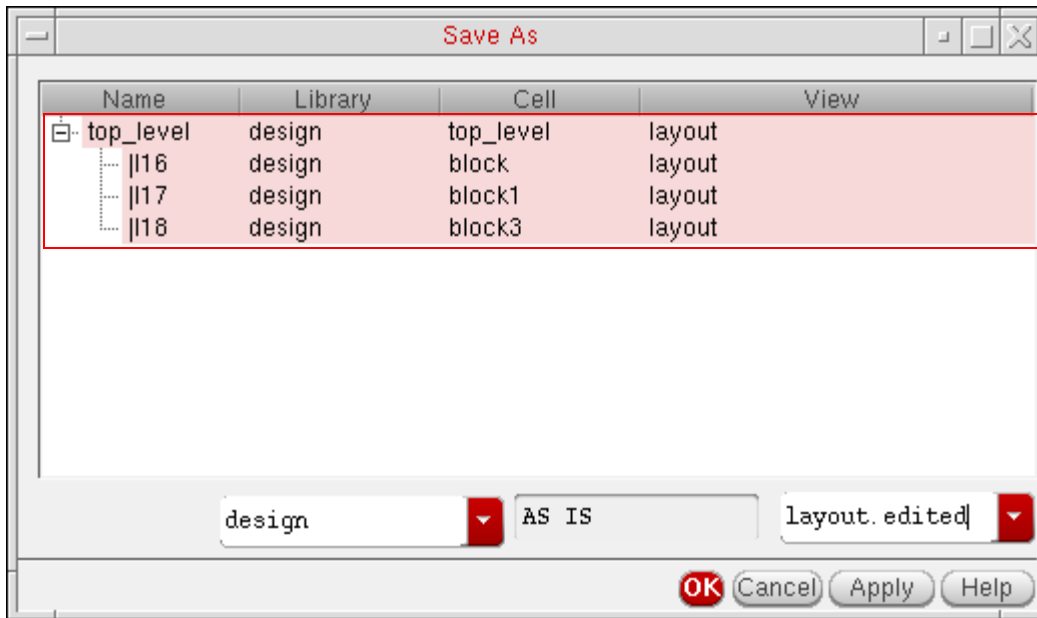
To save cellviews under a new name,

1. From the layout window menu bar, choose *File – Save As*.

Virtuoso Layout Suite XL User Guide

Editing the Layout

The Save As form is displayed, listing all the cellviews in the design hierarchically.



Name	Library	Cell	View
[-] top_level	design	top_level	layout
[-] I16	design	block	layout
[-] I17	design	block1	layout
[-] I18	design	block3	layout

design AS IS layout.edited

OK Cancel Apply Help

2. Select one or more rows from the list in the table.

The fields at the bottom of the form are populated based on the selection in the table. If multiple rows are selected, then the field displays the value if all the values in the selected rows are same. Otherwise it shows *AS IS*.

3. Type new *Library*, *Cell*, and *View* names you want to use. In the example above, the library and cell names are unchanged, but the view name for all the cells has been changed from *layout* to *layout.edited*.
4. Click *OK* or *Apply* to save the selected cellviews.



Tip

For general information on saving cellviews under a different name, see [Layout Editor L Basics](#) in the *Virtuoso Layout Suite L User Guide*.

Soft Block Pins

After you have configured the physical hierarchy, it might be necessary to create additional soft pins or feedthrough terminal block pins on your soft blocks to improve the connectivity of your design. The *Create – Soft Pins* command lets you create, delete, and update soft pins for your soft blocks. The *Create – Feedthrough Terminal Pins* lets you create feedthrough pins.



Tip

Both *Create Soft Pins* and *Create Feedthrough Terminal Pins* commands create strong pins. To create weak pins, use the *Create Pin* command instead.

Creating a Soft Pin

To create a soft pin:

1. In the layout canvas, select the soft block for which the soft pin is to be created.
2. Click *Create – Soft Pins*.

Virtuoso Layout Suite XL User Guide

Editing the Layout

The Add Soft Pin form is displayed.

Add Soft Pin

Soft block list: ether.top_1_port_pnr

Lib: ether

Cell: top_1_port_pnr

View: layout

Use: All Pins

Name: ==

Filter

Terminal name: ADC<0>

Name	Layer	Width	Height	Side
ADC<0>				

Name: ADC<0>.2

Layer: Bondpad dg

Width: 1.00

Height: 1.00

Side: None

Create Strong Pin Update Delete

Left Right Top Bottom

Rectangular and rectilinear shapes depicting their side constraints

OK Cancel Apply Help

Note: If you selected multiple blocks before you launched the form, all the selected

blocks are included in the soft block list at the top of the form. Select the one you want from the list.

3. Choose the *Terminal name* for which you want to create a soft pin from the drop-down list, which by default contains all the terminals in the currently selected block.

Note: You can filter the terminal names that are displayed by typing in the text box and filtering either *All Pins* or *Pins in Selected Set*.

To assist you, the canvas shows flight lines to represent the nets between the terminals you select.

The software automatically assigns a name for the new pin.

4. Choose the pin *Layer* purpose and specify the *Width* and *Height* of the pin in the fields provided.
5. Define the side constraints using an appropriate option from the *Side* drop-down list.

Note:

- ☐ You can assign a side (Left, Right, Top, or Bottom) constraint for each pin figure from the *Side* drop-down. This creates an alignment (side) constraint, which you can view in the Constraint Manager. The alignment constraint, so created, is honored by the pin optimizer.
- ☐ Selecting a value **None** from the *Side* drop-down implies that the pin figure is not constrained to any particular side.

6. Click *Create Strong Pin* to create a new pin.

The soft pin is created in the specified block.

Examples

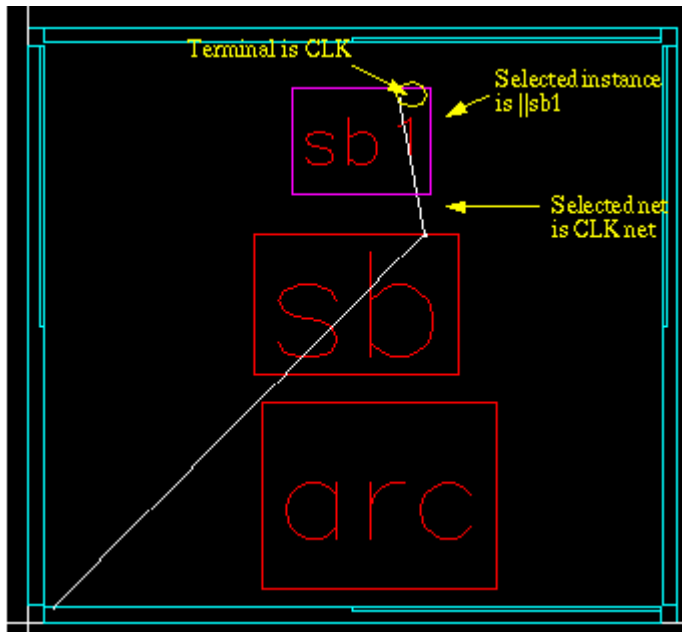
You want to create a soft pin for an existing terminal on a number of soft block instances.

To do this, select the soft block instances and the name of the existing terminal on the selected set to create an extra soft pin for the selected terminal on each instance in the selected set.

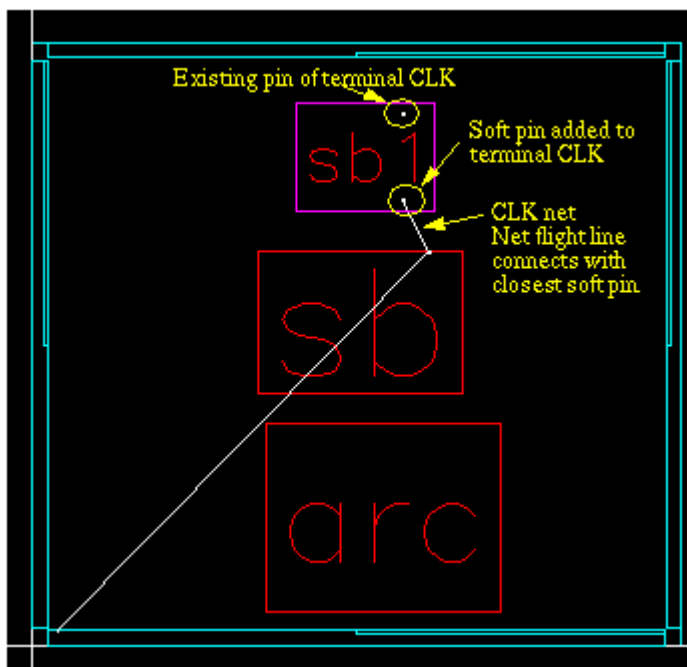
Virtuoso Layout Suite XL User Guide

Editing the Layout

Now you want to add a pin for the CLK terminal on instance | |sb1.



After creating another pin for the existing terminal CLK on the instance | |sb1, the CLK net changes as shown below.



Updating Parameters for a Soft Pin

To update the parameters for one or more soft pins,

1. In the layout canvas, select the soft block for which the soft pin is to be created.
2. Click *Create – Soft Pins*.
The Add Soft Pin form is displayed.
3. Select the pins you want to update from the list.
4. Use the controls provided to change the *Layer purpose* and specify the *Width* and *Height* of the pin, as required.
5. Define the side constraints using an appropriate option from the *Side* drop-down list.

Note:

- ☐ You can assign a side (Left, Right, Top, or Bottom) constraint for each pin figure from the *Side* drop-down. This creates an alignment (side) constraint, which you can view in the Constraint Manager. The alignment constraint, so created, is honored by the pin optimizer.
- ☐ Selecting a value **None** from the *Side* drop-down implies that the pin figure is not constrained to any particular side.

6. Click *Update*.

The selected pins are updated.

Deleting a Soft Pin

To delete one or more soft pins,

1. In the layout canvas, select the soft block for which the soft pin is to be created.
2. Click *Create – Soft Pins*.
The Add Soft Pin form is displayed.
3. Select the pins you want to delete from the list.
4. Click *Delete*.

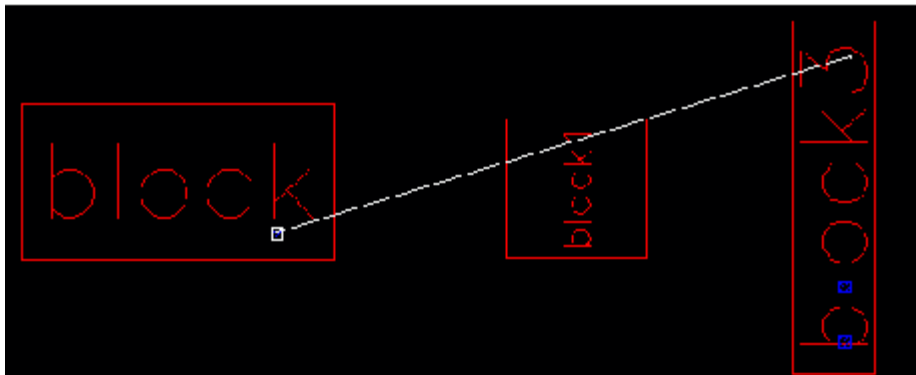
The selected pins are deleted.

Creating a Feedthrough Terminal Pin

To create a feedthrough terminal pin,

1. In the layout canvas, select the soft block for which the pin is to be created.

In this example, we select `block1` as shown in the picture below.



2. Click *Create – Feed Through Terminal Pins*.

Virtuoso Layout Suite XL User Guide

Editing the Layout

The Create Feed Through Terminal Block Pins form is displayed.

The dialog box is titled "Create Feed Through Terminal Block Pins". It contains the following sections:

- Soft block list:** A list box showing "ether.top_1_port_pnr.layout". To the right, fields for "Lib" (ether), "Cell" (top_1_port_pnr), and "View" (layout) are displayed.
- Select Net/Bundle:** A section with a "Search for" dropdown set to "Nets", a "Name" field with an equals sign, and a "Filter" button.
- Net/Bus/NetClass:** A list box containing "TRISTATE", "SYMBOL_MODE", "ANLG_TST", "CRS", and "LEDD".
- Define TTB pin parameter:** A section with a "User defined terminal name" text field.
- Table:** A table with columns "Layer", "Width", "Height", "Num", and "Side". It is currently empty.
- Form fields:** Below the table, there are input fields for "Layer:" (set to "Bondpad dg"), "Width:" (1.00), "Height:" (1.00), "Num:" (1), and "Side:" (None).
- Buttons:** "Add", "Update", and "Delete" buttons are located below the form fields.
- Align Pin:** A checkbox that is currently unchecked.
- Visual Aids:** Four icons labeled "Left", "Right", "Top", and "Bottom" showing rectangular and rectilinear shapes with their respective side constraints.
- Footer:** "OK", "Cancel", "Apply", and "Help" buttons.

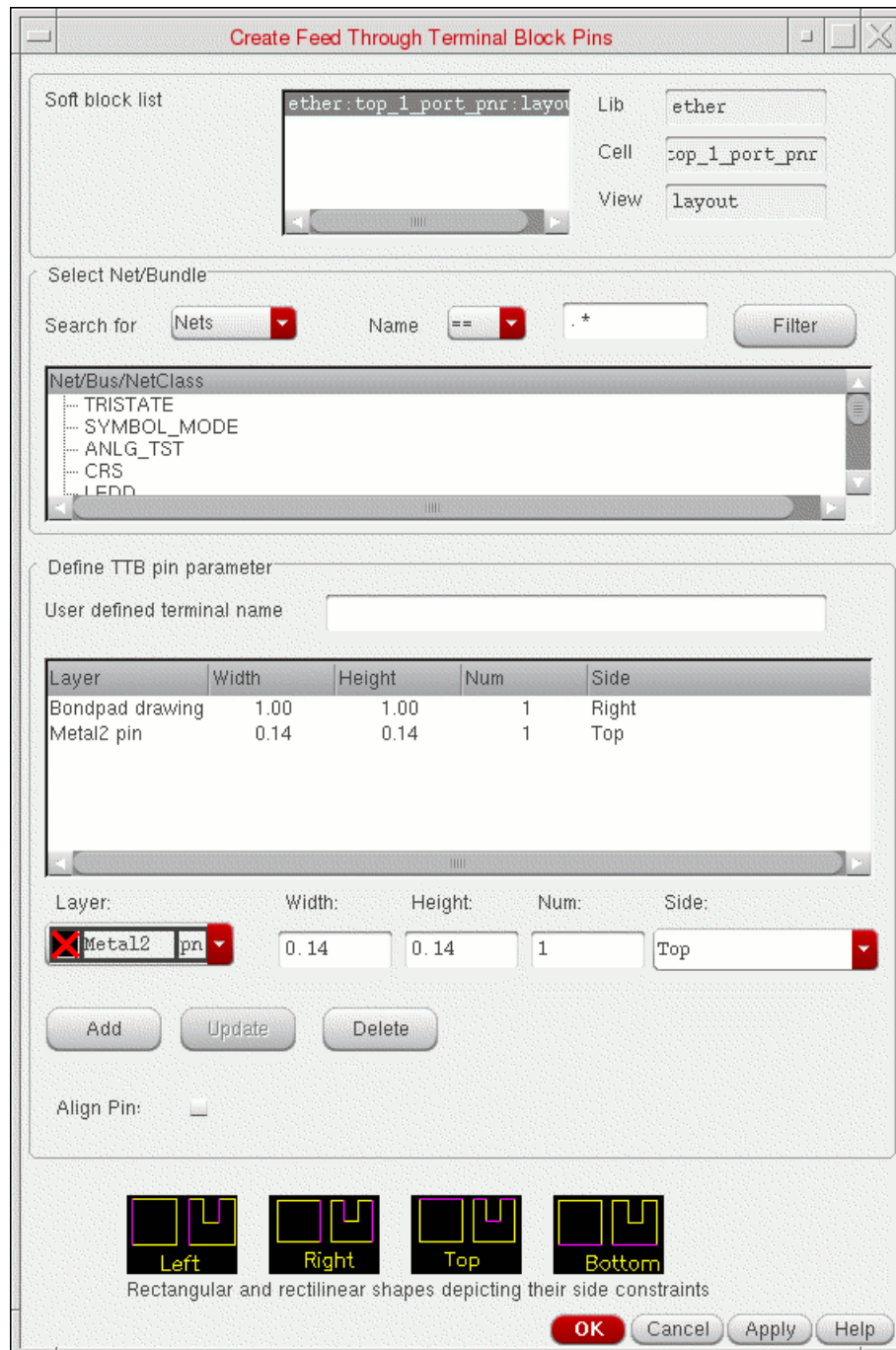
Note: If you selected multiple blocks before you launched the form, all the selected blocks are included in the soft block list at the top of the form. Select the one you want from the list.

Virtuoso Layout Suite XL User Guide

Editing the Layout

3. From the net table, select the net for which you want to create the feedthrough pin.

The system-generated terminal name is displayed as shown in picture below. The canvas shows flight lines to represent the nets between the terminals you select.



4. Choose the *Layer purpose* and specify the *Width* and *Height* of the pin in the fields provided.
5. Define the side constraints using an appropriate option from the *Side* drop-down list.

Note:

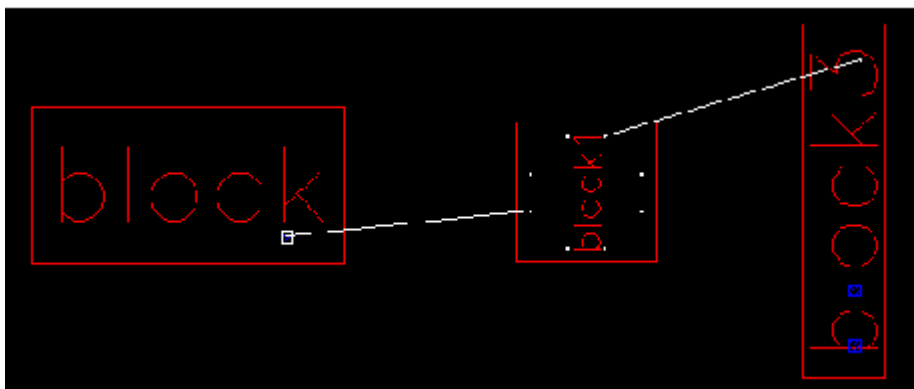
- ☐ You can assign a side (Left, Right, Top, or Bottom) constraint for each pin figure from the *Side* drop-down. This creates an alignment (side) constraint, which you can view in the Constraint Manager. The alignment constraint, so created, is honored by the pin optimizer.
- ☐ Selecting a value **None** from the *Side* drop-down implies that the pin figure is not constrained to any particular side.

6. Click *Add*.

The feedthrough pin parameters are added to the table as shown in the picture below.

7. Click *OK*.

The design view changes to show the feedthrough pin as shown in the picture below.



A new terminal `ftTerm_D` is created in soft block `block1` with two pins, Net `D` is broken into two parts; one part connecting `block` and `block1` and another connecting `block1` and `block3`.

Creating Feedthrough Terminal Pins for ordered nets

The steps to create feedthrough terminal pins for a group of ordered nets, which are grouped by Net Class or Bus constraints, are similar to the steps for creating a feedthrough terminal pin for a single net. The only difference is that you need to select a group of ordered nets in step 3 above.

Updating Parameters for a Feedthrough Terminal Pin

To update a parameter,

1. In the layout canvas, select the soft block for which the pin is to be created.
2. Click *Create – Feed Through Terminal Pins*.
The Create Feed Through Terminal Block Pins form is displayed.
3. Select the entry that you want to update in the table.
4. Change the values in the *Layer*, *Width*, *Height*, *Num*, and *Side* fields as required.
5. Click *Update*.

The parameter is updated in the table.

Deleting a Feedthrough Terminal Pin

To delete a pin,

1. In the layout canvas, select the soft block for which the pin is to be created.
2. Click *Create – Feed Through Terminal Pins*.
The Create Feed Through Terminal Block Pins form is displayed.
3. Select the entries that you want to delete in the table.
4. Click *Delete*.

Snapping Soft Block Pins to Grid

The *Move*, *Reshape*, *Stretch*, and *Chop* commands have options that let you snap boundaries and pins automatically to grid during interactive editing. For more information, see

- Moving an Object in Level-1 Editing Mode on page 387
- Stretching an Object in Level-1 Editing Mode on page 390
- Reshaping an Object in Level-1 Editing Mode on page 398
- Chopping an Object in Level-1 Editing Mode on page 400

Virtuoso Layout Suite XL User Guide

Editing the Layout

To snap pins without accessing these commands, use the *Connectivity – Pins – Snap Pins* command. This snaps both top-level and level-1 pins to the grid appropriate to the block type you are editing.

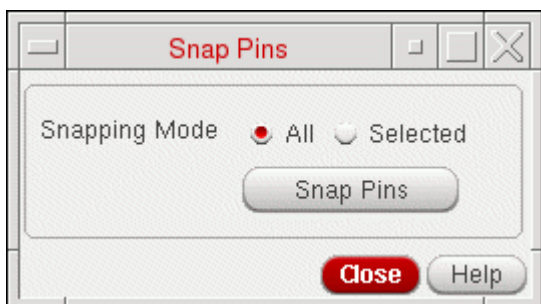
Block Type	Snap Behavior
<i>Analog</i>	Pin edges are snapped to the manufacturing grid.
<i>Digital</i>	Pin centers are snapped to the routing grid defined for the metal layer on which the pins are defined. Note: If the pins are not on a metal layer, they are snapped to the manufacturing grid.

Snapping All Pins to Grid

To snap all soft block pins to the appropriate grid,

1. From the layout window menu bar, choose *Connectivity – Pins – Snap Pins*.

The Snap Pins dialog appears.



2. Ensure that *Snapping Mode* is set to *All*.
3. Click *Snap Pins*.

All the top-level and level-1 pins are snapped to the appropriate grid.

In IC6.1.5 release, the pins with placement status as *Fixed* are also snapped and the status is retained as *Fixed*.

Snapping Selected Pins to Grid

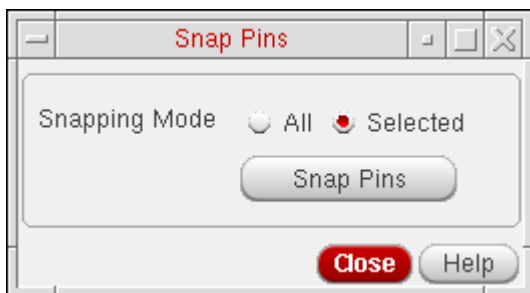
Snap Pins can also snap selected top-level and level-1 pins to grid. The command works with both pre- and postselected pins.

To snap preselected pins to grid,

1. Select the pins you want to snap in the layout window or Navigator assistant.
2. From the layout window menu bar, choose *Connectivity – Pins – Snap Pins*.

The Snap Pins dialog appears.

3. Ensure that *Snapping Mode* is set to *Selected*.



4. Click *Snap Pins*.

The selected top-level and level-1 pins are snapped to the appropriate grid.

To snap postselected pins to grid,

1. From the layout window menu bar, choose *Connectivity – Pins – Snap Pins*.

The Snap Pins dialog appears.

2. Ensure that *Snapping Mode* is set to *Selected*.
3. Select the pins you want to snap in the layout window or Navigator assistant.
4. Click *Snap Pins*.

The selected top-level and level-1 pins are snapped to the appropriate grid.

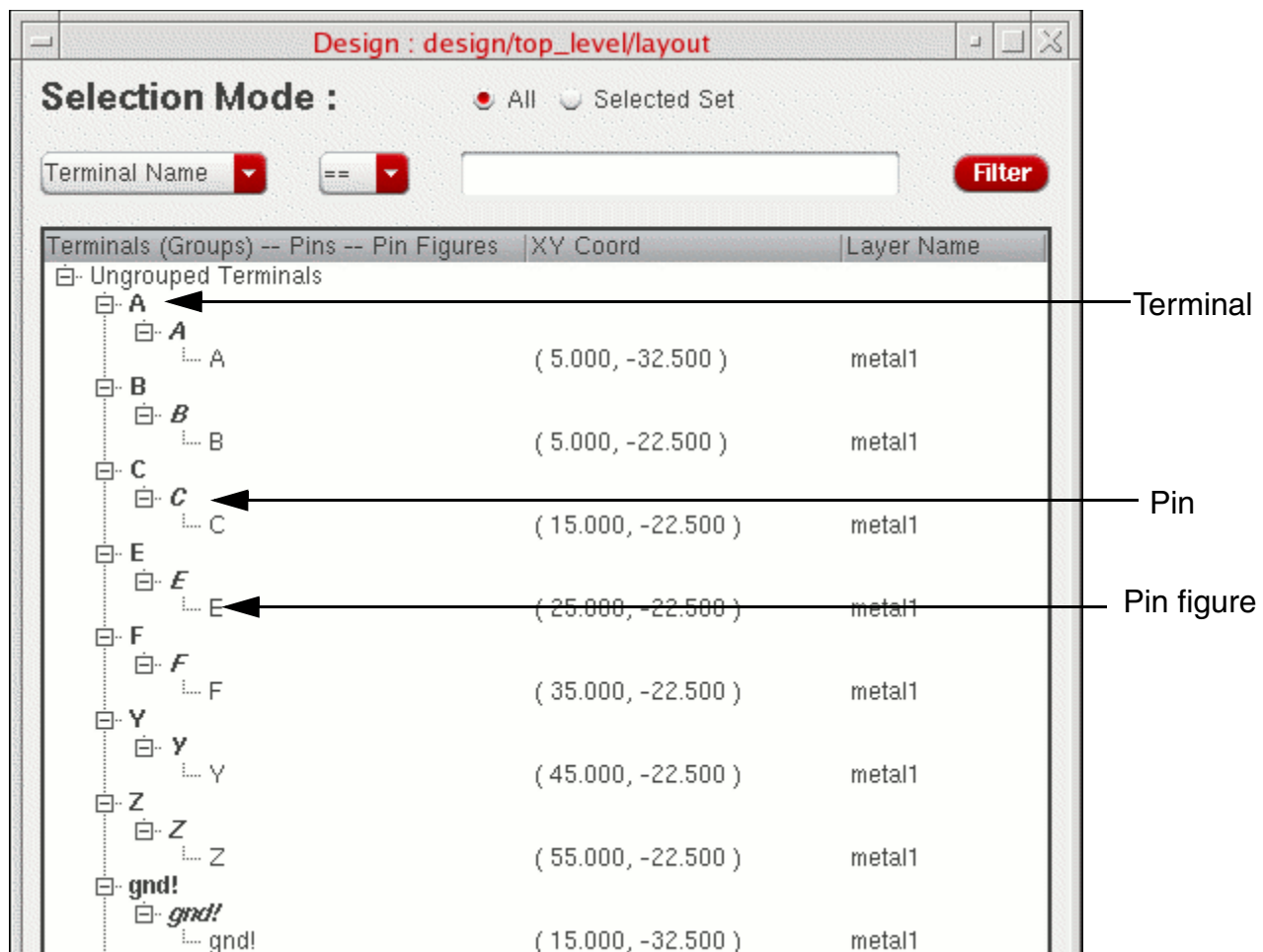
Note: In the *Selected* mode, when the pins with placement status as *Fixed* are snapped, the status of the pins is retained as *Fixed*.

Setting the Pin Connectivity Model

You can create and edit the connectivity models for the pins in your design using the Pin Connectivity Model dialog. This form lets you create, change and delete strong, weak, and must-connect pin connectivity settings.

- [Creating a Strong Connect Group](#) on page 424
- [Creating Separate Weak Connect Groups](#) on page 424
- [Creating a Single Weak Connect Group](#) on page 425
- [Creating a Must Connect Group](#) on page 425
- [Deleting a Must Connect Group](#) on page 425

You invoke the Pin Connectivity Model form using the *Connectivity – Pins – Pin Connectivity Setting* command.



The form lists all the terminals, pins, and pin figures in a tree structure. The *Expand All* and *Collapse All* buttons in the window expand and close the tree structure respectively.

- The first level in the tree structure lists the terminals in the design in bold, normal type; for example, **B, C, D, E, F, Y, Z, A, gnd!**, and **vdd!**
- When you expand the first level, the pins in each terminal are displayed in bold, italic type; for example, ***B, C, D, E, F, Y, Z, A, gnd!***, and ***vdd!***.
- When you expand the pins, the pin figures are displayed in normal type; for example, B, C, D, E, F, Y, Z, A, gnd!, and vdd!.

You can create and edit the connectivity model for *All* the pins in the design or for only those pins that were *Selected* when you opened the form. The form also provides a filter that you can use to display specific terminals and pins.



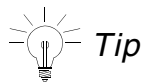
For a video demonstration on setting the pin connectivity model, see the [Pin Connectivity Model](#) video on SourceLink.

Creating a Strong Connect Group

You can create a strong-connect group on pins only. To create a strong-connect group,

1. Select the individual pins to be strongly connected.
2. Click the right mouse button and choose *Make Strong Connect Group*.

The pins are placed in a strong connect group.



You can also drag-and-drop a pin on another to create a strong connect group.

Creating Separate Weak Connect Groups

You can create a separate weak connect group on pin figures only. To create a separate weak connect group,

1. Select the pin figures in a strongly-connected group.
2. Click the right mouse button and choose click *Make Separate Weak Connect Group*.

The pins are weakly connected and placed in separate groups.

Creating a Single Weak Connect Group

You can create a single weak connect group on pin figures only. To create a separate weak connect group,

1. Select the pin figures in a strongly connect group.
2. Click the right mouse button and choose *Make Single Weak Connect Group*.

The pins are weakly connected and placed in single groups.

Creating a Must Connect Group

You can create a must connect group on terminals only. To create a must connect group,

1. Select the terminals that must be connected.
2. Click the right mouse button and choose *Make Must Connect Group*.

The terminals are connected and are placed together in the must connect group.

Deleting a Must Connect Group

To delete a must connect group,

1. Select a terminal in the group.
2. Click the right mouse button and choose *Delete Must Connect Group*.

Virtuoso Layout Suite XL User Guide

Editing the Layout

Preparing Your Design for Routing

This chapter explains how the Virtuoso® Layout Suite XL layout editor (Layout XL) processes connectivity information and presents several ways to prepare your design for routing.

This chapter discusses the following topics.

- [Connectivity Extraction](#) on page 428
- [Connectivity Extraction from Substrate and Well Layers](#) on page 442
- [Pseudoparallel Connections](#) on page 464
- [Working With Nets](#) on page 475
- [Layer Selection](#) on page 493
- [Flight Lines](#) on page 496
- [Markers](#) on page 497

For information about how to use the Virtuoso Chip Assembly Router to route your design, see the [Virtuoso Chip Assembly Router Guide](#).

Connectivity Extraction

The Layout XL connectivity extractor verifies that the connectivity of your physical implementation is legal and generates violation markers indicating open and short circuits, weak and must connect violations, and illegal overlaps in the design.

You can specify the types of markers that are created and limit the number of open violations displayed using the controls on the [Extraction Tab](#) form. You can view and manage the violation markers in the *Connectivity* tab of the [Annotation Browser](#) assistant.

This section discusses the following topics:

- [Connectivity Extraction Use Model](#)
- [Connections Recognized by the Extractor](#)
- [Extracting a Top-Level Design](#)
- [Extracting a Top-Level Design](#)
- [Extracting Hierarchical Cellviews](#)
- [Enabling and Disabling Incremental Extraction](#)
- [Avoiding False Overlap and Short Markers](#)
- [Assigning Connectivity to a Via](#)

Connectivity Extraction Use Model

Connectivity extraction is not run automatically by default when you open a design in Layout XL. If you are opening the design for the first time, you must run the extractor explicitly using the *Connectivity – Update – Extract Layout* command to generate an up-to-date connectivity model for the software to work from. The same applies if you have edited your design in a non-connectivity-aware environment such as Layout L.

At times, working in the non-connectivity aware environment is recommended for faster processing. This is when you may be working on very large designs, such as 32bit or 64bit designs. In fact, the tool evaluates your system's capability for processing such designs in the connectivity mode, and displays an appropriate message.

For example, when working on a 32bit design, the tool evaluates the virtual memory requirements for booting up the connectivity extractor and automatically shifts to a non-connectivity mode if the available virtual memory is less than the required. The following information message stating the shift to the non-connectivity aware mode is also displayed for the user:

Virtuoso Layout Suite XL User Guide

Preparing Your Design for Routing

The estimated memory needed to launch XL tier in connectivity mode is above what can be provided by the system. To be able to launch XL on this design, the connectivity extractor is going to be automatically disabled.

You are required to click the **OK** button to acknowledge the message and proceed.

Likewise, when working on a 64bit design, the tool displays the following message prompting you to choose the mode you would like to work in.

The estimated memory needed to launch XL tier in connectivity mode is very high. It could also require a long runtime to finish. Do you want to disable the connectivity extractor and launch XL in connectivity-less mode?

Depending on your preferences, you can choose **Yes** or **No** on the message pop up and proceed.

By default, Layout XL updates the connectivity and generates violation markers automatically when you create, delete, or otherwise modify the shapes in your design either manually or using the automatic tools. Changes to terminals and nets also cause the connectivity to be updated, including changing the name of an existing terminal in the Property Editor assistant or Edit Properties form. This incremental extraction capability is controlled by the *Update connectivity when design is modified* option on the [Extraction Tab](#) form.

For more information, see [Enabling and Disabling Incremental Extraction](#).

For more information on extracting the connectivity of a top-level design, see [Extracting a Top-Level Design](#).

The connectivity model and any violation markers generated are saved to the database when you close the current Layout XL session and are re-used the next time you open the design in Layout XL. Unless you edit your design outside of Layout XL, you do not need to run extraction again when you re-open a design.



Tip

To verify that all connections are made in the layout as specified in the schematic, use an Assura[®] verification product, a Dracula[®] verification tool, or an Assura hierarchical physical verification product to run a layout versus schematic (LVS) check on the completed design.

Hierarchical Connectivity Extraction

Top-level extraction considers the connectivity of objects and generates markers only at the top level of the design. To verify the connectivity for the entire hierarchy of your design, or for

selected lower-level cellviews, enable the *Extract Hierarchical Cellviews* on the Extract Layout form.

For more information on hierarchical extraction, see [Extracting Hierarchical Cellviews](#).

Automatic Connectivity Extraction

Some of the automatic placement and routing commands, such as *Analog Automatic Placement*, run connectivity extraction automatically as part of their normal processing.

Generate All From Source has an option to run connectivity extraction automatically after the components have been generated.

The Layout XL *Update Components And Nets* command only partially re-extracts a design. It corrects the connectivity of the changed instances in the design and re-extracts only those instances and objects connecting to them. It does not perform a full re-extraction.

You can use the [extractRunOnInit](#) environment variable to run connectivity extraction automatically when you launch Layout XL. However, this means that extraction is run even if it is not required and can adversely affect the performance of Layout XL during startup.



Tip

Cadence recommends that you do not enable this environment variable by default, but that you use the *Connectivity – Update – Extract Layout* command when you know it is required.

Connections Recognized by the Extractor

Layout XL recognizes connectivity only if all the shapes on a net have connectivity information. Otherwise, it reports opens (even though the router might report that the design is 100% routed).

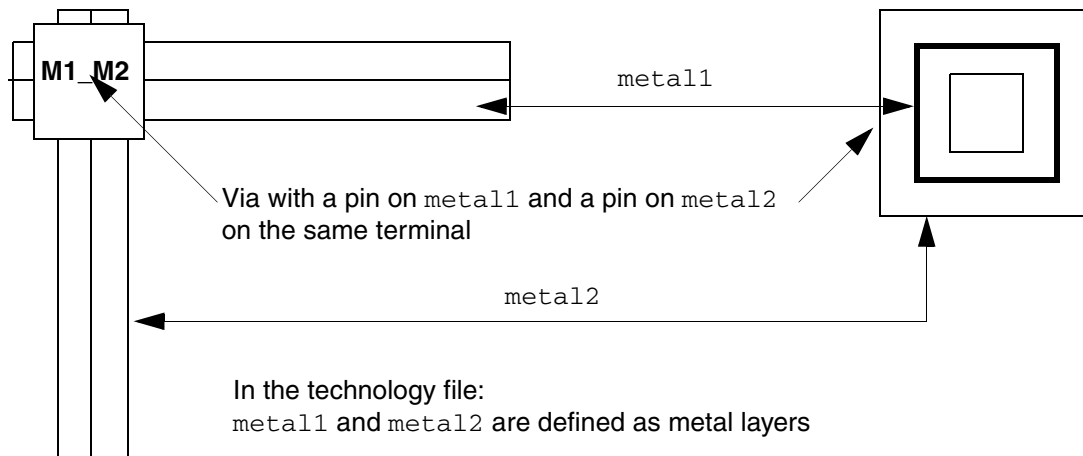
When extracting connectivity, Layout XL accepts connections:

- On layers that are listed in the connectivity stack as extractable layers. For more information, see [Connectivity Rules](#).
- Between two touching shapes or instance pins on the same layer.
- On microwave objects. For more information, see [Microwave Commands](#).

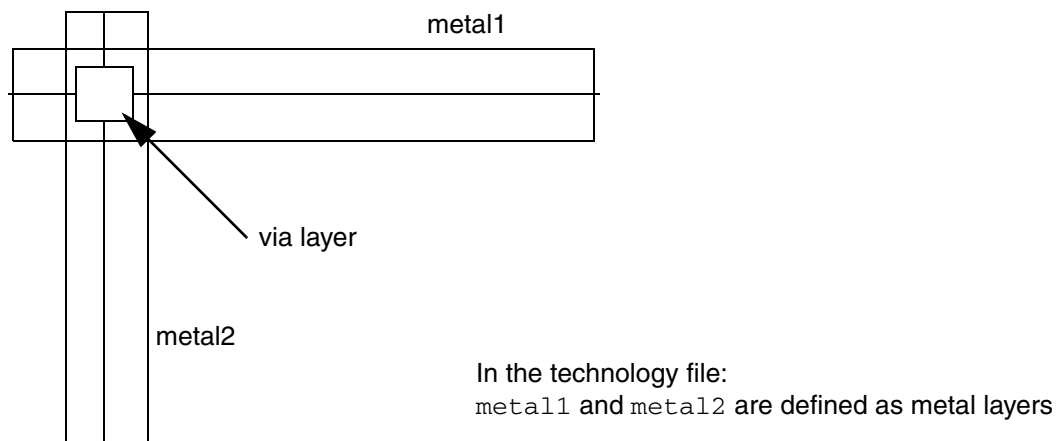
Virtuoso Layout Suite XL User Guide

Preparing Your Design for Routing

- Between two touching shapes or instance pins on different layers if the layers have been defined as equivalent layers or one of them is defined as a via layer. For more information, see [Connectivity Rules](#).
- Between two layers by means of a contact instance having a pin on each layer and both pins on the same terminal.



- Between two layers by means of a via. For more information, see [Valid Vias](#) and [Via Definitions](#).



Important

For performance reasons, Layout XL does not create shorts involving unassigned shapes in the hierarchy. It does create shorts for shapes that are assigned to a net, even if the shape in question has no terminal.

Note: If a net has no connectivity information and does not own a figure, an instTerm, a pin, a terminal, or a non-empty constraint; the net is called “empty” and can be deleted by the extractor. To determine if the current net is empty, the extractor also considers equivalent nets of the current net.

Instance Terminal Connectivity

The connectivity of an instance terminal can come from the schematic (if the corresponding schematic instance has connectivity) or from the first overlap detected in the layout. Once assigned, Layout XL considers the connectivity of an instance terminal as *sticky*, which means that when an instance terminal is assigned to a net, it stays assigned to that net unless you change it using, for example, the *Propagate Nets* command.

For example, assume you have an instance, I2, with no connectivity in Layout XL. When you move this instance to overlap a shape assigned to net2, Layout XL creates an instance terminal for I2 and assigns it to net2. If you now move I2 to overlap a shape on net1, a short is created.

The following exceptions apply.

- Where an instance terminal is assigned to a net with a must-join terminal at the current level, the extractor does not consider the instance terminal sticky. However, if the instance terminal is moved away from an overlap, it keeps its connectivity until the extractor finds an overlap with a different must-join net which is better suited for the connection. If such an overlap is found, the instance terminal is reassigned to the new net.

This behavior has been implemented to support the must-join pin model at different levels of hierarchy.

- When you move a physical-only instance (i.e., an instance for which the `lvIgnore` property is set to `t`) in the design canvas, Layout XL automatically reassigns the connectivity based on any new overlap. The only exception to this is where the physical-only instance has the `lxStickyNet` property set to `t`, in which case the extractor creates a short marker if the new connectivity does not match the existing connectivity.

If there is no new overlap after the move and the physical-only instance has the `lxStickyNet` property set to `t`, the existing connectivity is retained. If the

`lxStickyNet` property is not set to `t`, the existing connectivity is deleted.

Support for Diffusion Layers

Some manufacturing processes define diffusion shapes that must be explicitly implanted by n-type or p-type implant shapes to get their own type. If the implant is p-type, the diffusion shape is considered to be p-type. If the implant is n-type, the diffusion shape is considered to be n-type.

Other processes derive the implant type from the definition of the diffusion layer, “ndiff” or “pdiff”, in the technology file. This is the case when an explicit implant does not exist. The connectivity extractor supports both types of diffusion layers.

However, if the diffusion shape is not overlapped by an implant and the diffusion type is not defined in the technology file, the diffusion shape is ignored for extraction.

Note: Implant shapes never get any connectivity; they simply allow the diffusion to connect to wells and cuts.

- Analog processes typically feature a single diffusion layer and two or more layers for implants. The extractor recognizes shapes on layers with function `diff` as unimplanted diffusion: they need an explicit ‘n’ or ‘p’ implant shape to connect to the appropriate `pwell` or `nwell`.

The `functions` table for such a process might include the following definitions.

```
functions (
; ( layer          function          [maskNumber])
; ( -----          -)
; ( pwell          "pwell"          10          )
; ( nwell          "nwell"          20          )
; ( diff           "diff"           30          )
; ( poly           "poly"           40          )
; ( pimp           "pimp"           50          )
; ( nimp           "nimp"           60          )
);functions
```

Only `diff` diffusion is defined, so there are also `pimp` and `nimp` layers defined explicitly for the implants. A shape on `diff` must overlap with a shape on `pimp` in order to connect to a shape on `pwell` or with a shape on `nimp` in order to connect to a shape on `nwell`. The connectivity is derived as follows:

```
metal -> cut -> (diff & nimp) -> nwell or n-type substrate
metal -> cut -> (diff & pimp) -> pwell or p-type substrate
```

- Manufacturing processes used in digital design often define separate layers for n and p diffusion. A shape on these layers is considered to have an *implicit implant* whose type is derived from the layer function.

Here, the `functions` table might include the following definitions.

```
functions (
; ( layer          function          [maskNumber])
```

Virtuoso Layout Suite XL User Guide

Preparing Your Design for Routing

```
; ( -----)
( lv_nwell      "nwell"      10      )
( hv_nwell      "nwell"      20      )
( hv_pwell      "pwell"      30      )
( ndiff         "ndiff"      40      )
( ndiff_hv      "ndiff"      50      )
( pdiff         "pdiff"      60      )
( odiff_hv      "pdiff"      70      )
( poly          "poly"       80      )
);functions
```

There are no layers defined for implants, the extractor differentiates between p-type and n-type implants based on whether the function is `pdiff` or `ndiff`. An `ndiff` shape connects to shapes that have the `nwell` function; a `pdiff` shape connects to shapes that have the `pwell` or `substrate` function.

The connectivity is derived as follows:

```
metal -> cut -> ndiff -> nwell or n-type substrate
```

```
Metal -> cut -> pdiff -> pwell or p-type substrate
```

- Some processes have a single active diffusion layer, such as “`ndiff`”, which is used to connect to both types of wells and substrates. In addition, the process might have a single implant, such as “`pimp`”, which takes priority over the diffusion type.

The functions table for such a process might include the following definitions:

```
functions (
; ( layer      function      [maskNumber])
; ( -----)
( M1          "metal"       10      )
( CO          "cut"         20      )
( NOD         "ndiff"       30      )
( PP          "pimp"        40      )
( NW          "nwell"       50      )
( PW          "pwell"       60      )
);functions
```

A single diffusion layer, `ndiff`, and a single implant, `pimp`, is defined. An `ndiff` shape connects to shapes that have the `nwell` function. An `ndiff` shape when overlapped by a `pimp` shape connects to shapes that have the `pwell` function.

The connectivity is derived as follows:

```
metal -> cut -> ndiff -> nwell or n-type substrate
```

```
metal -> cut -> (ndiff & pimp) -> pwell or p-type substrate
```

Weakly-Connected Instance Pins

The extractor engine also considers the external loops of current that potentially exist in the connections made to weakly-connected instance pins. This means it is legal to connect two

weakly-connected instance pin figures with the same top-level net, provided the connections on these pins are made with top-level shapes that are, in turn, connected. If this is the case, the extractor does not create a marker because it is assumed that the current will not flow through the resistive path between the two pin figures inside the instance.

When the extractor detects an illegal weak-connect violation, the generated marker is a line marker drawn between the two instance pin figures that are overlapped by unconnected top-level shapes on the same net. This provides better visibility of the problem and eases the process of fixing it. The [Annotation Browser](#) highlights both the line marker and the two instance pin figures.

Incremental updates made by the user are also considered for the detection of illegal weak-connect violations. For example, if a shape on a given net is deleted anywhere in the design, the weak markers are recalculated on the instances connected to that net.

Legacy Contact Devices Converted from CDB

When a contact is converted from CDB to an `oaVia` object (standard or custom), its “stickiness” is derived from the `lxStickyNet` property (i.e., it behaves in the same way as a regular shape). If the via is part of a route, the `lxStickyNet` property is checked on the route too.

In some circumstances, a CDB contact can be converted to a regular instance. Layout XL recognizes such devices as vias using the `function` property, which can be set on the instance, its master, or in CDF.

- If `function` is set to `contact`, the stickiness of the instance is derived from `lxStickyNet`.
- If `function` is set to `via` or `substrateContact`, the contact is sticky by construction, meaning that the connectivity of the contact is the net attached to the instance terminal. When the contact is not sticky, it can be the case that an instance terminal is reassigned dynamically based on the overlaps (exactly like a non-sticky shape).

Unmatched Instance Terminals

When loading designs that have been edited outside Layout XL, the connectivity extractor silently cleans up any instance terminals that have no associated terminal in the cell master. This can mean that you are asked to save your design, even though nothing has visibly changed.

Extracting a Top-Level Design

When extracting a top-level design, Layout XL extracts

- Top-level objects—shapes, vias, and pin figures.
- Pin figures of top-level instantiated masters. These pin figures represent the interface between the top-level instances and the lower level instances in the design and are, therefore, considered to be at the top level. The connectivity of these pin figures in the context of top level extraction is represented by the nets attached to the instance terminals corresponding to these pin figures.

To extract the current top-level design;

1. From the CIW, open a layout cellview and choose *Connectivity – Update – Extract Layout*.

The Extract Layout form is displayed.



Extraction Stop Level

The `extractStopLevel` environment variable controls the **Extract connectivity to level** option on the *Extraction* tab, increasing the extractor's visibility into the design hierarchy.

For example, an overlap between a top-level shape and a hierarchical shape at level "N" is processed by the extractor only if the extraction stop level is equal to or greater than "N". However, if the hierarchical shape is a pin figure, it is visible from the top level even if the extraction stop level is "N-1".

Likewise, an overlap between a hierarchical shape at level "N" and a hierarchical shape at level "M" in two different hierarchies is processed by the extractor only if the extraction stop level is equal to or greater than the highest of the two levels, "N" and "M".

In the context of top-level extraction, the connectivity of a hierarchical shape is the net attached to the corresponding top-level instance terminal (if any). This top-level connectivity is established by recursively iterating in a bottom-up series from terminal to instance terminal, starting from the net attached to the hierarchical shape.

The greater the `extractStopLevel`, the more accurate is the extraction. For example:

- Top-level nets that are incomplete with `extractStopLevel = 0` may become complete with `extractStopLevel > 0`. For example, a top-level shape on a given net overlaps a hierarchical shape that has its top-level connectivity defined by an instance terminal on the same net. The resolution of the open violation, in this case, happens only if the hierarchical shape is visible, implying that the extraction stop level is sufficient, which is achieved at an extraction stop level greater than 0.
- There is a better detection of shorts between top-level and hierarchical shapes.
- There is a better detection of shorts between different instances.

Note: The extractor attempts to bind the instances that are in the hierarchical depth defined by the extract stop level. By default, the extract stop level is 0. Therefore, the extractor tries to bind all the top-level instances to their corresponding master.

Important

Whatever the `extractStopLevel`, the extractor only ever extracts the current top level; i.e., it only ever changes the connectivity of objects or creates markers at the top level of the design. For information on hierarchical extraction, see [Extracting Hierarchical Cellviews](#).

Caution

The full connectivity extraction of a design is not an undo-able operation because of the potentially high number of modifications made on the database. Therefore, the undo stack is cleared when the command is called. The undo mechanism is enabled for any subsequent editing.

Extracting Hierarchical Cellviews

Hierarchical extraction lets you extract either an entire hierarchical design or selected lower-level cellviews from the current top level of the design.

To perform a hierarchical extraction,

1. From the CIW, open a layout cellview and choose *Connectivity – Update – Extract Layout*.

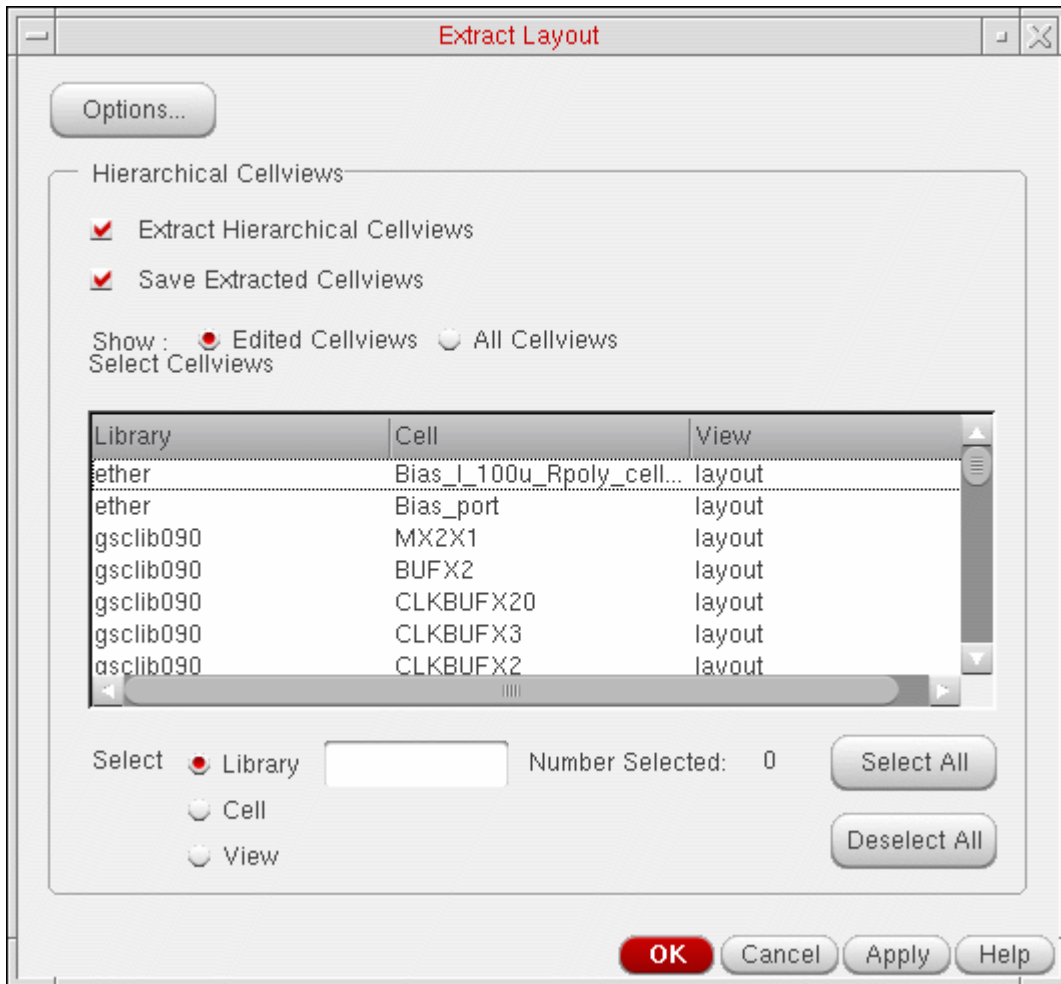
Virtuoso Layout Suite XL User Guide

Preparing Your Design for Routing

The Extract Layout form is displayed.

2. Switch on *Extract Hierarchical Cellviews*.

The hierarchical extraction controls are enabled.



If there are lower level cellviews that need to be re-extracted, they are listed in the *Select Cellviews* pane. You can choose whether to display *All Cellviews* or only those that have been edited since they were last saved.

3. Switch on *Save Extracted Cellviews* to automatically save each cellview after it has been extracted.
4. (Optional) To change the way the extraction is performed, click *Options* to open the Extraction Tab form where you can set the following extraction parameters for the current design.

- ❑ To increase or decrease the extractor's visibility into the hierarchy of the design, change the value of the *Extract connectivity to level* option.

For more information, see [Extracting a Top-Level Design](#).

- ❑ To change the list of extractable layers, choose a different constraint group from the *Derive extractable layers from constraint group* list.

For more information on how extractable layers are derived, see [Specifying Information Required by the Layout XL Connectivity Extractor](#).

- ❑ To show weak-connect and must-connect violations in the design, switch on the *Verify weak-connect violations* and *Verify must-connect violations* options.

- ❑ To show the opens in the design, switch on the *Verify open violations* option.

To improve performance, you can limit the numbers of violations displayed by net and by cellview using the *Maximum number of open violations by net* and *Maximum number of open violations by cellview* options.

You can also include or exclude instance terminals with non-extractable pin shapes using the *Verify unimplemented instance terminals* option.

5. In the *Select Cellviews* pane, choose the cellviews you want to re-extract.

You can filter the list by library name, cell name, and view name or click *Select All* to select all the cellviews listed.

Note: Note that the filter mechanism has no wildcard capability. For example, to select a cell called 'resistor', you must type 'r', 're', 'res', and so on. If you type 'sis', the cell is not selected.

6. Click *OK* or *Apply* to extract the selected lower level cellviews.

The selected cellviews are extracted all the way up to the top level of the design.

Parameterized Cells

The connectivity extractor does not extract pcell masters during hierarchical extraction. For more information on parameterized cells, see the [Virtuoso Parameterized Cell Reference](#).

Enabling and Disabling Incremental Extraction

Incremental extraction automatically updates the connectivity model whenever the design is modified either manually or using the automatic tools. This capability is switched on by default.

To enable and disable incremental extraction,

1. From the layout window menu bar, choose *Options – Layout XL* and click on the *Extraction* tab.
2. Check or uncheck the *Update connectivity when design is modified* option.
Note: The associated environment variable is called `extractEnabled`.
3. Click *OK* or *Apply* in the Layout XL Options form to accept the change.

Avoiding False Overlap and Short Markers

If you are working with a hierarchical design, you can set properties on instances at the lower levels of the design to avoid false overlap and short markers.

- Add the `lxBLOCKOverlapCheck` property to a shape, instance, or instance master.

This property tells the extractor not to check whether a nonoverlap layer of the shape, instance, or instance master is touching a nonoverlap layer at any cellview level.

For example, if you define the `lxBLOCKOverlapCheck` property on a shape in cellview “A”, you will not see any markers when overlapping the shape with an instance.

However, if cellview “A” is placed inside cellview “B”, and a *noOverlap* layer is drawn over the shape in cellview “A”, then a marker is displayed. This happens even though the shape in cellview “A” has the `lxBLOCKOverlapCheck` property defined.

- Add the `lxBLOCKExtractCheck` property to a shape, instance, or instance master.

This property tells the connectivity extractor not to check the connectivity between this shape and others it touches.

Assigning Connectivity to a Via

You can assign connectivity directly to a via in the same way as you assign connectivity to paths, polygons, and rectangles. The `lxBLOCKStickyNet` property is automatically added to the via, which retains the net assignment until you remove the property or override it with a new net assignment.

If you remove the `lxStickyNet` property, Layout XL assigns connectivity to the via based on whichever known nets it connects to.

Important

This behavior differs from previous releases where you were required to define a `function` property with value `substrateContact` on the via master or instance in order for the net assignment to the instance terminal of the via to be sticky. The `function` property is no longer supported.

Vias Implemented as Mosaics

The current version of Layout XL does not support vias and contacts that are implemented as mosaics.

Connectivity Extraction from Substrate and Well Layers

The extractor now fully recognizes soft connections to the substrate and well layers.

- The soft connections are established using tap vias. Tap vias can be defined as connections to the substrate or a well layer on which they are placed.
- The substrate and well layers form an area on which you can place devices. The extractor verifies the connections made to the bulk terminal of these devices.
- The extractor evaluates the soft connections to appropriately propagate the connectivity. The extractor supports connectivity extraction of the bulk terminal of a device even when the bulk is not implemented by a physical shape. This is typically the case when an N-type transistor is placed over a P-type substrate and no physical shape represents the bulk area of the transistor for connectivity extraction.
- The extractor allows the substrate and well layers to be partitioned into discrete areas. This enables users to limit the connectivity extraction to a local substrate or a well area created by a partition.

Before these enhancements were implemented, the connectivity of the substrate was not checked at all and any physical shape on a well layer was considered as metal in terms of connectivity propagation. Likewise, the connectivity of shapeless bulk terminals was not checked.

Now, the extractor checks connectivity for the substrate, well layers, and bulk terminals and subsequently creates appropriate open and short markers in the [Annotation Browser](#).

The following is a list of topics covered in this section:

Soft Connections: Definition

Bulk Area: Definition

Bulk Area: Connectivity Extraction

Connectivity Propagation through the Substrate and Well Areas

Soft Connections: Definition

A soft-connect object can be:

- The entire substrate or an isolated area of the substrate
- A physical shape on a well layer or an isolated portion of the well shape
- A bulk instance terminal

A soft connection is a physical overlap between a soft-connect object and a tap via connecting to it. For a well layer, a soft-connection can also be made directly from a physical layer connecting to it, such as a diffusion shape.

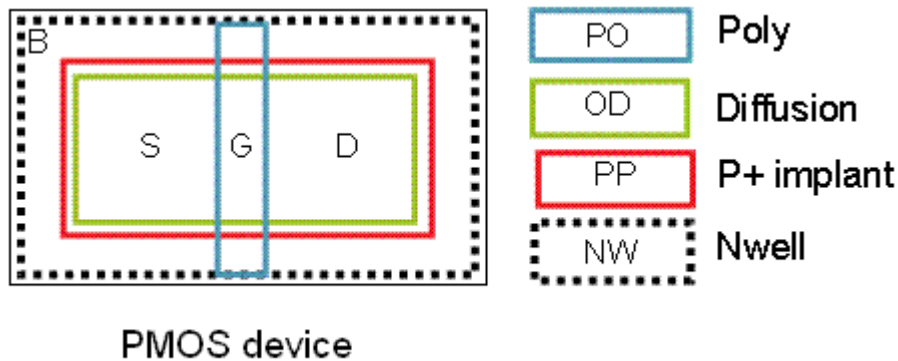
A soft-connect object must be polarized from a tap via or a connecting layer. The polarization in such a connection is unidirectional—from the polarization source to the soft-connect object.

In terms of connectivity checking, it is valid to connect several soft-connect objects together, such as two well shapes.

Note: The extractor does not report soft-connect design errors, such as a floating well shape or substrate area or a layout pattern that forces the current to flow out of a soft-connect object. To detect such errors, tools such as Cadence PVS (Physical Verification System) can be used.

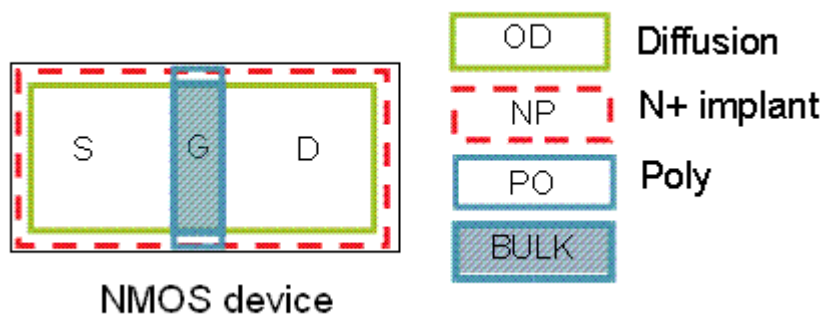
Bulk Area: Definition

The bulk area of a transistor can be defined using a physical and extractable pin figure or using a shapeless terminal. If the bulk is implemented using a physical shape, the corresponding pin figures define the bulk area and the extractor uses the corresponding layer(s) to determine the top-level connections. If the bulk is a shapeless terminal, the bulk area of the device is usually the active layer that lies under its gate, between the source and the drain. The following figure displays the bulk area of a PMOS device, which is a physical shape on the Nwell layer.



Implemented bulk terminal of a PMOS device for connectivity extraction

The following figure displays the bulk area of an NMOS device for a shapeless bulk terminal.



Bulk area of an NMOS device for

When the bulk terminal is shapeless, the corresponding bulk area needs to be derived by the extractor. For more information, see [Identify the Bulk Area](#).

Bulk Area: Connectivity Extraction

Bulk area connectivity extraction means that the extractor recognizes the bulk area as an extractable layer even when the terminal is shapeless.

Earlier, only partial extraction of the bulk area was possible as the extractor could recognize the bulk only when it was represented using a physical shape. Now, the bulk area is considered as “fully” extractable because the bulk is now recognized by the extractor even if it does not have a physical shape drawn.

For the extractor to derive connectivity information from the bulk area, you need to:

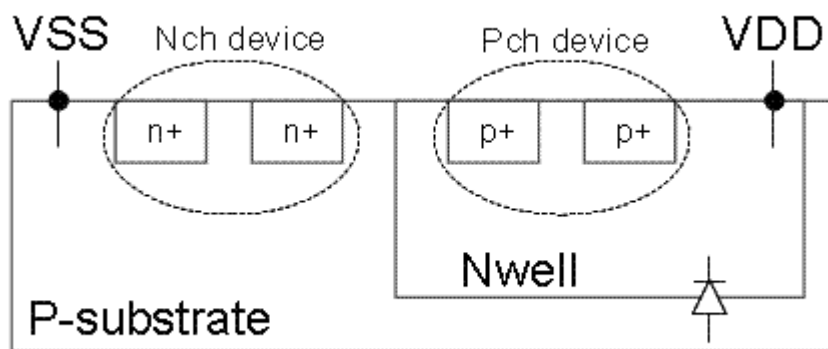
- Identify the Substrate Area
- Isolate the Substrate and Well Layers
- Identify the Bulk Area
- Identify the Substrate and Well Tap Vias

Identify the Substrate Area

Before we discuss why it is important to identify the substrate area for bulk area extraction, let us find out what a substrate means in the context of chip fabrication.

A substrate is the base layer of a chip on which the various devices are placed to form a functional model. The substrate can be N-type or P-type. Depending on the substrate type, devices can be placed directly on the substrate or inside a well shape.

For example, in the figure below, the substrate is P-type. So, the NMOS (Nch) devices are directly placed over the substrate. Conversely, the PMOS (Pch) devices are placed inside an N-type well (Nwell) drawn on the substrate. The P-substrate and the Nwell shape form a diode that must be reverse-biased to prevent the current to flow from one side to the other. This is why the P-type substrate is polarized to a net such as VSS, similar to the bulk terminal of the Nch devices. Likewise, the Nwell shape is polarized to a net such as VDD, similar to the bulk terminal of the Pch devices.

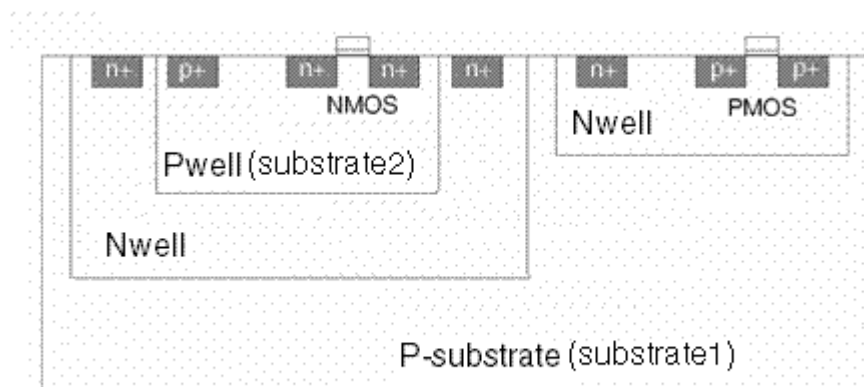


N-type process

Note:

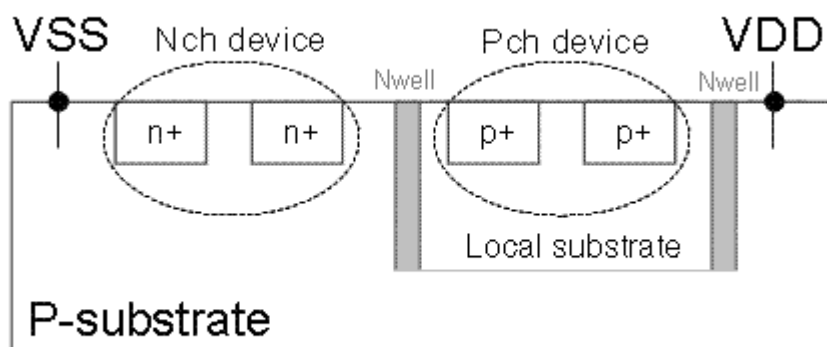
- The Nwell shape drawn over the substrate isolates the enclosed devices from the main substrate.

- The Nwell shape can be a guard ring. In such a case, the isolation created by the Nwell guard ring divides the substrate into two areas—substrate1 and substrate2—as displayed in the figure below.



Local and Global substrates

The main substrate is the *Global* substrate that belongs to the entire chip. The *Local* substrate, which is the area enclosed within the Nwell guard ring, as shown in the figure below. The local and global substrates are typically used in a design configuration to separate the digital and analog parts on a chip.



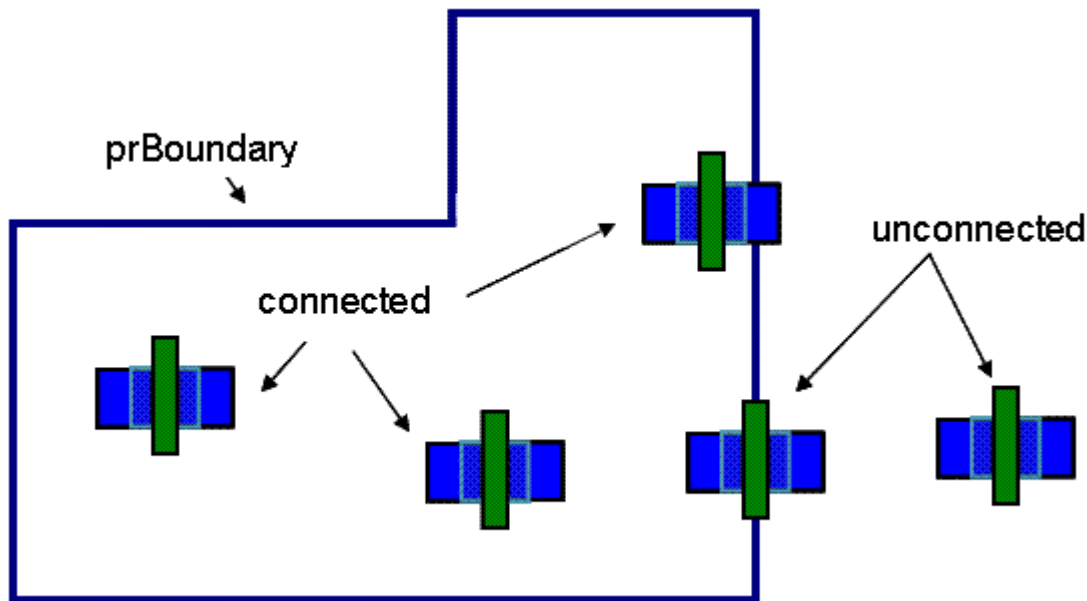
With this kind of isolation, the bulk instance terminals of devices placed on one substrate area are not connected to the bulk instance terminals of devices placed on the other substrate area. For more information, see [Isolate the Substrate and Well Layers](#).

Note: The Nwell guard ring can additionally be overlapped by a Deep Nwell layer. This increases the isolation between the main and the local substrate by preventing the current to flow vertically between the two substrate areas. For more information about

deep well layers, see [Substrate and Well Isolation Using a Deep Well Layer](#).

While extracting connectivity, you need to identify the substrate area by representing it using the prBoundary. You may then partition the global substrate into specific local substrates. Only those devices on the given substrate area that have their bulk area (physical shape or derived area) fully within the prBoundary are considered for extraction.

For example, in the figure below, the two devices that are labeled as “unconnected” do not have their bulk area fully inside the prBoundary. So, these devices are not considered for connectivity extraction.



Nch devices inside and outside the substrate area identified for extraction

Isolate the Substrate and Well Layers

Isolating a portion of the substrate requires that the technology file mentions a given well layer as “cutting” (or “stopping”) the substrate. Several well layers can stop the substrate, but they should all be of the same type— N-type or P-type. This helps the extractor recognize any shape drawn on a well layer as either:

- “Connecting” to the substrate. For example, a Pwell shape of type “pwell” in the technology file connects to a P-type substrate.

- “Stopping” the substrate—providing an area isolated from the substrate or cutting it into two parts. For example, an Nwell shape of type “nwell” in the technology file stops a P-type substrate.

Isolating a portion of a well layer is similar to isolating the substrate because it requires some other well layers to be defined as “stopping” it. These “stopping” well layers should all be of the same type, opposite to the type of the well layer to be stopped.

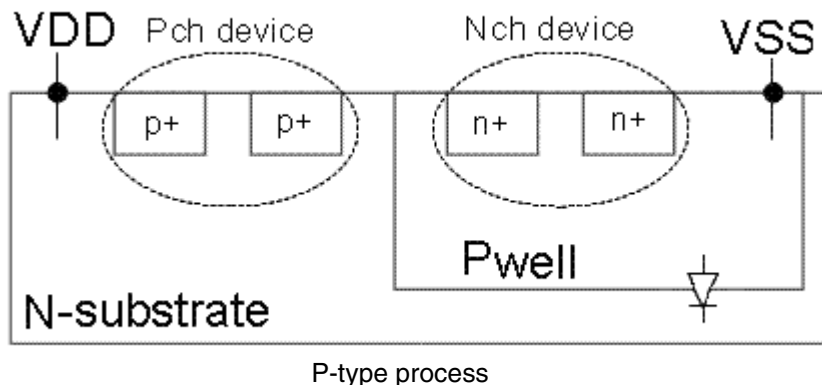
Note: You can use dedicated isolation layers to stop the substrate or well layers. In that case, these isolation layers should be defined as “recognition” layers in the technology file.

The details about a layer cutting or stopping another layer are specified in the technology file by using derived layers. For more information about derived layers, see [Specifying Information Required by the Layout XL Connectivity Extractor](#).

If the technology file does not define a well layer of the opposite type as the substrate as “stopping” the substrate, the extractor considers the corresponding well shapes as shorting to the substrate. In such a case, the extractor might produce some incorrect results during connectivity extraction. For more information, see [Checking Connectivity](#).

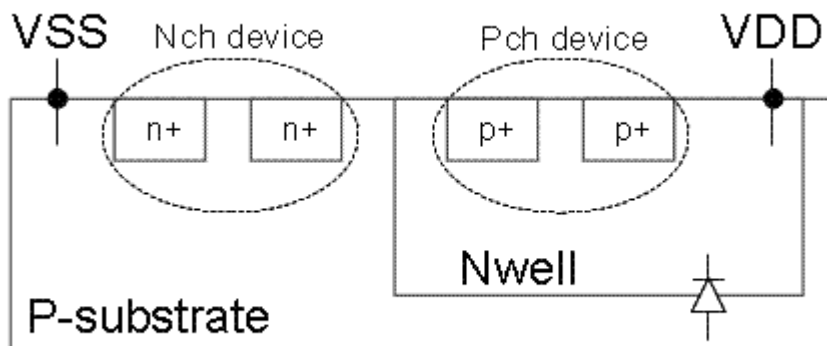
Let us consider the following example.

Consider a highly doped “P” region formed by a Pwell layer on top of an N-type substrate, as in the figure below. If you do not specify, by using appropriate derived layers, that the Pwell layer “cuts” the substrate, the extractor assumes that the Pwell layer “shorts” to it. The extractor, therefore, propagates the connectivity between the Pwell and the substrate, generating incorrect connectivity extraction results.



It is, therefore, required that the technology file defines any well layer, which is of the reverse doping type compared to the substrate, as “cutting” the substrate. For example, the figure

below displays a P-type substrate cut by an Nwell layer. For the extractor to establish correct connectivity between the two, the Nwell layer must be set as “cutting” the substrate.



To summarize, if a well layer is n-type and the substrate is p-type or vice versa, the technology file must indicate that the well layer “cuts” the substrate.

The various methods that you can use to isolate a substrate and well layer are:

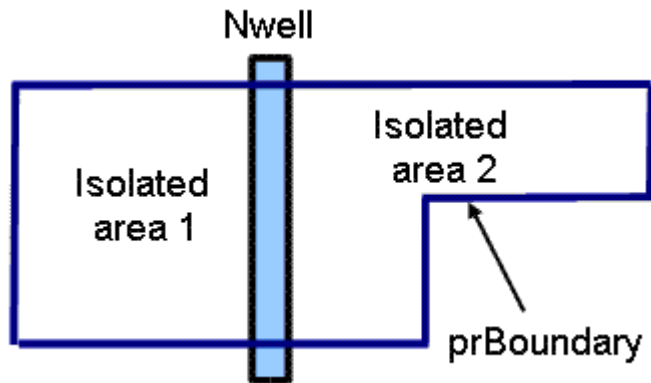
- Substrate and Well Isolation by Drawing a Physical Shape on a Layer
- Substrate and Well Isolation Using an Isolation Layer
- Substrate and Well Isolation Using a Pseudo (Dummy) Isolation Layer
- Substrate and Well Isolation Using a Deep Well Layer

Substrate and Well Isolation by Drawing a Physical Shape on a Layer

You can isolate an area of the substrate by creating a physical shape on a well layer. The well shape can be a simple rectangle or a guard ring. If you use a guard ring for isolation, the well shape that is formed isolates the substrate area lying inside the guard ring from the substrate area outside.

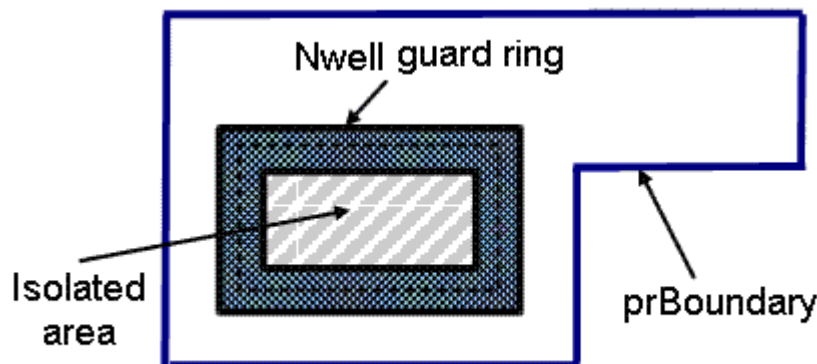
If an Nwell layer completely cuts the substrate into two distinct parts, the two substrate areas are electrically isolated. For example, in the following figure, the Nwell layer completely “cuts” through the substrate, partitioning it into two physically and electrically separated areas. If you now place “Nch” devices on the two substrate areas, there is no connectivity propagation across the bulk instance terminals of these devices.

Note: The two substrate areas must be physically distinct—without overlap or abutment—for the isolation to be effective and recognized by the extractor.



Nwell “cuts” substrate, defines two P-type areas

Let us now consider another example. In this case, an Nwell guard ring has been drawn on the substrate. This guard ring cuts the substrate and, therefore, provides a local substrate within, which is isolated from the global substrate. The bulk instance terminals of devices placed inside the guard ring are not connected to the bulk instance terminals of devices placed outside the guard ring.

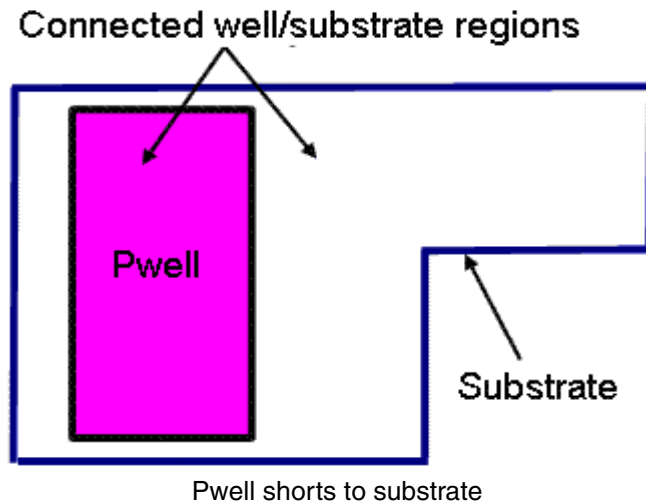


Nwell “cuts” substrate, defines a local substrate

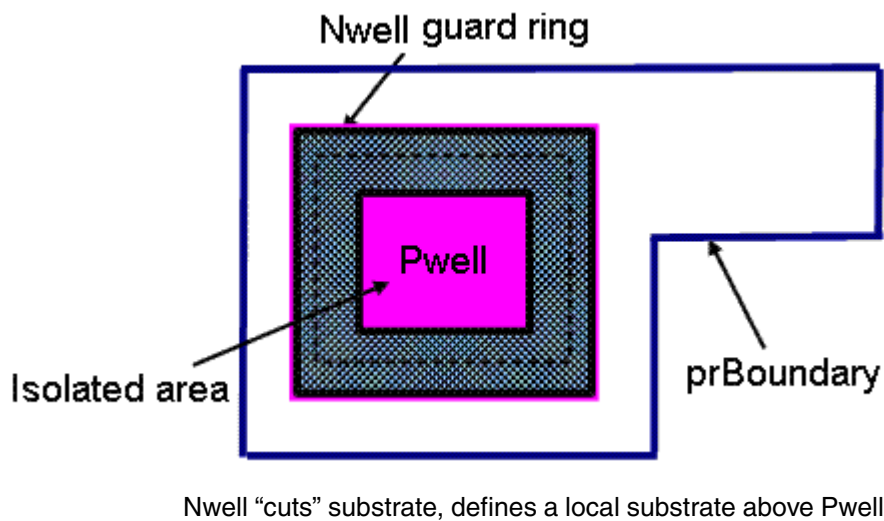
Note: If the Nwell guard ring does not form a closed ring, the two substrate areas formed by the guard ring are actually not electrically isolated. Therefore, the bulk instance terminals of devices placed inside the guard ring are, in this case, connected to the instance terminals of devices placed outside the guard ring.

Let us next consider an example where a Pwell shape of the same type as the substrate is placed over the substrate. Moreover, the well layer is not defined as “stopping” the substrate.

In this case, the extractor considers the well shape as connecting to the substrate and propagates connectivity between the two.



If you want to isolate the Pwell shape from the substrate, you can surround it with an Nwell guard ring as shown in the figure below.



Although not strictly required by the connectivity extractor for connectivity checking, you can additionally increase the isolation of the above Pwell shape by overlapping it with a Deep Nwell Layer. For more information about increased substrate isolation using deep wells, see [Substrate and Well Isolation Using a Deep Well Layer](#).

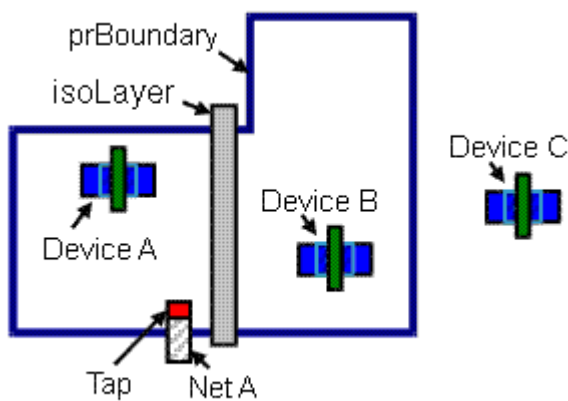
Related Topics

Bulk Area: Connectivity Extraction

Substrate and Well Isolation Using an Isolation Layer

In addition to using a well shape for isolating an area of the substrate or another well layer, you can use a dedicated isolation layer. An isolation layer can be a shape—a rectangle, a guard ring, or any other shape—that cuts the substrate or the well layer into discrete areas.

When isolating an area of the substrate or a well layer using an isolation layer, the bulk instance terminals of the devices on different substrate or well areas are not connected. Let us consider an example of a well-formed isolation made up of an isolation layer that completely cuts the substrate into two distinct parts.

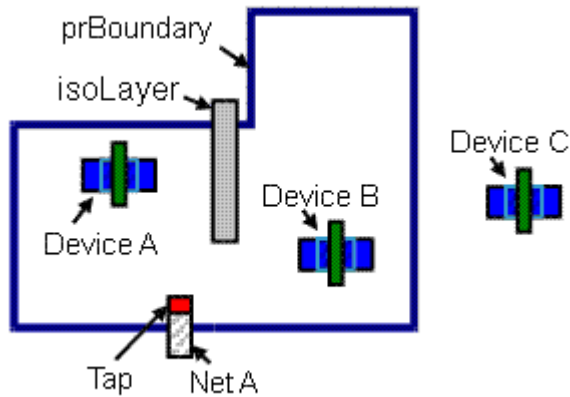


isoLayer cuts substrate, well-formed isolation

Here:

- The isolation layer cuts the substrate into two discrete areas.
- The bulk instance terminal of device A is assigned to net A from the tap via through the substrate.
- The bulk instance terminal of device B is left unassigned because it lies in a different substrate region.
- Device C is not considered for connectivity extraction of the bulk terminal because it lies outside the prBoundary. Therefore, the bulk instance terminal of the device is left unassigned.

Let us next consider an example of a malformed isolation in which the isolation layer does not completely cut the substrate into two different parts.



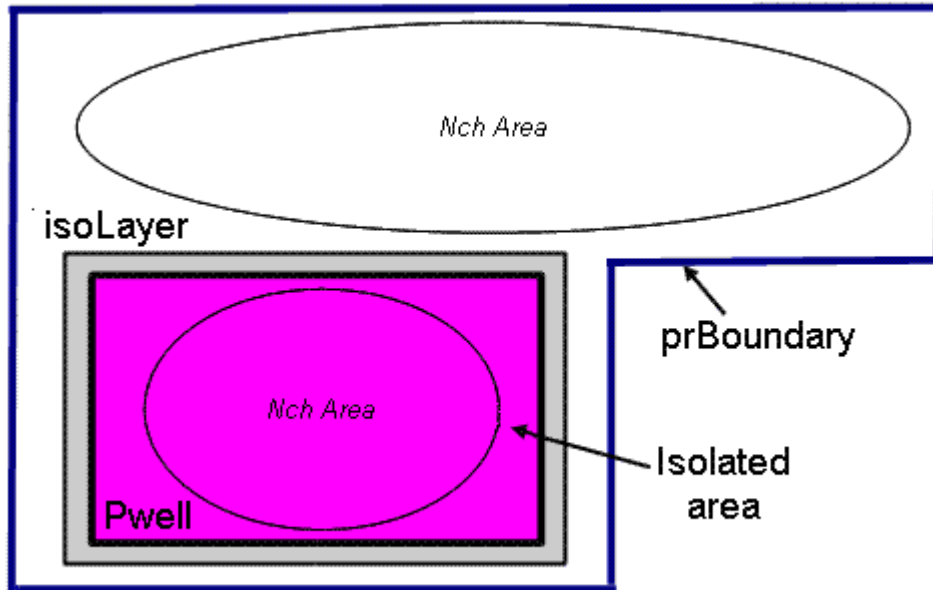
isoLayer partly cuts substrate, malformed isolation

Here:

- The isolation layer does not cut the substrate into two discrete areas.
- The bulk instance terminal of devices A and B is assigned to net A from the tap via through the substrate.
- Device C is not considered for connectivity extraction of the bulk terminal because it lies outside the prBoundary. Therefore, the bulk instance terminal of the device is left unassigned.

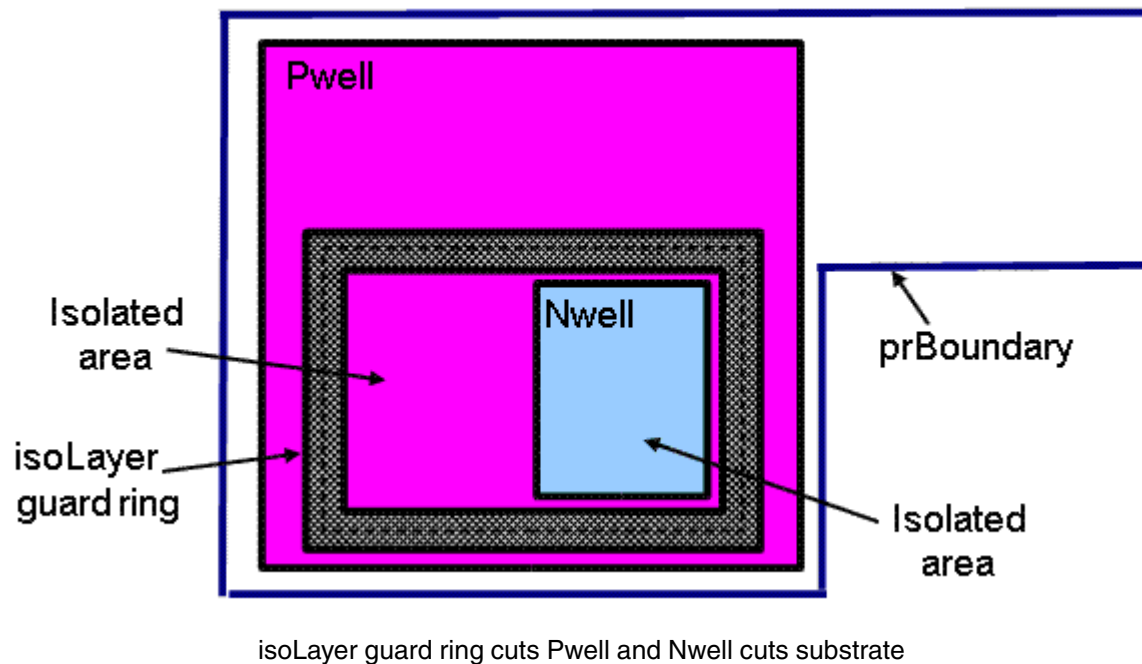
Note: During connectivity checking, the extractor does not consider the distances between a tap via placed over a substrate or well area and the bulk terminal of the devices placed within this area. All the devices that are placed on this substrate or well area are polarized, irrespective of their distance from the tap via.

An isolation layer can also be used to isolate a well layer from the substrate if the well layer is of the same type as the substrate. In the following figure, the Pwell shape is totally enclosed inside the isolation shape and it is, therefore, isolated from the substrate outside. In this example, the isolation layer is set as 'stopping' the substrate but the extractor only considers its edges as actually cutting the substrate. Therefore, the extractor considers the inner area as an isolated portion of the same doping type as the outer area.



isoLayer isolates Pwell by cutting substrate

For further isolation within a well area, you can use a guard ring. In the figure below, a guard ring containing an isolation layer is placed over a Pwell layer. This isolates the Pwell area inside the guard ring from the Pwell area outside. The Pwell outside the guard ring shorts to the substrate. However, the Pwell inside the guard ring forms an isolated area, which can be cut further by an Nwell shape to provide an isolated area of the opposite type.



Related topics

[Isolate the Substrate and Well Layers](#)

[Substrate and Well Isolation Using a Pseudo \(Dummy\) Isolation Layer](#)

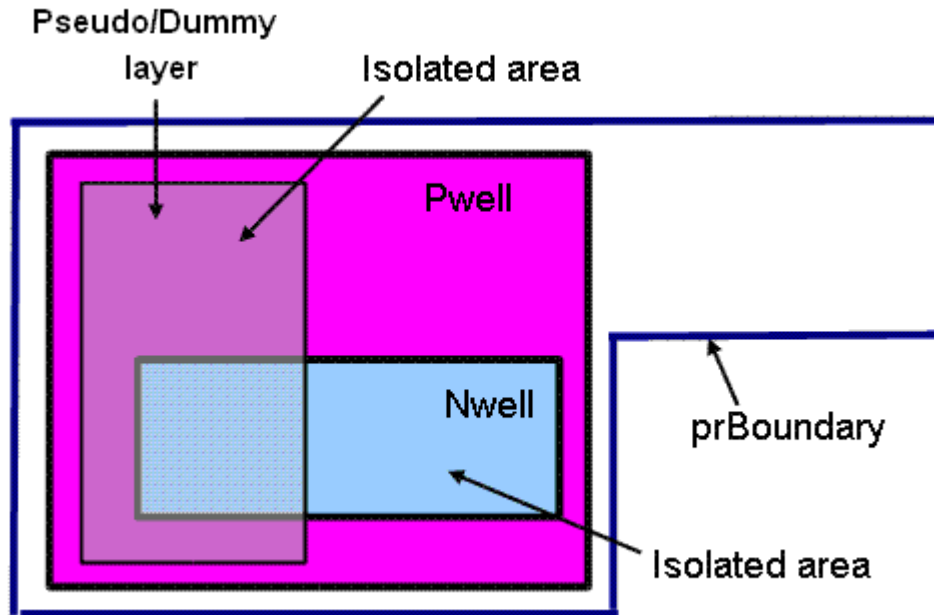
[Substrate and Well Isolation Using a Deep Well Layer](#)

[Bulk Area: Connectivity Extraction](#)

Substrate and Well Isolation Using a Pseudo (Dummy) Isolation Layer

This methodology is not recommended because the pseudo isolation layer is not manufactured. Therefore, some real short violations could potentially be left undetected by the tool. However, if a pseudo isolation layer is used, the same extraction requirements and connectivity checking results will be observed as for a real isolation layer.

A pseudo isolation layer is processed by the extractor the same way as a real isolation layer. For example, in the figure below, the pseudo isolation layer isolates the Pwell area below it from the Pwell area outside. The Nwell layer further cuts the P-type area and provides an isolated N-type area by itself.



Dummy layer isolates Pwell; Nwell cuts the substrate

Related Topics

[Isolate the Substrate and Well Layers](#)

[Substrate and Well Isolation Using an Isolation Layer](#)

[Substrate and Well Isolation Using a Deep Well Layer](#)

[Bulk Area: Connectivity Extraction](#)

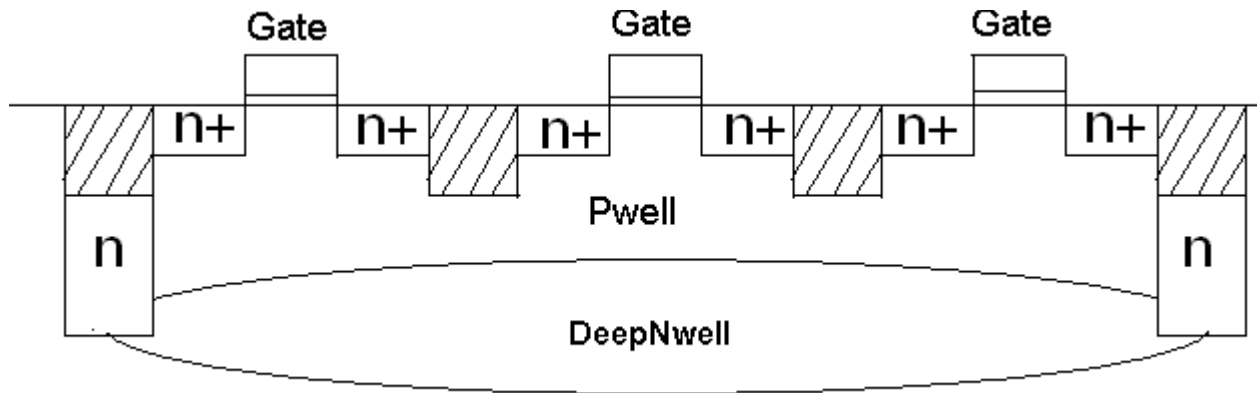
Substrate and Well Isolation Using a Deep Well Layer

Deep well layers allow users to provide a more efficient isolation when used in addition to the isolation methods based on physical shapes or isolation layers.

In the technology file, a deep well layer is a well layer of type “nwell” or “pwell” with the same function as its associated (non-deep) well layer but with a lower mask number. A deep well layer provides increased substrate isolation in triple-well processes, which comprise a buried Nwell layer that penetrates deep into the substrate to isolate the Pwell from the substrate.

Note: A deep well layer only connects to an associated (non-deep) well layer.

A well-formed local (P-type) substrate formed by using a deep well is a Pwell shape over a DeepNwell shape, as shown in the figure below, with the Pwell shape surrounded by an Nwell guard ring.



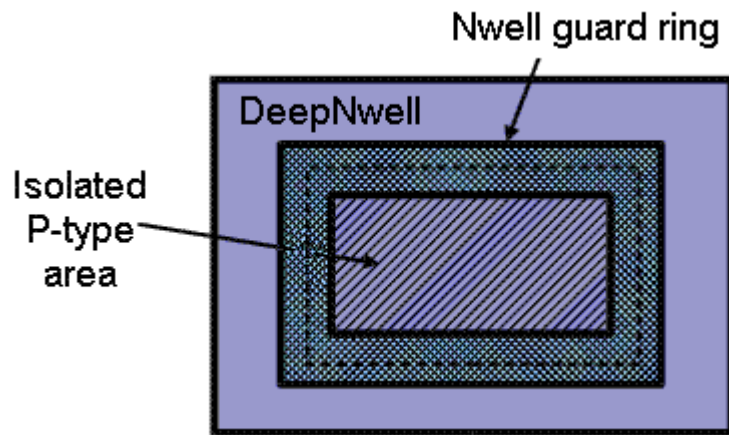
The Nwell guard ring provides sufficient isolation to prevent the current from flowing horizontally between the Pwell shape and the substrate. The deep well layer, DeepNwell, further increases the isolation by preventing the current from flowing vertically between the Pwell shape and the substrate. Therefore, if an “Nch” device is placed over the Pwell shape that lies above the DeepNwell, the bulk instance terminal of this “Nch” device is isolated from any other “Nch” device placed outside of the deep well.

Note: For the isolation due to the DeepNwell to be considered effective, the DeepNwell shape must be assigned to a net.



A deep well layer does not provide any isolation by itself because it does not form a diode with the substrate. Therefore, current can flow “horizontally” outside a deep well shape.

In the example below, the Nwell guard ring defines a local substrate, which is further isolated from the global one by a deep Nwell shape. Note that, in this example, the Pwell shape is not drawn. This could be intentional or because the technology file does not define the Pwell.



Local substrate with a Deep Well Layer

Related Topics

[Isolate the Substrate and Well Layers](#)

[Substrate and Well Isolation Using a Pseudo \(Dummy\) Isolation Layer](#)

[Bulk Area: Connectivity Extraction](#)

Identify the Bulk Area

For the extractor to recognize the bulk area of a device placed over the substrate or a well area, the bulk terminal should either have:

- Physically drawn shapes that represent the areas to be extracted.
- A derived region that represents the areas to be extracted.

In both cases, the bulk terminal must be explicitly set as soft-connect for the extractor to recognize it as a bulk connection.

You can create a soft-connect bulk terminal by using the `dbSetTermSoftConnect` SKILL API. Because the terminal is soft-connect, it does not draw current and is used only to polarize the bulk area of the corresponding device. The supported direction for polarization is from the substrate or well layer toward the bulk instance terminal. This allows the bulk area of the device to be polarized from an appropriate external source, but does not allow the connectivity to propagate from the bulk instance terminal to the outside of the device.

Note: As usual, the connectivity of a bulk terminal is provided by the associated top-level net attached to the instance terminal.

The connectivity extraction of a bulk terminal varies on a case basis:

- If the bulk terminal is implemented by at least one extractable pin figure, the extractor uses the corresponding geometries to detect any top-level overlap on these physical layers.
- If the bulk terminal is not implemented by any extractable pin figure (shapeless bulk terminal), the extractor derives the bulk area for that terminal using a derived layer. The derived layer represents the intersection between two or more physical layers, typically poly and diffusion. To set the connection between the bulk terminal and this derived layer, use the [`dbSetSoftConnectTermPinlessLayer`](#) API.

For example, the bulk area of an NMOS device placed on a P-type substrate can be a shapeless bulk for which the bulk area is derived based on the “gate area”. In that case, you can use the following definition to derive the bulk area:

```
techDerivedLayers((gateArea 10001 (PO 'and OD)))
```

Important

For the extractor to recognize the derived bulk area for connectivity extraction, the derived layer that represents the bulk area must be defined as a valid layer and used as a parameter for [`dbSetSoftConnectTermPinlessLayer`](#).

By default, a shapeless bulk terminal connects to the substrate. However, if you need to redefine this connection, you can do so using the [`dbSetSoftConnectTermConnectToLayer`](#) API.

- For a device having several bulk areas, either on the same bulk terminal or on different ones, the bulk areas should be physically connected so that they are properly polarized. Otherwise, the extractor creates open markers to indicate that the bulk areas should be connected. In the case of a device having several bulk terminals, such as when implementing multi-fingered devices, all the bulk terminals must be set as must-connect terminals to allow for proper detection of the unconnected bulk areas by the extractor.

Identify the Substrate and Well Tap Vias

A tap via (pVia or nVia) is a via that connects to a substrate area or a well layer and polarizes it.

In a P-type substrate, the technology file defines a pVia as a standard via that has its optional well or substrate layer set to “substrate”. A pVia connects to the P-type local or global substrate or to a drawn Pwell on which the via is placed.

The following is a sample via definition that allows a via to polarize the substrate from the `metall` layer:

```
metall - > cutLayer - > (diffusionLayer && PImplantLayer) - > substrate
```

Unlike a pVia, an nVia has its optional well or substrate layer set to a physical Nwell layer. An nVia connects to a well layer, such as Nwell, on which it is placed.

The following is a sample via definition that allows a via to polarize the Nwell shape on which it is placed:

```
metall - > cutLayer - > (diffusionLayer && NImplantLayer) - > Nwell
```

Note:

- The extractor recognizes an nVia defined as connecting to an Nwell layer even if the via master does not implement a physical Nwell shape.
- An nVia does not connect to a deep Nwell layer.

The definitions for an N-type substrate are similar to those for a P-type substrate except that it is an nVia that connects to an N-type substrate instead of a pVia. A pVia, on the contrary, is defined as connecting to a physical Pwell layer.

Related Topics

[Bulk Area: Connectivity Extraction](#)

[Connectivity Propagation through the Substrate and Well Areas](#)

Connectivity Propagation through the Substrate and Well Areas

As already mentioned, the extractor does not propagate the connectivity from a soft-connect object; such as a substrate area, a well shape, or a device bulk terminal to a polarizing object, such as a tap via or a diffusion shape. However, it does support connectivity to propagate between two soft connect objects that directly overlap.

For example, if you polarize a substrate or a well area and place multiple devices on it, you will observe that the connectivity propagates to the bulk instance terminals of all the devices. As earlier mentioned, no connectivity propagation is observed to the bulk instance terminals for the devices placed outside the polarized substrate or well area.

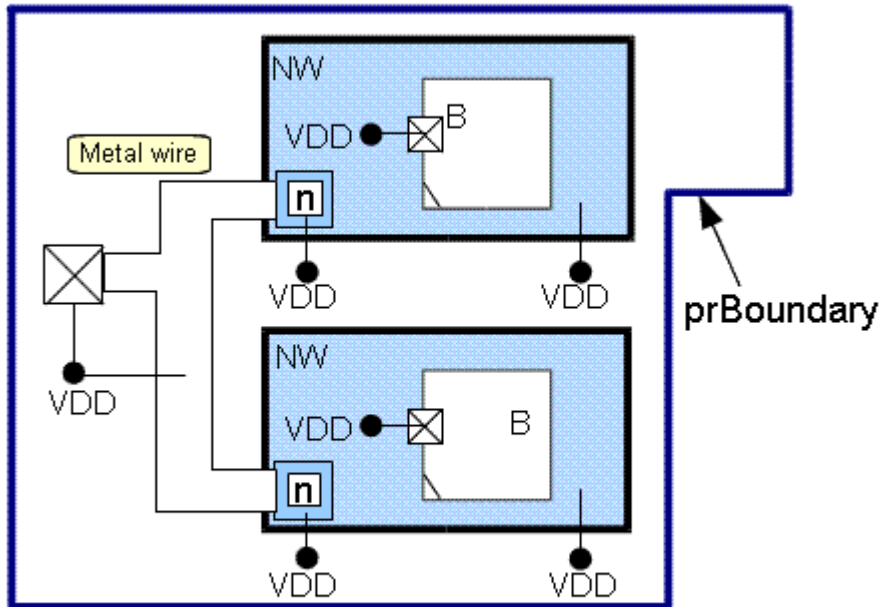
Conversely, during a *Pick From Schematic* command, if you place multiple devices whose bulk instance terminal is assigned to a net and place them over a floating substrate or a well area—an area which is not assigned to a net—you will observe no connectivity propagation from the bulk instance terminals to the substrate or well. This enforces the requirement that the substrate or the well must be polarized from an external source, such as a tap via or a diffusion shape.

Checking Connectivity

During connectivity extraction, if the extractor identifies any design violations related to substrate, well shapes, or bulk terminals, it generates appropriate open and short markers in the Annotation Browser.

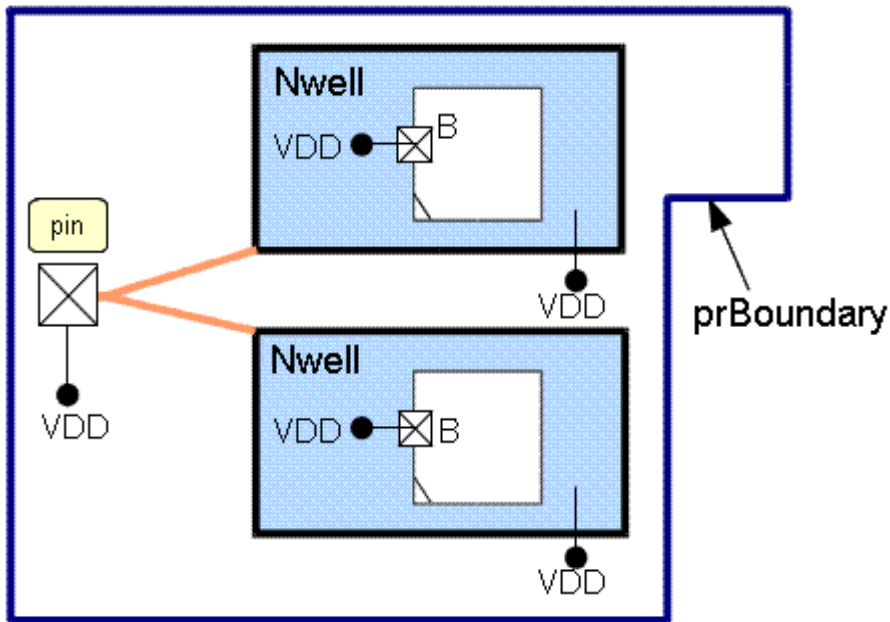
- All the devices placed on a polarized substrate or well area should have their bulk instance terminal assigned to the same net. If this is not done, the extractor creates short markers between the various nets.
- If a net polarizes the bulk instance terminal of a device through the substrate or a well, the connection is considered valid irrespective of the distance between the bulk terminal and the tap via on the net polarizing it. This means that for connectivity checking, a single tap via is sufficient to polarize the bulk of all the devices placed on a given substrate or a well area.

For example, in the figure below, a metal path on net VDD polarizes two well shapes from a pin on that net by using Nvias that connect the metal to the wells.



Well shapes properly polarized by a metal pin

Now, let's assume that the user deletes the VDD path. Although the two well shapes are close to each other, the extractor creates the open markers with the top-level VDD pin to indicate the preferred routing path. This is displayed in the figure below.

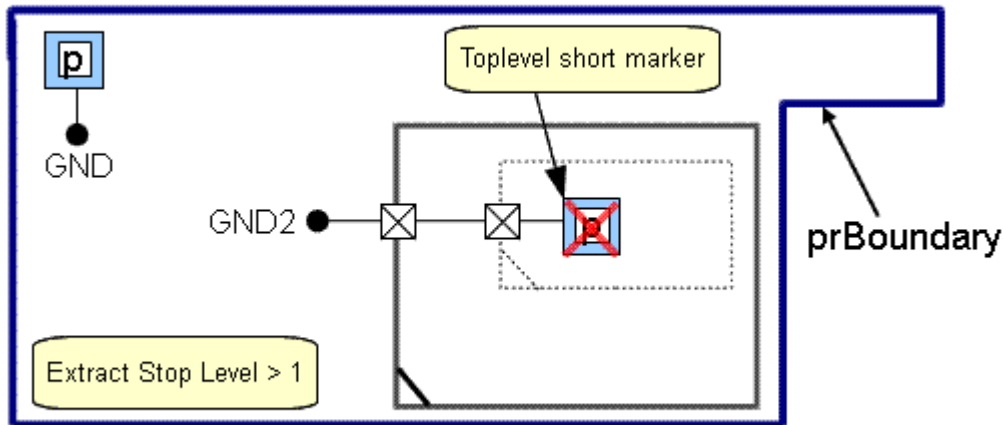


Open marker between a pin and two well shapes

Impact of the Hierarchy on the Top-Level Connectivity Checking

When the extract stop level is increased from 0 to a higher value, the extractor is able to create appropriate connectivity violation markers for objects located in the hierarchy and, in particular, for objects that create a soft connection with a top-level object.

For example, in the following figure, the substrate is polarized on net GND by a top-level pVia, as displayed in the upper left of the figure. This creates a short condition with the pVia located in the hierarchy and connected to the top-level net, GND2. The extractor, therefore, creates a short marker to report this design error.



Hierarchical pVia over a global substrate

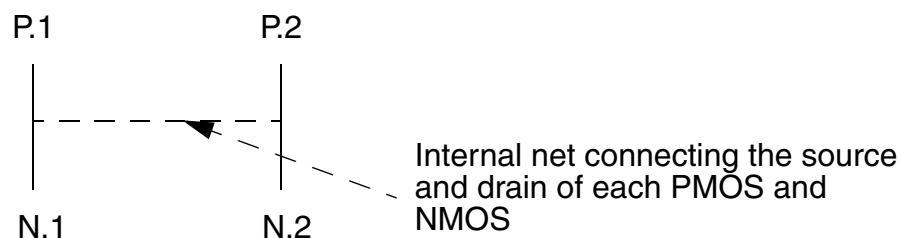
Pseudoparallel Connections

A pseudoparallel connection is defined as a group of instance terminals on the same net that must be physically connected at the current hierarchy level.

It is similar to a must-connect; the only difference being that it is implemented not using pins but using instance terminals that belong to a specified net. The main advantage of recognizing that a net can be made pseudoparallel is that it saves space because less routing is required to connect it.

What is a Pseudoparallel Connection

Consider a simple inverter comprising an NMOS and PMOS pair. If you set an mfactor of 2 on the inverter, when it is flattened in the layout, the connectivity is as represented below.

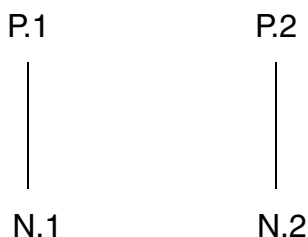


The broken line represents an internal net connecting the source and drain of each PMOS and NMOS. This net has four instance terminals; however, the router does not need to

actually make a physical connection from the P . 1 / N . 1 pair to the P . 2 / N . 2 pair in order for the circuit to function correctly. Because it is mfactored, the nets connecting to all the other terminals of P . 1 and P . 2 are exactly the same, as are the nets connecting to all the other terminals of N . 1 and N . 2. Therefore, the voltage on the node between P . 1 and N . 1 is the same as that on the node between P . 2 and N . 2, removing the need to connect them up with a wire.

To represent this situation in the database, the instance terminals are partitioned into sub-subnets. Each sub-subnet on a pseudoparallel net contains two instance terminals which must be connected together, but no connection is necessary between the instance terminals on different sub-subnets.

In the example above, there would be two sub-subnets. The instance terminals from P . 1 / N . 1 would be contained in one of them, representing the fact that they must be connected in order for the circuit to work. The instance terminals from P . 2 / N . 2 would be in the other, again representing the fact that they must be connected in order for the circuit to work. But no connection is required between the two. The circuit would work equally well without the connecting wire, as shown below.

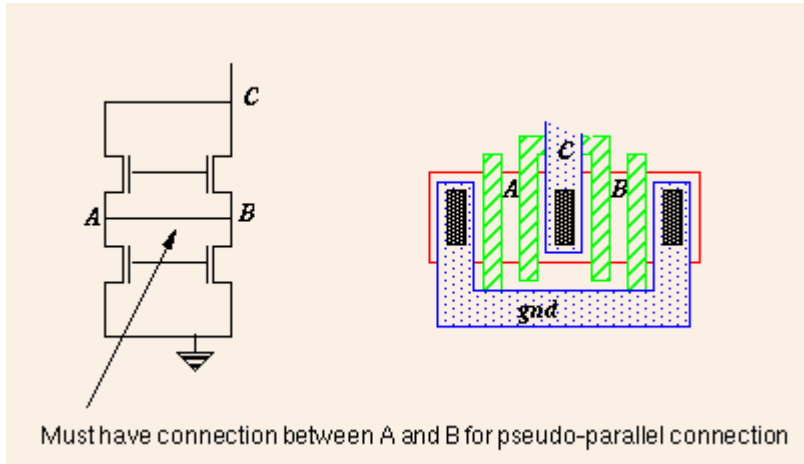


The model extends depending on the `mfactor` employed. In the example above, if the `mfactor` is `x`, then for the “internal” net connecting all the sources and drains together there are a total of `x` sub-subnets each containing two instance terminals.

Layout XL does not consider the `mfactor` in the source when determining whether a net is pseudoparallel or not. Instead it considers the net, the instances attached to it, and all the other connections to those instances. In general, any time there is a pair of devices in series in the schematic, each with the same `mfactor` value, then the net implementing the series connection can be made pseudoparallel in the layout.

Examples of Pseudoparallel Connections

Pseudoparallel connections are useful in situations like the one shown below.

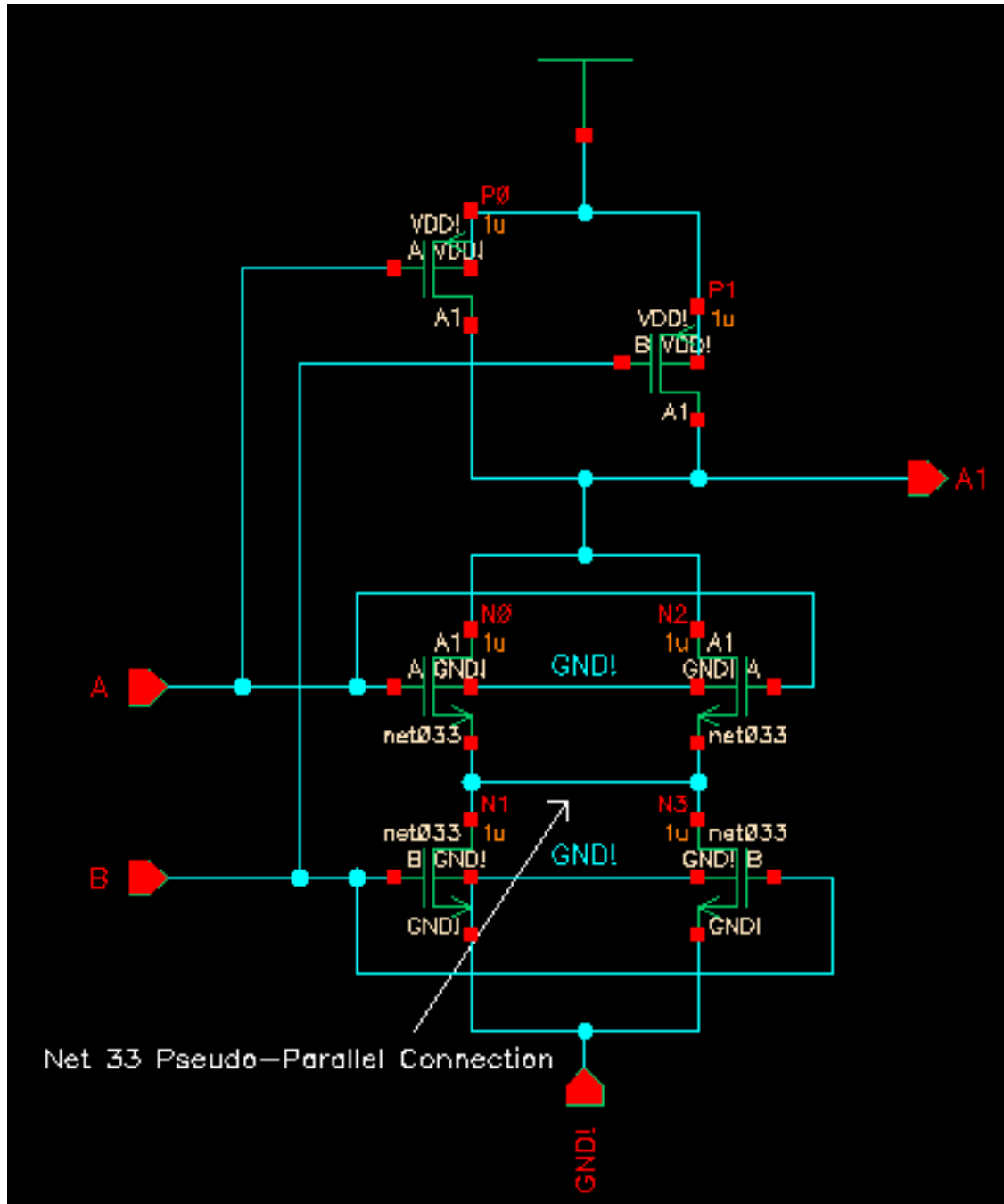


In this example, nodes A and B can be considered equipotential, i.e., there is no current flowing through them. You can save area in your design if this connection is skipped, even though the connectivity reference requires all instance terminals on the same net to be physically connected. With pseudoparallel connections on A and B, no connection is required between the two instance terminals even though they are on the same net.

Virtuoso Layout Suite XL User Guide

Preparing Your Design for Routing

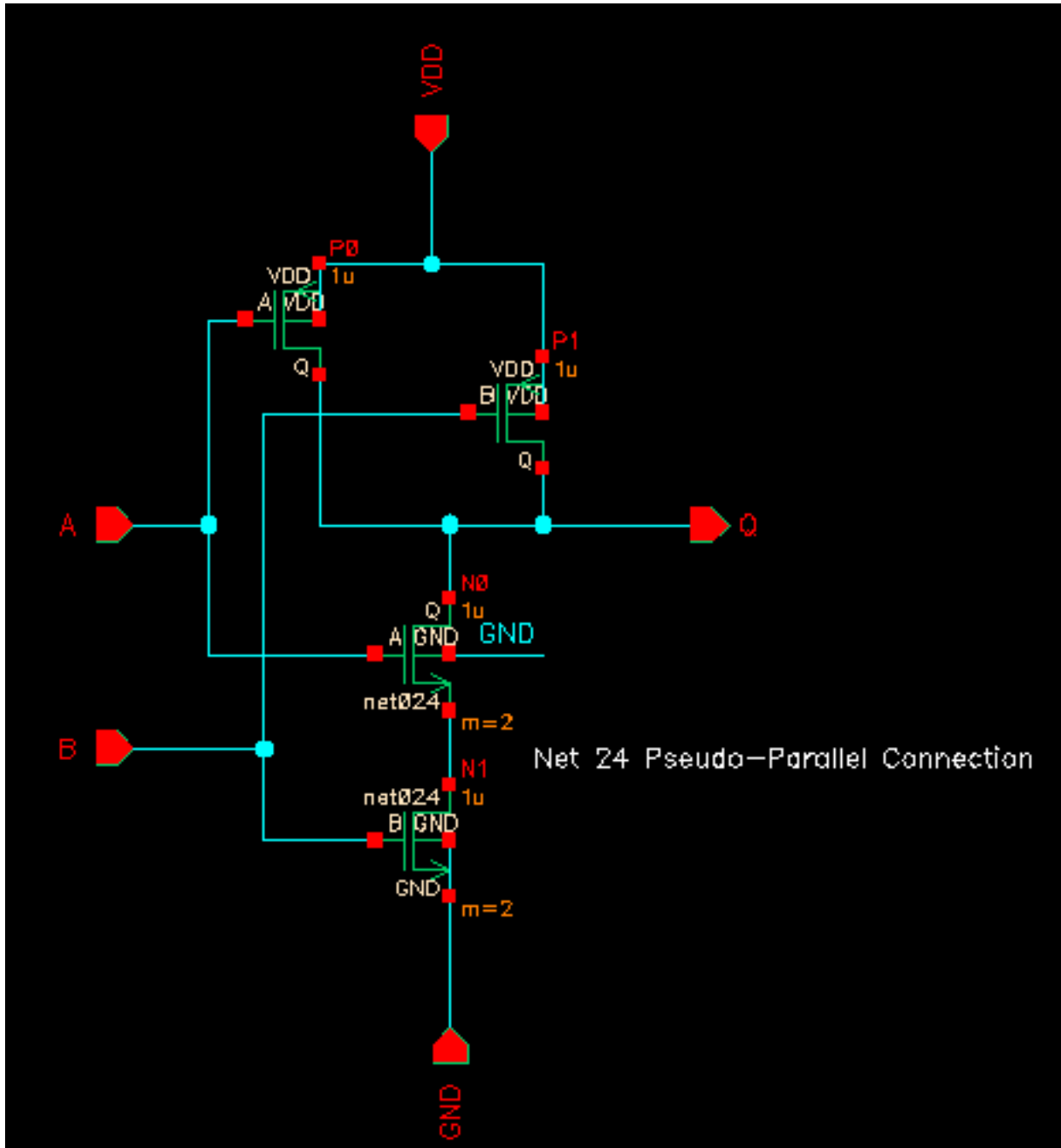
The picture below shows an explicit pseudoparallel connection of two or more groups of parallel devices in series.



Virtuoso Layout Suite XL User Guide

Preparing Your Design for Routing

The picture below shows an implicit pseudoparallel connection of two or more groups of mfactored devices in series.



Layout XL automatically finds possible pseudoparallel connections and treats them as such when you generate the layout using the *Generate All From Source* command.

If the layout exists already, you can create pseudoparallel connections automatically by running the *Update Components And Nets* command.

For information on other methods of creating pseudoparallel connections, see [Defining a Pseudoparallel Connection](#).

Defining a Pseudoparallel Connection

There are two ways in which you can define a pseudoparallel connection.

- [Using the Pseudoparallel Connect Command](#)
- [Using the setPPConn Environment Variable](#)

Using the Pseudoparallel Connect Command

To define a pseudoparallel connection for a set of instance terminals on a net,

1. From the layout window menu bar, choose *Connectivity – Pins – Pseudo Parallel Connect*.

A message in the layout window status banner prompts you to click to select a net.

2. Click on the net containing the instance terminals you want to connect.

Flight lines appear between the pins on the specified net that can be pseudoparallel connected.

Note: If you were showing incomplete nets, those flight lines are removed automatically.

A message in the layout window status banner prompts you to select an instance terminal.

3. Click on one or more pins of the selected net to select the instance terminals you want to connect in a pseudoparallel connection.

The pseudoparallel connection is defined and the flight lines updated to show the connection.

4. When you are done, press `ESC` to cancel the command.

Instead of selecting nets by clicking in the layout canvas, you can use the Define Pseudo Parallel Connected Net form.

To open the form,

1. Press `F3` while the *Pseudo Parallel Connect* command is active.

The Define Pseudo Parallel Connected Net form is displayed. The form lists all the nets with no I/O pins.

2. In the form, type the name of a net in the *Net* field or select the net names from the list.



Tip

You can work on more than one net at a time by choosing multiple nets in the form or dragging your cursor in the design window to select an area when the command line prompts you to select a net.

Note: You must define pseudoparallel connectivity within a net. You cannot define pseudoparallel connectivity between nets.

3. When you are done, press `ESC` to cancel the command.



Tip

You can change between the *Must Connect*, *Strongly Connected*, *Weakly Connected*, and *Pseudo Parallel Connect* commands using the right mouse button.

Using the `setPPConn` Environment Variable

Setting the `setPPConn` environment variable to `t` automatically identifies nodes that qualify as pseudoparallel connections and defines them as such.

To use the `setPPConn` environment variable,

- Add the following line to the `.cdsenv` file.

```
layoutXL setPPConn boolean t
```

Resetting a Pseudoparallel Connection

To reset a pseudoparallel connection,

1. Delete one of the instances in the pseudoparallel connection.
2. Set the `setPPConn` environment variable to `nil`.
3. Run either *Update Components And Nets* or *Generate All From Source* to regenerate the deleted instance.

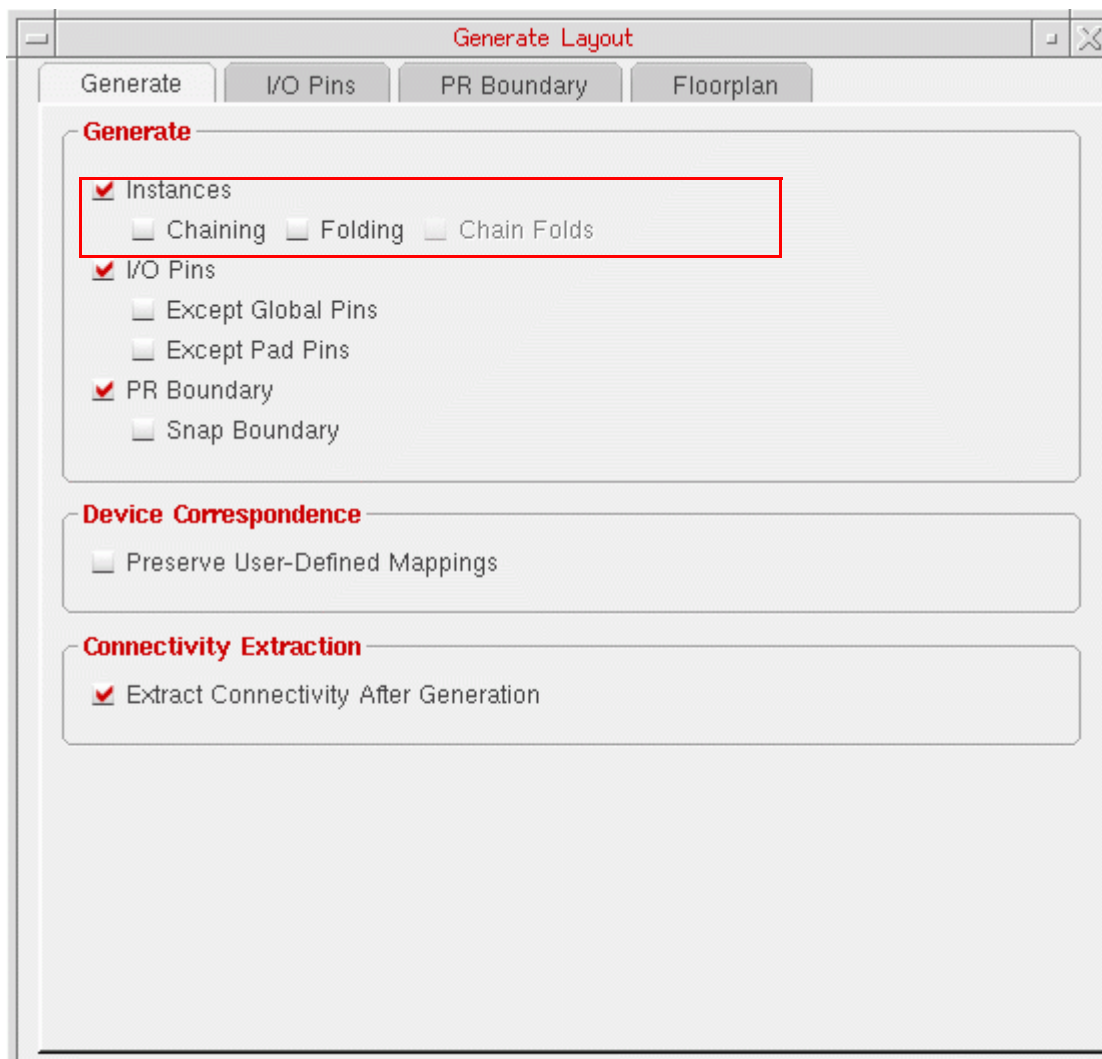
Generating a Layout without Chaining and Folding on Pseudoparallel Nets

Pseudoparallel nets are shown as must-connects when the `setPPConn` environmental variable is set to `t` and the layout is generated without chaining and folding.

To do this,

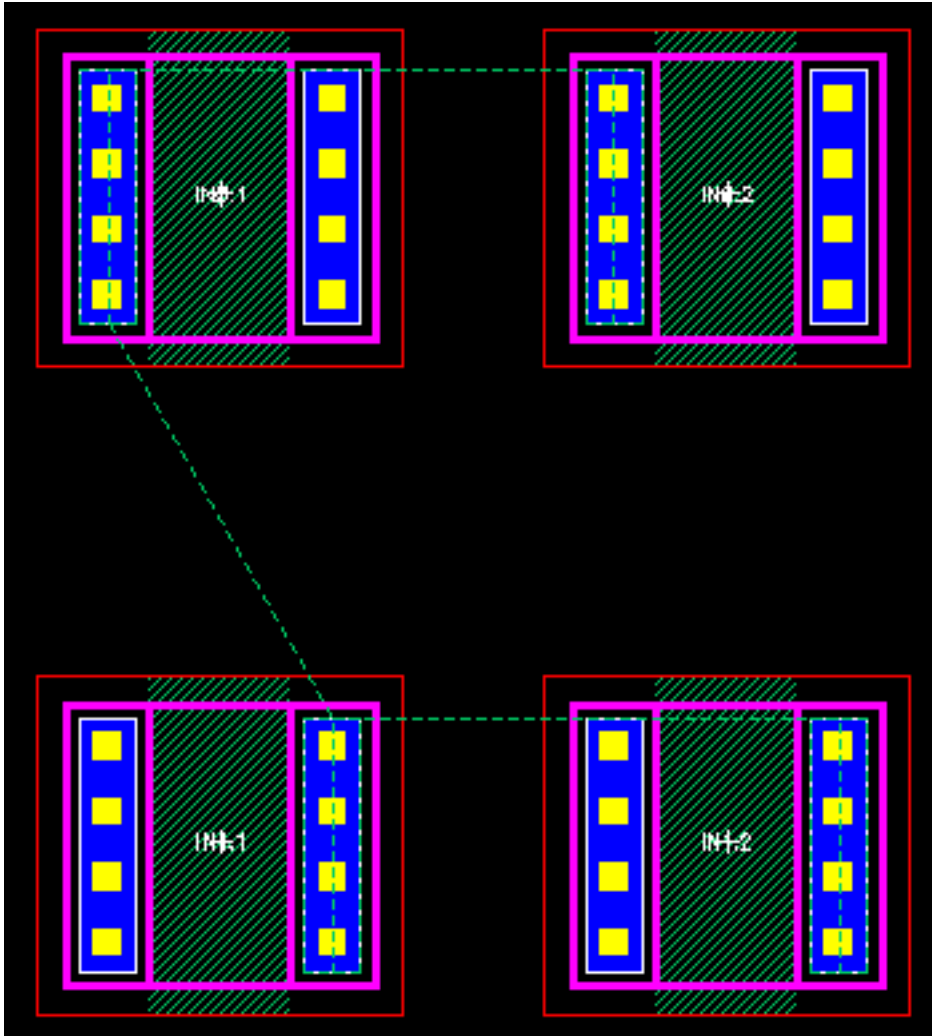
1. From the layout window menu bar, choose *Connectivity – Generate – All From Source*.
2. On the Generate Tab, turn off the *Chaining* and *Folding* options.

Note: The *Chain Folds* option is automatically deselected when the *Folding* check box is deselected.



3. Set the rest of the options for your layout and click *OK*.

4. Open the Annotation Browser assistant, go to the *Connectivity* tab and select the pseudoparallel net.

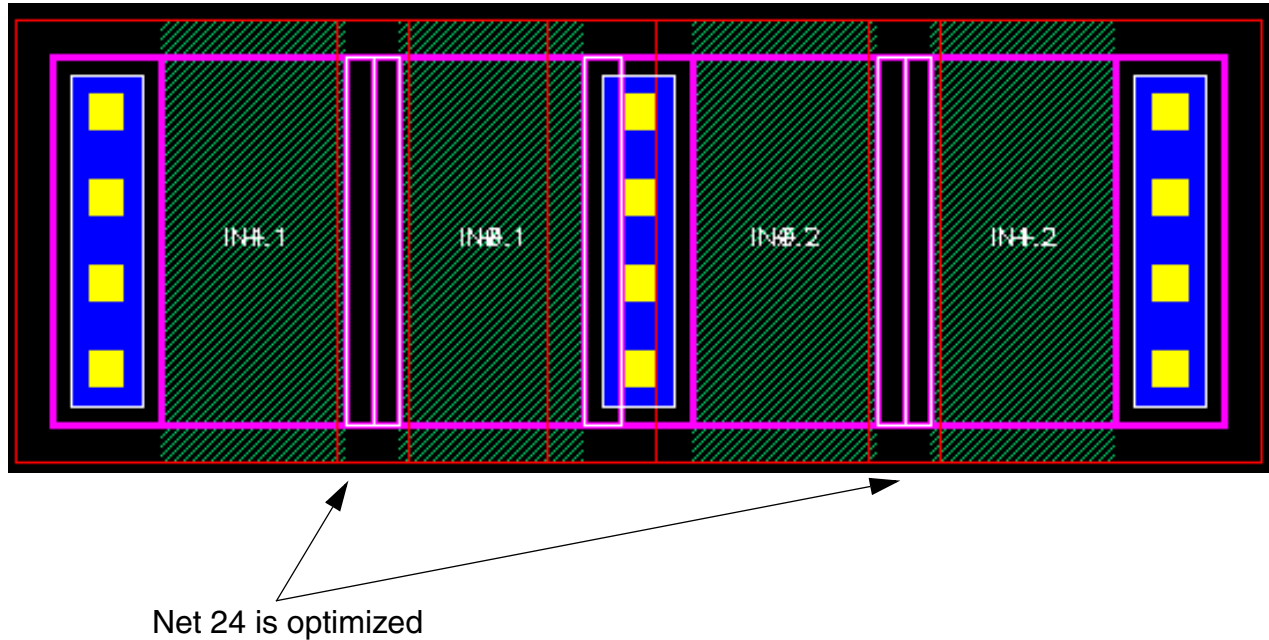


Abutting instances with pseudoparallel nets is order-dependent. If you manually abut instances with pseudoparallel nets they will abut correctly. If you then abut instances that do not have pseudoparallel nets they will not abut and the pseudoparallel nets that were previously abutted will no longer be pseudoparallel nets.

Generating a Layout with Chaining and Folding on Pseudoparallel Nets

When the `lxAllowPseudoParallelNets` environment variable is set to `t`, layout generation optimizes pseudoparallel nets for implicit pseudoparallel connections.

Note: This also works with the *Generate Selected From Source* command.



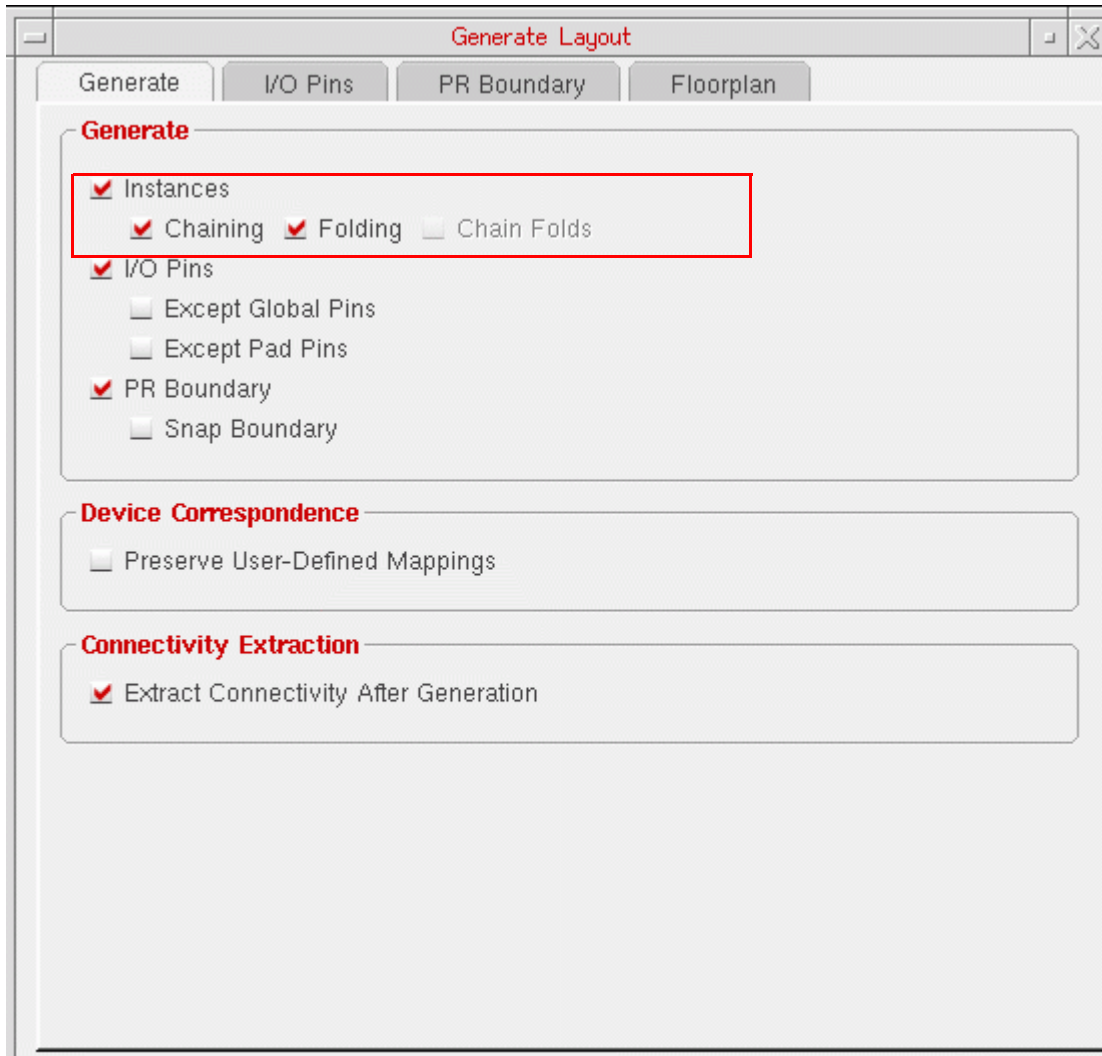
To do this,

1. Add the following line to your `.cdsenv` file.

```
layoutXL lxAllowPseudoParallelNets boolean t
```

2. From the layout window menu bar, choose *Connectivity – Generate – All From Source*.

3. On the Generate Tab, turn on the *Chaining* and *Folding* options.



4. Set the rest of the options for your layout and click *OK*.
5. To verify that the pseudoparallel nets are complete, open the Annotation Browser assistant and go to the *Connectivity* tab.

The pseudoparallel nets in question are no longer listed in the *Incomplete Nets* category in the browser.

Generating Selected Components with Pseudoparallel Nets

You can set pseudoparallel nets when using the *Generate Selected From Source* command if

- The *Group As In Schematic* option is turned on.
- The nets are internal to the instances created in the scratch cellview.
- The `setPPConn` environment variable is set to `t`.
- All the rules for pseudoparallel nets are met.

For more information, see [Pseudoparallel Connections](#) on page 464.

Working With Nets

You can use the Navigator assistant to select and deselect the nets in your design and the Property Editor and Constraint Manager assistants to create and edit attributes, properties, and constraints on those nets.

When you select a net in the Navigator assistant, Layout XL selects the net (or its flight line if it is not yet routed) and all the shapes associated with that net in the design.

- [Selecting and Deselecting Nets](#)
- [Routing a Net](#)
- [Deleting Routing on a Net](#)
- [Locking and Unlocking Nets](#)
- [Editing Net Attributes and Properties](#)
- [Creating and Editing Net Constraints](#)

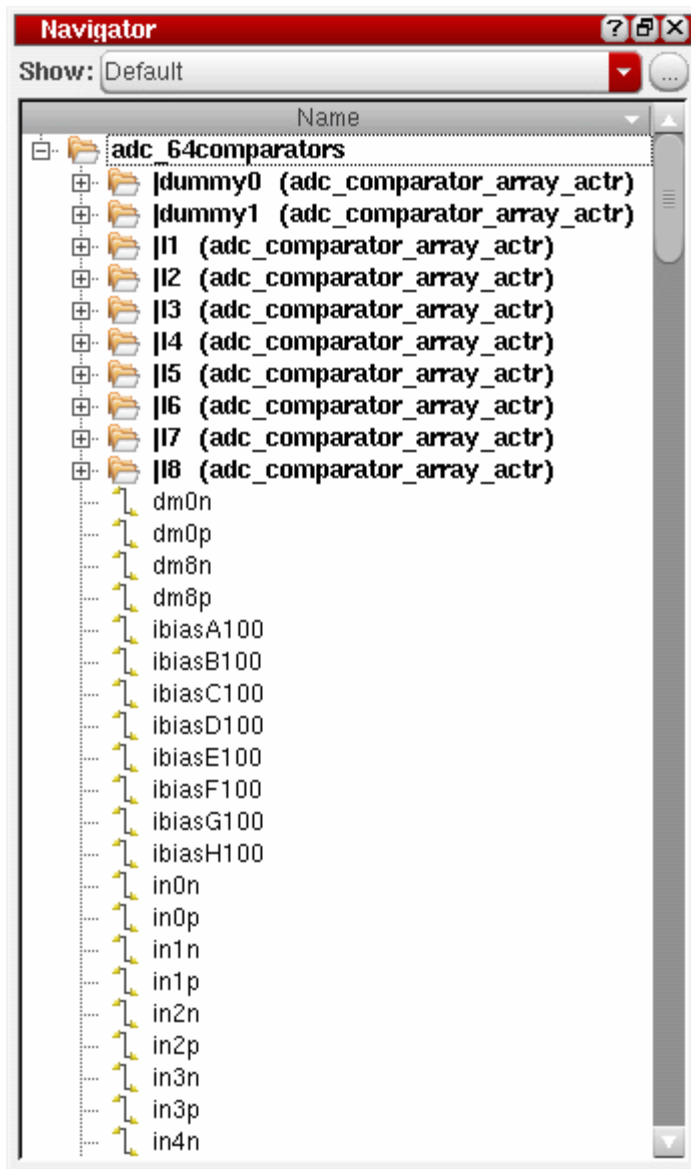
Selecting and Deselecting Nets

You need to select nets to perform pin optimization tasks, edit net criticality, and so on.

To select a net,

1. From the layout window menu bar, choose *Windows – Assistant – Navigator*.

The Navigator assistant is displayed.



2. Use the *Show* drop down in the toolbar to display only the *Nets* in the design.
3. Locate the net you want to work on in the list and click on it to select it.

The net you selected (or its flight line if it is not yet routed) is highlighted in the design window, along with all the shapes on that net.

Note: The option to probe flight lines is off by default. You set it on the [Highlight Options](#) form available using the *Options – Highlight* menu pick.

To deselect all nets,

- Click in an empty area of the design window.

For more information on the Navigator assistant, see [The Navigator Assistant](#) in the *Virtuoso Schematic Editor XL User Guide*.

Routing a Net

To route a selected net, you can choose from the following routing commands, as appropriate:

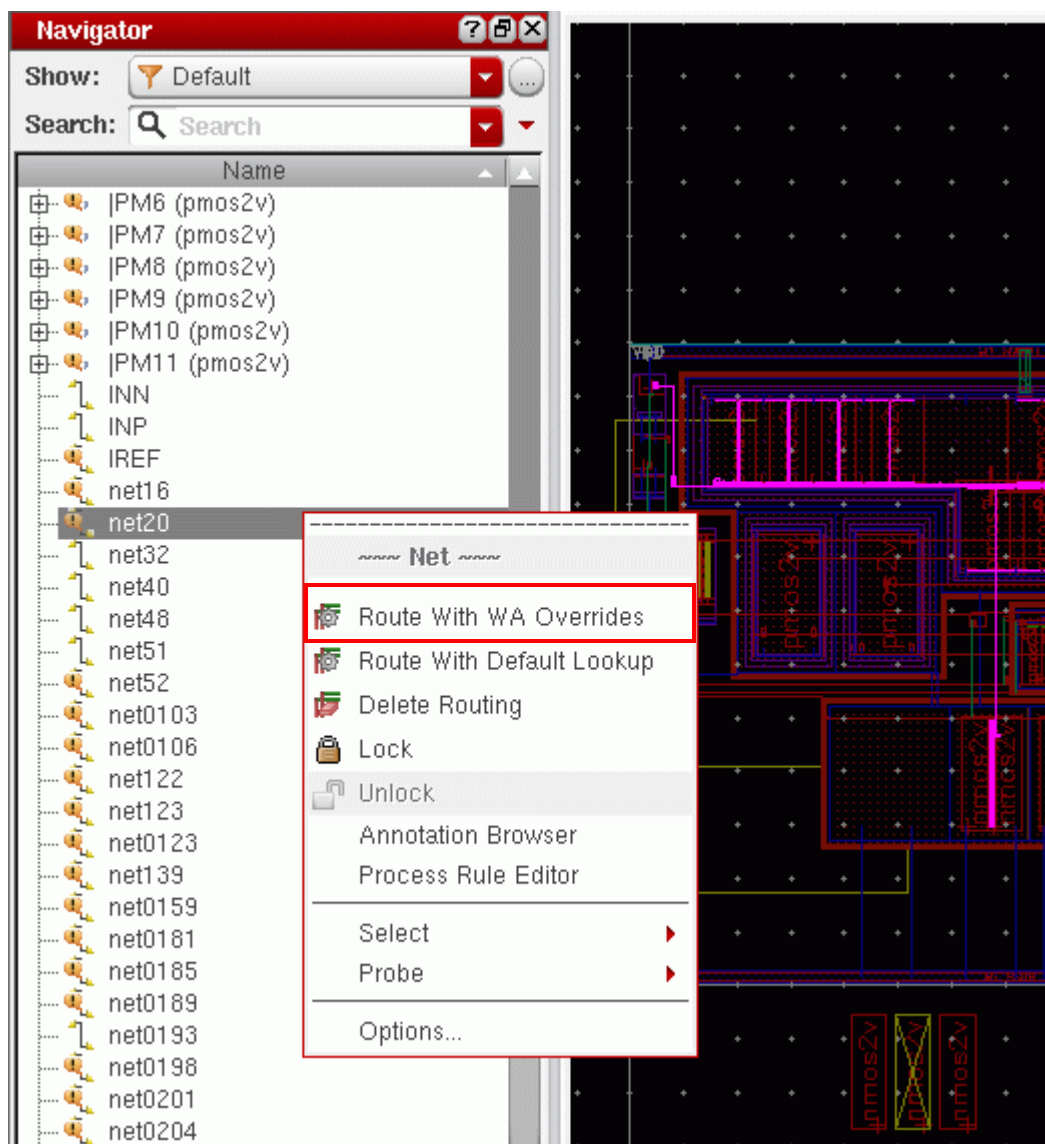
Routing With WA Overrides

Use this command to route the selected nets with the Wire Assistant Overrides taking precedence over the default constraint settings as defined by the Wire Editor Default Constraint Group, or any Net Specific Constraints.

For more information on the Wire Assistant Override Constraints, see [Override Constraints](#) under the Wire Assistant section of the Routing Assistants chapters in the *Virtuoso Space-based Router User Guide*.

Virtuoso Layout Suite XL User Guide

Preparing Your Design for Routing

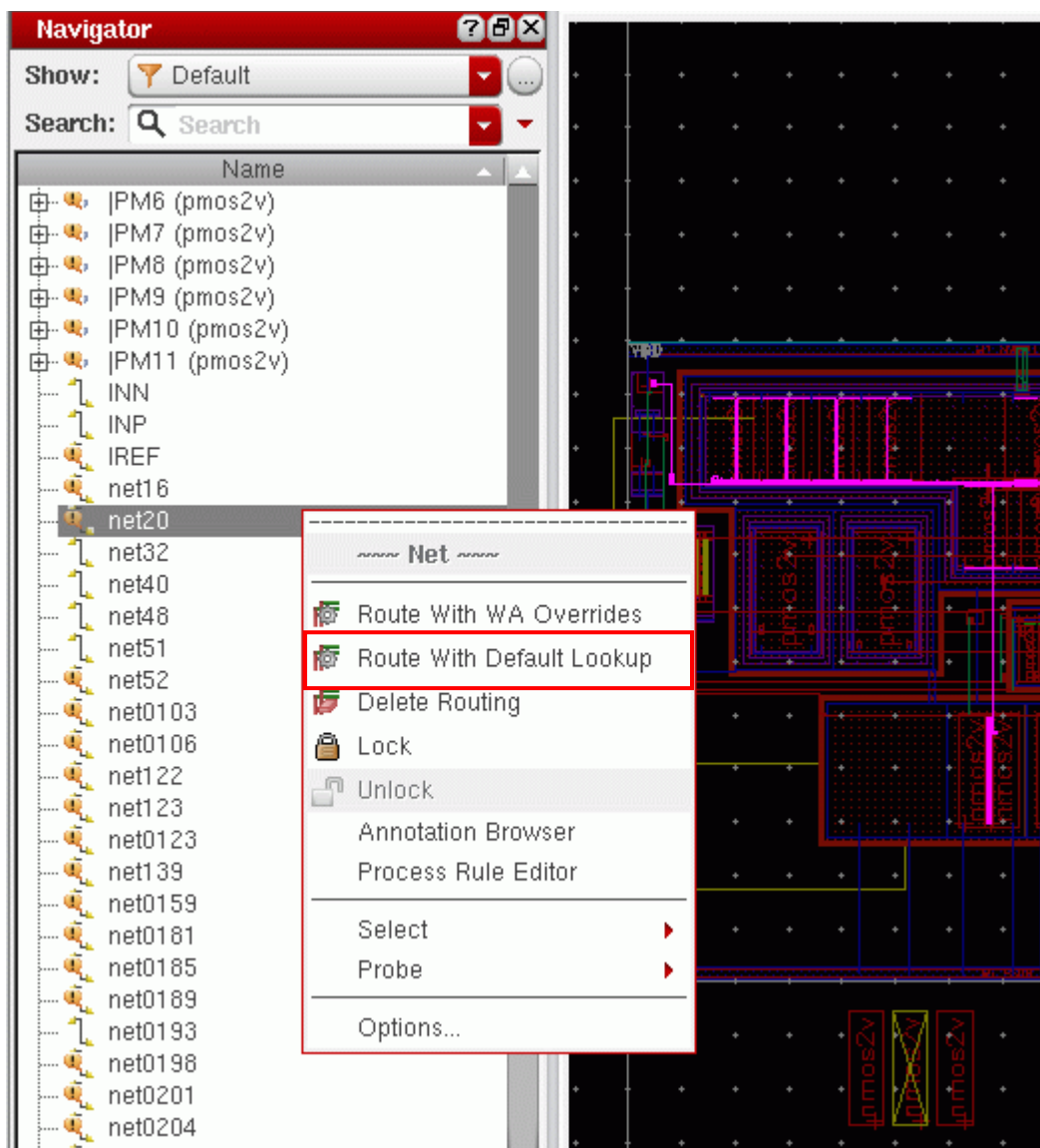


Routing With Default Lookup

Use this command to route the selected net by using the default constraint lookup for routing.

Virtuoso Layout Suite XL User Guide

Preparing Your Design for Routing



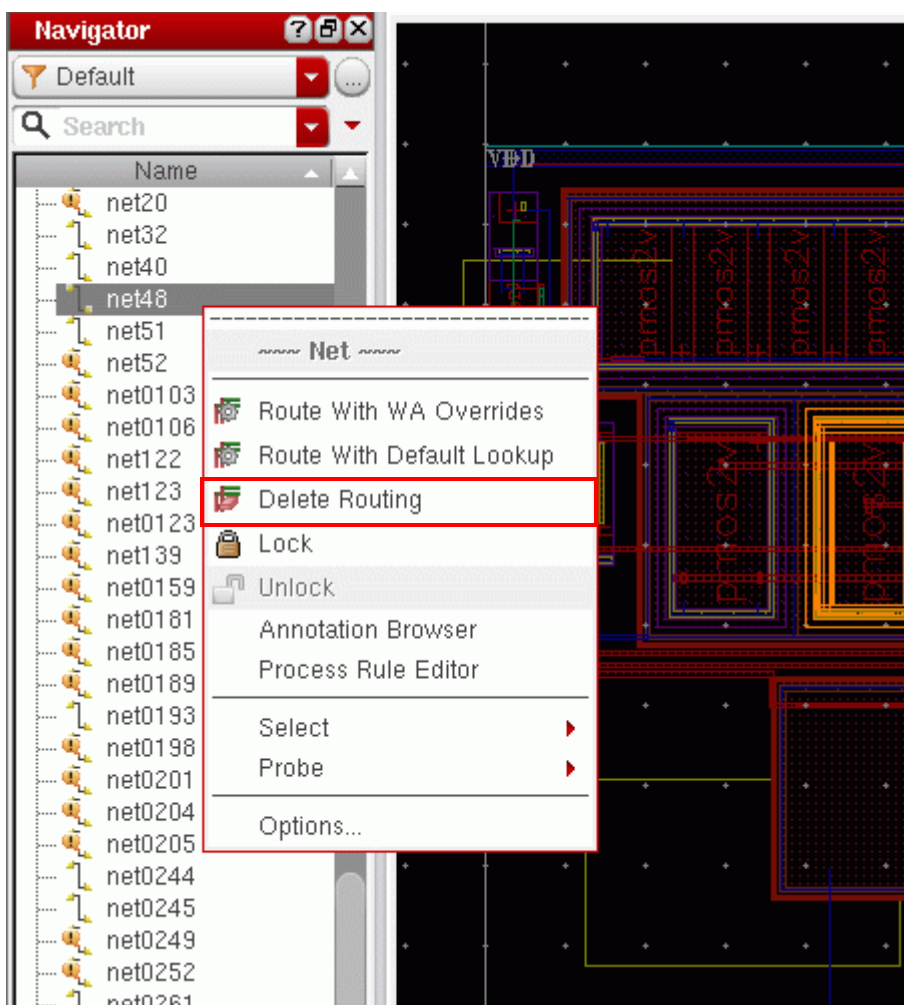
Important

Cadence recommends that you lock the nets after performing pin-to-trunk routing to ensure that any subsequent routing by assisted routing commands does not rip the nets and re-route them as regular nets. For information on locking a net, see [Locking and Unlocking Nets](#).

Deleting Routing on a Net

To delete the routing information and shielding wires on a net, you can right-click the net in the navigator and choose *Delete Routing* from the shortcut menu. Alternatively, you can access the *Delete Routing* command from the *Route* menu. For more information about the command, see [Deleting Routes](#).

Note: Any routes or nets that are locked, are not deleted. For more information about locked nets, see [Locking and Unlocking Nets](#).



Locking and Unlocking Nets

You can lock and unlock nets to set the routing status on paths, pathSegs, and vias. In other words, by locking a net, you can implement a constraint on its position in the design. For more information on the *Locked* constraint, see [Locked](#).

To lock or unlock a net, you can use the appropriate shortcut command from the **Navigator Assistant** or the **Canvas**.

Locking and Unlocking a Net from the Navigator Assistant

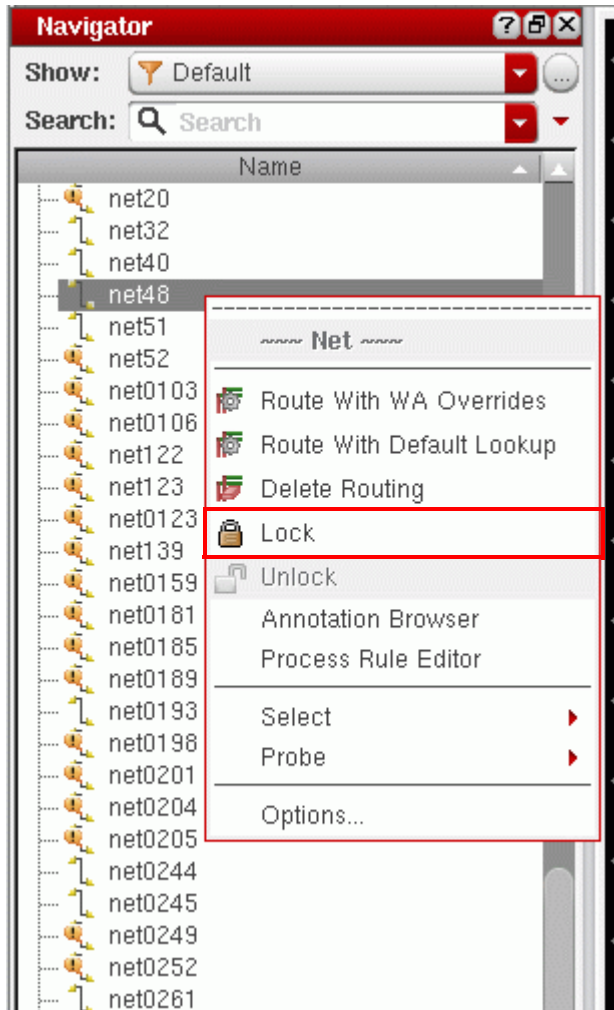
To lock a net by using the Lock command in the Navigator Assistant shortcut menu:

1. Right-click a net or pin in the Navigator Assistant.

Virtuoso Layout Suite XL User Guide

Preparing Your Design for Routing

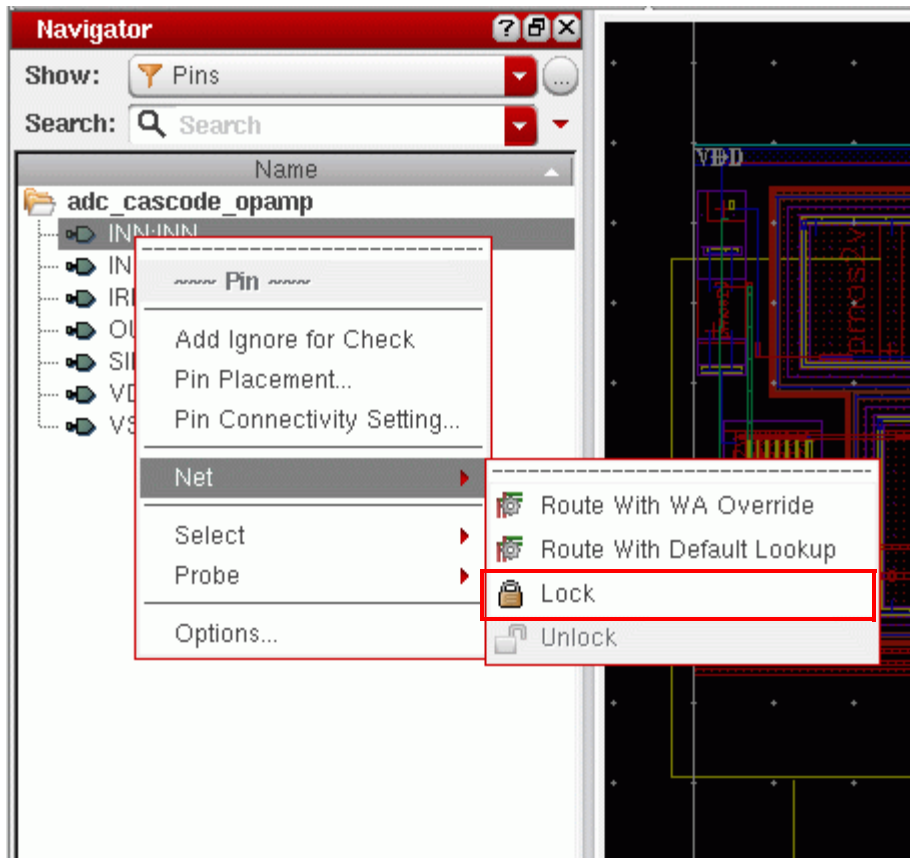
If you selected a net, the *Lock* and *Unlock* commands are displayed in the Navigator Assistant shortcut menu.



Virtuoso Layout Suite XL User Guide

Preparing Your Design for Routing

If you selected a pin, the *Lock* and *Unlock* commands appear under the **Net** sub menu.



2. To lock the net, select the *Lock* command.

This locks the position of the selected net on the canvas, making it impossible to move the net to a new position unless the net is unlocked.

Note: To unlock a net, select the *Unlock* command. Unlocking makes the net open for relocation along the canvas.

See also [Locking and Unlocking a Net from the Canvas](#).

Locking and Unlocking a Net from the Canvas

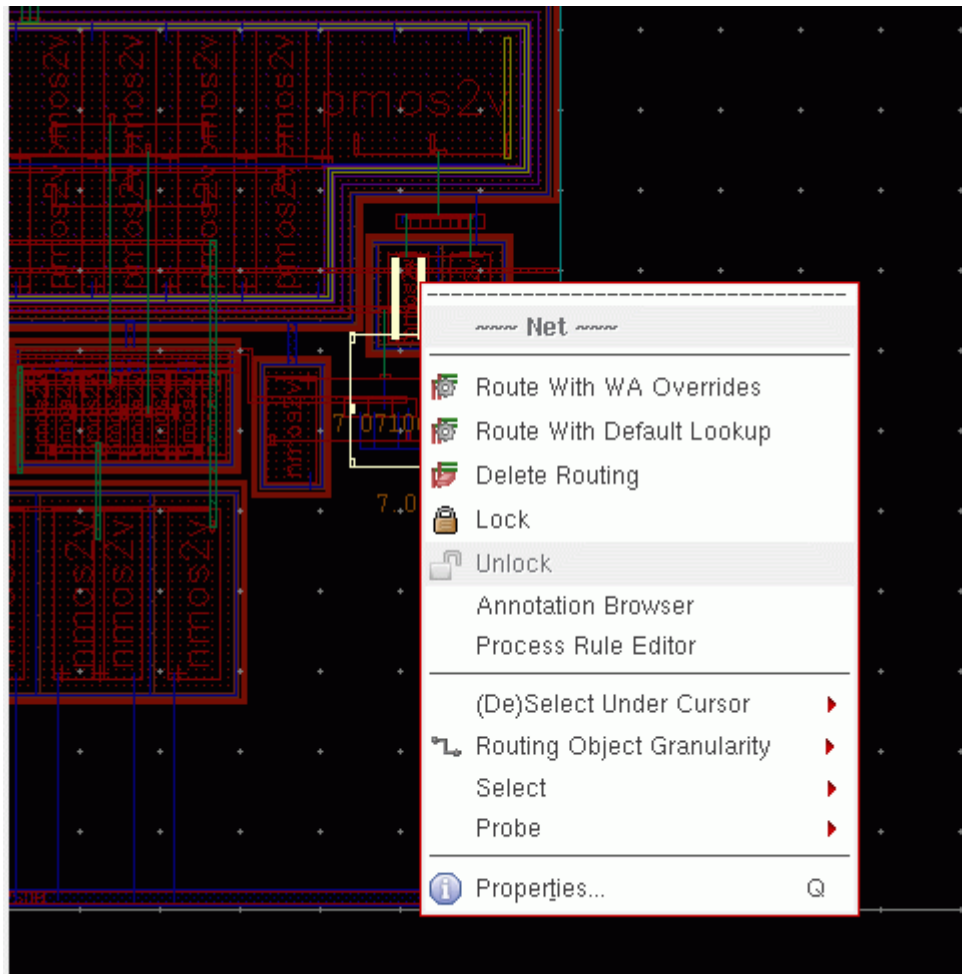
To lock a net from the shortcut menu available on the Canvas:

1. Right-click a net, shape, or a pin on the canvas.

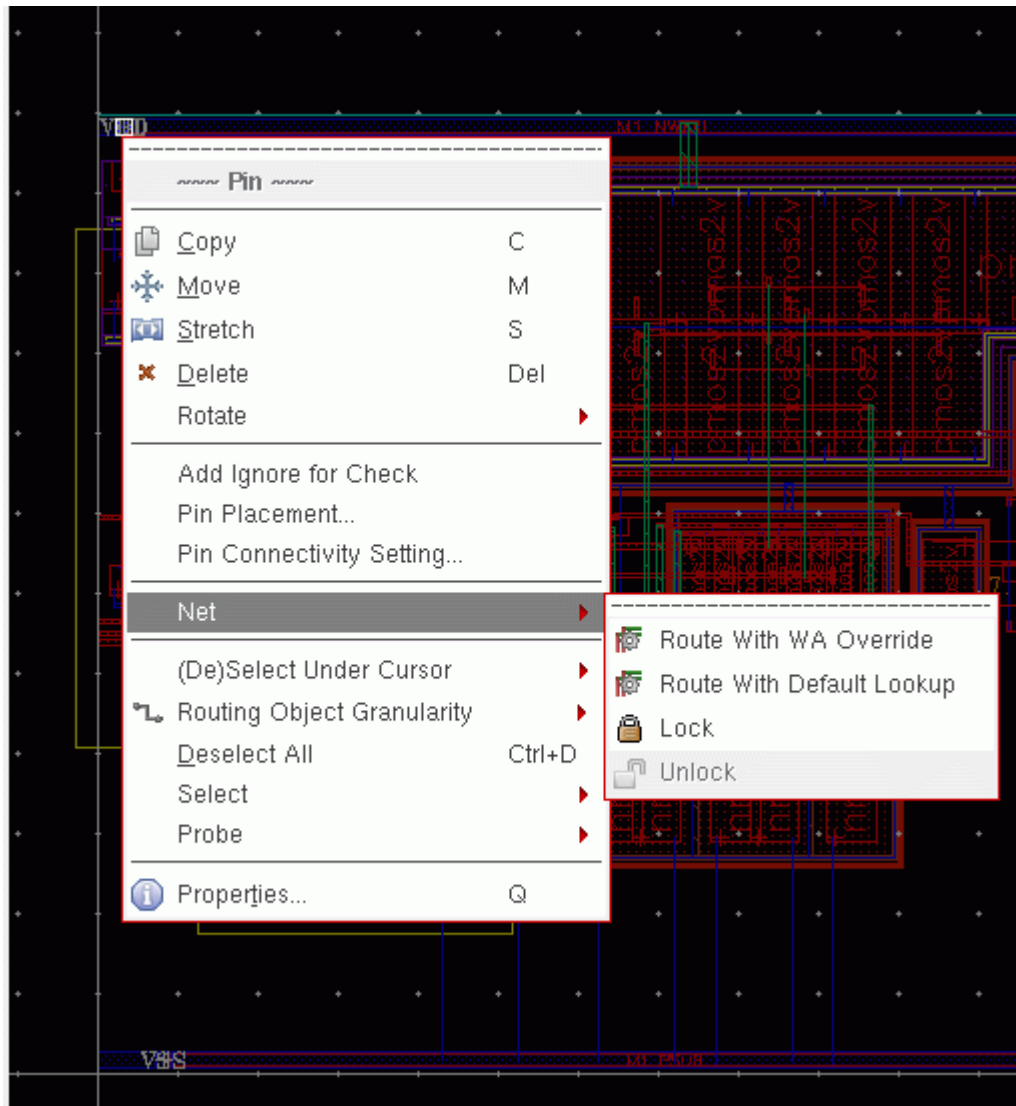
Virtuoso Layout Suite XL User Guide

Preparing Your Design for Routing

The shortcut menu is displayed.



If you selected a pin, the *Lock* and *Unlock* commands appear under the **Net** submenu.



2. To lock the net, select the *Lock* command.

This locks the position of the selected net on the canvas, making it impossible to move the net to a new position unless the net is unlocked.

Note: To unlock a net, select the *Unlock* command from the shortcut menu. Unlocking makes the net open for relocation along the canvas.

See also [Locking and Unlocking a Net from the Navigator Assistant](#).

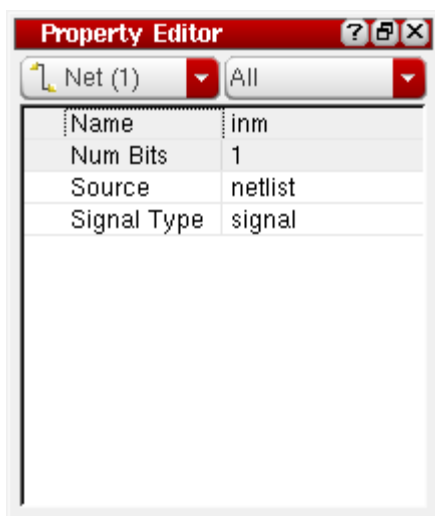
Editing Net Attributes and Properties

You can create and edit the attributes and properties on existing nets using the Property Editor assistant.

To edit net attributes,

1. In the Navigator assistant, select the net whose properties you want to edit.
2. From the layout window menu bar, choose *Windows – Assistants – Property Editor*.

The Property Editor assistant is displayed.



3. To change the signal type for a net,
 - a. Click in the value column next to the *Signal Type* label to reveal a drop-down list of all the available signal types.
 - b. Choose the signal type you want from the list.
4. To change the net assignment for a selected object, click in the *Net Name* field and type in the new net name for the object.

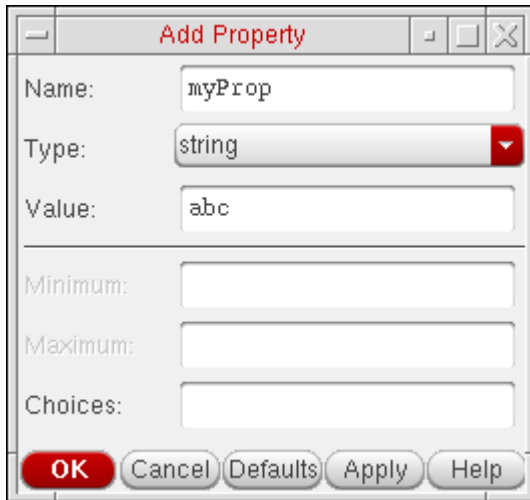
Note: When the selected object is a route element (i.e., a via or a pathSeg) the *Net Name* field is relabeled *Route Net Name*. This indicates that changing the net name for that object also changes the net name for the entire route.
5. To delete a net assignment, remove the name of the net from the *Net Name* field.
6. To add a new property for a selected net,

Virtuoso Layout Suite XL User Guide

Preparing Your Design for Routing

- a. Click the right mouse button in the Property Editor assistant and choose *Add property to...*

The Add Property dialog is displayed.

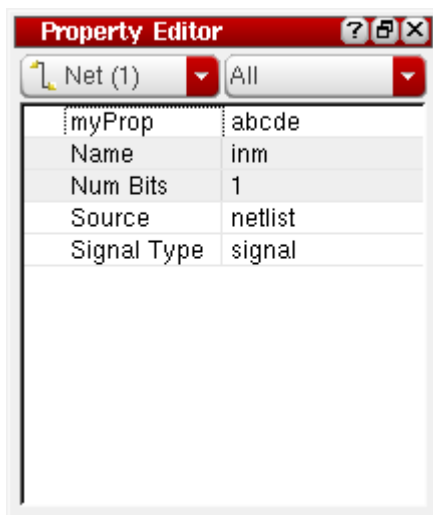


The Add Property dialog box is shown with the following fields and buttons:

- Name: myProp
- Type: string (dropdown menu)
- Value: abc
- Minimum: (empty text box)
- Maximum: (empty text box)
- Choices: (empty text box)
- Buttons: OK, Cancel, Defaults, Apply, Help

- b. Define the new property and click *OK* or *Apply* to add it to the net.

The Property Editor is updated to show the property you just added.



The Property Editor dialog box is shown with the following fields and table:

- Net (1) (dropdown menu)
- All (dropdown menu)

myProp	abcde
Name	inm
Num Bits	1
Source	netlist
Signal Type	signal

For more information on Property Editor functionality, see [The Property Editor Assistant](#) in the *Virtuoso Schematic Editor XL User Guide*.

Creating and Editing Net Constraints

You can use a combination of the navigator assistant and the Constraint Manager assistant to create and edit *Net Priority* and *Net Class* constraints on existing nets in your design.

Creating a Net Priority Constraint

A *Net Priority* constraint is an ordered net constraint which defines the order of priority when routing a net. A higher priority implies that the net is more critical and therefore needs to be routed more optimally (in terms of wire length and any other user-defined constraint on it), even if that makes routing of the lower priority nets less optimal.

Net Priority constraints are honored by the Virtuoso Floorplanner, Virtuoso Analog Placer, Virtuoso Chip Assembly Router, and the Virtuoso Space-based Router.

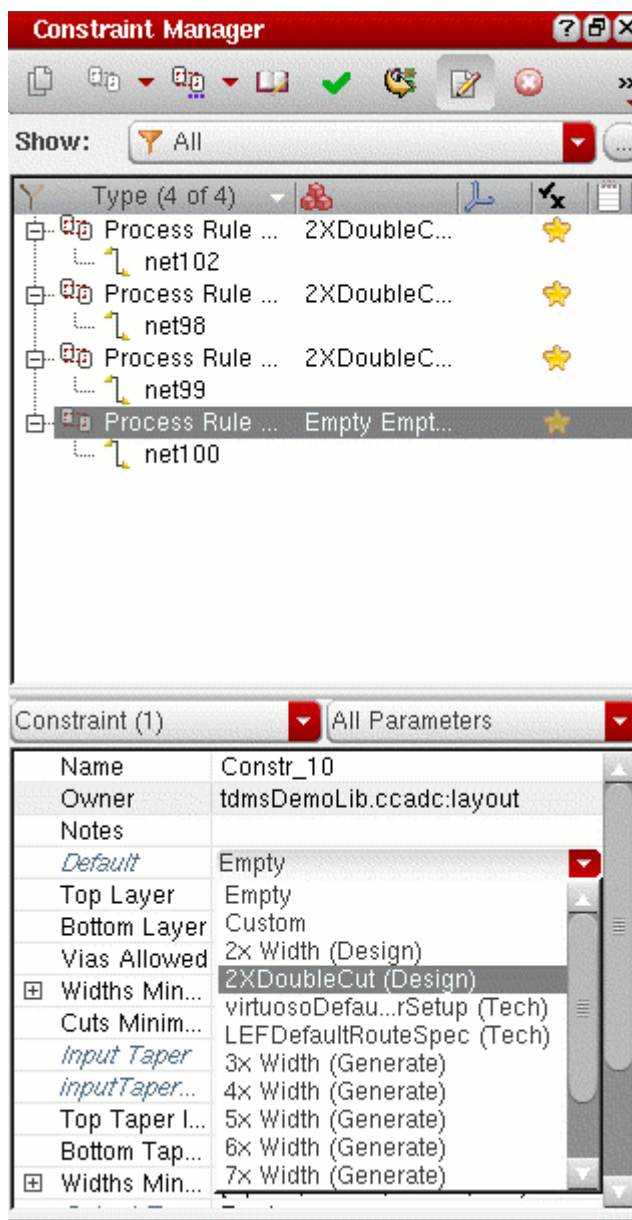
To create a *Net Priority* constraint,

1. From the layout window menu bar, choose *Windows – Assistants – Constraint Manager*.

Virtuoso Layout Suite XL User Guide

Preparing Your Design for Routing

The Constraint Manager assistant is displayed.

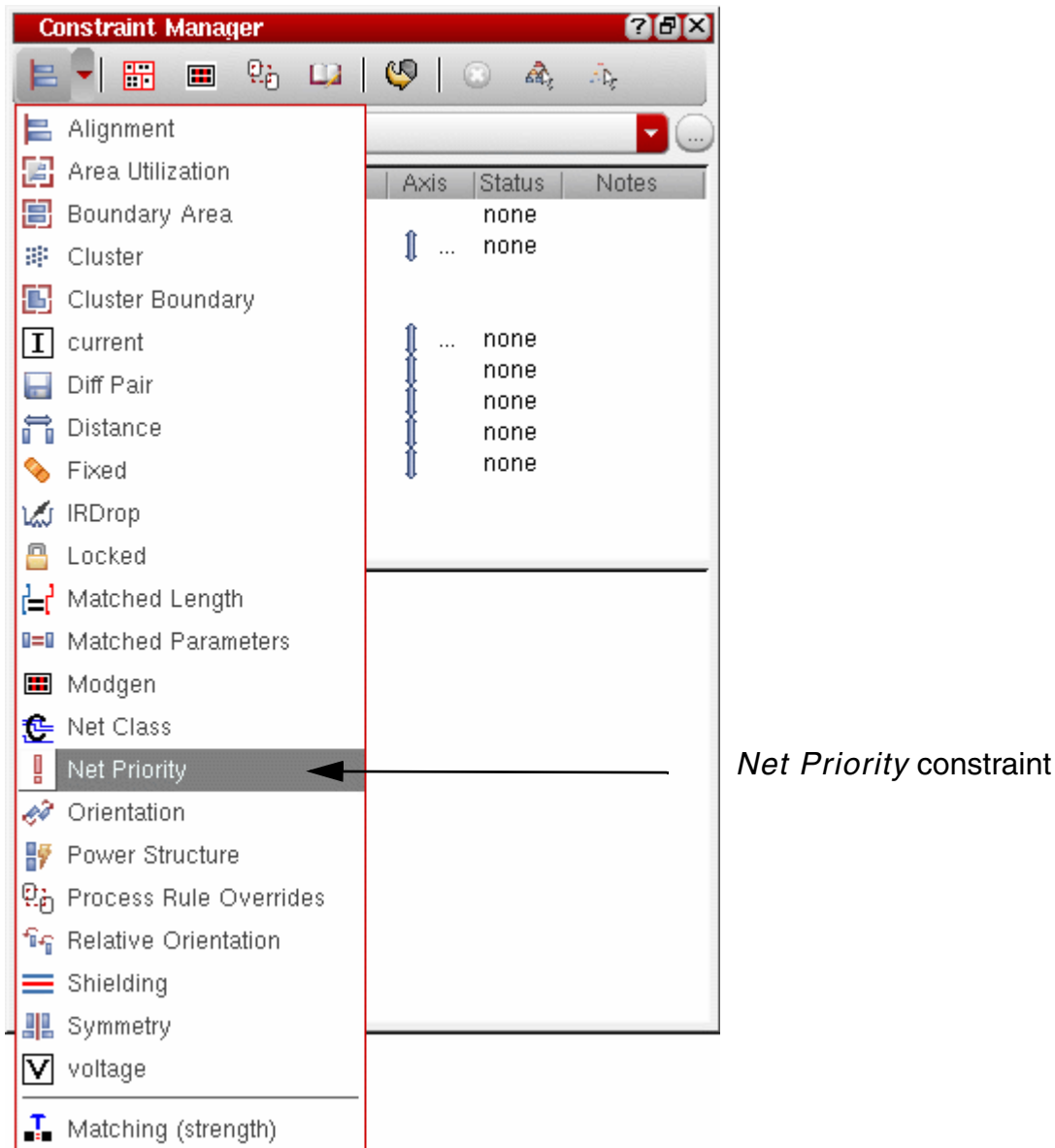


Constraint Manager table

Constraint Editor

2. In the Navigator assistant, select the first net in your ordered net constraint.

3. In the Constraint Manager assistant, pull down the constraint list and choose *Net Priority*.



The Create Priority dialog is displayed.

4. Type in the priority for the selected net (for example, 1).

Note: Zero is not a valid value for the net priority. Enter a non-zero integer between -128 and 128.

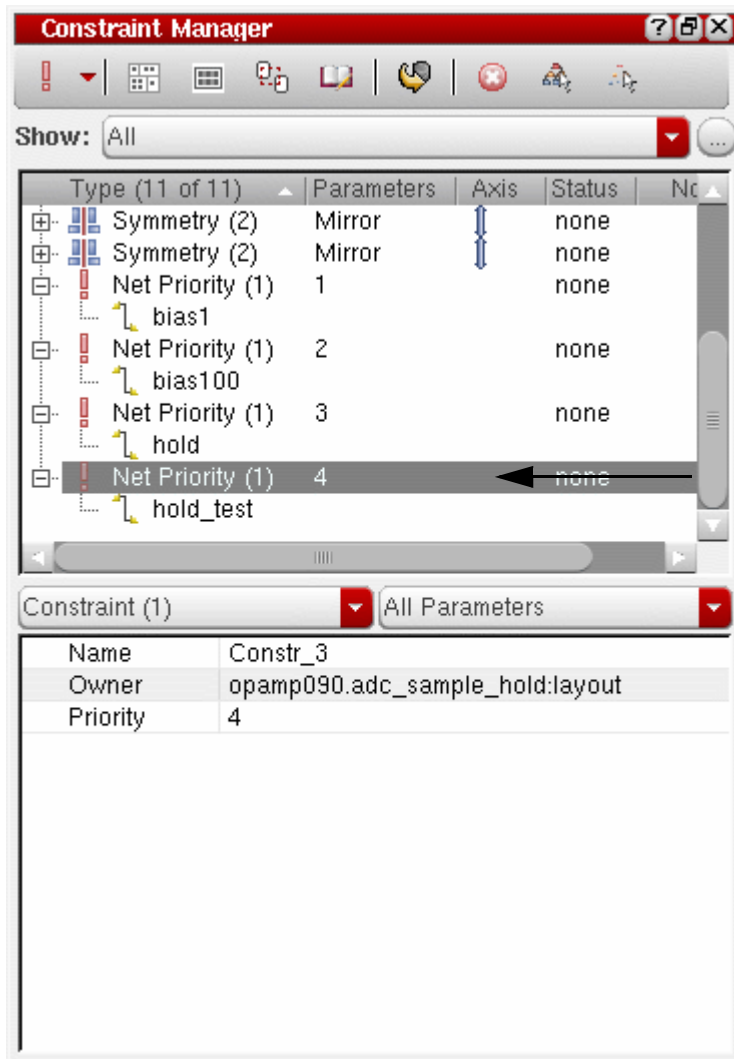
5. Click *OK* to create the constraint.

Virtuoso Layout Suite XL User Guide

Preparing Your Design for Routing

It is added to the list in the Constraint Manager assistant.

- Repeat steps 2 thru 5 for each additional net you want to add to the ordered net constraint.



Constraints defined

Note: The nets need not belong to the same logical bus. If you assign the same priority to two nets, both nets are added to a single *Net Priority* constraint.

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

Creating a Net Class Constraint

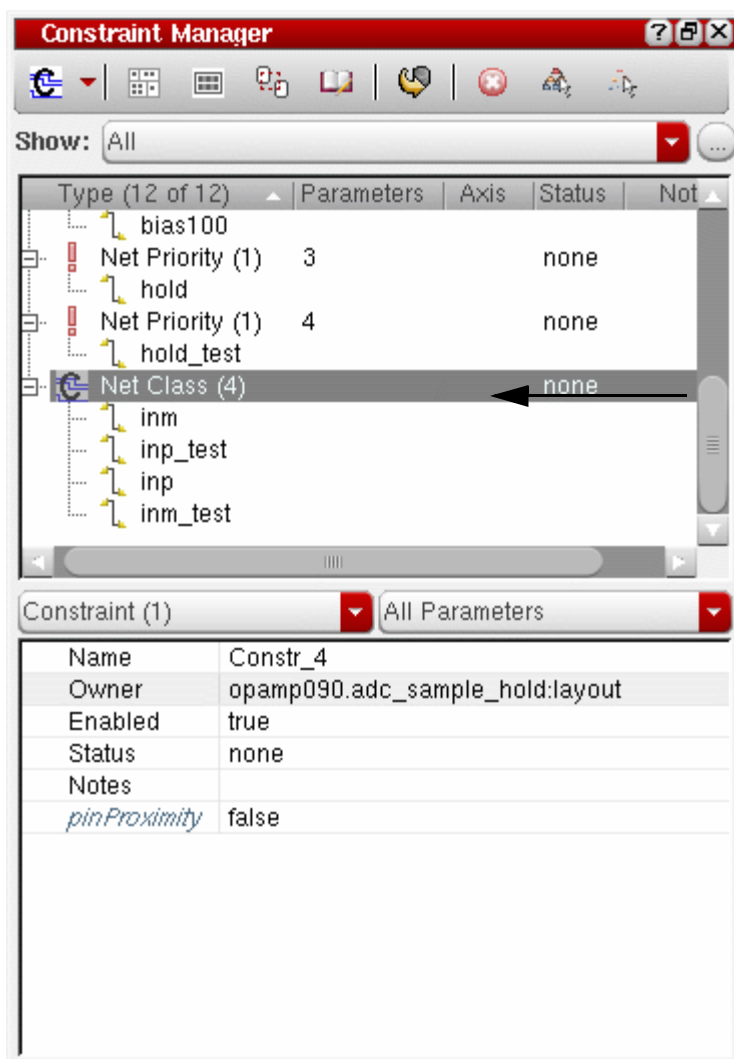
A *Net Class* constraint defines a group of nets that have a specific set of common constraints (typically process rule overrides) on or between its members, or with respect to other nets or groups of nets in the design. *Net Class* constraints are honored by the Virtuoso Floorplanner.

To create a *Net Class* constraint,

1. In the Navigator assistant, select the nets you want to group into a *Net Class* constraint.
2. From the layout window menu bar, choose *Windows – Assistants – Constraint Manager*.

The Constraint Manager assistant is displayed.

3. In the Constraint Manager assistant, pull down the constraint list and choose *Net Class*.



Constraints defined

The *Net Class* constraint is created with the members you specified and is added to the list in the Constraint Manager assistant.

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

Making Net Connections

There are two main ways to create interconnect in Layout XL.

- Create a wire using the *Create – Wire* command.

For more information, see the [Interactive Wire Editing](#) in the *Virtuoso Space-based Router User Guide*.

- Create objects such as paths, rectangles and polygons using the *Create Path*, *Create Polygon*, and *Create Rectangle* commands.

For more information, see [Creating Objects](#) in the *Virtuoso Layout Suite L User Guide*.

Probing Nets

To highlight nets while you are creating interconnect, switch on the *Probe nets during object creation* option on the [Display Tab](#) of the Layout XL Options form.

Note: This option is also available in the Create Path, Create Polygon, Create Rectangle, and Create Wire forms.



You can also trace the physical connectivity of a net visually using the *Mark Net* command. For more information, see [Using Connectivity](#) in the *Virtuoso Layout Suite L User Guide*.

Layer Selection

When working in the layout window, you can select a layer purpose pair to work on directly by using the [Layers Assistant](#) in the Palette Assistant or you can switch on the automatic tapping options to specify how Layout XL chooses layers during interactive routing using wires, paths, and shapes.

Selecting a Layer in the Palette Assistant

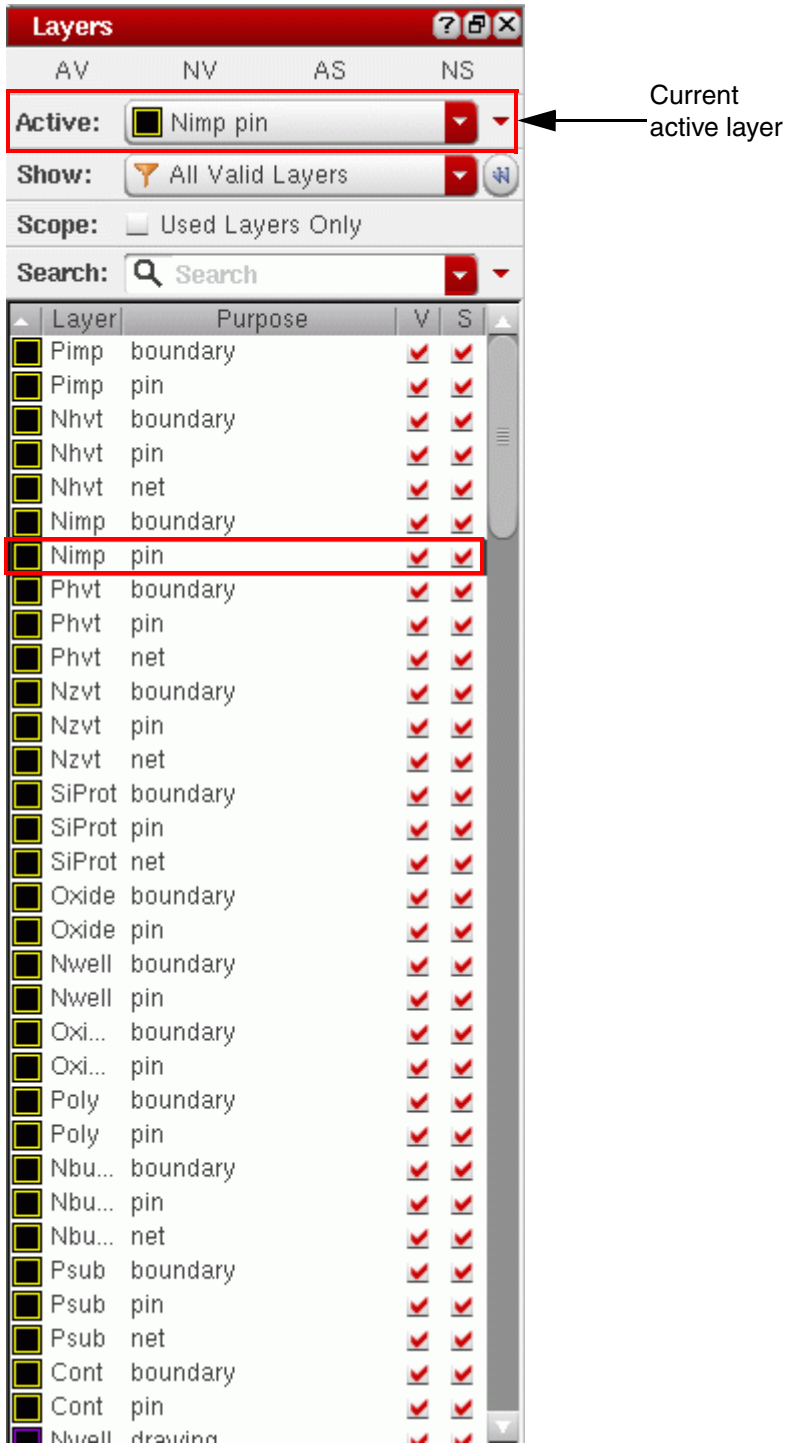
To select the layer you want to use to connect design components,

- In the Layers Assistant, select the layer purpose pair you want to use.

Virtuoso Layout Suite XL User Guide

Preparing Your Design for Routing

The layer purpose pair you selected appears as the current active layer.



Selecting a Layer Automatically when Creating a Wire

The *Wire Auto Tap* option in the Layout Editor Options form controls which layer is chosen automatically when using the *Create – Wire* command.

By default this option is switched on when you are using Layout XL. This means that when you click on an existing net on a particular layer, Layout XL changes to the same layer and creates the wire on that layer.

If you click on overlapping layers, one of two things will happen depending on whether the *Select from Overlaps* option in the Layout Editor Options form is on or off.

- If *Select from Overlaps* is on, a dialog pops up allowing you to choose which layer to use.
- If *Select from Overlaps* is off, Layout XL chooses the layer to use based on the order in which the layers are defined in the technology file.

For more information, see the [Interactive Wire Editing](#) in the *Virtuoso Space-based Router User Guide*.

Selecting a Layer Automatically when Creating a Path or Shape

The *Shape Auto Tap* option provides the same functionality in relation to the *Create Path*, *Create Polygon*, and *Create Rectangle* commands. *Shape Auto Tap* is off by default in Layout XL.

For more information, see [Creating Objects](#) in the *Virtuoso Layout Suite L User Guide*.

Flight Lines

For information on how to display flight lines depicting incomplete nets in the layout canvas, see [Working with Incomplete Nets](#).

Layout XL draws flight lines showing incomplete electrical connections between the devices on each net. The CIW reports how many incomplete nets there are.

If you draw a path between two components that completes the connection, the flight lines disappear. If the path does not complete the connection, the flight lines remain.

Markers

Markers are flashing boxes in the layout canvas that indicate electrical shorts or invalid overlaps.

Finding Markers

To find markers,

1. From the layout window menu bar, choose *Verify – Markers – Find*.

The Find Marker form is displayed.

The screenshot shows the 'Find Marker' dialog box with the following sections:

- Severity:** A group box containing nine checkboxes, all of which are checked: 'all', 'none', 'fatal error', 'error', 'acknowledged warning', 'critical error', 'signed off error', 'info', 'signed off critical error', 'warning', and 'annotation'.
- Search Scope:** A group box containing three radio buttons: 'cellview' (selected), 'hierarchy', and 'library'. Below 'hierarchy' is a 'starting from:' label with two radio buttons: 'top cellview' (selected) and 'current cellview'. Below 'library' is an empty text input field.
- Options:** A group box containing three controls: 'Zoom To Markers' with an unchecked checkbox; 'Cellview Access Mode:' with 'read' and 'write' radio buttons, where 'write' is selected; and 'Find by:' with 'master cell only' and 'all occurrences' radio buttons, where 'all occurrences' is selected.

At the bottom of the dialog are six buttons: 'Apply' (highlighted in red), 'Cancel', 'Previous', 'Next', 'Delete', and 'Help'.

For more information on this form, see [Finding Markers](#) in the *Virtuoso Layout Suite L User Guide*.

2. Turn on *Zoom To Markers* and click *Apply*.

The layout window zooms in on the first marker with one of the selected severity levels.

3. To move to the next marker, click *Next*. To revisit the last marker, click *Previous*.

4. To exit the command, click *Cancel*.

Explaining Markers

To find out what each marker means, follow these steps.

1. From the layout window, choose *Verify – Markers – Explain*.

The layout window prompts you to point at a marker.

2. Click on the marker you want to know more about.

You see an information window identifying the location of the marker and explaining why it was generated.

3. Click on another marker to explain or press `ESC` to exit the command.

Deleting a Single Marker

To delete a single marker,

1. From the layout window menu bar, choose *Verify – Markers – Delete*.

The layout window prompts you to point at the marker you want to delete.

2. Click on the marker to delete.

The marker disappears.

3. Click on another marker to delete or press `ESC` to exit the command.

Deleting All Markers

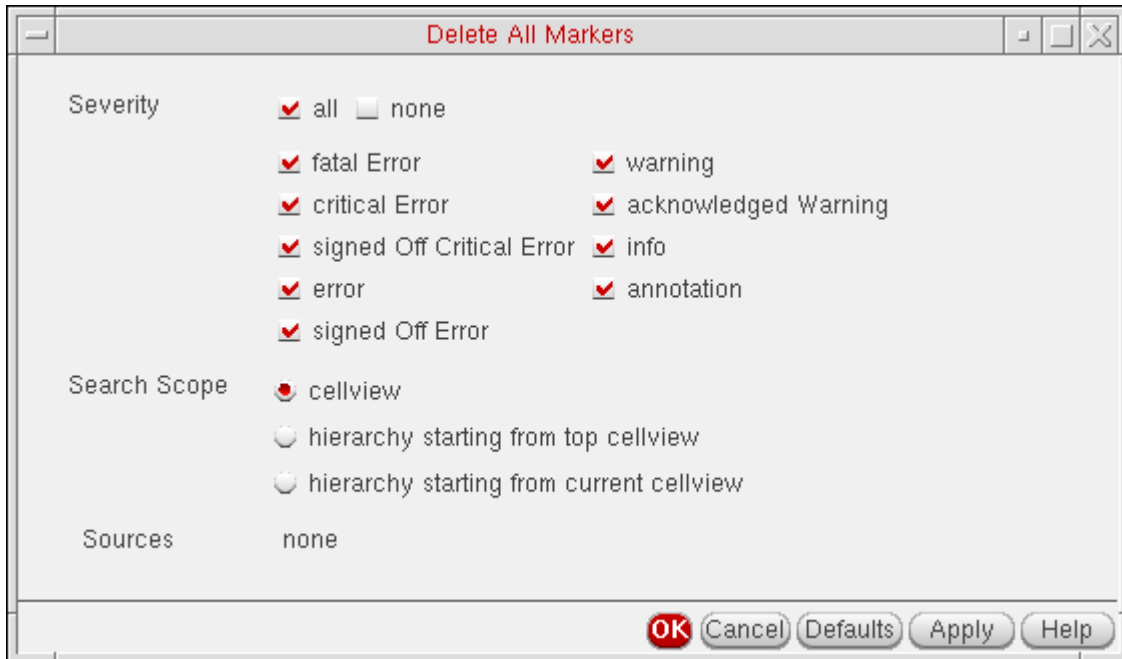
To delete all the markers in your design,

1. From the layout window menu bar, choose *Verify – Markers – Delete All*.

Virtuoso Layout Suite XL User Guide

Preparing Your Design for Routing

The Delete All Markers form is displayed.



2. Choose the types of markers to delete and the levels of hierarchy from which you want to delete them.

For more information, see [Deleting All Markers](#) in the *Virtuoso Layout Suite L User Guide*.

3. Click *OK*.

All the specified markers disappear.

Virtuoso Layout Suite XL User Guide

Preparing Your Design for Routing

Checking Design Data

This chapter explains how to use the Virtuoso® Layout Suite layout editor (Layout XL) to check your design as you work.

This chapter discusses the following topics.

- [Probing](#) on page 502
- [Checking Shorts, Incomplete Nets, and Illegal Overlaps](#) on page 517
- [Check Against Source](#) on page 519
- [Checking Manufacturability](#) on page 525

Probing

Probing lets you select an instance, net, or pin in the layout or schematic window to highlight the corresponding element in the other window.

- Use the options in the right mouse button menu to specify how nets, instances, and terminals are probed and which layer and colors are used to draw the probes. Also use the right mouse button to remove existing probes.
- Use the options in the XL Probe form to filter the objects that can be selected by clicking in a window and to locate a specific object to probe.
- Use the Highlight Options form to specify how nets, instances, and terminals are probed and which layer and colors are used to draw the probes.

The probes remain even after probing is cancelled, meaning that you can use the results of probing as the basis for further operations in the layout, such as moving and stretching connected objects, increasing contact sizes, or adding well ties.

Note: To display flightlines when probing an instance or a net, select the Net granularity using the right mouse button contextual menu or the Selection Options form. Then, choose the *Options - Highlight Options* command to display the Highlight Options form. From the Net section, select the **FlightLines** option. You should now be able to see flightlines on selecting the instance in the canvas.

General Probing Behavior

You can select the object you want either by clicking in the schematic or layout window, or by choosing it from the list in the XL Probe form. For example,

- If you click on an instance in the layout, the layout instance and the corresponding schematic instance are both highlighted using the display color specified in the Highlight Options form, and a message like the one below is printed in the Information Panel in the XL Probe form.

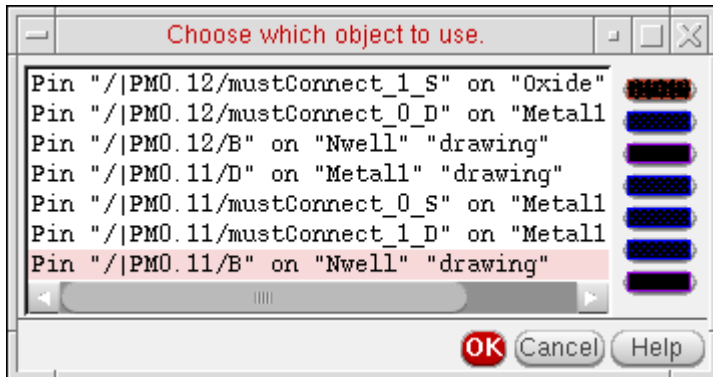
```
inst: (lay) Q12-> (sch) Q12
```

- If Layout XL cannot find the corresponding schematic object, you see a question mark (?) at the end of the message in the XL Probe form.

```
inst: (lay) Q12-> (sch) ?
```

- If you click on a location where there is more than one object, Layout XL highlights the selected objects in the following order: pins; nets; instances.

- If you click on a location where there is more than one of the same kind of object, a message window opens asking which one you want to probe.



- The probes remain highlighted even after probing is cancelled.

Probing Terminals and Nets in Must-Connect Relationships

The following behavior applies when you probe terminals and nets in must-connect relationships.

- If you probe a terminal in the schematic, all the corresponding terminals (including must-connect terminals) in the layout are cross-probed using the same display color.
- If you probe a net in the schematic, all corresponding nets (including those connected by must-connect terminals) are cross-probed in the layout using the same display color.
- If you probe a terminal in the layout, the corresponding terminal in the schematic is cross-probed using the same display color and the other must-connect terminals in the layout are cross-probed in a different color.
- If you probe a net in the layout, the corresponding net in the schematic is cross-probed using the same display color and the other must-connect nets in the layout are cross-probed in a different color.

Probing Components in OTM, MTM, and MTO Relationships

When you probe a component bound by a one-to-many, many-to-many, or many-to-one mapping, Layout XL highlights all the components in the mapping group.

If you probe an external net of a mapping group, the corresponding external net in the other window is highlighted. If you probe an internal net of a mapping group, the entire mapping group is highlighted.

Probing with Multiple Layout Cellviews

If you have multiple layout cellviews open and you select a design element in one part of a cellview pair (a connectivity source and a layout) Layout XL highlights the corresponding element in any other implementations of the other part. For example, if you have several versions of a layout open, selecting R14 in the schematic highlights R14 in each of the layouts of that schematic.

However, if you have a schematic and two different layouts open, the *XL Probe* command applies to only the layout from which you selected the command.

Probing Objects using the Right Mouse Button

The context menu available via the right mouse button (RMB) provides only relevant probing commands for a selected object. This is a significant usability enhancement that allows faster access to the appropriate commands.

For example, if the selected net does not have a probe added; the RMB menu displays the **Probe – Add Probe** option. After the probe is added, the **Add Probe** option is disabled and the **Remove Probe** option is enabled instead.

The probe options available in the right mouse button context menu are:

- **Add Probe:** Adds a probe to a selected object. The color of the probe added is determined based on the color selected on the Highlight Options form. See [Adding a Probe using the Right Mouse Button](#).
- **Remove Probe:** Removes the probe from a selected object. See [Removing a Probe using the Right Mouse Button](#).
- **Remove All Probes:** Removes all the probes from all the objects in a design. This option also removes the cross probes in the schematic. See [Removing All Probes using the Right Mouse Button](#).
- **hilite drawing <0-9>:** If the selected object is not yet probed, sleeting a hilite drawing color adds a probe in the selected color. If the selected object has already been probed, selecting a hilite drawing will change the color of the probe to the selected color. See [Adding a Probe using the Right Mouse Button](#), [Modifying a Probe using the Right Mouse Button](#), and [Dynamic Probing](#).

Adding a Probe using the Right Mouse Button

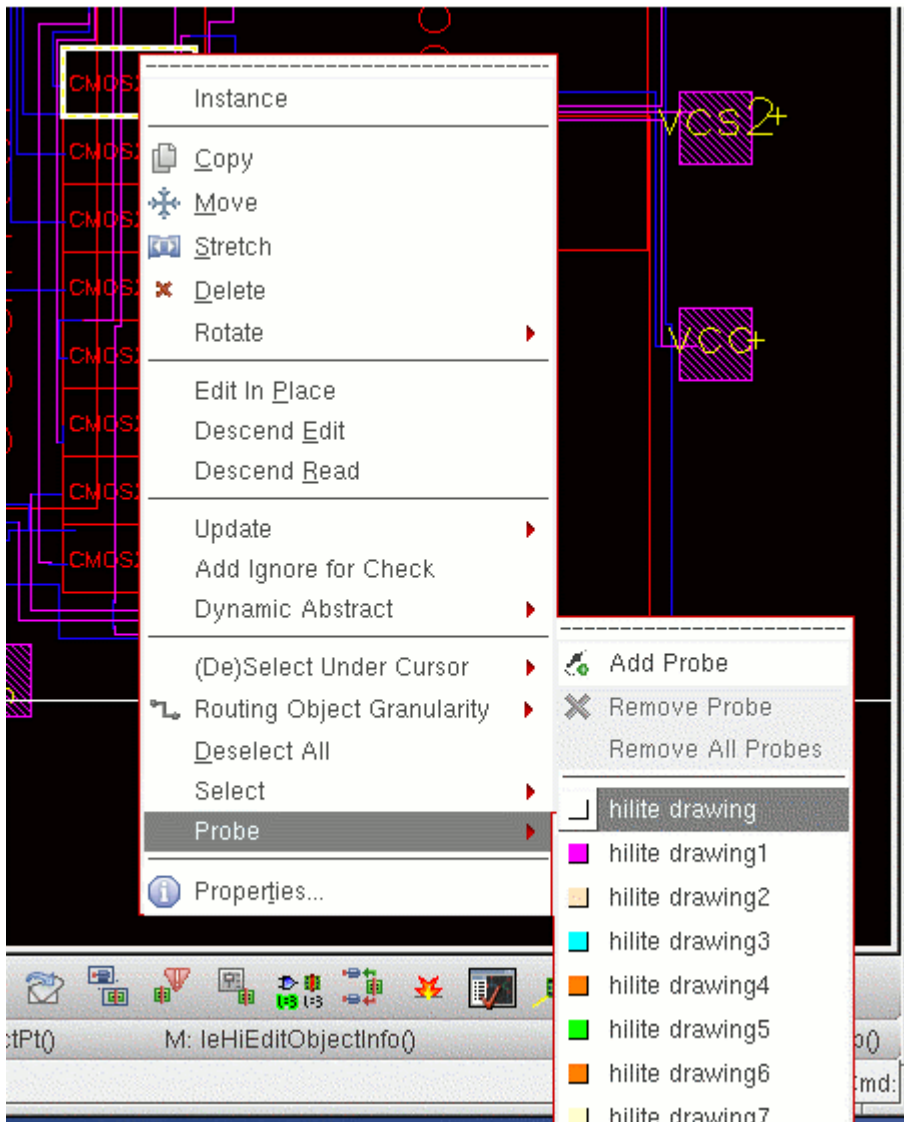
To probe an object by using the RMB:

Virtuoso Layout Suite XL User Guide

Checking Design Data

1. Select the object in the canvas or the navigator and click the right mouse button.

A context menu appears displaying the probing options. A context menu appears displaying the probing options.

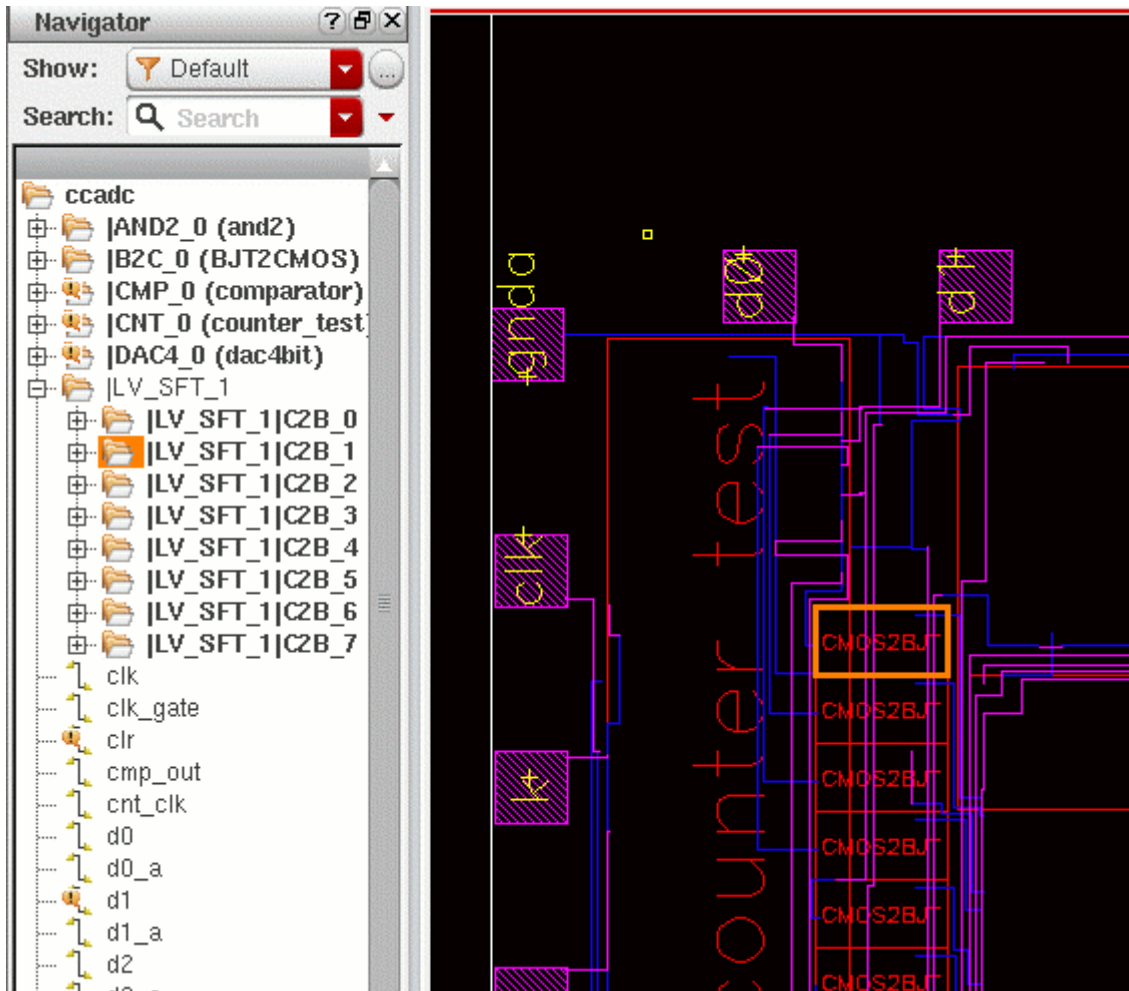


2. Click **Add Probe**.

A colored probe is added to the selected object in the layout view. A similar, corresponding probe is also added in the schematic view.

Note: To probe an object only in the schematic view, you must select the object in the schematic navigator.

The new probe you added also displays in the Navigator Assistant.



The orange highlight in both the views indicates the probe

The Navigator Assistant not only displays new probes added in the canvas, but you can also use it to add new probes to the design. In fact, a probe you add to a higher level in the navigator tree automatically applies to the lower levels also. Therefore, you can use the Navigator Assistant for probing objects across the design hierarchies. This implies that probing a net at any level in the hierarchy will probe that net anywhere else it appears in the navigator, up or down the tree.

A probe on a lower instance will probe itself “up” the way. So, if instance I2, lies within instance I1; a probe you place on I2 will automatically propagate “up” to display itself on I1 also.

To probe an object, select the Navigator tree object you want to probe and click the right mouse button. From the options displayed, select the appropriate probing command.

The right mouse button probing options displayed in the Navigator tree are the same as those displayed in the canvas. Depending on the hierarchical level you select, the probe may be applied across the levels in a design. However, only probing an object at the current level of hierarchy will cross probe in the schematic. Probing at any other level of the hierarchy will only probe the layout.

Note:

- ☐ To add a probe in a specific color, select **Probe - hilite drawing<number>** from the color palette in the right mouse button menu.
- ☐ If you want the tool to automatically display subsequent probes in different colors, select the **Options - Highlight** command. Under the **Display Layer** section, select the **Cycle** check box. This ensures that any new probes added take their color from the palette. The first probe added gets the color at the top of the palette. Any subsequent probes added, get the color next in the sequence.
- ☐ If you want all the new probes to appear in a specific color, you must uncheck the **Cycle** option and specify the color to be used for the highlight.
- ☐ To update a highlight color of an existing probe, see [Modifying a Probe using the Right Mouse Button](#).

Removing a Probe using the Right Mouse Button

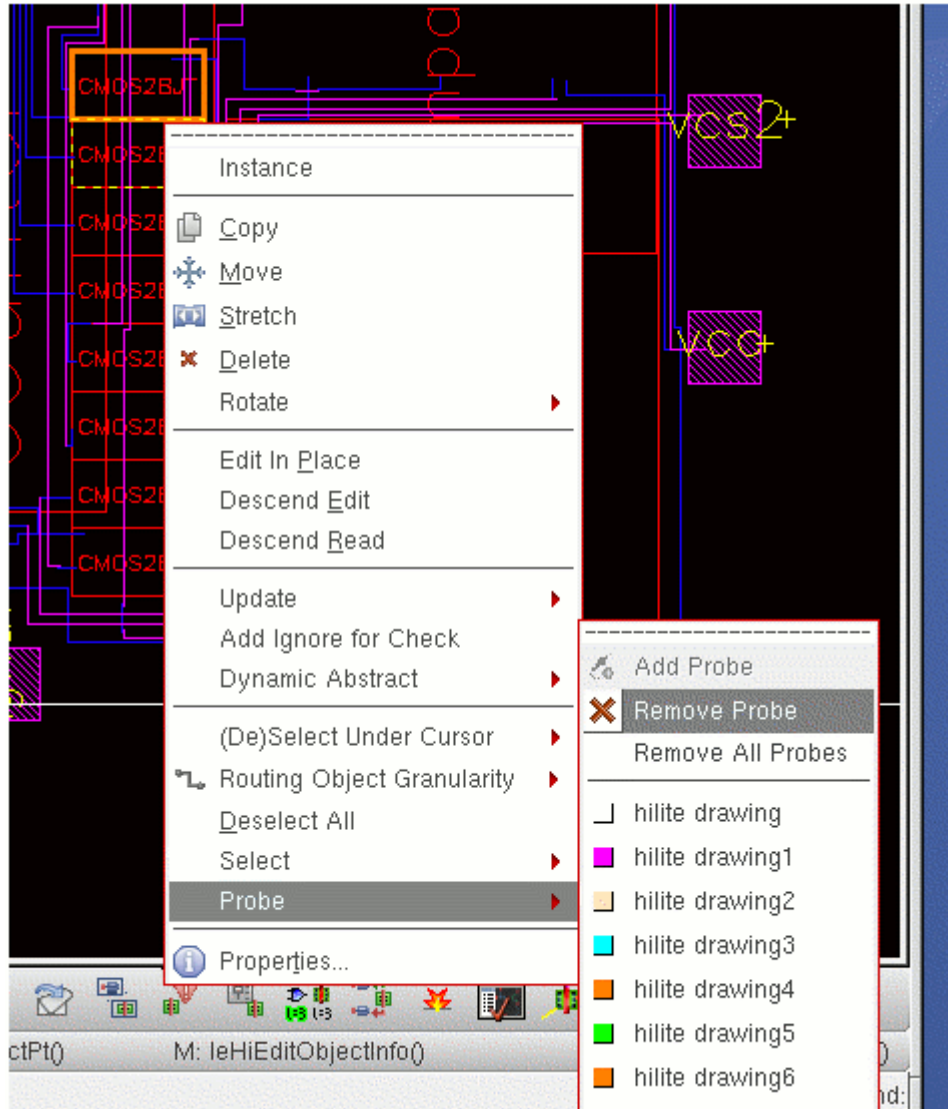
An object may have two types of probes—the one added directly and the ones inherited from objects at a higher level. For the inherited probes to be deleted from the object, you need to remove the probe from where it originates.

For the direct probe to be removed from an object, you can use the context menu available through the right mouse button.

To remove a probe on an object using the right mouse button:

1. Select the object in the layout canvas or the navigator and click the right mouse button.

A context menu appears displaying the probing options.



2. Click **Remove Probe**.

The probe on the selected object is deleted.

Removing All Probes using the Right Mouse Button

Unlike removing probes from specific objects, if you want to remove all the probes in a design with a single click; you can use the **Removes All Probes** option. This removes all the probes in the design, irrespective of the object selected. In fact, the option can also be invoked using the right mouse button with no object selected.

Note: The **Remove All Probes** option is enabled in the context menu only when a design carries one or more probes. This option allows for a single command to be run on several objects, improving usability and performance. In addition, the same command can be used for removing probes across the schematic and layout views. This not only improves performance but also helps ensure consistency across the two views.

Modifying a Probe using the Right Mouse Button

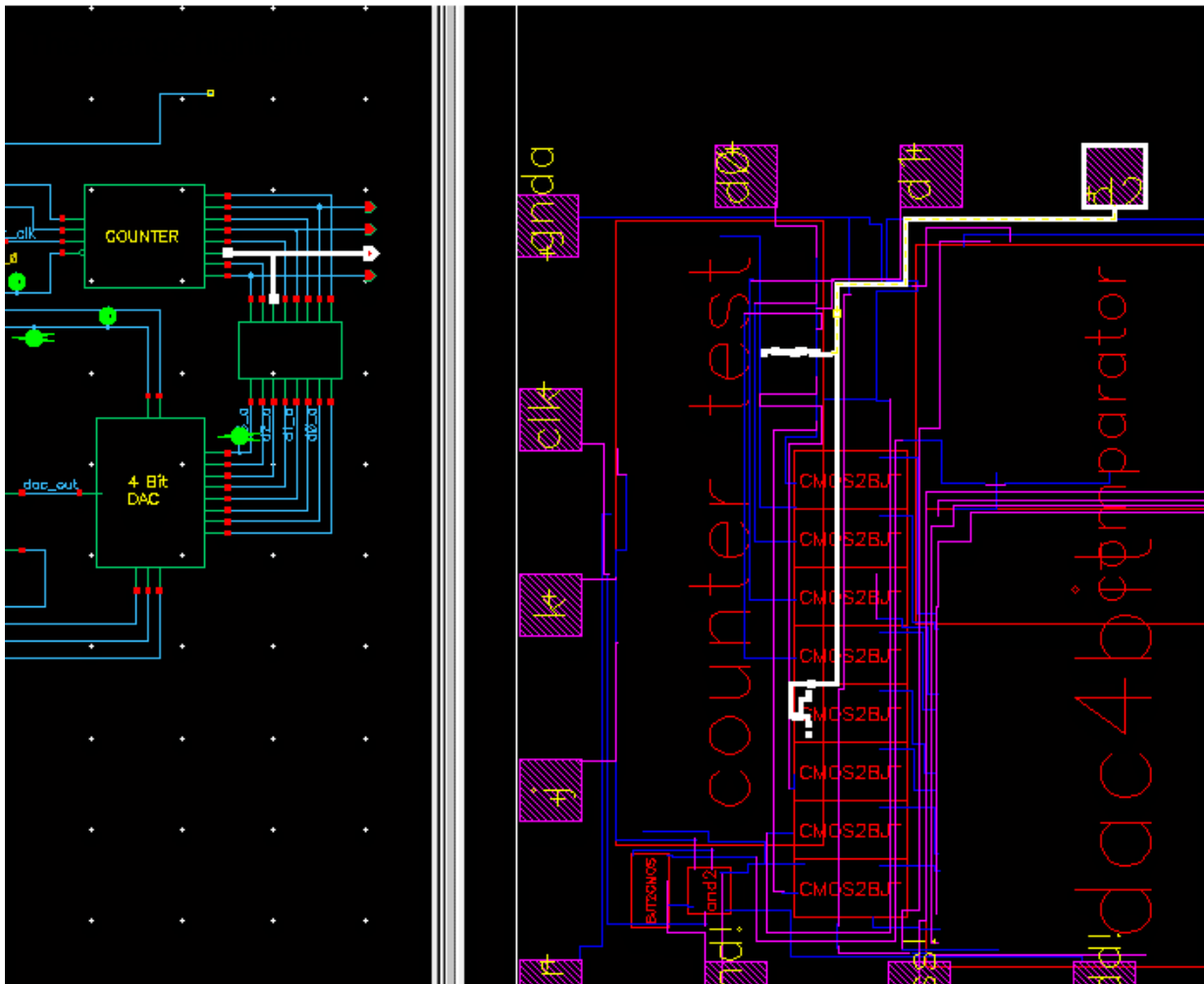
In the context of probing, “modification” implies a change in the probe highlight color. For a probe to be modified, you must first select the object from which the probe originates. Then, select an appropriate color from the palette in the right mouse button menu.

Note: To identify the object from which a probe originates, you can view the tooltip information on the probed object.

To know more about the highlight options available, see [Adding a Probe using the Right Mouse Button](#) and [Dynamic Probing](#).

Dynamic Probing

The Dynamic Probing feature allows for probing an object on mouse over. For each probe created in the canvas, a corresponding probe in the schematic view is also created. However, as soon as the mouse moves off the object; the probe is removed.



The probe in white is a dynamic probe.

The rules for probe highlighting and color selection are the same as those for adding a new probe. For more information, see [Adding a Probe using the Right Mouse Button](#).

To invoke the dynamic probing feature, access **Options – Highlight** and select the **Dynamic Probe** option.

For the probes to be dynamically highlighted in the navigator; make sure the **Display Probe** option on the **Customize Navigator Filters** form is selected. To access the Customize Navigator Filters form, click the ellipsis (...) button adjacent to the **Show** drop-down in the Navigator Assistant.

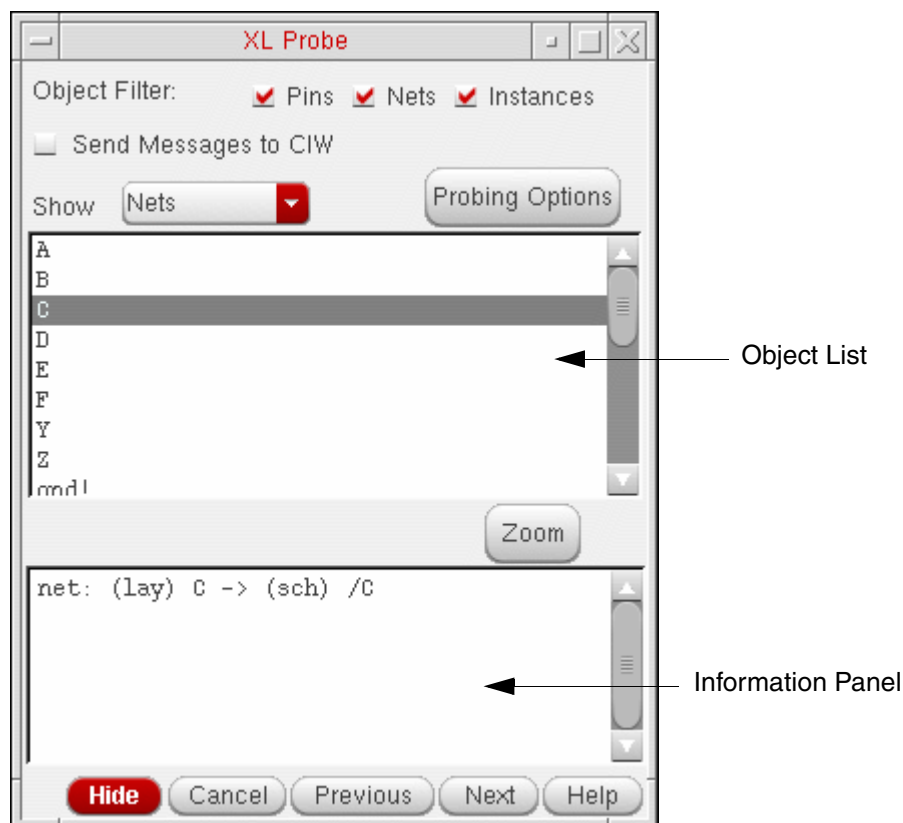
For detailed information on the Navigator, see [The Navigator Assistant](#) in the *Virtuoso Schematic Editor XL User Guide*.

Probing Objects by Clicking in a Window

To probe an object by clicking in either the layout or schematic window,

1. From the layout window menu bar, choose *Connectivity – XL Probe*.

The XL Probe form is displayed.



2. In the *Object Filter* section, specify the types of objects that can be probed when you click in the layout or schematic window.

For example, if you check only *Pins*, you can create probes only for pin objects by clicking in either window. Clicking on a net or instance has no effect. You can use this

option to prevent certain types of objects from being probed in congested areas of your design.

Note: This option has no effect on probe creation using the object list in the XL Probe form, only on probe creation by clicking in the layout or schematic window.

3. Click on the object you want to probe in either the schematic or layout window.

Note: If you click on a pin, Layout XL probes only the `pinFigs` associated with the pin. If you click on a net, Layout XL probes all the shapes on the specified net, including routing shapes and `pinFigs`.

Information on the probed object is printed to the Information Panel in the XL Probe form and the relevant objects are highlighted in the schematic and layout windows.

4. Adjust the probing behavior as required using the options on the form.

- ☐ To display information on the selected device in the CIW, turn on *Send Messages to CIW*.
- ☐ To specify how nets, instances, and terminals are probed and which layer and colors are used to draw the probes, click *Probing Options*.
- ☐ To zoom in on the bounding box of the probed objects, click *Zoom*.

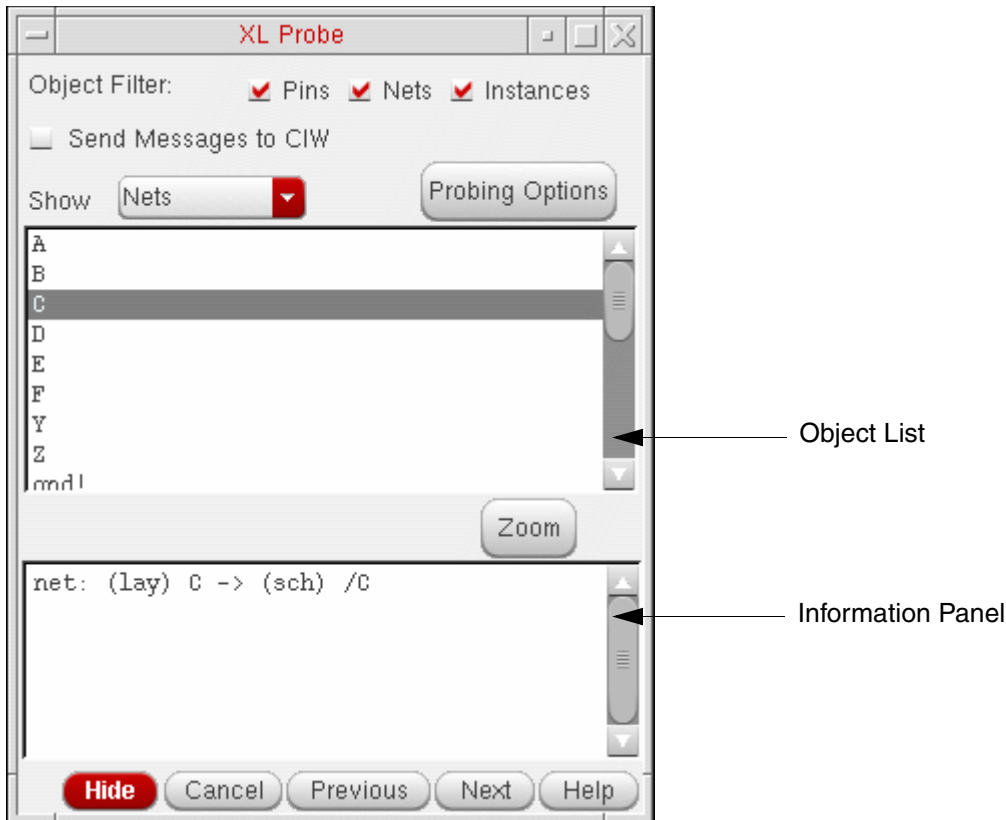
For more information, see [Highlight Options](#) in the *Virtuoso Layout Suite L User Guide*.

Probing Objects using the XL Probe Form

To probe an object from the XL Probe form,

1. From the layout window menu bar, choose *Connectivity – XL Probe*.

The XL Probe form is displayed.



2. From the *Show* drop-down, choose the type of layout object you want to probe: *Pins*, *Nets*, *Net Classes*, or *Instances*.

The Object List is updated to show only layout objects of the specified type.

Note: The *Object Filter* option has no effect on this list, only on probe creation by clicking in the layout or schematic window.

Important

CMX legacy net classes are listed in the XL Probe form but are not shown under the *Net Class* category in the Constraint Manager assistant.

3. Choose the layout object you want to probe from the list.

Information on the probed object is printed to the Information Panel in the XL Probe form, the relevant object is highlighted in the layout window, and its schematic counterpart in the schematic window.

4. Adjust the behavior as required using the options on the form.

- ❑ To display information on the selected device in the CIW, turn on *Send Messages to CIW*.
- ❑ To specify how nets, instances, and terminals are probed and which layer and colors are used to draw the probes, click *Probing Options*.
- ❑ To zoom in on the bounding box of the probed objects, click *Zoom*.

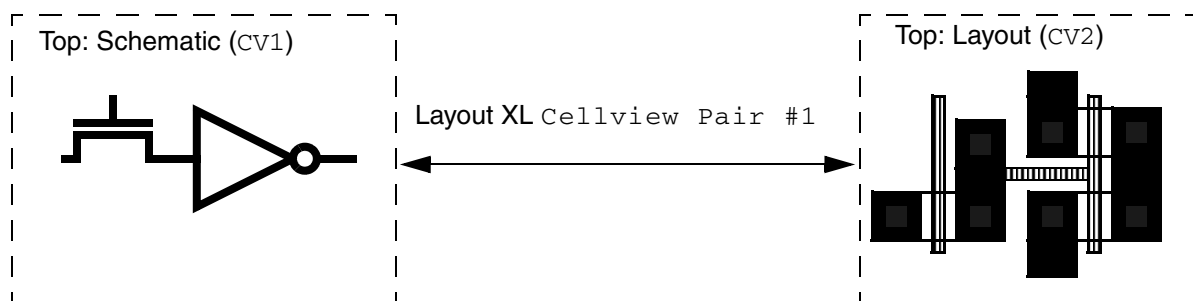
For more information, see [Highlight Options](#) in the *Virtuoso Layout Suite L User Guide*.

Probing a Hierarchical Design

To probe a hierarchical design,

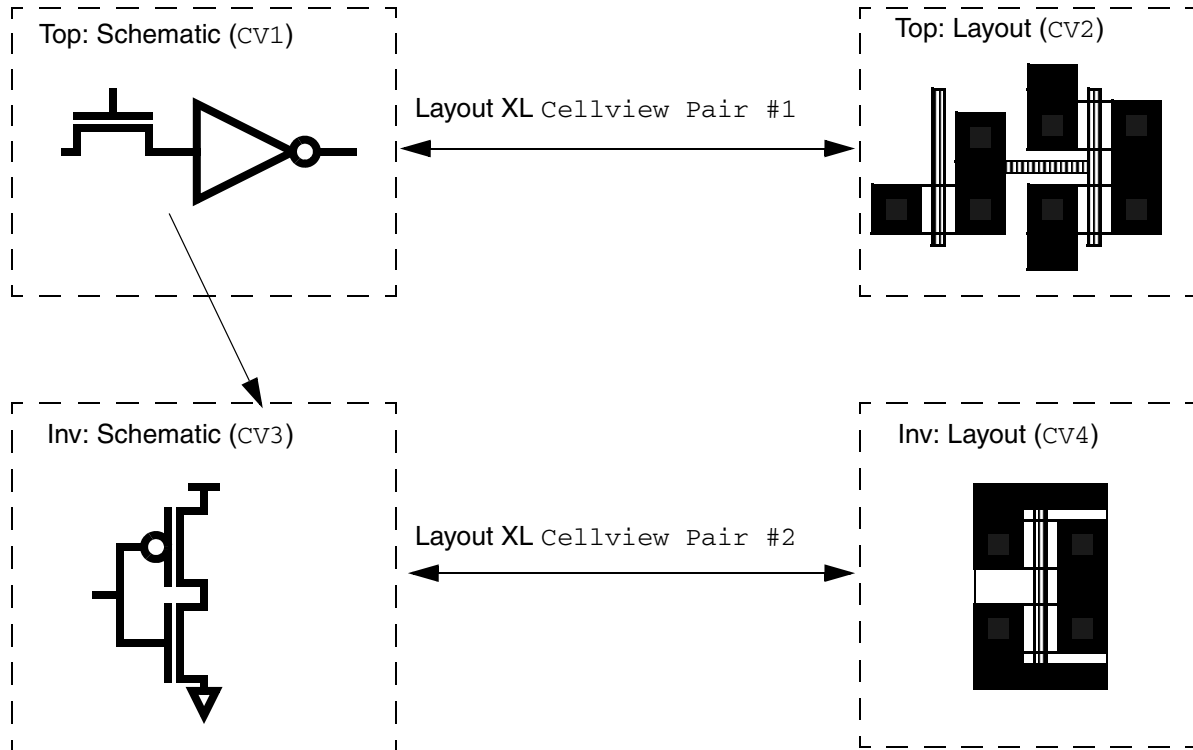
1. Open a hierarchical design in Layout XL.

In the example below, this is Cellview Pair #1.



2. From the layout window menu bar, choose *Connectivity – XL Probe*.
3. Choose the instance to probe; for example, CV1, which is represented in the layout as pair of transistors (a flat representation).
4. Choose *Edit – Hierarchy – Edit in Place*.

The schematic view of the inverter opens (CV3 in the diagram below).



If you probe one of the transistors in this schematic, the corresponding transistor in the layout in the level above is highlighted (CV2 in the diagram).

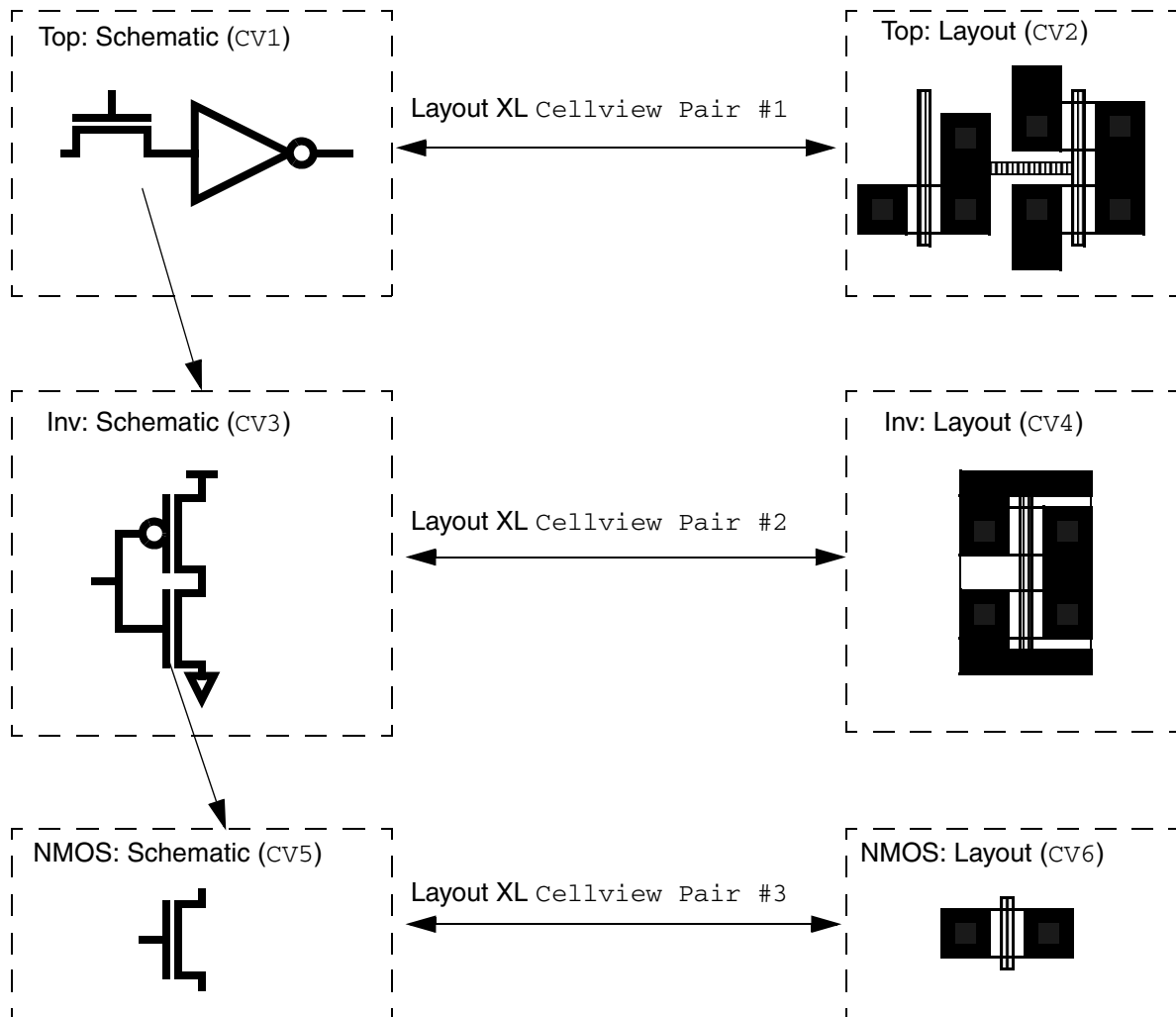
5. In the inverter schematic (CV3), select one of the two NMOS instances.
6. In the schematic window, choose *Launch – Layout XL* to open the layout view for the inverter in Layout XL. This is CV4 in the diagram above.

This creates another cellview pair, Cellview Pair #2.

Virtuoso Layout Suite XL User Guide

Checking Design Data

7. To descend into the schematic of the inverter, select one of the NMOS instances and from the schematic window choose *Edit – Hierarchy – Edit in Place*.



The schematic of the NMOS opens, CV5 in the diagram.

If you probe the transistor in this schematic, the corresponding transistor in the layout in the level above is highlighted (CV4 in the diagram).

8. In the schematic window, choose *Launch – Layout XL* to open the layout of the transistor (CV6 in the diagram) in Layout XL.

This creates another cellview pair, Cellview Pair #3. You can probe from NMOS: Schematic back to Top: Layout, Inv: Layout, and NMOS: Layout.

Removing Probes

To remove an individual probe from the layout,

- Hold down the `Control` key and click on the probe to delete.

To remove all probes from the layout and schematic windows, do one of the following.

- Click on an empty space in the layout canvas.
- Press `Ctrl+I` or type the following SKILL command in the CIW.

```
lxProbeRemoveAll()
```

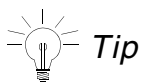
Exiting the Probe Command

To exit the *XL Probe* command,

- Click *Cancel* in the XL Probe form or press `Esc`.

Checking Shorts, Incomplete Nets, and Illegal Overlaps

Use the [Annotation Browser](#) assistant to check the number of incomplete nets, shorts, and illegal overlaps in your design.



Tip

The command relies on connectivity extractor markers. If you have deleted these markers, you must re-extract your design before checking for shorts and opens. For more information, see [Connectivity Extraction](#).

To check the number of shorts, opens, and illegal overlap markers,

1. Extract the design if required by doing one of the following.
 - ☐ From the layout window menu bar, choose *Connectivity – Update – Extract Layout*.
 - ☐ Click the *Extract Layout* button on the Layout XL toolbar.



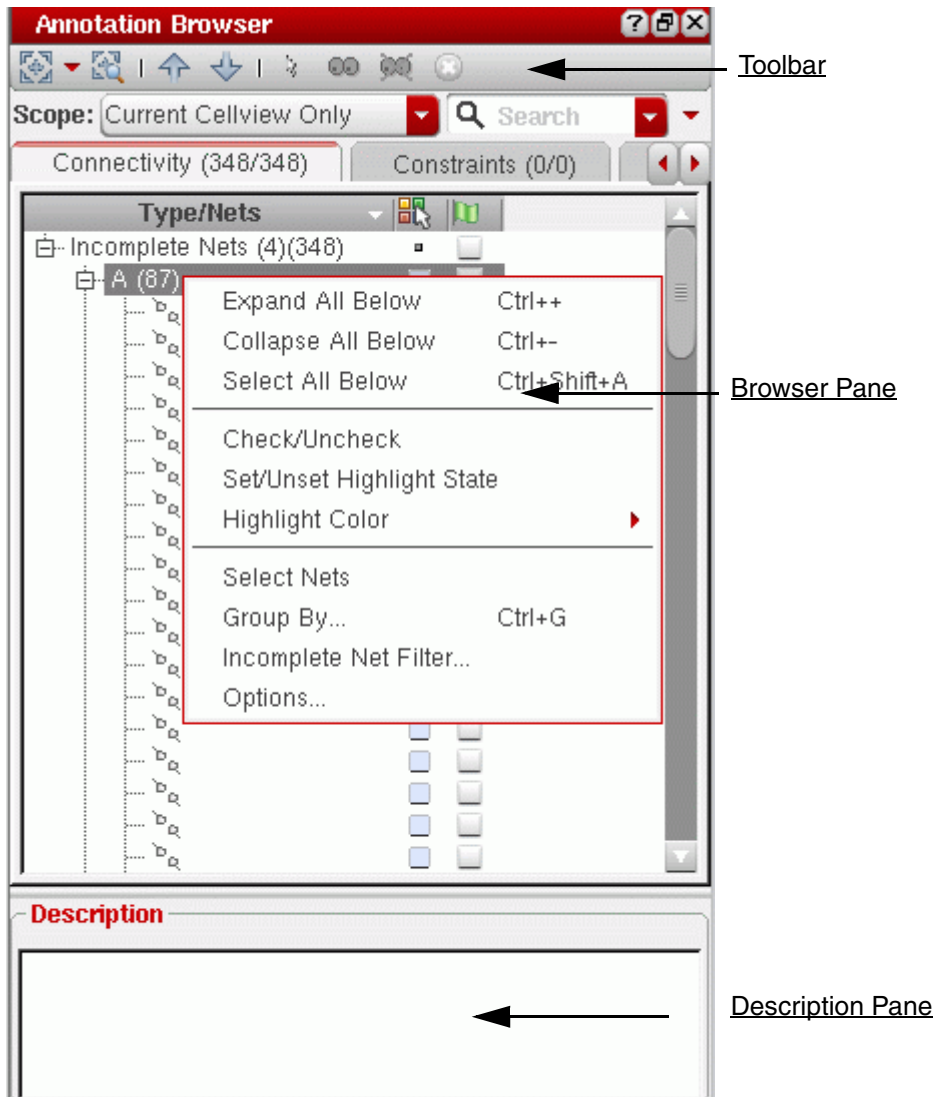
Virtuoso Layout Suite XL User Guide

Checking Design Data

2. Click the *Annotation Browser* button on the Layout XL toolbar.



The Annotation Browser assistant's *Connectivity* tab is displayed.



The tab label indicates the total number of connectivity violations in the design. These are separated into different categories for *Illegal Layer Overlaps*, *Incomplete Nets*, and *Shorts* in the browser pane.

3. Click in the *Set Highlight State* column for the entry whose flight line you want to see.

4. Click in the *Set Highlight Color* column to set the color used to draw the corresponding marker in the design window. Choose *cycle* to let Layout XL select the color automatically by cycling through a predefined list.
5. Click in the *Set Marker Check State* column to set the *Checked* state of the corresponding marker. When a marker is set to *Checked*, you can use the *Hide Checked Markers* button to hide it in the Annotation Browser without deleting the marker from the design window.

Note: Clicking in these columns for a particular node applies the setting for all the entries under that node. Clicking in these columns next to the *Incomplete Nets* category shows all the incomplete net flight lines in the design.

Check Against Source

The *Check Against Source* command lets you check that

- There are no missing or mismatched cellview masters between schematic and layout.
- There are no devices present in the schematic but missing in the layout (and vice versa).

Note: The command does not report missing I/O pins and instances which have a property specified in the propsUsedToIgnoreObjsForCheck environment variable. See Ignoring Components with no Schematic Counterpart.

- There are no connectivity issues on top level pins and global nets; mismatched or missing terminals and instance terminals; or unbound nets in the layout.

Note: *Check Against Source* does not report physical-only terminals (`physOnly=t`) created to support an implicit inherited connection defined using a net expression in the schematic.

- The CDF parameters and cell and instance properties listed for devices in the schematic are associated with equivalent parameters or properties on devices in the layout; and that the values of parameters and properties on the layout and schematic views are consistent.

You specify how the parameters and properties are compared on the Parameters Tab of the Layout XL Options form. The default settings check parameter values only and ignore missing parameters or properties in either the schematic or the layout. This reduces the number of messages in the Check Against Source report.

- The bus terminal information in the layout cellview matches the bus terminal information in the schematic cellview. This behavior is controlled by the createImplicitBusTerminals environment variable.



- *Check Against Source* does not report any issues found in user-defined bindings created using the Define Device Correspondence form.
- When *Check Against Source* is run, the information about the schematic nets does not include any hierarchical information in the name.

Using CDF Callbacks with Check Against Source

Check Against Source uses the schematic list of parameters, but the layout value of the `width` (for folding), `sParam` (for `sfactor`), and `mfactor=1` and `sfactor=1` to verify the list of parameters.

The total `width/sParam` is compared before applying the callback, all other parameters are compared with the corresponding master after applying the callback.

Checking a Layout Against a Schematic

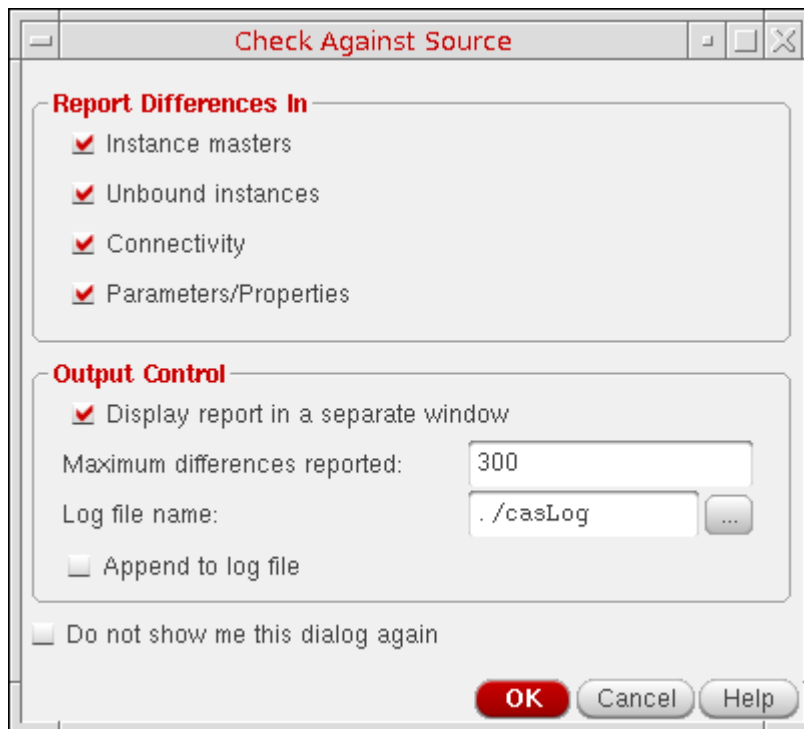
To check the components in your layout view against the schematic,

1. From the layout window menu bar, choose *Connectivity – Check – Against Source*.

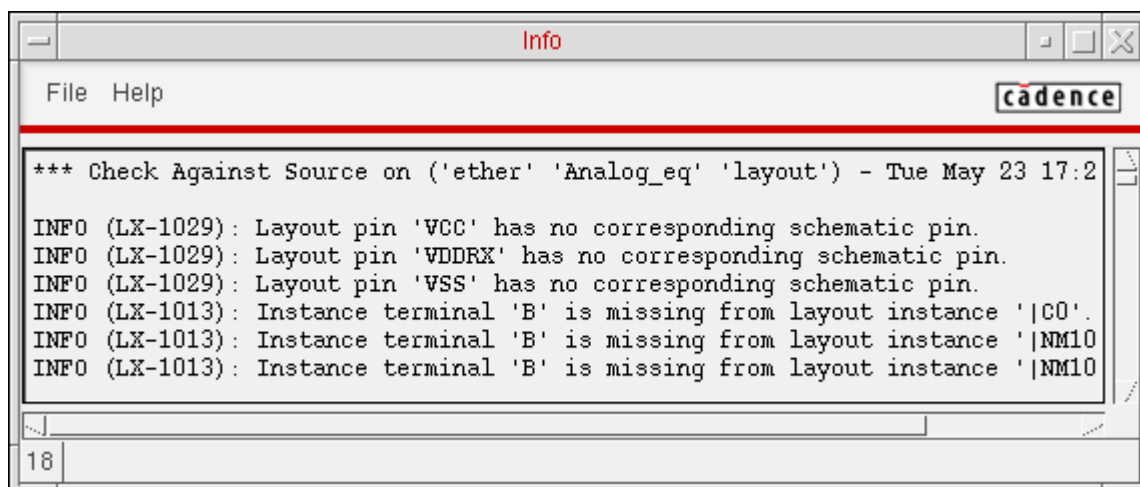
Virtuoso Layout Suite XL User Guide

Checking Design Data

The Check Against Source form is displayed.



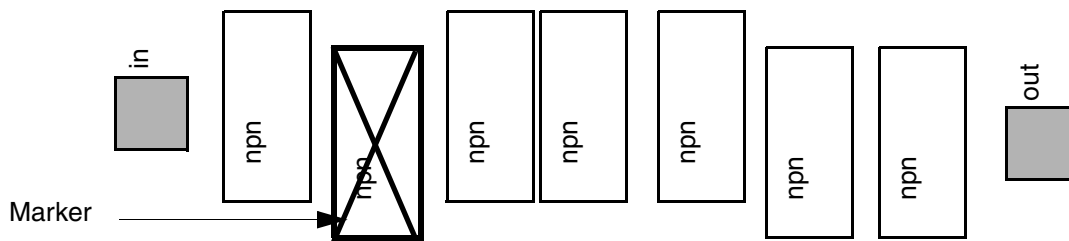
2. In the *Report Differences In* group box, choose the differences you want to report.
3. In the *Output Control* group box, choose where you want to display the report, set the limit for the numbers of differences reported, and specify a name for the log file.
4. Click *OK* to run the check.
5. An Info window appears, reporting problems in the categories you specified in the form.



6. Use the Info window's *File – Save As* command to save the report to an ASCII file and the *File – Close Window* command to dismiss the report from your desktop.

Changes in the Layout Window

If a device that is not in the schematic is present in the layout, it appears with a blinking marker in the layout canvas.



If a device parameter in the layout is different from the corresponding parameter in the schematic, a text window appears and lists the device names and properties (unless the device is ignored).

The command also reports shapes that are on redundant or unmatched nets which no longer exist in the schematic, but it does not put markers on these shapes in the layout canvas. You can check the shapes and delete them manually as required.

Suppressing the Check Against Source Form

To run *Check Against Source* without seeing the Check Against Source dialog, do one of the following.

- Type the following command in the CIW.

```
envSetVal("layoutXL" "disableCASOptionsPopUp" 'boolean "t")
```

- Check the *Do not show me this dialog again* box before you click *OK* on the form.

To see the Check Against Source dialog the next time you run the command,

- Type the following command in the CIW.

```
envSetVal("layoutXL" "disableCASOptionsPopUp" 'boolean "nil")
```

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

Ignoring Components with no Schematic Counterpart

If your layout contains components (for example, template cells or dummy devices) with no schematic counterpart, you can have *Check Against Source* ignore them using the *Properties used to ignore objects during check option* on the Layout XL Parameters Options form. To do this,

1. Add a user-defined boolean property (for example, `ignoreDummyDevice` set to `t`) to the layout instance master, or selectively to each layout instance, to be ignored.

For more information on how to do this, see [Setting Layout XL Properties](#).

Note: If you add the property to the master, each instance of that master that you create will be ignored by *Check Against Source*.

2. From the layout window menu bar, choose *Options – Layout XL* and choose the *Parameters* tab.

The [Parameters Tab](#) of the Layout XL Options form is displayed.

3. Check the radio button marked *Properties used to ignore objects during check*. (The equivalent environment variable is called `propsUsedToIgnoreObjsForCheck`.)
- 4.

Virtuoso Layout Suite XL User Guide

Checking Design Data

5. Type in the property name string (for example, `ignoreDummyDevices`) in the field next to the *Add* button, and click *Add* to add the property to the list of ignored properties.

The screenshot shows the 'Layout XL Options' dialog box with the 'Parameters' tab selected. The 'Ignore' section has four radio buttons, with 'Properties used to ignore objects during check' selected. Below this are two lists: 'Inherited from Generation' containing 'lxRemoveDevice', 'ignore', and 'nlAction'; and a list containing 'ignoreDummyDevices' and 'lvsIgnore'. An 'Add' button is next to an empty text field, and a 'Remove' button is next to the list. The 'Schematic Parameter Names' section has a dropdown menu set to 'Parallel-connected factor' and a text field containing 'm M'. The 'Parameter/Property Comparison' section has three radio buttons, with 'Ignore missing parameters or properties' checked, and a 'Tolerance' field set to '1e-06'. The 'Component Description Format' section has two checkboxes, both unchecked. At the bottom are buttons for 'OK', 'Cancel', 'Defaults', 'Apply', and 'Help'.

Layout XL Options

General Display Extraction Generation **Parameters**

Ignore

Edit ☐ Parameters to ignore during check
☐ Parameters to ignore during generation
☒ Properties used to ignore objects during check
☐ Properties used to ignore objects during generation

Inherited from Generation

lxRemoveDevice ignore nlAction	ignoreDummyDevices lvsIgnore

Add Remove

Schematic Parameter Names

Parallel-connected factor m M

Parameter/Property Comparison

☒ Consider parameters only
☐ Consider parameters and properties
☒ Ignore missing parameters or properties

Tolerance: 1e-06

Component Description Format

☐ Evaluate CDF callbacks during generation and update
☐ Use 0 significant digits when calculating parameter values

OK Cancel Defaults Apply Help

6. Click *OK* in the Layout XL Options form.

The next time you run *Check Against Source*, cellviews with the specified property will not be reported as mismatches.

Note: Connectivity is traced as normal; i.e, the pins on the dummy device are propagated to the top level and assume correct connectivity.

Checking Manufacturability

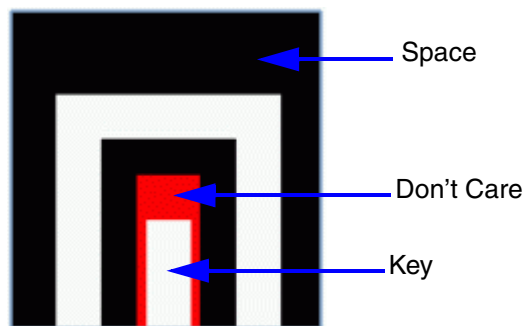
To check manufacturability of a design for optimum device yield and performance, the Virtuoso custom design environment now comes integrated with various verification and fixing flows. You can use these flows to evaluate a layout design against specific schematic parameters to detect any violations and fix them interactively. The advantage of using these flows is that any design violations that can impact the device performance can be detected and fixed well before the design is sent for manufacturing.

- Match and Fix Flow
- Litho Fixing Flow
- Litho/LDE Analysis

Match and Fix Flow

The *Match and Fix Flow* uses a *layout pattern* and an associated fixing *rule* that is defined in the *rule deck* to determine any inconsistencies that need to be fixed before the design is manufactured.

A *layout pattern* can be described as a layout area that is defined using one or many layers, as shown in the figure below:



The “space” area is inferred as the region within the bounding box that is not designated as “key” or “don’t care”.

A region of the target search layout is reported as “matching” the pattern if the following conditions are fulfilled:

1. The “key” area is fully covered by the layout geometry
2. The “space” area is fully uncovered

The list of layout patterns to be used for matching and the list of associated fixing rules are contained within a *rule deck*.

Therefore, a *rule deck* comprises the following:

- The list of layout patterns (such as stream file, layer definitions, and filtering rules)
- The list of Match and Fix rules (such as pattern, layers, and fixing rules)

The format for writing a rule deck is XML and the syntax is as shown below:

```
<?xml version="1.0" ?>
<MatchAndFixRules version="0.1">
  <!-- Pattern definitions section -->
  <!-- Match and fix rule definitions section -->
</MatchAndFixRules>
```

Important

Cadence recommends that you consult with your foundry for availability of a compatible rule deck to run the Match and Fix flow.

Pattern Definition

A single-layer layout pattern can be described in the rule deck as shown below:

```
0 <!-- Single layer Pattern -->
1 <Pattern name="single_pattern1" version="v2">
2 <Source type="oasis">
3 <File name="pattern/ single_pattern1.oas" />
4 <BBoxPurpose>5000</BBoxPurpose>
5 <Purposes type="key">10</Purposes> <!-- Key Purpose -->
6 <Purposes type="dc">50</Purposes> <!-- Don't care Purpose -->
7 </Source>
8 <Description>
9 Necking Pattern 1
10 </Description>
11 <Severity>1</Severity>
```

Virtuoso Layout Suite XL User Guide

Checking Design Data

```
12 <FilteringRule type="CatenaDRC" name="minSpaceRule">
13 <DRCRuleValue type="UserUnits">90 nm</DRCRuleValue>
14 </FilteringRule>
15 </Pattern>
```

The description of each line in the pattern definition is given in the table below.

Line #	Description
1	Pattern name (mandatory). Engine version (optional; default is v2).
2	Pattern source type (mandatory). Pattern source can be either “oasis” or “gdsii”.
3	Source file name (mandatory).
4	Bounding box purpose (mandatory).
5 – 6	Purpose mapping for “key” and “don’t care” areas. ■ “Key” purpose is mandatory ■ “Don’t care” purpose is optional
8 – 10	Detailed description (optional).
11	Severity of the pattern (optional).
12 – 14	Filtering rule (optional).

The *Filtering* rule is used by the Match and Fix flow to filter out and return only matching regions that also fail the given rule. Currently, the following filtering rules are supported:

■ *Min space rule*

```
<FilteringRule type="CatenaDRC" name="minSpaceRule">
<DRCRuleValue type="UserUnits">90 nm</DRCRuleValue>
</FilteringRule>
```

■ *Min notch rule*

```
<FilteringRule type="CatenaDRC" name="oaMinNotchSpacing">
<DRCRuleValue type="UserUnits">90 nm</DRCRuleValue>
```

Let us now consider a sample pattern definition in the rule deck for a multi-layer layout pattern.

```
1 <!--Multi Layer Pattern -->
2 <Pattern name="multi_pattern2" version="v2">
3 <Source type="gdsii">
```

Virtuoso Layout Suite XL User Guide

Checking Design Data

```
4 <File name="pattern/ multi_pattern2.gds2" />
5 <BBoxPurpose>5000</BBoxPurpose>
6 <PatternLayer name="custom_M02">
7 <Purposes type="key">20</Purposes>
8 <Purposes type="dc">50</Purposes>
9 </PatternLayer>
10 <PatternLayer name="custom_M03">
11 <Purposes type="key">30</Purposes>
12 </PatternLayer>
13 </Source>
14 <Description>
Multi layer pattern 1
15 </Description>
16 <Severity>1</Severity>
17 <FilteringRule type="CatenaDRC" name="minSpaceRule">
18 <DRCRuleValue type="UserUnits">90 nm</DRCRuleValue>
19 </FilteringRule>
20 </Pattern>
```

Notice that the lines 6-9 and 10-13 illustrate how the individual layers of a two-layer pattern can be specified.

Match and Fix Rule Definition

In addition to carrying a layout pattern definition, the rule deck carries associated Match and Fix rule definitions. If the target search layout is not found to match the defined pattern, the layout pattern is fixed according to the associated fixing rules defined in the rule deck.

Note that a single match rule is often associated with several fixing rules. If a match rule does not have an associated fixing rule, the fixing engine uses the default fixing flow.

A sample Match and Fix rule definition for a single-layer layout pattern is given below:

```
1 <MatchAndFixRule name="Match&Fix Rule 1 on M2">
2 <MatchRule>
3 <Pattern name=" single_pattern1" />
4 <Layer>M2</Layer>
5 </MatchRule>
6 <FixRule name="moveEdges">
7 <Move unit="nm">194 156 194 208 WEST 130</Move>
8 </FixRule>
9 <FixRule name="fixFilteringRule" />
```

Virtuoso Layout Suite XL User Guide

Checking Design Data

```
10 <FixRule name="decongest">
11 <Option name="ripupEngine">detail</Option>
12 </FixRule>
13 </MatchAndFixRule>
```

The description of each line in the Match and Fix rule definition is given in the table below.

Line #	Description
1	Match and fix rule name (mandatory).
2- 5	Match rule description (mandatory).
3	Name of the pattern to match (mandatory).
4	Target layout layer (or layers) on which to search for the pattern (mandatory).
6 - 12	Fixing rule description (optional)
6 – 8	First fixing rule (name is a mandatory parameter).
7	Parameter for the first fixing rule (specification of an edge to be moved).
9	Second fixing rule.
10 - 12	Third fixing rule.
11	Option for the third fixing rule.

Currently, Virtuoso supports the following fixing strategies:

■ *fixFilteringRule*

Apply internal heuristics, such as moving the shapes, to fix the pattern by using the filtering rule. Note that to be usable, either the pattern or the match rule should have a filtering rule.

■ *decongest*

Use the rerouting engine to decongest the area where the pattern has been found.

■ *moveEdges*

Use the move engine to move one or several edges.

Let us now consider a Match and Fix rule definition for a multi-layer layout pattern as given below:

```
1 <MatchAndFixRule name="Multi-layer rule on M2-M3">
```

```
2 <MatchRule>
3 <Pattern name=" multi_pattern2" />
4 <PatternLayerMap name="custom_M02">M2</PatternLayerMap>
5 <PatternLayerMap name="custom_M03">M3</PatternLayerMap>
6 </MatchRule>
7 </MatchAndFixRule>
```

The individual layers specified in the pattern are mapped to search layers in the target layout. Note that Lines 4 and 5 aim at mapping the pattern layers, `custom_M02` and `custom_M03` to the layout layers `M2` and `M3`.

To run the Match and Fix flow, choose *Optimize – Match and Fix*.

For more information, see the [Match And Fix](#) form.

Litho Fixing Flow

The *Litho Fixing Flow* reads and fixes the hotspots that are detected during a litho physical analysis. The Litho Fixing Flow is primarily a “fixing” flow that takes an HIF (Hotspots Interchange Format) file as the input for identifying any violations, also called hotspots, and enables checking them using a [Litho Checker](#) and fixing the hotspots interactively.

The HIF file includes hotspot details such as error type, error location, severity of the error, and the hints on how to correct the hotspot. Virtuoso is able to use these hints embedded in the HIF file to fix the lithography violations and check if any new violations have been introduced while using the integrated LPA (Litho Physical Analyzer) engine, which is accessible via the *Litho/LDE* menu.

To check the litho violations, you can use the

For more information on the integrated Litho Physical Analyzer, see [Litho/LDE Analysis](#) and the [LDE Analysis](#) form.

For information on using the Litho Fixing Flow, see the [Litho Fixing](#) form.

Litho/LDE Analysis

To facilitate a fast and accurate electrical analysis of the entire chip, Virtuoso supports the new *Litho/LDE* menu that provides the following litho analysis, hotspot checking, and hotspot browsing tools:

- [Litho Checker](#)

- [LDE Analysis](#)
- [Violation Browser](#)



To display the *Litho/LDE* menu, choose the *Launch – DFM* command.

Litho Checker

Use the Litho Checker to run a litho physical analysis on a selected region in a design. The results that are generated during the analysis can then be loaded into the [Violation Browser](#).

To run the Hotspot Checker, choose *Litho/LDE – Litho Checker*.

For more information about the Litho Checker, see [Litho Physical Analyser \(LPA\) Hotspot Browser Application Note](#).

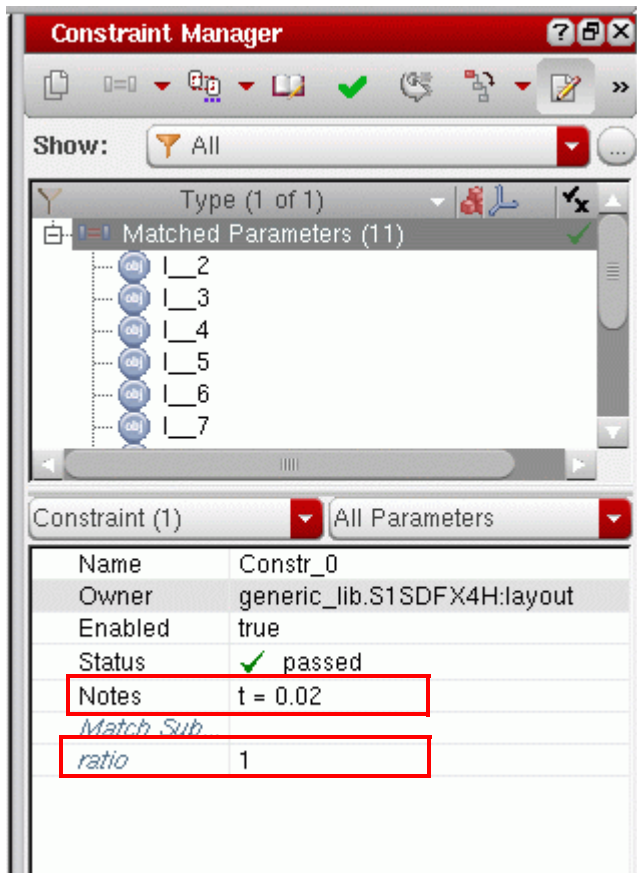
LDE Analysis

Use the LDE Analysis or the Litho Electrical Analysis to compare the drive current (I_{on}) of the layout devices in a design against the desired schematic value to evaluate the optimum device performance. Any violations or overdrives that can possibly impact the device performance are highlighted in the [DRC/DFM tab](#) of the Annotation Browser, and can be interactively fixed.

To be able to perform a matching analysis of the drive current across the schematic and the layout views, you will first need to apply a [Matched Parameters](#) constraint to the design.

Virtuoso Layout Suite XL User Guide

Checking Design Data



The Matched Parameters constraint is created through a CDF parameter called “stress_current”. The fields within this constraint that carry information required for LDE analysis are:

- **Notes:** Indicates the tolerance percentage within which the drive current of the devices in a constraint is considered permissible. For example, a value of $t = 0.02$ in the *Notes* field indicates that the listed devices in the constraint must match currents within 2 percent of each other.

For more information about the *Notes* parameter, see [Constraint Notes](#) in the *Virtuoso Unified Custom Constraints User Guide*.

- **ratio:** Indicates the expected drive current of the devices in a given constraint.

For more information about the *ratio* parameter, see [Ratio Parameter](#) in the *Virtuoso Unified Custom Constraints User Guide*.

To run the LDE Analysis, choose *Litho/LDE – LDE Analysis*.

For more information, see the [LDE Analysis](#) form.

Violation Browser

Use the Violation Browser to inspect all the hotspots reported during a litho physical or LDE analysis and review any contour predictions available and the associated guidelines, if any. You can then use the information gathered during the review to identify the hotspots that need to be fixed and fix them interactively using the Violation Browser or the [DRC/DFM tab](#) of the Annotation Browser.

To run the Violation Browser, choose *Litho/LDE – Violation Browser*.

For more information about the Violation Browser, see the [Litho Physical Analyser \(LPA\) Hotspot Browser Application Note](#).

Virtuoso Layout Suite XL User Guide

Checking Design Data

Updating Design Data

This chapter explains how to use the Virtuoso® Layout Suite XL layout editor (Layout XL) to make design changes and update your design as you progress.

It contains information on the following areas.

- [Updating Components and Nets](#) on page 536
- [Updating Layout Constraints](#) on page 547
- [Updating Layout Parameters](#) on page 547
- [Updating Schematic Parameters](#) on page 554
- [Updating Device Correspondence](#) on page 560
- [Updating the Connectivity Reference](#) on page 570

Updating Components and Nets

Use the *Update Components And Nets* command to automatically update your layout to take account of instances, pins, and connectivity that you have changed in the schematic.

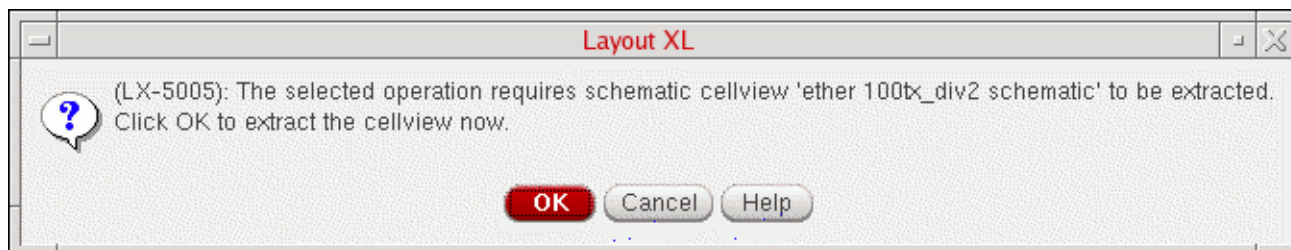
In contrast to *Generate All From Source*, which deletes all existing components in the layout view and regenerates everything from scratch, *Update Components And Nets* updates only the components that have changed. It

- Adds new instances and pins.
- Removes old instances (including any unbound vector and mosaic instances) and pins, along with any empty nets left as a result of the removal.
- Updates instance masters, connectivity and instance, terminal, and net names to match those in the schematic.

To update the components and nets in a layout view,

1. From the layout window menu bar, choose *Connectivity – Update – Components And Nets*.

If the schematic has been changed since it was last extracted, you see a dialog box indicating that it needs to be re-extracted.



Click *OK* in the message box to extract the schematic and all its reference libraries. (Note that the extracted cellviews are not automatically saved.)

Note: To specify a different extraction behavior, click *Cancel* in the message box and choose the *Check – Hierarchy* command from the schematic window menu bar. For more information, see [Checking a Design Hierarchy](#) in the *Virtuoso Schematic Editor L User Guide*.

Virtuoso Layout Suite XL User Guide

Updating Design Data

When the schematic is extracted, the Update Components and Nets form is displayed.

The screenshot shows the 'Update Components and Nets' dialog box with the following settings:

- Update**
 - ☐ Update Selected Layout Components Only
 - ☐ Update Nets and Instance Name Mismatches Only
 - ☒ Update Net Signal Type
 - ☒ Update Instance Masters by **Replacing the Old** Instance
 - ☒ Delete Unmatched Pins ☒ Delete Unmatched Instances
 - ☐ Update Layout Parameters
 - ☐ Move Changed Overlapping Instances Below PR Boundary
- Generate**
 - ☒ Instances
 - ☐ Chaining ☐ Folding ☐ Chain Folds
 - ☒ I/O Pins
 - ☐ Except Global Pins
 - ☐ Except Pad Pins
 - ☒ PR Boundary
 - ☐ Snap Boundary
- Device Correspondence**
 - ☒ Preserve User-Defined Mappings

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

The form remembers values set previously in the current Virtuoso session. When you open it for the first time, it shows the default values set in your `.cdsenv` file.

2. Set the options on the form as needed. For more information, see
 - a. Updating Selected Layout Components
 - b. Updating Nets and Instance Name Mismatches Only
 - c. Specifying the Components to be Updated
 - d. Specifying the I/O Pins to be Updated

e. Specifying How the PR Boundary is Updated

3. Click *OK* in the Update Components and Nets form.

Layout XL updates the layout design as specified in the form. For detailed information on these settings, see the sections listed in step 2.

Limitations of Update Components and Nets

Update Components And Nets does not correct any connectivity errors introduced while defining a many-to-many device mapping using the *Define Device Correspondence* command. These errors are reported by *Check Against Source* and must be fixed manually using the Define Device Correspondence form.

Troubleshooting Update Components and Nets

Any problems encountered during the update are reported in the CIW and in a Layout XL Info text window. Use Check Against Source to get more information on the problems encountered and how to resolve them.

Updating Selected Layout Components

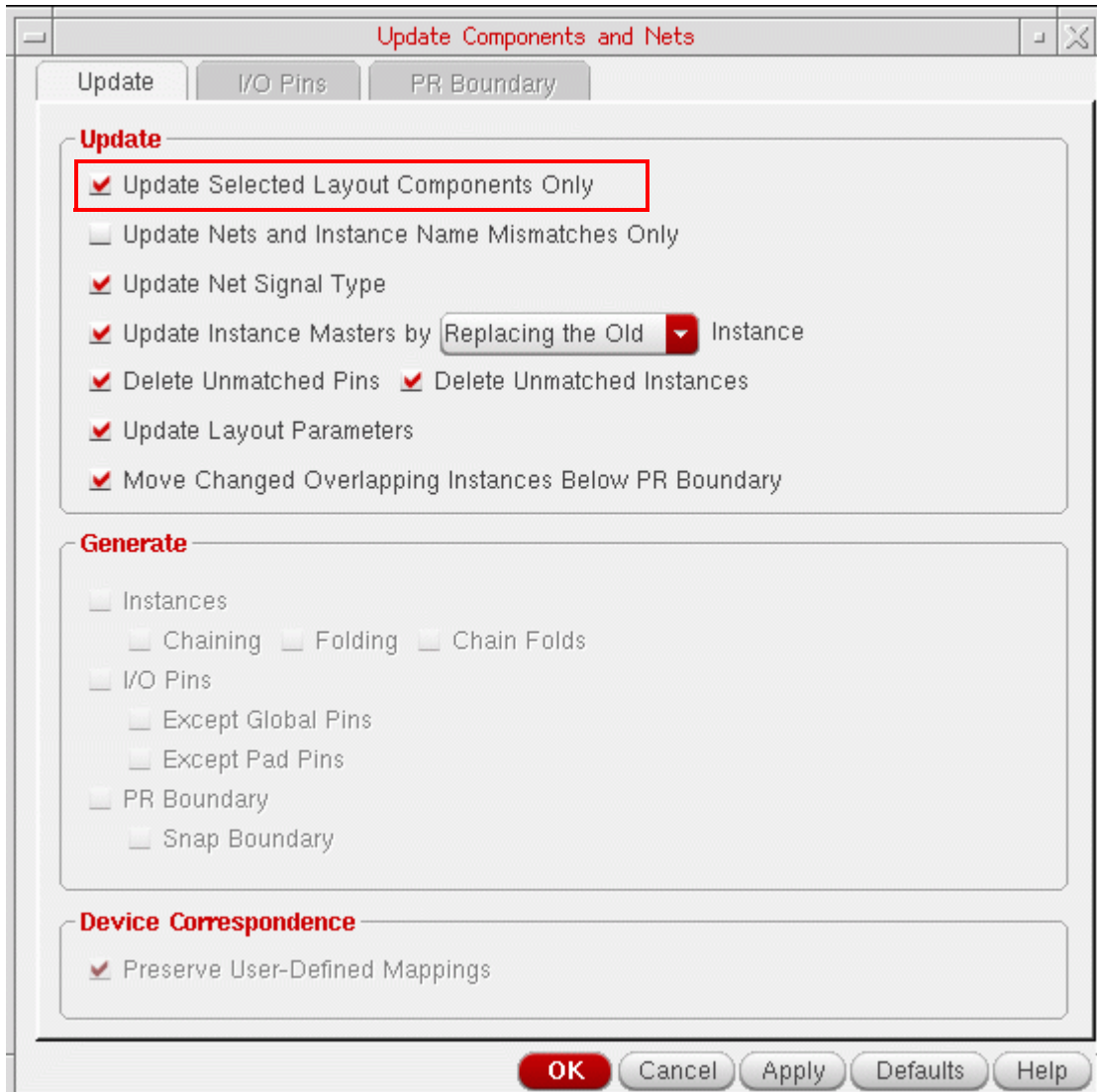
To update only selected instances and pins,

1. Select the Update Tab of the Update Components and Nets form.

Virtuoso Layout Suite XL User Guide

Updating Design Data

2. In the *Update* group box, select the *Update Selected Layout Components Only* check box.



The options to generate missing components, and consequently the *I/O Pins* and *PR Boundary* tabs, are turned off and grayed out.

Preserve User-Defined Mappings is switched on and grayed out.

3. To update only the connectivity information for the selected instances and pins, select the *Update Nets and Instance Name Mismatches Only* check box and proceed to [step 9](#).

For more information, see [Updating Nets and Instance Name Mismatches Only](#).

4. To update the signal types assigned to nets from the schematic view, select the *Update Net Signal Type* check box.
5. To update instances with incorrect masters, select the *Update Instance Masters* check box and specify how an incorrect instance is handled:
 - ☐ By default, it is removed and replaced by an instance of the correct master in the same location.
 - ☐ When set to *Creating a New*, the command puts a marker on the instance with the incorrect master and renames it *name_old*. It then creates a new instance with the correct master and places it below the PR boundary.

Note: Because you are creating a new instance, you can optionally enable *Chaining*, *Folding*, or *Chain Folds* in this mode.
6. To delete layout pins and instances that are not present in the schematic, select the *Delete Unmatched Pins* and *Delete Unmatched Instances* check boxes. When you delete unmatched pins, redundant nets and terminals are also deleted.
7. To update parameters and parameter values in the layout instances to match those on their schematic counterparts, select the *Update Layout Parameters* check box.
8. To automatically place instances below the prBoundary if they overlap other instances during the update, select the *Move Changed Overlapping Instances Below PR Boundary* check box. The overlapping instances are the ones that have changed in size, and if placed at their original location, may overlap other unchanged instances.
9. Click *OK* to update the selected instances and pins.

Updating Nets and Instance Name Mismatches Only

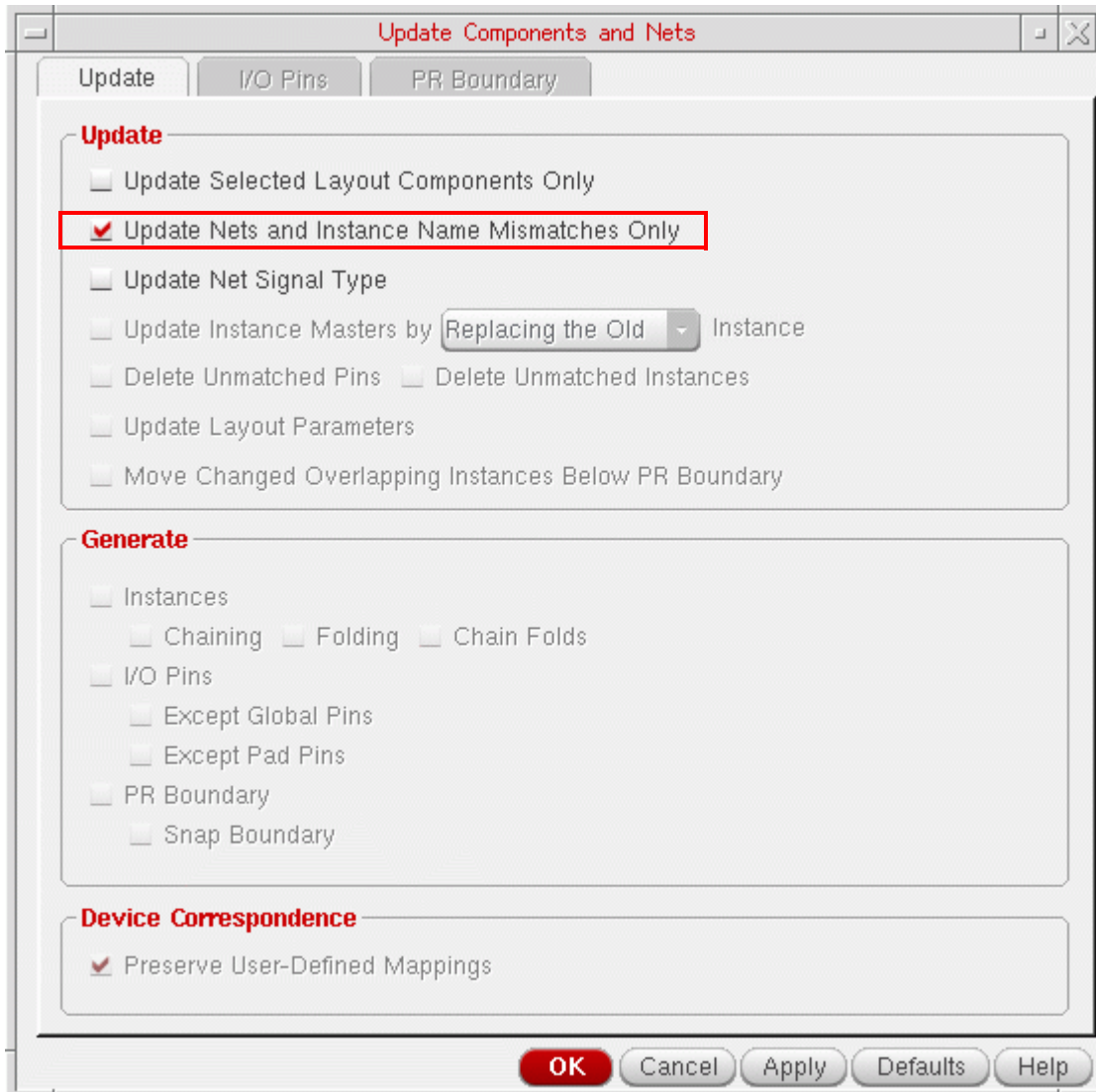
To update only the net assignments and instance and net names in your layout design without changing any other components,

1. Select the Update Tab of the Update Components and Nets form.

Virtuoso Layout Suite XL User Guide

Updating Design Data

2. In the *Update* group box, select the *Update Nets and Instance Name Mismatches Only* check box.



Preserve User-Defined Mappings is selected but inactivated, ensuring that these mappings are maintained. All other options on the form are inactive, except *Update Selected Layout Components Only* and *Update Net Signal Type*.

3. To update only selected instance and pins, select the *Update Selected Layout Components Only* check box.
4. To update the signal types assigned to nets from the schematic view, select the *Update Net Signal Type* check box.
5. Click **OK**.

Virtuoso Layout Suite XL User Guide

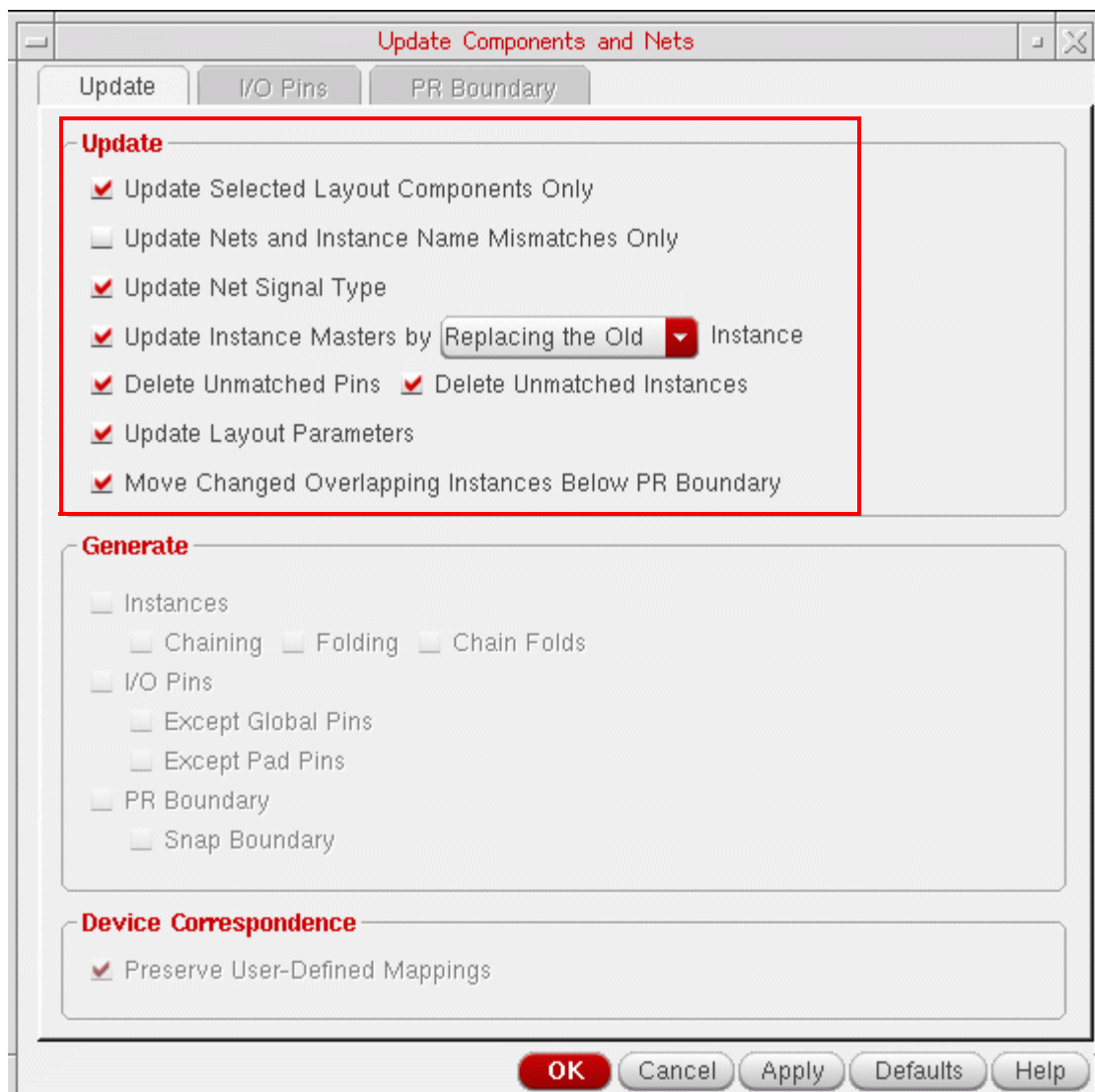
Updating Design Data

Layout XL updates net assignments and instance, terminal, and net names to match those in the schematic.

Specifying the Components to be Updated

When updating the whole design, you can specify that missing components are generated and existing components are regenerated. To do this,

1. Select the Update Tab of the Update Components and Nets form.
2. Select the appropriate options in the *Update* group box to specify how the layout components that need to be changed or removed should be handled.



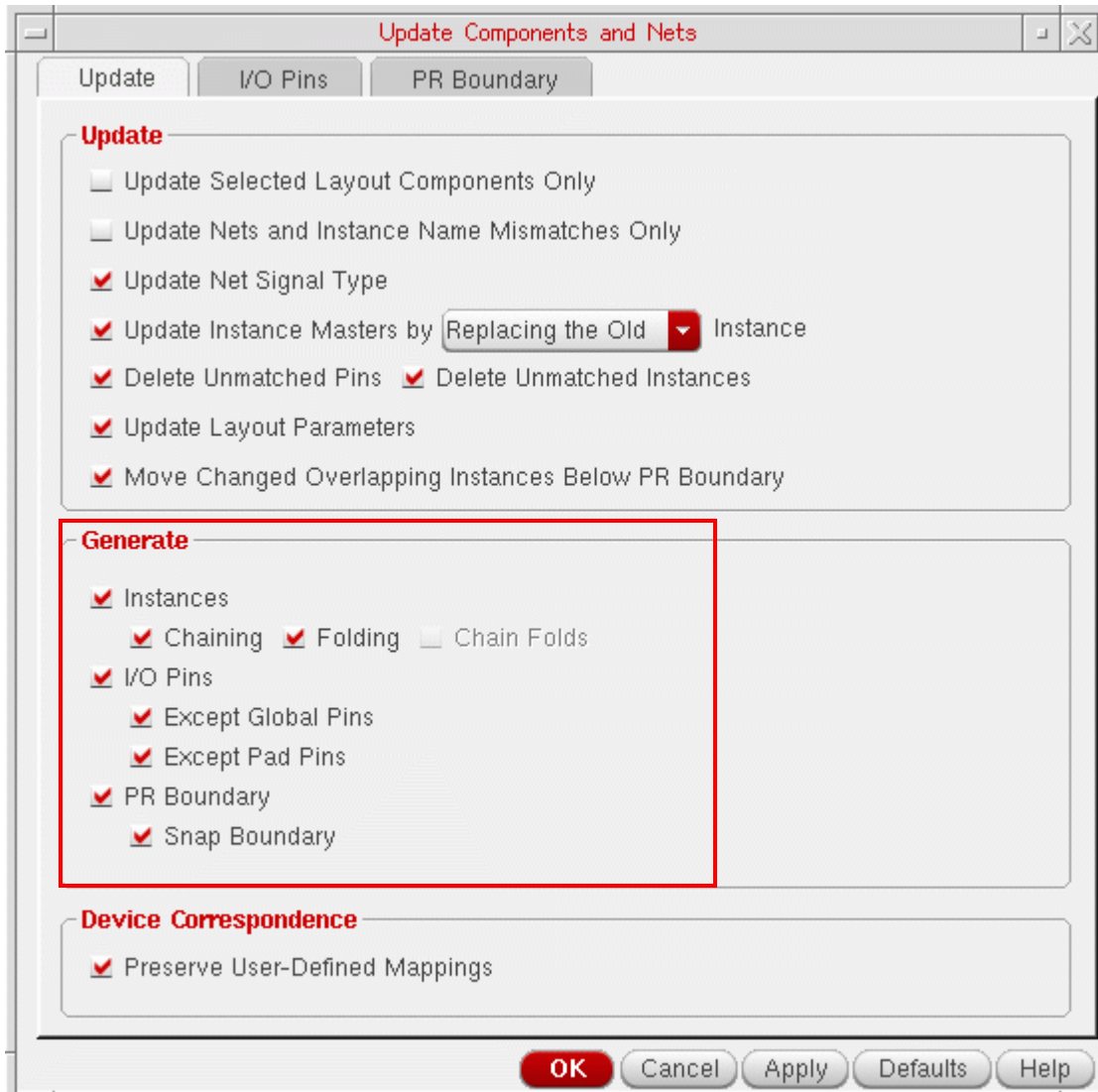
- a. To update only selected instances and pins, see Updating Selected Layout Components.
- b. To update only the connectivity information for the selected instances and pins, see Updating Nets and Instance Name Mismatches Only.
- c. To update the signal types assigned to nets from the schematic view, select the *Update Net Signal Type* check box.
- d. To update instances with incorrect masters, turn on *Update Instance Masters* and specify how the incorrect instance should be handled.
 - ☐ By default, it is removed and replaced by an instance of the correct master in the same location.
 - ☐ When set to *Creating a New*, the command puts a marker on the instance with the incorrect master and renames it *name_old*. It then creates a new instance with the correct master and places it below the PR boundary.

Note that because you are creating a new instance, you can optionally enable *Chaining*, *Folding*, and *Chain Folds* in this mode.
- e. To delete layout pins and instances that are not present in the schematic, select the *Delete Unmatched Pins* and *Delete Unmatched Instances* check boxes. When you delete unmatched pins, redundant nets and terminals are deleted at the same time.
- f. To update parameters and parameter values in the layout instances to match those on their schematic counterparts, select the *Update Layout Parameters* check box.
- g. To automatically place instances below the prBoundary if they overlap other instances during the update, select the *Move Changed Overlapping Instances Below PR Boundary* check box. The overlapping instances are the ones that have changed in size, and if placed at their original location, may overlap other unchanged instances.

Virtuoso Layout Suite XL User Guide

Updating Design Data

3. Use the options in the *Generate* group box to regenerate instances, pins, and PR and snap boundaries.



- a. Select the *Instances*, *I/O Pins*, and *PR Boundary* options as required.

Layout XL puts any missing instances and pins below the design boundary and updates their connectivity. Pins are automatically snapped to the placement grid.

- b. To chain any new transistors, select the *Chaining* check box.

New chains are not attached to existing chains. Devices that were chained prior to running the command are not moved from their existing positions; missing devices are added in the correct position in the chain and not under the place and route boundary.

- c. To divide new transistors in the design into two or more folds, select the *Folding* check box.
- d. To chain the individual folds of a transistor, select the *Chain Folds* check box.

Note: The *Chain Folds* check box is active only when the *Chaining* check box is deselected and the *Folding* check box is selected.

If the *Chaining* and *Folding* check boxes are deselected, and you have deleted all the folds of a folded transistor, *Update Components And Nets* generates a single device for that transistor. If you have deleted only some folds of a folded device, Layout XL creates markers only if the folds in a numbered sequence are missing. Otherwise, Layout XL does nothing about the missing folds of folded transistors.

Note: All devices to be chained or folded must belong to a component type which has *Component class* set to *PMOS* or *NMOS* and values set for the *Active layer*, *Width parameter*, and *Folding threshold* parameters. See [Component Types Mode](#).

- e. To avoid generating layout pins for global nets in the schematic, select the *Except Global Pins* check box.
 - f. To avoid generating layout pins for schematic pins that are connected to I/O pads, select the *Except Pad Pins* check box.
 - g. To generate a snap boundary, select the *Snap Boundary* check box.
4. To preserve any user-defined mapping of devices between the schematic and the layout, select the *Preserve User-Defined Mappings*.

This option preserves user-defined one-to-one, many-to-many, many-to-one, and one-to-many device correspondence defined in the [Define Device Correspondence](#) form. It does not report missing devices or shapes within a mapped group. See [Updating Device Correspondence](#).

Specifying the I/O Pins to be Updated

You specify the pins to be generated on the [I/O Pins Tab](#) of the Update Components and Nets form. The options and use model are the same as for the *I/O Pins* tab on the generate Layout form. For more information, see [Specifying the I/O Pins to be Generated](#).

Layout XL generates any missing pins, places them below the design boundary, and updates their connectivity appropriately. If a layout instance includes a terminal but no pin for a global layout net (because, for example, you have deleted the pin), the pin is regenerated. All pins are automatically snapped to the placement grid.

Pin permutation is not preserved during the update. If any component in the design needs to be updated, the command resets the permutation status of all the cells to that originally stored in the schematic.

Physical-only terminals (those with the `physOnly` property set to `t`) created to support an implicit inherited connection defined using a net expression in the schematic are not updated. Physical-only terminals are deleted only if they have no connections.

Specifying How the PR Boundary is Updated

You specify how the place and route boundary is updated on the PR Boundary Tab of the Update Components and Nets form. The controls and use model are the same as for the *I/O Pins* tab on the Generate Layout form. For more information, see Specifying the Place and Route Boundary.

If you uncheck the *PR Boundary* option on the form, the existing place and route boundary is retained and is not automatically resized to take into account the updated design.

Updating Components and Nets Using SKILL

You can also use the public SKILL API to run *Update Components And Nets* using either of the following methods.

- The `lxUpdateComponentsAndNets` SKILL function.
- A combination of the following functions, which let you specify what is generated and updated directly in SKILL.

`lxUpdateComponentsAndNetsStart`

`lxGetAvailablePinLPPs`

`lxGetPinNets`

`lxSetBoundaryOptions`

`lxSetAreaEstimationOptions`

`lxSetGenerateOptions`

`lxSetNetPinSpecs`

`lxSetUpdateOptions`

`lxUpdateComponentsAndNetsFinish`

For more information, see the sections on the above functions in the *Virtuoso Layout Suite SKILL Reference*.

Updating Layout Constraints

To update constraints from schematic to layout, do one of the following

- From the layout window menu bar, choose *Connectivity – Update – Layout Constraints*.
- In the Constraint Manager toolbar, click the *Update Layout Constraints* button.



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

Updating Layout Constraints after Folding

If you fold instances in the layout view using one of the methods listed below, you must use the *Update Layout Constraints* command to ensure that all the transferred constraints have the correct folded members.

- *Generate Selected From Source*
- *Generate Folded Devices*
- Placement Planning incremental layout generation

Note: The command does not update any constraints you created in the layout. These constraints might have incorrect folded members.

Updating Layout Parameters

Use the *Update Layout Parameters* command to update the parameter values of devices in the layout to match the values in the schematic. The software checks the parameter values on the specified devices in the schematic against the values in the layout and updates the layout parameters when it finds differences (unless you have set the lvslgnore or ignore property on a device).

The updates are reported in a Layout XL Info window, along with any layout devices that contain properties not on the schematic counterpart. If updating parameters results in a

change in the layout, Layout XL updates the connectivity and indicates any problems by drawing markers in the layout window.

If you are using CDF callbacks, the command applies the callback first then applies all differing parameters on the schematic device.



Tip

To see a report on what would be updated without making any changes to your design, run *Check Against Source* before you run *Update Layout Parameters*. For more information, see [Checking a Layout Against a Schematic](#).

Specifying the Parameters and Properties to Update

You specify how the parameters and properties are updated on the [Parameters Tab](#) of the Layout XL Options form.

- *Consider parameters only* compares only the CDF parameters in the schematic against the parameters and properties in the layout and updates values that do not match.

By default, parameters that are present in the schematic but missing from the layout are ignored; switch off *Ignore missing parameters or properties* to propagate the missing parameters to the layout.

- *Consider parameters and properties* compares CDF parameters and cell and instance properties in the schematic against the parameters and properties in the layout and updates values that do not match. Choose this option to update in the layout user-defined schematic properties, or when you know that a device has specific properties that you are interested in updating.

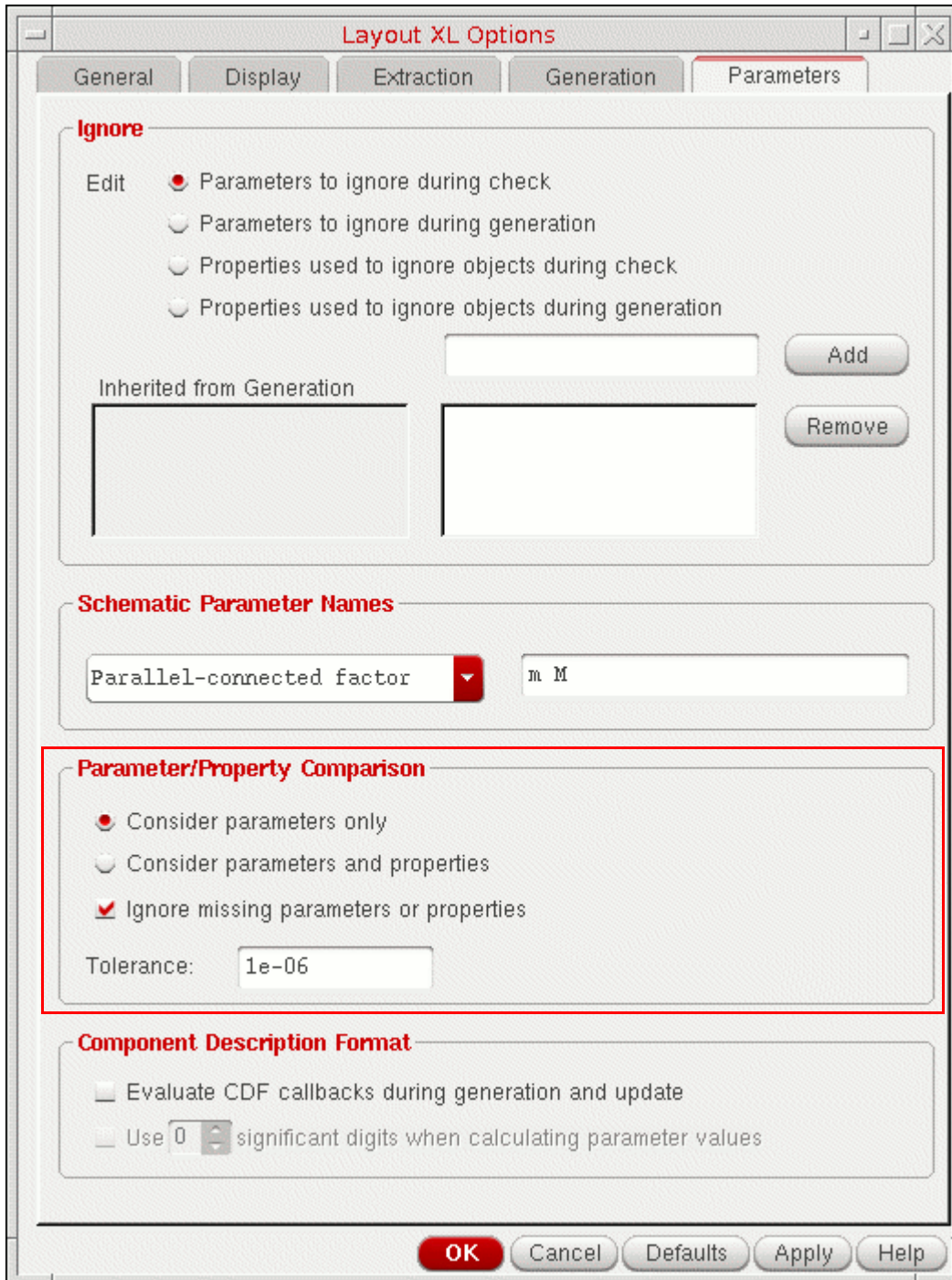
By default, parameters and properties that are present in the schematic but missing from the layout are ignored; switch off *Ignore missing parameters or properties* to propagate these missing parameters and properties to the layout.

- *Ignore missing parameters or properties* ignores parameters (and properties, if *Consider parameters and properties* is checked) that are present in the schematic but missing in the layout. Switch off this option if, for example, your schematic has additional properties that you want to propagate to the layout.

Virtuoso Layout Suite XL User Guide

Updating Design Data

By default, the software updates layout parameter values only and ignores parameters that are present in the schematic but missing from the layout.



Virtuoso Layout Suite XL User Guide

Updating Design Data

To propagate any missing parameters from the schematic to the layout, switch off *Ignore missing parameters or properties*.

The screenshot shows the 'Layout XL Options' dialog box with the 'Parameters' tab selected. The 'Ignore' section has four radio buttons: 'Parameters to ignore during check' (selected), 'Parameters to ignore during generation', 'Properties used to ignore objects during check', and 'Properties used to ignore objects during generation'. Below these are two empty list boxes labeled 'Inherited from Generation' and an 'Add' button. The 'Schematic Parameter Names' section has a dropdown menu showing 'Parallel-connected factor' and a text field containing 'm M'. The 'Parameter/Property Comparison' section is highlighted with a red rectangle and contains three radio buttons: 'Consider parameters only' (selected), 'Consider parameters and properties', and 'Ignore missing parameters or properties'. Below these is a 'Tolerance' field set to '1e-06'. The 'Component Description Format' section has two checkboxes: 'Evaluate CDF callbacks during generation and update' and 'Use 0 significant digits when calculating parameter values'. At the bottom are buttons for 'OK', 'Cancel', 'Defaults', 'Apply', and 'Help'.

Layout XL Options

General Display Extraction Generation **Parameters**

Ignore

Edit ☒ Parameters to ignore during check
☐ Parameters to ignore during generation
☐ Properties used to ignore objects during check
☐ Properties used to ignore objects during generation

Inherited from Generation

Add Remove

Schematic Parameter Names

Parallel-connected factor m M

Parameter/Property Comparison

☒ Consider parameters only
☐ Consider parameters and properties
☐ Ignore missing parameters or properties

Tolerance: 1e-06

Component Description Format

☐ Evaluate CDF callbacks during generation and update
☐ Use 0 significant digits when calculating parameter values

OK Cancel Defaults Apply Help

Virtuoso Layout Suite XL User Guide

Updating Design Data

To update parameters and properties, including missing properties, set the options as shown below.

The screenshot shows the 'Layout XL Options' dialog box with the 'Parameters' tab selected. The 'Ignore' section has four radio buttons: 'Parameters to ignore during check' (selected), 'Parameters to ignore during generation', 'Properties used to ignore objects during check', and 'Properties used to ignore objects during generation'. Below these are two empty list boxes labeled 'Inherited from Generation' and an 'Add' button. The 'Schematic Parameter Names' section has a dropdown menu showing 'Parallel-connected factor' and a text field containing 'm M'. The 'Parameter/Property Comparison' section is highlighted with a red rectangle and contains three radio buttons: 'Consider parameters only', 'Consider parameters and properties' (selected), and 'Ignore missing parameters or properties'. Below this is a 'Tolerance' field set to '1e-06'. The 'Component Description Format' section has two checkboxes: 'Evaluate CDF callbacks during generation and update' and 'Use 0 significant digits when calculating parameter values'. At the bottom are buttons for 'OK', 'Cancel', 'Defaults', 'Apply', and 'Help'.

Layout XL Options

General Display Extraction Generation **Parameters**

Ignore

Edit ☒ Parameters to ignore during check
☐ Parameters to ignore during generation
☐ Properties used to ignore objects during check
☐ Properties used to ignore objects during generation

Inherited from Generation

Add Remove

Schematic Parameter Names

Parallel-connected factor m M

Parameter/Property Comparison

☐ Consider parameters only
☒ Consider parameters and properties
☐ Ignore missing parameters or properties

Tolerance: 1e-06

Component Description Format

☐ Evaluate CDF callbacks during generation and update
☐ Use 0 significant digits when calculating parameter values

OK Cancel Defaults Apply Help

Limitations

Update Layout Parameters does not

- Delete any parameters or properties from the layout view.
- Update the parameters of complex devices defined using the `lxCombination` property, because this might involve deleting and creating instances in the layout view. Use *Update Components And Nets* instead.
- Update split transistors if the sizes of the resulting transistors are different, because such transistors may be the result of manual splitting. If the layout figure selected for update is a transistor that has been folded, Layout XL displays a message that says that it is not updating the parameters.
- Copy `netSet` properties, which are used to specify the new value of a global signal when you are using inherited connections to assign more than one value to a global net.

Updating Layout Parameter Values in Preselection Mode

To update parameter values for devices already selected,

1. In the layout canvas, select the devices you want to update.
2. From the layout window menu bar, choose *Connectivity – Update – Layout Parameters*.

If you are using CDF callbacks, *Update Layout Parameters* applies the callback first and then updates the parameters on the selected layout devices to match the values of the corresponding devices in the schematic.

3. Press `ESC` to exit the command.

Updating Layout Parameter Values in Postselection Mode

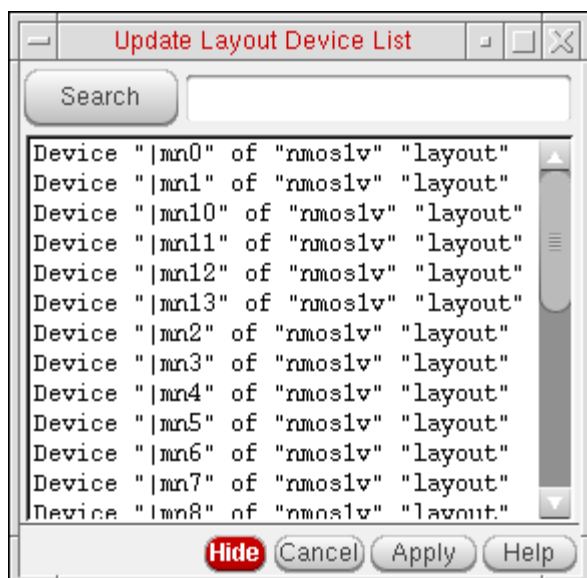
To update parameter values in postselection mode,

1. From the layout window menu bar, choose *Connectivity – Update – Layout Parameters*.

The CIW and the layout window prompt you to select the devices whose parameter values you want to update.

2. To see a list of the layout devices in the design, move the cursor into the layout canvas and press `F3`.

The Device List form is displayed.



3. Select a device in the form, in the layout canvas, or in the schematic window.

Layout XL highlights the device in all three locations.

4. Click *Apply* in the form or move the cursor into the layout canvas and press `Return`.

If you are using CDF callbacks, *Update Layout Parameters* applies the callback first and then updates the parameters on the selected layout devices to match the values of the corresponding devices in the schematic.

5. Press `ESC` to exit the command.

Updating Parameter Values for Folded Devices

To update parameter values for folded devices,

- Switch on the *Update width parameter method* option on the Generation Tab of the Layout XL Options form before you run *Update Layout Parameters*.



Tip

Check the size of folded devices manually to avoid overriding the specified size.

Excluding a Specific Parameter from the Update

To prevent a specific layout parameter from being updated by the *Update Layout Parameters* command,

1. From the layout window menu bar, choose *Options – Layout XL* and click the *Parameters* tab.
2. Select *Parameters to ignore during generation*, type the name of the parameter you want to ignore in the field provided, and click *Add*.

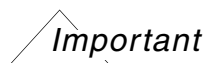
The parameter name is added to the list and will not be updated when you run *Update Layout Parameters*.

3. From the layout window menu bar, choose *Connectivity – Update – Layout Parameters*.

The layout parameters are updated to match the values in the schematic; however, the parameter you specified to be ignored is not updated.

Updating Schematic Parameters

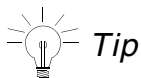
Use the *Update Schematic Parameters* command to update the parameter values of devices in the schematic to match the values in the layout.



You must have the schematic window open in edit mode to use this command.

The software checks the values of the parameters of the specified devices in the schematic against the values in the layout and updates the schematic parameters when it finds differences (unless you have set the *lvsignore* or *ignore* property on a device).

The updates are reported in a Layout XL Info window. If you are using CDF callbacks, the command applies the callback first then applies all differing parameters on the layout device.



To see a report on what would be updated without making any changes to your design, run *Check Against Source* before you run *Update Schematic Parameters*. For more information, see [Checking a Layout Against a Schematic](#).

Specifying the Parameters and Properties to Update

You specify how the parameters and properties are updated on the [Parameters Tab](#) of the Layout XL Options form.

- *Consider parameters only* compares only the CDF parameters in the schematic against the parameters and properties in the layout and updates values that do not match.

By default, parameters that are present in the layout but missing from the schematic are ignored; switch off *Ignore missing parameters or properties* to propagate the missing parameters to the schematic.

- *Consider parameters and properties* compares CDF parameters and cell and instance properties in the schematic against the parameters and properties in the layout and updates values that do not match.

By default, parameters that are present in the layout but missing from the schematic are ignored; switch off *Ignore missing parameters or properties* to propagate the missing parameters to the schematic.

- *Ignore missing parameters or properties* ignores parameters (and properties, if *Consider parameters and properties* is checked) that are present in the layout but missing in the schematic. Switch off this option if, for example, your layout has additional properties that you want to propagate to the schematic.

Virtuoso Layout Suite XL User Guide

Updating Design Data

By default, the software updates schematic parameter values only and ignores parameters that are present in the layout but missing from the schematic.

The screenshot shows the 'Layout XL Options' dialog box with the 'Parameters' tab selected. The dialog has five tabs: General, Display, Extraction, Generation, and Parameters. The 'Parameters' tab contains several sections:

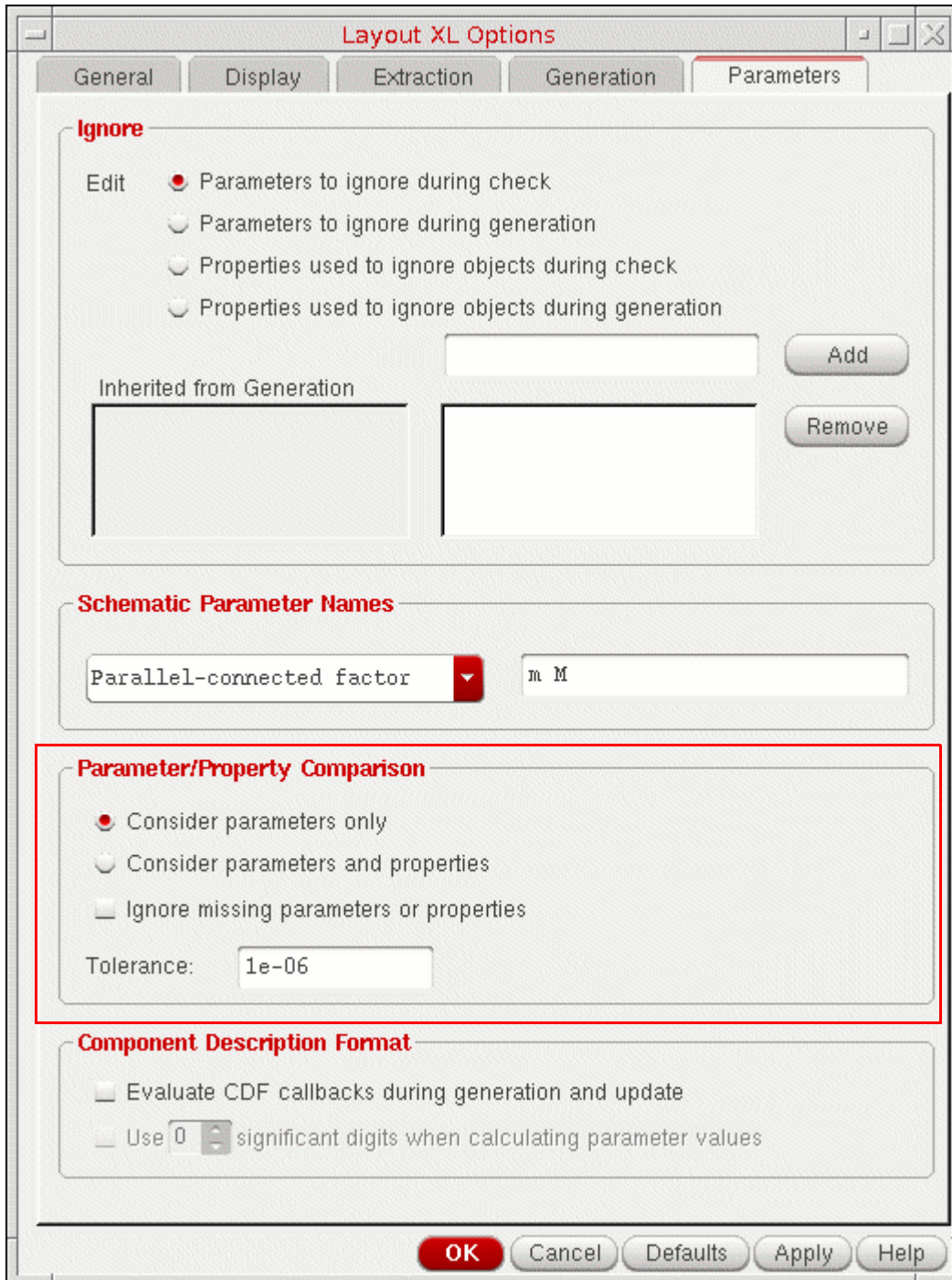
- Ignore:** This section has an 'Edit' button and four radio buttons:
 - ☒ Parameters to ignore during check
 - ☐ Parameters to ignore during generation
 - ☐ Properties used to ignore objects during check
 - ☐ Properties used to ignore objects during generationBelow these are two empty list boxes, one labeled 'Inherited from Generation', and 'Add' and 'Remove' buttons.
- Schematic Parameter Names:** This section has a dropdown menu showing 'Parallel-connected factor' and a text field containing 'm M'.
- Parameter/Property Comparison:** This section is highlighted with a red border and contains three radio buttons:
 - ☒ Consider parameters only
 - ☐ Consider parameters and properties
 - ☒ Ignore missing parameters or propertiesBelow these is a 'Tolerance:' label and a text field containing '1e-06'.
- Component Description Format:** This section contains two checkboxes:
 - ☐ Evaluate CDF callbacks during generation and update
 - ☐ Use 0 significant digits when calculating parameter values

At the bottom of the dialog are five buttons: 'OK' (highlighted in red), 'Cancel', 'Defaults', 'Apply', and 'Help'.

Virtuoso Layout Suite XL User Guide

Updating Design Data

To propagate any missing parameters from the layout to the schematic, switch off *Ignore missing parameters or properties*.



Virtuoso Layout Suite XL User Guide

Updating Design Data

To update parameters and properties, including missing properties, set the options as shown below.

The screenshot shows the 'Layout XL Options' dialog box with the 'Parameters' tab selected. The 'Ignore' section has four radio buttons: 'Parameters to ignore during check' (selected), 'Parameters to ignore during generation', 'Properties used to ignore objects during check', and 'Properties used to ignore objects during generation'. Below these are two empty list boxes labeled 'Inherited from Generation' and an 'Add' button. The 'Schematic Parameter Names' section has a dropdown menu showing 'Parallel-connected factor' and a text field containing 'm M'. The 'Parameter/Property Comparison' section is highlighted with a red border and contains three radio buttons: 'Consider parameters only', 'Consider parameters and properties' (selected), and 'Ignore missing parameters or properties'. Below these is a 'Tolerance' field set to '1e-06'. The 'Component Description Format' section has two checkboxes: 'Evaluate CDF callbacks during generation and update' and 'Use 0 significant digits when calculating parameter values'. At the bottom are buttons for 'OK', 'Cancel', 'Defaults', 'Apply', and 'Help'.

Layout XL Options

General Display Extraction Generation **Parameters**

Ignore

Edit ☒ Parameters to ignore during check
☐ Parameters to ignore during generation
☐ Properties used to ignore objects during check
☐ Properties used to ignore objects during generation

Inherited from Generation

Add Remove

Schematic Parameter Names

Parallel-connected factor m M

Parameter/Property Comparison

☐ Consider parameters only
☒ Consider parameters and properties
☐ Ignore missing parameters or properties

Tolerance: 1e-06

Component Description Format

☐ Evaluate CDF callbacks during generation and update
☐ Use 0 significant digits when calculating parameter values

OK Cancel Defaults Apply Help

Updating Schematic Parameter Values in Preselection Mode

To update the schematic parameters of devices already selected,

1. In the schematic or layout window, select the devices you want to update.
2. From the layout window menu bar, choose *Connectivity – Update – Schematic Parameters*.

If you are using CDF callbacks, Layout XL applies the callback first and then updates the parameters on the selected schematic devices to match the values of the corresponding devices in the layout (unless you have set the lvsignore or ignore property on a device).

An Information window reports any changes that are made.

3. Press `ESC` to exit the command.

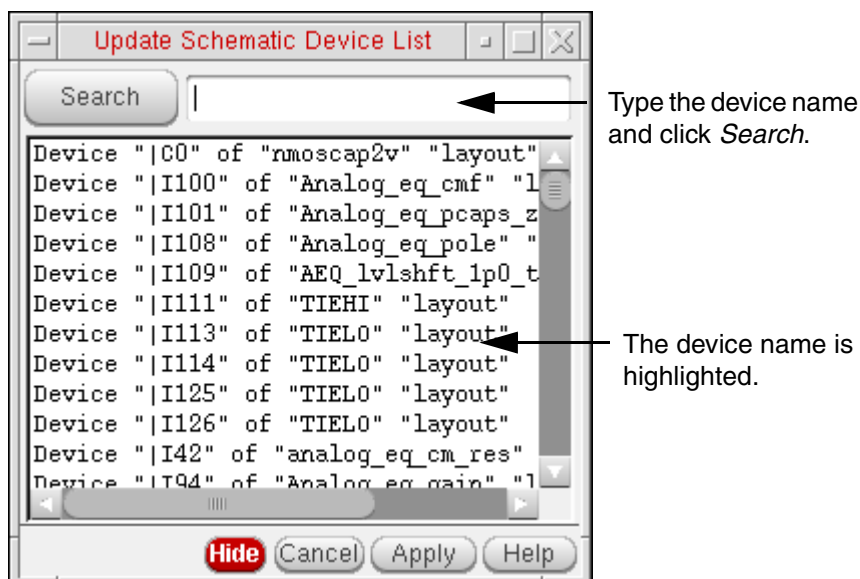
Updating Schematic Parameter Values in Postselection Mode

1. From the layout window menu bar, choose *Connectivity – Update – Schematic Parameters*.

The CIW and layout window prompt you to select the devices whose parameters you want to update.

2. To see a list of the layout devices in design, move the cursor to the layout canvas and press `F3`.

The Device List form is displayed.



3. Select a device in the form, in the layout window, or in the schematic window.

Layout XL highlights the device in all three locations.

4. Click *Apply* in the form or move the cursor into the layout canvas and press `Return`.

If you are using CDF callbacks, Layout XL applies the callback first and then updates the parameters on the selected schematic devices to match the values of the corresponding devices in the layout (unless you have set the `lvignore` or `ignore` property on a device).

An Information window reports any changes that are made.

5. Press `ESC` to exit the command.

Updating Device Correspondence

For each schematic and layout pair, the correspondence between the components in the views is determined by the Layout XL binder engine based primarily on the connectivity of the design.

The binder starts from the terminals and builds up the correspondence by following the logical connectivity through the design. When it encounters a net, the binder searches for the instances connected to the net and binds instances with matching connectivity and master cellviews. The binder is also physical-connectivity-aware. Instances connected by shapes that are not assigned to a net are now recognized as interconnected. When incrementally binding such instances, the Layout XL automatically creates and assigns nets to instance terminals if required.

When it finds no more matches based on connectivity, the binder binds instances with matching names and cellview masters. It then restarts the connectivity search from these instances. This lets Layout XL handle legacy layout views with mismatched names, and facilitates a smooth flow for Engineering Change Orders that involve the renaming of instances and terminals.

Initial Binding

When Layout XL is started, the system explores the connectivity of the schematic design creating an initial mapping between instances and nets, then instances connected to the same nets with the expected name, or with a master that satisfies the rules defined during physical hierarchy configuration.

Note: When you edit a design in Layout L and then load it in Layout XL, the bindings may be incomplete; this is because no connectivity extraction takes place by default when Layout XL

Virtuoso Layout Suite XL User Guide

Updating Design Data

is started. In such cases, you may need to perform a full batch extraction of the design for all possible bindings to be found. See [Connectivity Extraction](#) for more information.

When components are generated in the layout, the system propagates the connectivity to the layout view and sets the correspondence between the generated instances and their schematic counterparts. This correspondence is reported in the CIW when you start Layout XL; for example, the report below shows perfect correspondence between schematic and layout.

```
\o INFO (BND-1002): Binder initialization report:
\o   Schematic instances with corresponding layout instances:    40
\o   Layout instances with corresponding schematic instances:    40
\o   Layout instances with no corresponding schematic instance:  0
\o   Instances present in schematic but missing from layout:     0
```

The report below indicates that there is a problem with the correspondence.

```
\o INFO (BND-1002): Binder initialization report:
\o   Schematic instances with corresponding layout instances:    0
\o   Layout instances with corresponding schematic instances:    0
\o   Layout instances with no corresponding schematic instance:  40
\o   Instances present in schematic but missing from layout:     9
```

In the case above, the software was unable to find the correct layout view because it was not included in the *Physical view list* on the [Generation Tab](#) form.

Incremental Binding

Device correspondence and connectivity information are updated incrementally by the system whenever required (for example, when instances are created in the design). This saves having to rebuild device correspondence every time there is a change, significantly improving performance, especially during operations such as *Undo*.

Note: When you run the schematic editor *Check and Save* command during a Layout XL session, the binder is reinitialized to take account of any changes in schematic connectivity. This can lead to the creation of new bindings between schematic and layout components.

User-Defined Device Correspondence

If you need to change the system-generated device correspondence manually, you can do so using the *Define Device Correspondence* command. User-defined device correspondence is persistent between Layout XL sessions, and can be preserved during *Generate All From Source* and *Update Components And Nets* using the *Preserve User-Defined Mappings* option in the [Generate Layout](#) form.

The *Define Device Correspondence* command and, where required, the Assign Layout Instance Terminals form correctly propagate connectivity to the layout implementation. There is no need to run the *Update Components And Nets* command after defining device correspondence. To update the connectivity of devices that are already correctly mapped, you need to only use the *Update Components And Nets* command.

Important

Check Against Source does not report any issues found in user-defined bindings created using the Define Device Correspondence form.

The procedures that follow describe how to manually define mappings between instances, but apply equally for terminals as well.

- Defining a One-to-One Device Correspondence
- Defining a One-to-Many Device Correspondence
- Defining a Many-to-One Device Correspondence
- Defining a Many-to-Many Device Correspondence
- Assigning Layout Instance Terminals to Nets
- Deleting a Device Correspondence

Dynamic Updates to the Define Device Correspondence Display

The Define Device Correspondence form shows you the current correspondence between instances and terminals in the schematic and layout views. You can leave the form open on your desktop while you manipulate your design in the layout window; the form is dynamically updated to reflect the changes you make.

Note the following:

- For performance reasons, the form does not update dynamically if you select more than 100 objects from the design canvas. To define device correspondence for a large number of objects, select them in the form itself.
- The *All Terminals* and *Unbound Terminals* filters show the terminals in the design, not the pin figures that typically represent them in the layout window.
- If you delete a pin figure in the layout window, the corresponding terminal is removed from the Define Device Correspondence form only if the net connected to this terminal has no other connections. If the net has other connections, the terminal is not removed.

Similarly, if you delete an instance, the terminals connected to the instance terminal's nets are deleted only if they are no longer connected in the design.

Defining a One-to-One Device Correspondence

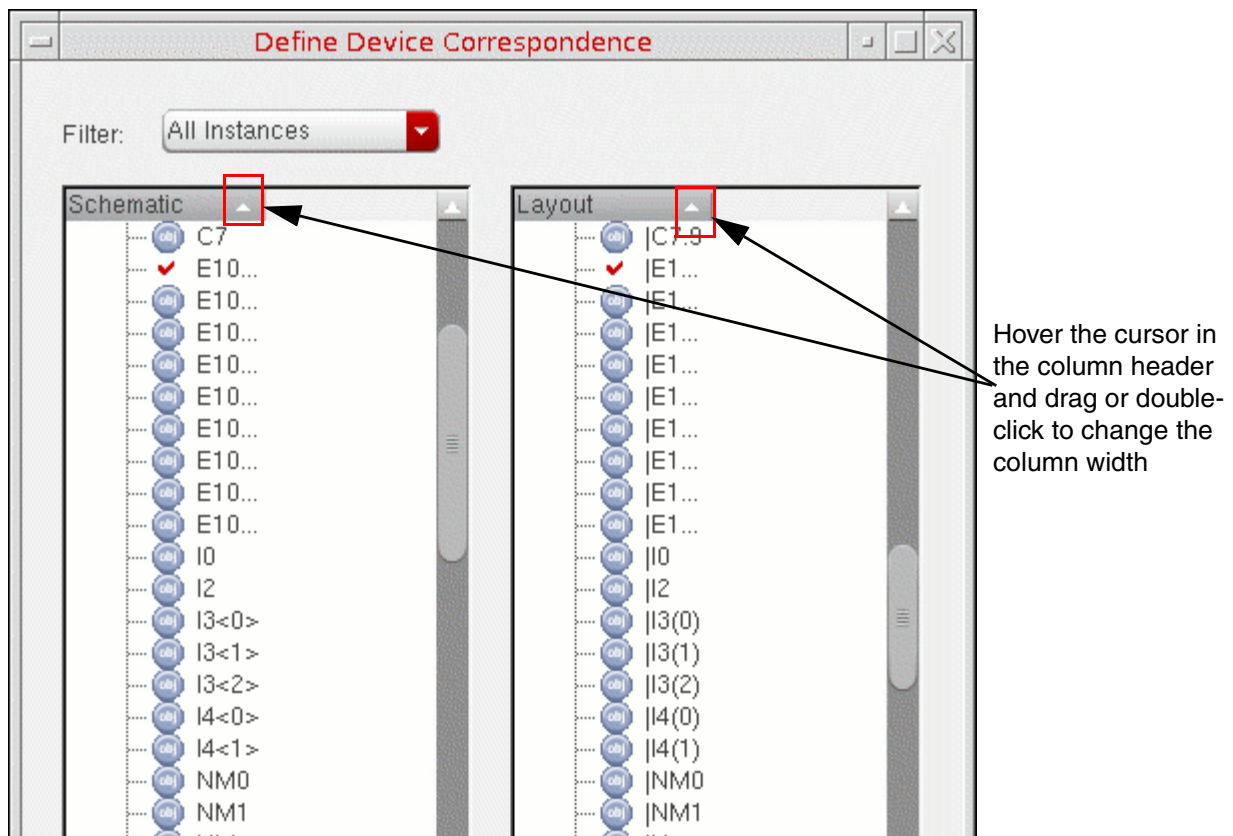
Important

To create a one-to-one device correspondence, you must select only one schematic device and one layout device at a time and click *Map*. If you select multiple schematic and layout devices and click *Map*, Layout XL creates a single many-to-many correspondence and NOT multiple one-to-one mappings.

To define a correspondence between a single schematic instance (or terminal) and a single layout instance (or terminal),

1. From the layout window menu bar, choose *Connectivity – Define Device Correspondence*.

The Define Device Correspondence form appears.

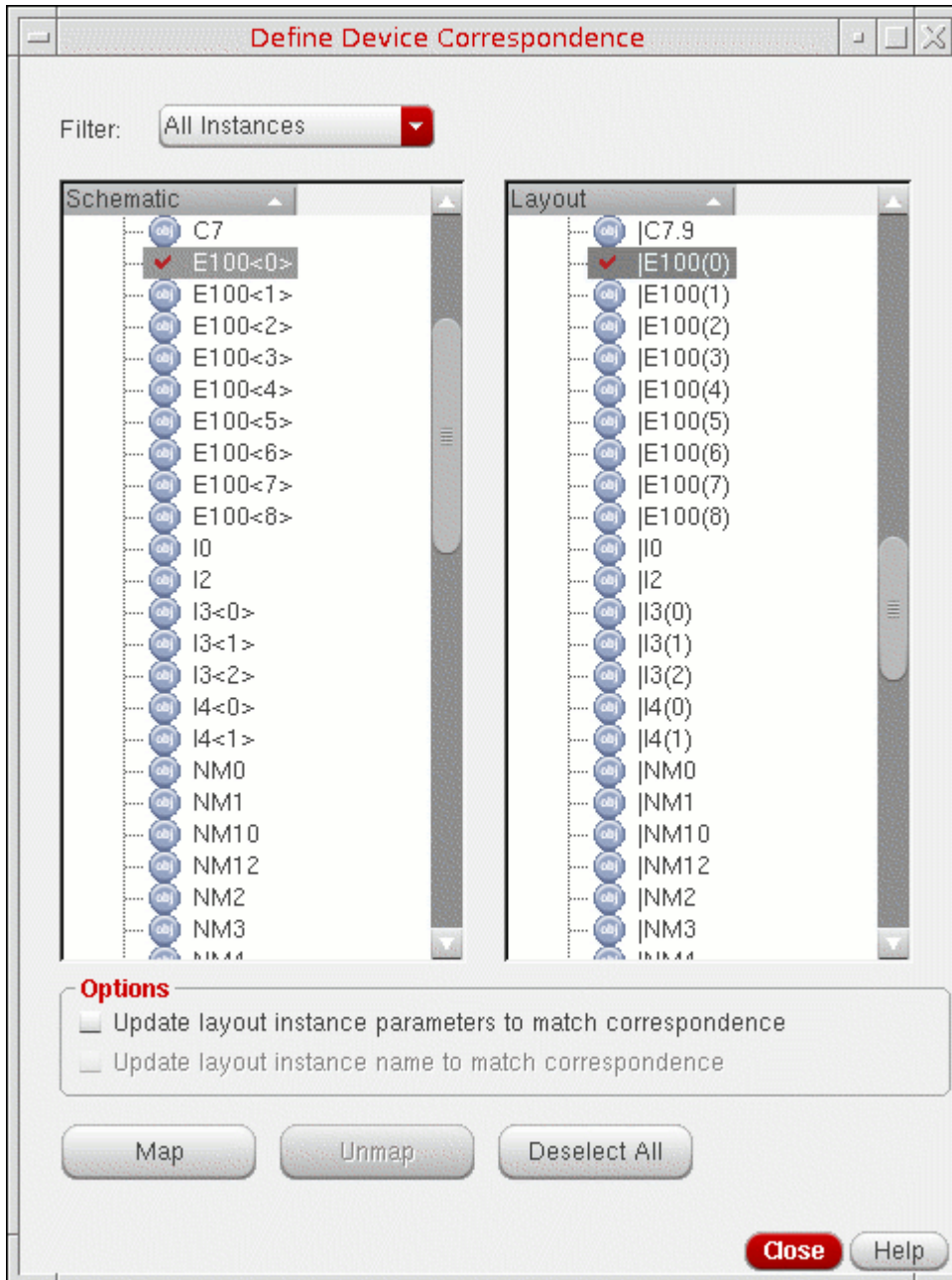


Virtuoso Layout Suite XL User Guide

Updating Design Data

If you cannot see the entire instance or terminal name, hover your cursor over the column header until it changes to show the Resize cursor. You can then drag to resize the column to the width you want, or double-click to expand the column to fit the longest name.

2. Select an unmapped instance from the *Schematic* tree and an unmapped instance from the *Layout* tree.



You can map only unmapped devices. To see only the unmapped instances, choose *Unbound Instances* from the *Filter* cyclic field. To see only the unmapped terminals, choose *Unbound Terminals*.

3. (Optional) Use the two *Update* options at the bottom of the form to update the name and parameters values of the layout instance to match those in the schematic instance.
4. Click *Map* to create a correspondence between the selected schematic instance and the selected layout instance.
 - ☐ If the instance masters match, the correspondence is updated and the connectivity from the schematic is propagated to the layout.
 - ☐ If the instance masters do not match, the Assign Layout Instance Terminals form is displayed. This form lets you assign layout nets to the unbound instance terminals in the selected layout instance. For more information, see Assigning Layout Instance Terminals to Nets.

Defining a One-to-Many Device Correspondence

To define a correspondence between a single schematic instance and multiple layout instances,

1. From the layout window menu bar, choose *Connectivity – Define Device Correspondence*.

The Define Device Correspondence form appears.

2. Select an unmapped instance from the *Schematic* tree and the unmapped instances from the *Layout* tree to which it is to be mapped.

Note: You can map only unmapped devices. To see only the unmapped instances, choose *Unbound Instances* from the *Filter* cyclic field; to see only unmapped terminals, choose *Unbound Terminals*.

3. (Optional) Use the two *Update* options at the bottom of the form to update the name and parameters values of the layout instance to match those in the schematic instance.
4. Click *Map* to create a correspondence between the selected schematic instance and the selected layout instances.

Layout XL binds the instances, but needs more information in order to propagate the connectivity correctly.

The Assign Layout Instance Terminals form is displayed. This form lets you assign layout nets to the unbound instance terminals in the selected layout instance. For more information, see Assigning Layout Instance Terminals to Nets.

Defining a Many-to-One Device Correspondence

To define a correspondence between a multiple schematic instances and a single layout instance (for example, mapping two schematic transistors to a single inverter in the layout),

1. From the layout window menu bar, choose *Connectivity – Define Device Correspondence*.

The Define Device Correspondence form appears.

2. Select the unmapped instances from the *Schematic* tree and the unmapped instance from the *Layout* tree to which they are to be mapped.

Note: You can map only unmapped devices. To see only the unmapped instances, choose *Unbound Instances* from the *Filter* cyclic field; to see only unmapped terminals, choose *Unbound Terminals*.

3. (Optional) Use the two *Update* options at the bottom of the form to update the name and parameters values of the layout instance to match those in the schematic instance.
4. Click *Map* to create a correspondence between the selected schematic instances and the selected layout instance.

Layout XL binds the instances, but needs more information in order to propagate the connectivity correctly.

The Assign Layout Instance Terminals form is displayed. This form lets you assign layout nets to the unbound instance terminals in the selected layout instance. For more information, see Assigning Layout Instance Terminals to Nets.

5. When you have finished assigning layout instance terminals, click *Close* to dismiss the form.

The correspondence is updated and the connectivity you defined is propagated to the layout.

Defining a Many-to-Many Device Correspondence



Connectivity errors introduced while defining a many-to-many device correspondence are reported by Check Against Source but cannot be corrected automatically by Update Components And Nets. This is because it is impossible to determine the correct terminal mapping in a many-to-many relationship.

To define a correspondence between multiple schematic instances and multiple layout instances,

1. From the layout window menu bar, choose *Connectivity – Define Device Correspondence*.

The Define Device Correspondence form appears.

2. Select the instances to be mapped from the *Schematic* and *Layout* trees.

Note: You can map only unmapped devices. To see only the unmapped instances, choose *Unbound Instances* from the *Filter* cyclic field; to see only unmapped terminals, choose *Unbound Terminals*.

3. (Optional) Use the two *Update* options at the bottom of the form to update the name and parameters values of the layout instance to match those in the schematic instance.
4. Click *Map* to create a correspondence between the selected schematic instances and the selected layout instances.

Layout XL binds the instances, but needs more information in order to propagate the connectivity correctly.

The Assign Layout Instance Terminals form is displayed. This form lets you assign layout nets to the unbound instance terminals in the selected layout instance. For more information, see Assigning Layout Instance Terminals to Nets

Assigning Layout Instance Terminals to Nets

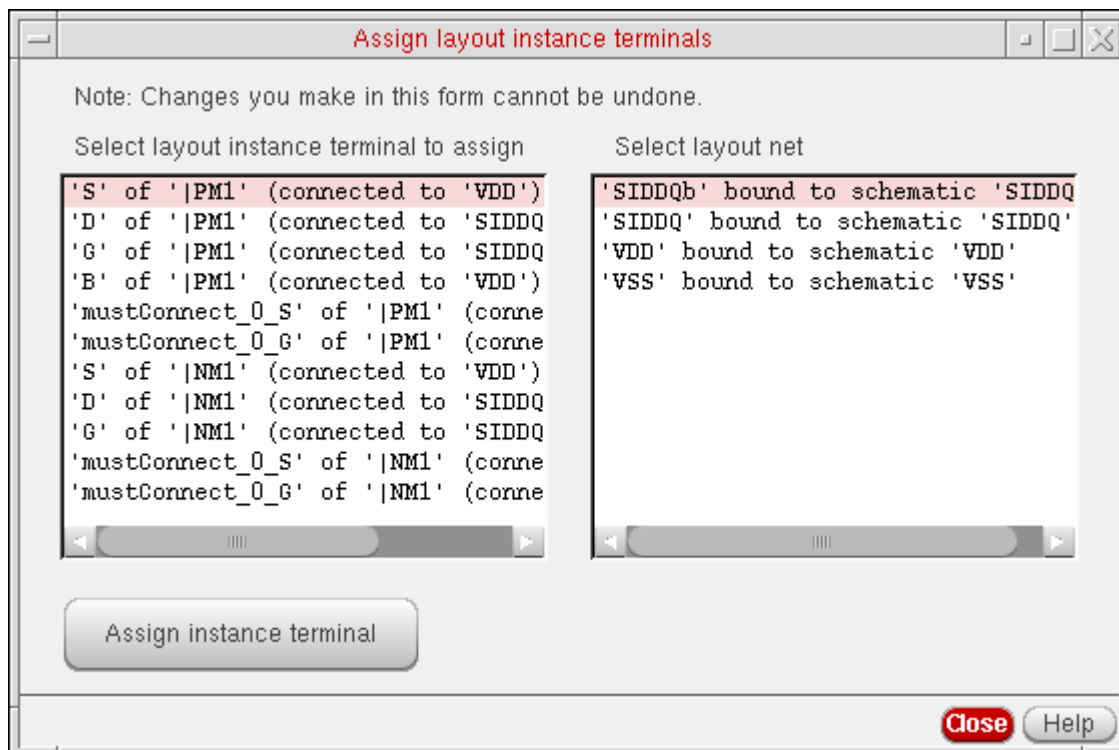
When creating a one-to-one device correspondence between schematic and layout instances with matching masters, *Define Device Correspondence* is typically able to assign layout instance terminals to the correct nets automatically. However, in some cases, the command is unable to assign all the instance terminals automatically; for example,

- If the instance masters in a one-to-one correspondence do not match.
- If you are creating a one-to-many, many-to-one, or many-to-many device correspondence.

In these cases, the Assign Layout Instance Terminals form is displayed. This form lets you manually assign layout nets to the unbound instance terminals in the selected layout instances.

To assign layout instance terminals,

1. In the Assign Layout Instance Terminals form, select a layout net from the list on the right and one or more instance terminals to which it is to be assigned from the list on the left.



2. Click *Assign instance terminal*.

The instance terminals are assigned to the net and removed from the list.

3. Click *Close* to dismiss the form.

The device correspondence is updated and the connectivity you defined is propagated to the layout.

If you close the form without manually assigning all the terminals, the connections of the remaining terminals are not modified.



Tip

You cannot unassign instance terminals in the form after you have clicked *Assign instance terminals*.

However, you can reverse all the assignments for the selected instances by closing the Assign Instance Terminals form and clicking *Unmap* for the instances, which are still selected, in the Define Device Correspondence form.

Deleting a Device Correspondence

To delete a schematic to layout device correspondence,

1. From the layout window menu bar, choose *Connectivity – Define Device Correspondence*.

The Define Device Correspondence form appears.

2. Choose either *All Instances* or *All Terminals* from the *Filter* cyclic field, depending on which type of object you want to unmap.
3. Select the device you want to unmap from either the *Schematic* or *Layout* tree.

The devices to which it is mapped are selected in the other tree.

4. Click *Unmap*.

All the devices that formed the selected device correspondence are unmapped and the binding is updated.

Define Device Correspondence SKILL API

The following public SKILL functions let you set and access device correspondence information in your design. For more information, see Layout XL SKILL Functions in the *Virtuoso Layout Suite SKILL Reference*.

To get device correspondence information, use

- bndGetBoundObjects
- bndGetSiblingBoundObjects

To add, set, and replace device correspondence information for instances only, use

- bndAddInstsBindingByName
- bndRemoveInstBindingByName
- bndReplaceInstsBindingByName
- bndSetInstsBindingByName

To add, set, and replace device correspondence information for terminals only, use

- bndRemoveTermBindingByName
- bndReplaceTermsBindingByName

■ bndSetTermsBindingByName

To add, set, and replace device correspondence information for either terminals, shapes or instances, use

■ bndAddObjectsBinding

■ bndRemoveObjectBinding

■ bndReplaceObjectsBinding

■ bndSetObjectsBinding

Updating the Connectivity Reference

Use the *Update Connectivity Reference* command to update the source schematic for your layout view and to change the physical configuration which specifies how the layout is generated from the schematic.

Updating the Schematic View

In Layout XL, the default schematic is always the schematic view with the same cell name as the layout view. To update your layout using a different schematic, you need to update the connectivity reference,

1. Choose *Connectivity – Update – Connectivity Reference*.

The Update Connectivity Reference form is displayed.

2. Check the *Use schematic view* option and do one of the following.

- ☐ Type the *Library*, *Cell*, and *View* names of the schematic in the fields provided. You can type in the name of a different library. The default view name is `schematic`.
- ☐ Use *Set by Cursor* and click on an open schematic window
- ☐ Use the *Browse* button to browse your file system for the schematic you want.

3. Switch on *Update layout cellviews hierarchically* to update the library setting for any lower-level cellviews in a hierarchical cellview.

4. Click *OK*.

Layout XL sets the connectivity reference to the schematic you selected.

Updating the Physical Configuration View

To change the physical configuration in use for a particular schematic,

1. Choose *Connectivity – Update – Connectivity Reference*.

The Update Connectivity Reference form is displayed.

2. Check the *Use configuration view* option and do one of the following.

- ☐ Type the *Library*, *Cell*, and *View* names of the view in the fields provided. You can type in the name of a different library. The default view name is `physConfig`.
- ☐ Use the *Browse* button to browse your file system for the view you want.

3. Click *OK*.

Layout XL sets the physical configuration to the view you specified.

Virtuoso Layout Suite XL User Guide

Updating Design Data

Troubleshooting Layout XL

This chapter tells you what to do if unexpected results occur while using the Virtuoso® Layout Suite XL layout editor (Layout XL). It covers the following topics.

Troubleshooting Cloning

- [Structure Already Exists in the Layout](#) on page 575
- [Connectivity Structure is Different](#) on page 576
- [Master Cells are Different](#) on page 576
- [More Source Components are Selected than Target Components](#) on page 577
- [Properties are Different](#) on page 578
- [Automatic Parameter Update Causes Different Submasters \(Message LX-2149\)](#) on page 579

Troubleshooting Configure Physical Hierarchy

- [Schematic Properties Not Present in physConfig](#) on page 580
- [NLP Expressions Cannot be Converted to the New Schema](#) on page 580
- [Global or Inherited Nets Beyond Physical Leaf Cells are not Detected](#) on page 581

Troubleshooting Connectivity Extraction

- [Connections Not Made](#) on page 582
- [Layout XL Does Not Recognize Connected Pins and Nets](#) on page 582
- [Markers for Nonexistent Overlaps and Shorts Appear](#) on page 582
- [Path Ends Not Accepted](#) on page 583
- [Placement and Routing Do Not Run](#) on page 583

- [Layout XL Does Not Recognize Physical Vias](#) on page 583
- [Moving Software Executables To a New Location](#) on page 584
- [Unmatched Instance Terminals are Automatically Removed](#) on page 584
- [Layout XL Commands Requiring Schematic Extraction](#) on page 584

Troubleshooting Constraint Migration

- [CMX Legacy Net Classes](#) on page 584

Troubleshooting Editing

- [Components Move Slowly](#) on page 585
- [Extra Probes Appear](#) on page 585
- [Generate Layout Form Does Not Keep Values from the Last Entry](#) on page 585
- [Parameters Not Updated](#) on page 585
- [Schematic Not Editable](#) on page 585
- [Warning to Update Your Design Appears at Startup](#) on page 586

Troubleshooting Generating Selected Components From Source

- [Hierarchical Designs](#) on page 586
- [Global Terminals and Implicit Inherited Connection Terminals](#) on page 587

Troubleshooting Parameter Evaluation

- [Netlisting Mode](#) on page 587
- [Evaluating CDF Callbacks by Default](#) on page 587

Troubleshooting Performance

- [Design Fails to Open or Takes a Long Time to Open](#) on page 588
- [Layout XL Is Slow](#) on page 588

Troubleshooting Startup

- [Invalid Markers from Previous Software Versions](#) on page 589
- [Options Form Does Not Appear](#) on page 590

Troubleshooting Cloning

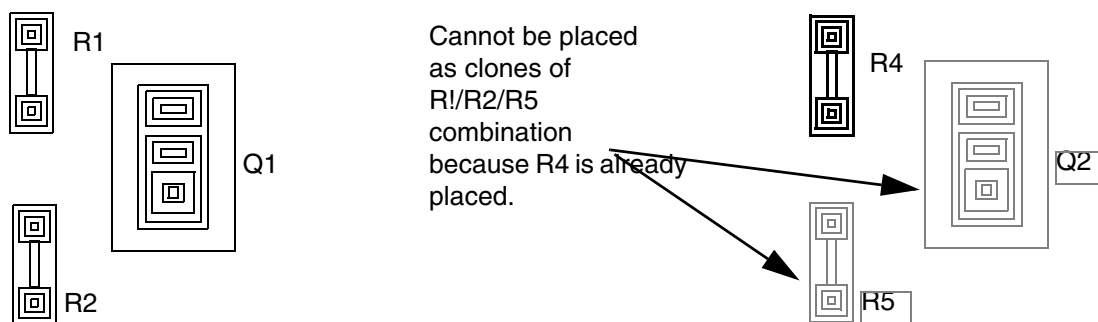
If a target cannot be cloned, no outline of the layout structure appears and the software displays the following message.

```
Cloning could not be completed.  
Check for:  
    Target partially implemented  
    Connectivity Structure mismatch  
    Mismatch of parameters or masters
```

Check the following.

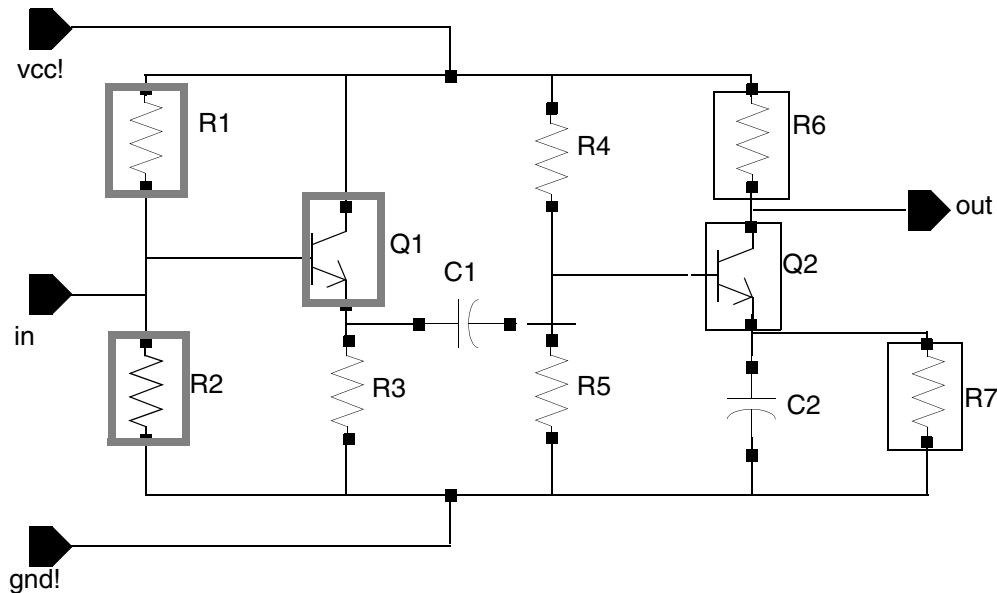
Structure Already Exists in the Layout

The structure or part of the structure that you selected as the target already exists in the layout. For example, here R1, R2, and Q1 in the layout were the source components and R4 (which is already implemented), R5, and Q2 were the target components.



Connectivity Structure is Different

The connectivity structure of the source and the target is not the same; for example, if you chose R1, R2, and Q1 as source components and R6, R7, and Q2 as target components.

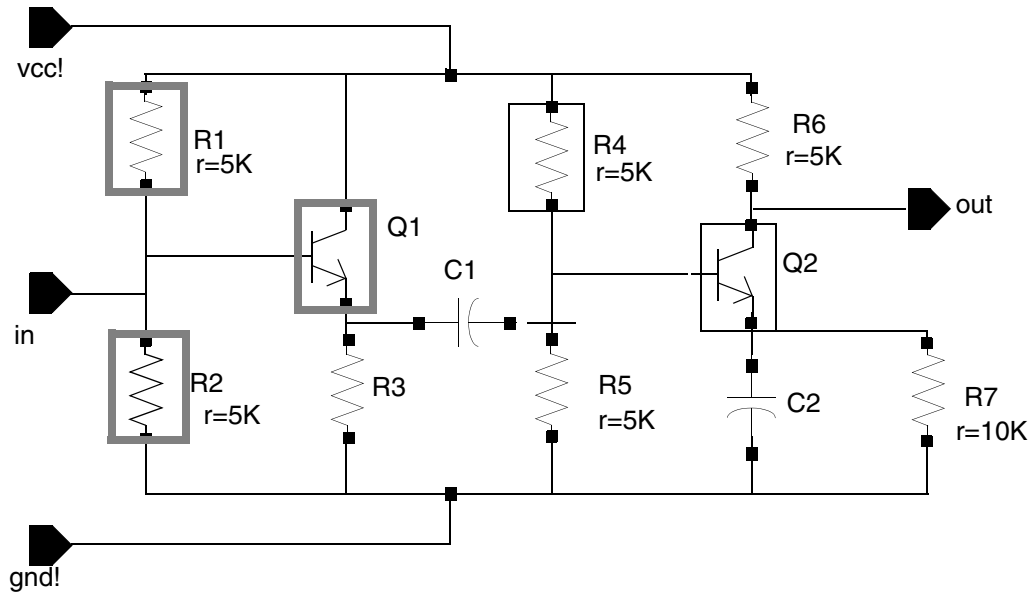


Master Cells are Different

The master cells of the source and the target instances are not the same.

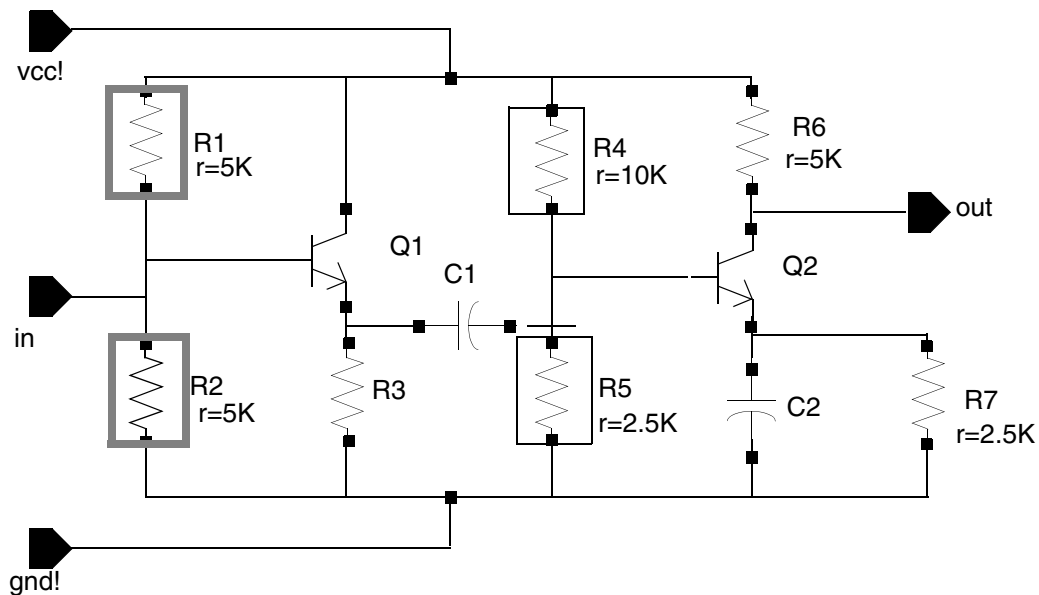
More Source Components are Selected than Target Components

More source components are selected than target components. For example, if you selected R1, R2, and Q1 as source components and selected only R4 and Q2 for target components.



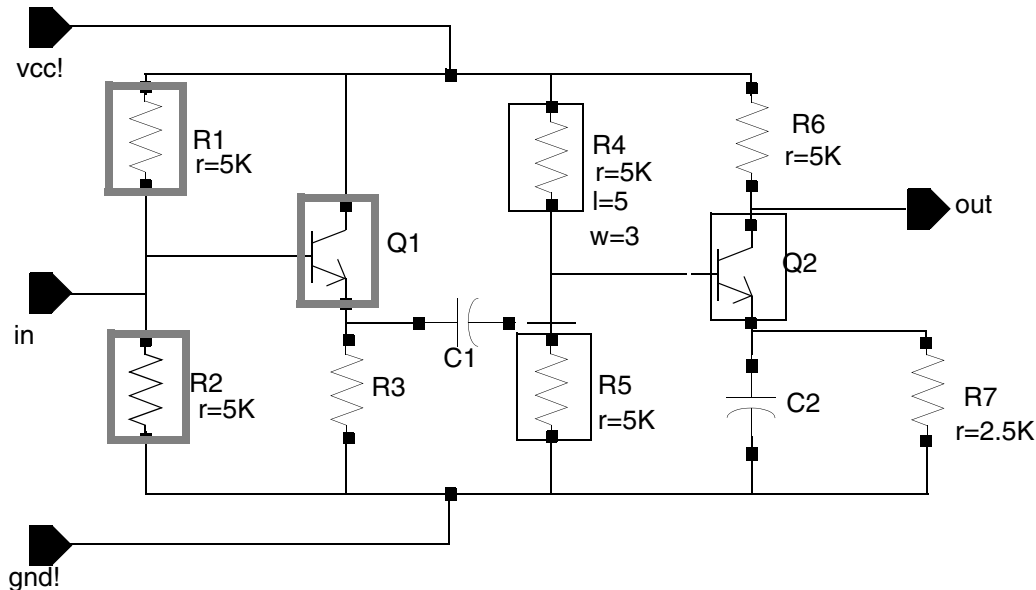
Properties are Different

The properties of the source and target instances are not the same. For example, you could not use R1 and R2 in this diagram to clone R4 and R5 because their resistance value properties are different.



The target instance can have properties that are additional to the properties of the source instance, but the target instance must have all the same properties with the same values as the source instance has.

In the example below, you can use R1, R2, and Q1 as source components to clone R4, R5, and Q2 because the targets have the same properties and values as the sources, even though one of the targets has additional properties.



You can assign the property `lxParamsToIgnore` to source instances with properties that are different from the target instance properties, so that the source instances can be cloned successfully as target instances.

Automatic Parameter Update Causes Different Submasters (Message LX-2149)

When you search for clones with *Exact Parameter Match* turned off, Layout XL also lists targets with components that do not have same parameter values. When you generate a clone from one of these targets, the software automatically updates the parameters and parameter values in the generated clone to match those defined on its schematic counterparts.

This automatic parameter update can lead to the creation of different submasters for the cloned instances. This in turn can cause problems when you try to generate synchronous clones and with existing abutments in the design.

You can avoid this situation by running the *Update Layout Parameters* command before selecting the clone source and then *Find Matching Targets* with *Exact Parameter Match* turned on. If the different submasters are causing problems with abutment, add the abutment

properties to the list of properties to be ignored in the Parameters Tab of the Layout XL Options form.

Troubleshooting Configure Physical Hierarchy

Schematic Properties Not Present in physConfig

A schematic instance contained in a cell must have its master translated first in order for the properties to be converted correctly. The `cdb2oa` and `dfIIoa20222` translators attempt to ensure that this always happens, however, it is not guaranteed. If the properties for a particular schematic instance do not appear in the physical configuration view, Cadence recommends that you translate your data to OpenAccess 2.2 first and then perform the conversion to use the IC 6.1 Layout XL schema as a separate step.

NLP Expressions Cannot be Converted to the New Schema

The conversion commands cannot convert the following schematic properties if they are defined as an NLP expression on a cell.

- `lxRemoveDevice`
- `lxRounding`
- `lxStopList`
- `lxUseCell`
- `lxViewList`

When the system encounters this situation it issues the following message.

```
Cannot convert the 'oldPropName' property with value 'nlpExpr' on instance
'instName' in logical cellview 'libName/cellName/viewName' because NLP
expressions are not supported. Use the Configure Physical Hierarchy window to
specify explicitly the newOptionName either on a per cell or a per instance
basis.
```

newOptionName is the name of the relevant field in the Configure Physical Hierarchy window's *Cells* or *Instances* table or the name of the relevant option field in the Attributes

pane. The mapping between *oldPropName* and *newOptionName* in the message is as follows.

oldPropName	newOptionName
<code>lxRemoveDevice</code>	<i>Remove device option</i>
<code>lxRounding</code>	<i>Rounding option</i>
<code>lxStopList</code>	<i>Inherited Stop List</i>
<code>lxUseCell</code>	<i>Physical Cell</i>
<code>lxViewList</code>	<i>Inherited View List</i>

Global or Inherited Nets Beyond Physical Leaf Cells are not Detected

The default behavior of the Configure Physical Hierarchy design elaboration was changed in version **IC 6.1.2.500.14** and subsequent releases.

Previously, Configure Physical Hierarchy always elaborated the entire logical design hierarchy beyond the nodes which mapped to physical leaf cells. Now, the software stops the elaboration when it reaches a logical node that maps to a physical view with one of the view names specified in the *Physical stop view list* field.

This lets you use the physical stop view list to limit the logical elaboration for large hierarchical designs that would otherwise take a long time to open or might not open at all in Layout XL. However, it also means that global or inherited nets below the leaf instance in the schematic are not considered during the elaboration.

If your design relies on such nets, Cadence recommends that you switch off this environment and re-elaborate your design. To do this,

1. In the CIW, type

```
envSetVal("layoutXL" "cphStopLogicalElabAtPhysLeaf" 'boolean nil)
```

2. Force a full re-elaboration by changing the *Logical switch view list* in the *Global Bindings* pane in the Configure Physical Hierarchy window.

Troubleshooting Connectivity Extraction

Connections Not Made

If the software does not accept an electrical connection you make, make sure that the following information is set in the technology file. For more information, see [Technology File Requirements for Layout XL](#) and in particular [Connectivity Rules](#).

- Connectivity information in the appropriate constraint group, for example `virtuosoDefaultExtractorSetup`.
 - ❑ The `validLayers` specifies all the layers you want to extract through.
 - ❑ The `errorLayer` constraint, along with the `techDerivedLayers` section, specifies layers that are not permitted to overlap.
- Layer information in the `layerRules` section.
 - ❑ The `functions` section, which specifies the layer function and assigns a mask number to each layer.
 - ❑ The `equivalentLayers` section defines layers that you want to be considered electrically equivalent.

Also check that the layer properties were set when you made the connection.

Layout XL Does Not Recognize Connected Pins and Nets

If a pin is connected to a net but has the `lxBlockExtractCheck` property on it, Layout XL is unable to recognize that the pin is connected.

If a layer you use for pcells is not listed in the technology file in the `validLayers` constraint, Layout XL does not recognize incomplete nets noted on that layer. Furthermore, the probe function does not highlight any layer that is not listed in `validLayers`.

Markers for Nonexistent Overlaps and Shorts Appear

If there are markers for nonexistent overlaps and shorts in the layout, make sure that the appropriate layer was active when the path was created.

If this does not fix the problem, make sure that the layer information and connectivity information are set correctly in the technology file. For more information, see [Technology File Requirements for Layout XL](#)

Path Ends Not Accepted

If the software rejects the final segment of a path or wire connection and displays the following message.

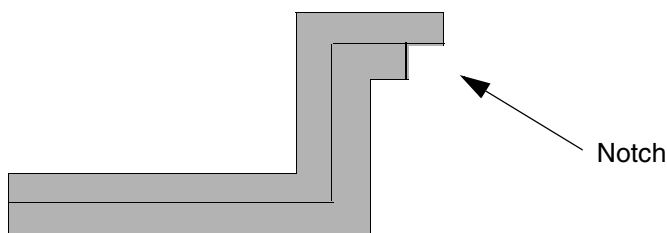
First or last segment of created path has length less than or equal to half the path width

- Press `Return` to end a path instead of double clicking the mouse (do not double click to end a path).

or

- From the layout window menu bar, choose *Options – Display* and set the *Snap Mode* for *Create* commands to something other than *L90XFirst* or *L90YFirst*.

When you use the *L90XFirst* or *L90YFirst* snap modes, the double click often makes a notch in the path, which triggers the error message and can cause errors in mask layout.



Placement and Routing Do Not Run

Layout XL does not let you place or route the elements of a design unless you have defined which layers of your layout design are conducting layers.

If you try to place or route your design before defining the conducting layers, you get an error message.

Define the layers you want to be conducting layers in the `viaLayers` constraint in the technology file. For more information, see [Constraint Groups](#).

Layout XL Does Not Recognize Physical Vias

If you have an existing design in which Layout XL does not recognize physical vias, you need to add a property with name `function` and value `via` to the master cellview of the via or to the component description format (CDF) for via layout.

For information on adding a via, see [Editing and Defining Properties](#) in the *Virtuoso Layout Suite L User Guide*.

Moving Software Executables To a New Location

If you move software executables to a different location other than specified by manufacturing, the custom placer function may take longer to start because it is searching for the executable.

Unmatched Instance Terminals are Automatically Removed

When loading designs that have been edited outside Layout XL, the connectivity extractor silently cleans up any instance terminals that have no associated terminal in the cell master. This can mean that you are asked to save your design, even though nothing has visibly changed.

Layout XL Commands Requiring Schematic Extraction

Certain Layout XL commands, such as *Update Components And Nets*, *Pick From Schematic*, *Generate All From Source*, check that the schematic connectivity is up-to-date before proceeding. If the schematic has been changed since it was last extracted, you see a message indicating that it needs to be re-extracted.

If you see this message every time you run one of these commands, it is likely that there are errors in the schematic that have prevented the previous extraction from completing successfully.

If this is the case, you need to run extraction again, fix any errors highlighted, and save the schematic. The next time you run a Layout XL command, you will not be prompted to re-extract the schematic.

Troubleshooting Constraint Migration

CMX Legacy Net Classes

CMX legacy net classes are listed in the XL Probe form but are not shown under the *Net Class* category in the Constraint Manager assistant. This does not impact the individual constraints migrated.

Troubleshooting Editing

Components Move Slowly

If you find that the *Move* command works too slowly, especially when you have many iterated instances, turn off the display of incomplete net flight lines or display only nets that are relevant to the current task.

Extra Probes Appear

To delete unwanted probes in the schematic or layout canvas while the Probe command is active,

- Click on an empty space in the design.

or at any time

- Press `Control-Shift-L`.

Generate Layout Form Does Not Keep Values from the Last Entry

The Generate Layout form displays default values each time it opens, it does not carry over values from the previous time the form was displayed.

To use predefined settings, set the appropriate environment variables in your `.cdsenv` file before you start Layout XL. Alternatively, you can load predefined cellview information from another OpenAccess cellview. For more information, see on page 296.

Parameters Not Updated

If the parameters are not updated the way you expect, check for the presence of the `lxParamsToIgnore` property on components of your design.

Schematic Not Editable

If you cannot make changes to the schematic,

- Check that the schematic is editable. (If it is, the word “Editing” is displayed in the window banner; if it is not, the word “Reading” is displayed.)

If it is not editable, make it so using the *File – Make Editable* command.

- Check with your system administrator that you have a license for the Virtuoso Schematic Editor (product 302).

Warning to Update Your Design Appears at Startup

When you open a design in Layout XL and an Info window appears, it means that the schematic has been changed after the last time the layout was changed, so the layout does not represent the latest version of the design.

You can check the layout against the changed schematic using the layout window *Check Against Source* command and then update the layout view using either the *Update Components And Nets* or *Define Device Correspondence* commands as needed.

Troubleshooting Generating Selected Components From Source

If the *Generate Selected From Source* command fails, check that

- The instance is not already placed in the layout.

If it is already placed, you cannot use *Generate Selected From Source* to generate it again.

- All the required masters and reference views are present, including
 - The default layout master for the specified schematic component.
 - Any layout master specified by the `useCell` attribute for the schematic instance or master.
 - The required layout view as derived from the `stopList`.

Hierarchical Designs

If you push into a hierarchical schematic view beyond the layout stop point, you cannot use the *Generate Selected From Source* command to generate lower-level instances inside that top-level cell.

For example, assume there is a layout view derived from the specified `stopList` corresponding to a `cell1` instantiated in the schematic. If you descend into an instance of `cell1` and try to generate the lower-level instances, the command fails.

Global Terminals and Implicit Inherited Connection Terminals

Generate Selected From Source does not generate global terminals or implicit inherited connection terminals (i.e., wires with `netSet` properties).

Troubleshooting Parameter Evaluation

Netlisting Mode

To ensure that CDF parameters are always evaluated correctly, make sure that the `CDS_Netlisting_Mode` shell environment variable is set to `Analog`.

To set `CDS_Netlisting_Mode`,

- Type the following commands in the CIW.

```
setShellEnvVar("CDS_Netlisting_Mode=Analog")  
cdsSetNetlistMode()
```

To check which mode is currently set,

- Type the following command in the CIW.

```
cdsGetNetlistMode()
```

For more information on `CDS_Netlisting_Mode`, see [Customizing the Simulation Environment](#) in the *Open Simulation System Reference*.

Evaluating CDF Callbacks by Default

To ensure that SKILL callbacks defined on CDF parameters (for example, `lxCombination`) are evaluated by default, you must set the `lxEvalCDFCallbacks` environment variable to `t`.

`lxEvalCDFCallbacks` causes all SKILL callbacks defined on CDF parameters to be evaluated by default when you run the following Layout XL commands. Note that callbacks defined on buttons are never evaluated.

- *Generate All From Source*
- *Generate Folded Devices*
- *Update Components And Nets*

- *Update Layout Parameters*
- *Update Schematic Parameters*

The callbacks are evaluated in the order in which they are listed in the CDF, so it is important that you set the appropriate callback to the appropriate place in the list. For more information, see [Defining Parameters](#) in the *Component Description Format User Guide*.



Enabling this environment variable can adversely impact performance, especially on large designs with many callbacks.

Troubleshooting Performance

Design Fails to Open or Takes a Long Time to Open

If your design takes a long time to open in Layout XL, or fails to open at all, try the following.

- If your design has many devices which overlap but have not been abutted previously, and you start Layout XL with full connectivity extraction, by default the software attempts to automatically abut the devices in question. To prevent this, set the [lxLocalAbutment](#) environment variable to `t`.
- If your design is hierarchical, it might be that the logical elaboration performed by Configure Physical Hierarchy is causing the problem.

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

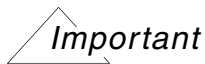
Note: When `cphStopLogicalElabAtPhysLeaf` is set to `t`, global nets below the leaf instance in the schematic are not considered during the elaboration. If your design relies on such nets, Cadence recommends that you do not switch on this environment variable.

Layout XL Is Slow

The following considerations can help you optimize Layout XL performance speed.

- Layout XL is targeted for use in designs with fewer than 10,000 devices. The performance of the online extractor degrades substantially beyond that limit.

- Instead of displaying flight lines for all the incomplete nets in the design, use the highlighting features of the path command or probing to determine connections or display only nets you are working on or that are relevant to the current task.
- If you need extra pins in the layout for feedthrough nets or substrate connections, consider adding those after you have completed the main portion of the design. Adding extra pins increases the number of nets the extractor manages and degrades performance.
- Specify only layers to be used as interconnect in the `validLayers` constraint. Using more layers than necessary causes the extractor to check all shapes on those layers, which slows performance. If you do not have interconnect to wells, do not specify the well in the `validLayers` constraint.



For more information, see [Constraint Groups](#) on page 69.

Troubleshooting Startup

Invalid Markers from Previous Software Versions

If you are opening layouts you developed using previous versions of Virtuoso software, you can clean up the layout canvas and get rid of invalid markers.

1. From the layout window menu bar, choose *Connectivity – Update – Components And Nets*.

Layout XL resynchronizes the layout connectivity with the schematic connectivity.

2. From the layout window menu bar, choose *Verify – Markers – Delete All*.

The markers disappear.

3. Choose *File – Save*.

4. From the layout window menu bar, choose *Window – Close*.

The layout window closes.

5. From the schematic window, choose *Window – Close*.

6. The schematic window closes.

7. From the Command Interpreter Window (CIW), choose *File – Open*.

The Open File form is displayed.

8. From the *Library Name* cyclic field, choose the library name of your design.
9. In the *Cell Name* field, type or select from the *Cell Names* list the cell name of your design.
10. From the *View Name* cyclic field, choose the view name of your design.
11. Click *OK*.

The layout window appears, displaying the updated layout configuration.

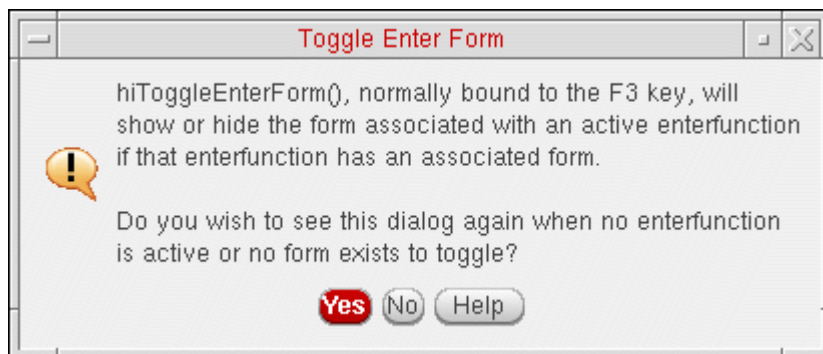
12. From the layout window menu bar, choose *Launch – Layout XL*.

The Layer Selection Window (LSW) and the schematic view appear.

Note: If you still have problems, check that your technology is correctly defined. For more information, see [Technology File Requirements for Layout XL](#).

Options Form Does Not Appear

When you press **F3** to see an options form listing possible choices for a command and the command does not have an options form, the message box shown below appears:



If you want the message box to appear every time you press **F3** for a command that has no options form, click *Yes*.

If you do not want the message box to appear every time you press **F3** for a command that has no options form, click *No*.

Layout XL Environment Variables

This appendix provides information on the names, descriptions, and graphical user interface equivalents for Virtuoso® Layout Suite XL layout editor (Layout L) environment variables.

Note: Only the environment variables documented in this chapter are supported for public use. All other Layout XL environment variables, regardless of their name or prefix, and undocumented aspects of the environment variables described below, are private and are subject to change at any time.

Inherited Environment Variable Settings

Many of the environment variables honored by Layout XL are set in Layout L. Information on these environment variables is not duplicated in this section.

For more information on Layout L environment variables, see [Environment Variables](#) in the *Virtuoso Layout Suite L User Guide*.

Related Topics

[List of Layout XL Environment Variables](#)

[Setting Environment Variables](#)

[GUI Options and Corresponding Environment Variables](#)

[Hidden Environment Variables with No Corresponding GUI Option](#)

List of Layout XL Environment Variables

[autoAbutment](#) on page 598

[autoArrange](#) on page 599

[autoMirror](#) on page 601

[autoMirrorChains](#) on page 603

[autoSpace](#) on page 606

[autoZoomIsFixed](#) on page 608

[bndLargeBinderNetThreshold](#) on page 609

[capacitanceParamNames](#) on page 610

[chainDummyFlexBothEndNets](#) on page 612

[chainExtendSelection](#) on page 613

[chainFolds](#) on page 614

[chainPreserveExistingChains](#) on page 615

[chainUseDeviceOrder](#) on page 616

[checkMissingParamsOrProps](#) on page 617

[checkParamsOnly](#) on page 618

[checkStateConfirmModeChange](#) on page 620

[ciwWindow](#) on page 621

[cloningDoExactMatch](#) on page 622

[colorDraglines](#) on page 624

[constraintAwareEditing](#) on page 626

[cphPromptConfigOrSchematic](#) on page 628

[cphStopLogicalElabAtPhysLeaf](#) on page 629

[createBoundaryLabel](#) on page 631

[createImplicitBusTerminals](#) on page 632

[crossSelect](#) on page 634

[deleteConfirmModeChange](#) on page 635

[deleteUnmatchedInsts](#) on page 636

[deleteUnmatchedPins](#) on page 637

[deviceExtractType](#) on page 638

[disableCASOptionsPopUp](#) on page 640

[drdUseNetName](#) on page 644

[extractShowMustConnectMarkers](#) on page 646

[extractShowUnimplementedInstTerms](#) on page 647

[extractShowWeakMarkers](#) on page 648

[flightLineEnable](#) on page 649

[foldOptimizeSupplyAndGroundNets](#) on page 651

[hideDraglinesForGlobalNets](#) on page 652

[hideMarkersWhenBrowserHidden](#) on page 654

[hierarchyDepth](#) on page 655

[highlightedIsSelectable](#) on page 656

[incNetHiliteLayer](#) on page 657

[inductanceParamNames](#) on page 658

[infoWindow](#) on page 660

[initAspectRatio](#) on page 662

[initAspectRatioOption](#) on page 663

[initCreateBoundary](#) on page 664

[initCreateInstances](#) on page 665

[initCreateMTM](#) on page 666

[initCreatePadPins](#) on page 668

[initCreatePins](#) on page 669

[initCreateSnapBoundary](#) on page 670

[initDoFolding](#) on page 671

[initDoStacking](#) on page 672

[initEstimateArea](#) on page 673

[initGlobalNetPins](#) on page 674

[initIOLabelType](#) on page 675

[initIOPinLayer](#) on page 677

[initIOPinName](#) on page 679

[initPinHeight](#) on page 680

[initPinMultiplicity](#) on page 682

[initPinWidth](#) on page 683

[initPrBoundaryH](#) on page 685

[initPrBoundaryW](#) on page 686

[initUtilization](#) on page 687

[labelOrient](#) on page 688

[layoutWindow](#) on page 690

[lswWindow](#) on page 691

[lxAllowPseudoParallelNets](#) on page 692

[lxChainAlignNMOS](#) on page 694

[lxChainAlignPMOS](#) on page 696

[lxDeltaWidth](#) on page 698

[lxEvalCDFCallbacks](#) on page 699

[lxFingeringNames](#) on page 701

[lxGenerationOrientation](#) on page 703

[lxGenerationTopLevelOnly](#) on page 710

[lxGetSignifDigits](#) on page 711

[lxGroundNetNames](#) on page 713

[lxInitResetSource](#) on page 714

[lxLocalAbutment](#) on page 715

[lxPositionMinSep](#) on page 717

[lxPositionPinsOnBoundary](#) on page 718

[lxRetainFoldOrient](#) on page 719

[lxSchematicDefaultApp](#) on page 720

[lxSchExtractTopLevelOnly](#) on page 721

[lxStackMinimalFolding](#) on page 722

[lxStackPartitionParameters](#) on page 723

[lxSupplyNetNames](#) on page 725

[lxUpdateFoldedWidth](#) on page 726

[lxUseLibList](#) on page 728

[lxValidateXLParameterEvaluation](#) on page 730

[lxWidthTolerance](#) on page 732

[mfactorNames](#) on page 733

[mfactorSplit](#) on page 734

[moveAsGroup](#) on page 735

[openConnRef](#) on page 736

[openConnRefTab](#) on page 737

[openLocation](#) on page 738

[openMode](#) on page 739

[paramsToIgnore](#) on page 740

[paramsToIgnoreForCheck](#) on page 742

[paramTolerance](#) on page 743

[pinTextSamePurpose](#) on page 745

[prefixLayoutInstNamesWithPipe](#) on page 746

[preserveAreaBoundary](#) on page 747

[preserveBlockages](#) on page 748

[preserveClusters](#) on page 749

[preserveClusterBoundaries](#) on page 750

[preserveRows](#) on page 751

[preserveTrackPattern](#) on page 752

[probeCycleHilite](#) on page 753

[probeDevice](#) on page 754

[probeDuringCreate](#) on page 755

[probeHiliteLayer](#) on page 756

[probeInfoInCIW](#) on page 757

[probeNet](#) on page 758

[probePin](#) on page 759

[propsUsedToIgnoreObjs](#) on page 760

[propsUsedToIgnoreObjsForCheck](#) on page 762

[resistanceParamNames](#) on page 763

[schematicWindow](#) on page 765

[scopeLevel](#) on page 766

[setPPConn](#) on page 768

[sfactorNames](#) on page 769

[showDraglinesForDistantConns](#) on page 771

[snapCpaToPlacementGrid](#) on page 773

[stopList](#) on page 774

[suppressExpansion](#) on page 776

[transistorWidthParamNames](#) on page 778

[turnInfixOffWhenSmartSnapping](#) on page 780

[updateLayoutParameters](#) on page 781

[updateNetSigType](#) on page 782

[updateNetsOnly](#) on page 783

[updateOneToOneMappings](#) on page 784

[updatePlacementStatus](#) on page 785

[updatePlacementStatusInBoundary](#) on page 786

[updateReplacesMasters](#) on page 787

[updateSelectedComponents](#) on page 789

[updateWithMarkers](#) on page 790

[viewList](#) on page 791

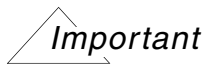
autoAbutment

```
layoutXL autoAbutment 'boolean { t | nil }
```

Description

Turns on automatic abutment, which abuts prepared transistors so that they can share pins. Abutment is switched on for both automatic and interactive layout generation and during automatic placement using the Virtuoso custom digital placer.

The default is `t`, which means that devices that have not been abutted but are overlapping can be abutted. If the devices are already abutted, they are not reabutted.



Auto-abutment works only when automatic extraction is on; i.e, when the extractEnabled environment variable is set to `t`.

Arguments

None.

GUI Equivalent

Command: *Options – Layout XL (Generation tab)*

Field: *Abut transistors*

Examples

```
envGetVal("layoutXL" "autoAbutment")  
envSetVal("layoutXL" "autoAbutment" 'boolean t)  
envSetVal("layoutXL" "autoAbutment" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

autoArrange

```
layoutXL autoArrange 'boolean { t | nil }
```

Description

Controls whether Layout XL automatically rearranges its four windows on your desktop when you launch the application.

The default is `t`, meaning that the windows are positioned based on the values specified for the following environment variables.

- [ciwWindow](#)
- [layoutWindow](#)
- [lswWindow](#)
- [schematicWindow](#)

If those environment variables are not set, or they are set to the default value – "`((0.0 0.0) (0.0 0.0))`" – Layout XL tiles the four windows based on the height and width of your screen.

Arguments

None.

GUI Equivalent

Command: *Options – Layout XL (Display tab)*

Field: *Auto arrange windows*

Examples

```
envGetVal("layoutXL" "autoArrange")  
envSetVal("layoutXL" "autoArrange" 'boolean t)  
envSetVal("layoutXL" "autoArrange" 'boolean nil)
```

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

Additional Information

If you are using dual monitors and you do not want Layout XL to occupy the entire space available, set the `USE_SINGLE_MONITOR` global environment variable before you start Virtuoso from the command line.

Related Topics

[List of Layout XL Environment Variables](#)

autoMirror

```
layoutXL autoMirror 'boolean { t | nil }
```

Description

Specifies that if a short violation is created during abutment, then Layout XL first mirrors the device in question in order to resolve the short. Only if this is unsuccessful does Layout XL attempt to resolve the short by permuting the pins.

If you switch the option off, Layout XL uses only pin permutation when attempting to resolve shorts.

The default is `t`.

Arguments

None.

GUI Equivalent

Command: *Options – Layout XL (Generation tab)*

Field: *Mirror transistors*

Examples

```
envGetVal("layoutXL" "autoMirror")  
envSetVal("layoutXL" "autoMirror" 'boolean t)  
envSetVal("layoutXL" "autoMirror" 'boolean nil)
```

Additional Information

- If one of the devices is in any type of group (including a synchronous clone), the other device must be in the same group for automatic mirroring to occur.
- Automatic mirroring is disabled when constraint-aware editing is on and constraints exist between the instances and chains under consideration.

Related Topics

[List of Layout XL Environment Variables](#)

autoMirrorChains

```
layoutXL autoMirrorChains 'boolean { t | nil }
```

Description

Specifies that if a short violation is created during abutment of a chain of devices, then Layout XL first mirrors the chain in question in order to resolve the short. Only if this is unsuccessful does Layout XL attempts to resolve the short by permuting the pins.

The default is `t`.

Arguments

None.

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "autoMirrorChains")  
envSetVal("layoutXL" "autoMirrorChains" 'boolean t)  
envSetVal("layoutXL" "autoMirrorChains" 'boolean nil)
```

Additional Information

Chains can be mirrored only if

- The autoMirror environment variable is set to `t`
- They have no physical connectivity or routing defined
- The cells in the chain all belong to the same group (including synchronous clones) or no group
- The chain contains no multiply-abutted cells
- Any end-pin violations are valid left/right or top/bottom pairs

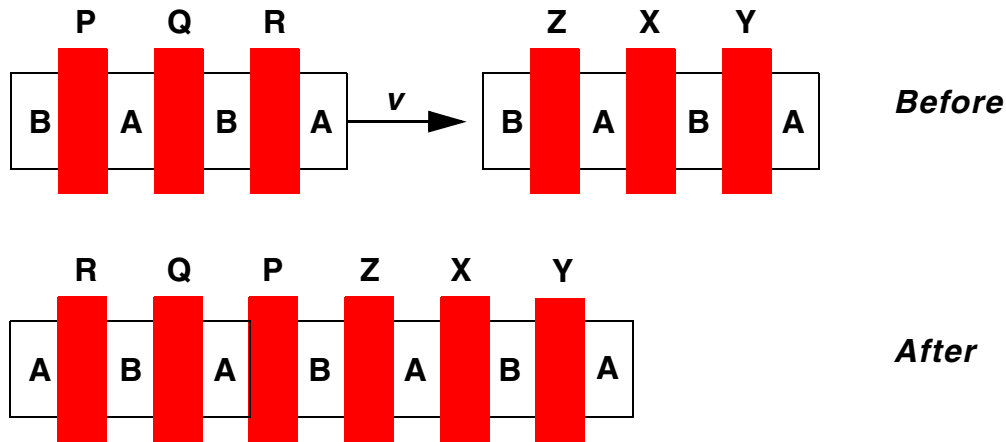
Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

The following transformations are possible. Which of the three is employed is controlled internally by the abutment code. You are not required to choose a particular transformation. No transformation is performed which would break the chain.

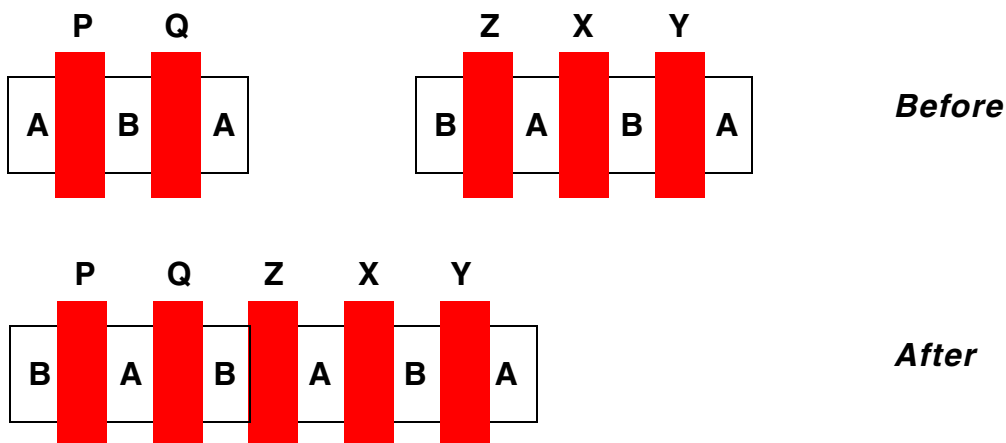
■ *Chain Mirror*

Mirrors the complete moving chain about its central Y axis to allow abutment.



■ *Cascade Mirror*

Mirrors each individual device in the chain to allow abutment.

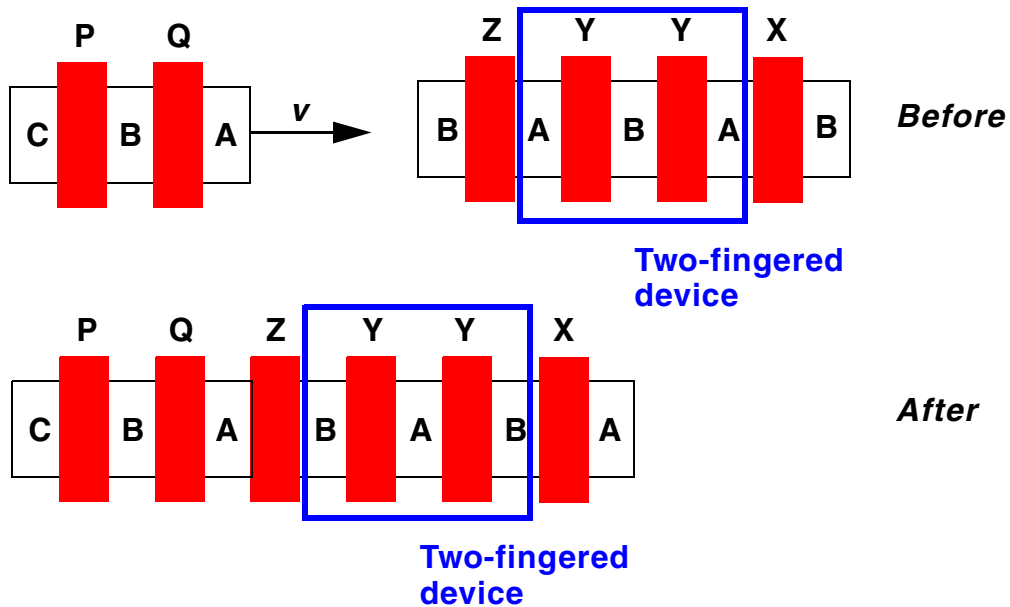


■ *Cascade Mirror Permute (2 nets)*

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

Neither *Chain Mirror* nor *Cascade Mirror* alone can deliver a transformation that allows abutment. Instead, a mixture of *Cascade Mirror* and *Permute* is employed in order to achieve the desired result.



Related Topics

[List of Layout XL Environment Variables](#)

autoSpace

```
layoutXL autoSpace 'boolean { t | nil }
```

Description

Switches on automatic spacing, which allows components with the properties vxInstSpacingDir and vxInstSpacingRule to be spaced based on the values specified in the properties.

The default is `t`.

Arguments

None.

GUI Equivalent

Command: *Options – Layout XL (Generation tab)*

Field: *Space components*

Examples

```
envGetVal("layoutXL" "autoSpace")  
envSetVal("layoutXL" "autoSpace" 'boolean t)  
envSetVal("layoutXL" "autoSpace" 'boolean nil)
```

Additional Information

Automatic spacing is run only on pairs of instances that are not already abutted. This includes devices which overlap but which are not abutted (for example, instances where abutment has been attempted but has failed).

To establish whether an instance is abutted or not, type the following into the CIW.

```
inst~>groupMembers~>group~>name
```

This returns the names of the abutment groups to which the instance belongs. If the instance is not abutted, it returns `nil`.

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

If one of the devices is in any type of group (including a synchronous clone), the other device must be in the same group for automatic spacing to occur.

Related Topics

[List of Layout XL Environment Variables](#)

autoZoomIsFixed

```
ab autoZoomIsFixed 'boolean { t | nil }
```

Description

Controls whether the Annotation Browser's *Auto Zoom* function defaults to Fixed mode (t) or Minimal mode (nil).

The default is nil (Minimal mode).

Note: For more information on the Annotation browser *Auto Zoom*, see the section on the toolbar in [Annotation Browser Graphical User Interface](#).

Arguments

None.

GUI Equivalent

Command: *Windows – Assistants – Annotation Browser (Auto Zoom toolbar button)*

Field: *Fixed* and *Minimal* radio buttons

Examples

```
envGetVal("ab" "autoZoomIsFixed")  
envSetVal("ab" "autoZoomIsFixed" 'boolean t)  
envSetVal("ab" "autoZoomIsFixed" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

bndLargeBinderNetThreshold

layoutXL bndLargeBinderNetThreshold 'int *integer*

Description

Specifies the minimum number of instance terminals that can be attached to a net before the net is considered as large.

Large nets are not used in the binding process for performance reasons. As a result, instances connected to large nets can be left unbound, for example, if the only connection possible is through a large net.

The default is 50.

Arguments

integer

A non-zero, positive integer that specifies the number of instance terminals attached to a net, beyond which the net is considered as large.

GUI Equivalent

None.

Examples

```
envGetVal ("layoutXL" "bndLargeBinderNetThreshold")  
envSetVal ("layoutXL" "bndLargeBinderNetThreshold" 'int 50)
```

Related Topics

[List of Layout XL Environment Variables](#)

capacitanceParamNames

```
layoutXL capacitanceParamNames 'string "list_of_parameters"
```

Description

Lists the names of the schematic parameters used to specify capacitance.

The default is "c C".

The capacitance value must be split among the generated series-connected devices during the following commands.

- *Generate All From Source*
- *Check Against Source*
- *Update Layout Parameters.*

Layout XL checks each schematic instance for one of the listed parameter names and updates the capacitance values of the matching layout parameters accordingly.

Arguments

list_of_parameters

A list of parameter names each separated by a space. The list must be enclosed in quotation marks; for example,

"c C"

GUI Equivalent

Command: *Options – Layout XL (Parameters tab)*

Field: *Capacitance*

Examples

```
envGetVal("layoutXL" "capacitanceParamNames")  
envSetVal("layoutXL" "capacitanceParamNames" 'string "c C")
```

Related Topics

List of Layout XL Environment Variables

chainDummyFlexBothEndNets

```
layoutXL chainDummyFlexBothEndNets 'boolean { t | nil }
```

Description

Ensures that dummy instances can change their nets on both sides during interactive chaining to support abutment on both sides.

The default is `nil`, which means dummy instances can change their net on only one side.

Arguments

None.

Examples

```
envGetVal("layoutXL" "chainDummyFlexBothEndNets")
envSetVal("layoutXL" "chainDummyFlexBothEndNets" 'boolean t)
envSetVal("layoutXL" "chainDummyFlexBothEndNets" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

chainExtendSelection

```
layoutXL chainExtendSelection 'boolean { t | nil }
```

Description

When only part of a chain is selected, extends the selection to include the whole chain, including any mfactored instances and legs of folded devices.

The default is `nil`, which means you can select individual devices, mfactors, or folded legs within a chain without selecting the entire chain.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – Chained Devices*

Field: *Extend Selection*

Examples

```
envGetVal("layoutXL" "chainExtendSelection")
envSetVal("layoutXL" "chainExtendSelection" 'boolean t)
envSetVal("layoutXL" "chainExtendSelection" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

chainFolds

```
layoutXL chainFolds 'boolean { t | nil }
```

Description

Lets Layout XL about the folds of newly folded devices into chains.

The default is `t`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – Folded Devices*

Field: *Chain Folds*

Examples

```
envGetVal("layoutXL" "chainFolds")  
envSetVal("layoutXL" "chainFolds" 'boolean t)  
envSetVal("layoutXL" "chainFolds" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

chainPreserveExistingChains

```
layoutXL chainPreserveExistingChains 'boolean { t | nil }
```

Description

Ensures that any existing chains in the design are not broken during interactive chaining.

The default is `nil`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – Chained Devices*

Field: *Preserve Existing Chains*

Examples

```
envGetVal("layoutXL" "chainPreserveExistingChains")  
envSetVal("layoutXL" "chainPreserveExistingChains" 'boolean t)  
envSetVal("layoutXL" "chainPreserveExistingChains" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

chainUseDeviceOrder

```
layoutXL chainUseDeviceOrder 'boolean { t | nil }
```

Description

Maintains the relative positions of instances in the generated chain by sorting the instances to be chained in X and Y axes and then abutting each device in the sorted list.

The default is `nil`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – Chained Devices*

Field: *Use Device Order*

Examples

```
envGetVal("layoutXL" "chainUseDeviceOrder")  
envSetVal("layoutXL" "chainUseDeviceOrder" 'boolean t)  
envSetVal("layoutXL" "chainUseDeviceOrder" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

checkMissingParamsOrProps

```
layoutXL checkMissingParamsOrProps 'boolean { t | nil }
```

Description

When set to `nil`, ignores parameters (and properties, if `checkParamsOnly` is also set to `nil`) that are present in one view but missing from the other.

The default is `nil`.

Set this to `t` if, for example, your layout has additional properties that you want to propagate to the schematic using the *Update Schematic Parameters* command.

This environment variable is honored by the following commands in the *Connectivity* menu.

- *Check Against Source*
- *Update Layout Parameters*
- *Update Schematic Parameters*

Arguments

None.

GUI Equivalent

Command: *Options – Layout XL (Parameters tab)*

Field: *Ignore missing parameters or properties*

Examples

```
envGetVal("layoutXL" "checkMissingParamsOrProps")
envSetVal("layoutXL" "checkMissingParamsOrProps" 'boolean t)
envSetVal("layoutXL" "checkMissingParamsOrProps" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

checkParamsOnly

```
layoutXL checkParamsOnly 'boolean { t | nil }
```

Description

Specifies whether the *Check Against Source*, *Update Layout Parameters*, and *Update Schematic Parameters* commands consider CDF parameters only or CDF parameters and cellview properties when comparing values in schematic and layout.

- When set to `t`, the commands check only CDF parameters in the schematic against the parameters and properties in the layout.
- When set to `nil`, the commands check both CDF parameters and cellview properties in the schematic against parameters and properties in the layout. Use this setting to check or update user-defined properties in the layout and schematic or when you know that a particular instance has certain properties that you are interested in comparing or updating.

In both cases, the commands then report (or update) values that do not match, parameters that cannot be checked, and parameters that are missing from the layout view.

The default is `nil`.

Arguments

None.

GUI Equivalent

Command: *Options – Layout XL (Parameters tab)*

Field: *Consider parameters only*
 Consider parameters and properties

Examples

```
envGetVal("layoutXL" "checkParamsOnly")
envSetVal("layoutXL" "checkParamsOnly" 'boolean t)
envSetVal("layoutXL" "checkParamsOnly" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

checkStateConfirmModeChange

```
ab checkStateConfirmModeChange 'boolean { t | nil }
```

Description

Controls whether or not you must confirm explicitly that a read-only cellview is to be re-opened in edit mode before setting the checked state of a marker in the Annotation Browser assistant.

The default is `t`, which means that when you try to use the *Check/Uncheck* command (or its browser pane equivalent) in a read-only cellview, the software prompts you to confirm that you want to re-open the cellview in edit mode before performing the requested operation. When set to `nil`, the software automatically re-opens the cellview in edit mode and performs the requested operation.



Tip

The `deleteConfirmModeChange` environment variable performs the same function for the Annotation Browser's *Delete* command.

Arguments

None.

GUI Equivalent

None.

Examples

```
envGetVal("ab" "checkStateConfirmModeChange")
envSetVal("ab" "checkStateConfirmModeChange" 'boolean t)
envSetVal("ab" "checkStateConfirmModeChange" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

ciwWindow

```
layoutXL ciwWindow 'string "pair_of_coordinates"
```

Description

Specifies the position and size of the command interpreter window (CIW) on the screen. It is used by the autoArrange environment variable when rearranging the Layout XL windows on startup.

By default, the CIW is positioned in the bottom left corner of the screen.

Arguments

pair_of_coordinates

A pair of coordinates enclosed in quotation marks specifying the lower left and upper right corners of the CIW; for example,

```
"((0.0 0.0) (600.0 300.0))"
```

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "ciwWindow")  
envSetVal("layoutXL" "ciwWindow" 'string "((3.0 3.0) (750.0 180.0))")
```

Related Topics

[List of Layout XL Environment Variables](#)

cloningDoExactMatch

```
layoutXL cloningDoExactMatch 'boolean { t | nil }
```

Description

Requires that the parameter values on target components match exactly the parameter values on the source components. When switched off, Layout XL will also suggest targets that have different parameters and values.

Note: If required, you can update the parameters and values when you generate a clone by setting the [cloningAutomaticUpdateLayoutParameters](#) environment variable to `t`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – Clones*

Field: *Exact Parameter Match*

Examples

```
envGetVal("layoutXL" "cloningDoExactMatch")  
envSetVal("layoutXL" "cloningDoExactMatch" 'boolean t)  
envSetVal("layoutXL" "cloningDoExactMatch" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

cloningAutomaticUpdateLayoutParameters

```
layoutXL cloningAutomaticUpdateLayoutParameters 'boolean { t | nil }
```

Description

Automatically updates the parameters and parameter values on the devices in a generated clone to match those on their counterparts in the schematic clone source. Parameters that are set on layout devices but are not present on their schematic counterparts are not removed.

When switched off, the parameters and values are taken from the layout clone source.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – Clones*

Field: *Update Layout Parameters to Match Schematic*

Examples

```
envGetVal("layoutXL" "cloningAutomaticUpdateLayoutParameters")
envSetVal("layoutXL" "cloningAutomaticUpdateLayoutParameters" 'boolean t)
envSetVal("layoutXL" "cloningAutomaticUpdateLayoutParameters" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

colorDraglines

```
layoutXL colorDraglines 'boolean { t | nil }
```

Description

Displays each dragline in a different color. This environment variable is honored by the *Generate Selected From Source*, *Generate Clones*, *Move*, and *Stretch* commands.



Tip

For more information on displaying draglines, see also [hideDraglinesForGlobalNets](#) and [showDraglinesForDistantConns](#).

Arguments

None.

GUI Equivalent

Command: *Options – Layout XL (Display tab)*

Field: *Initial Display – Draglines – Use colors*

Examples

```
envGetVal("layoutXL" "colorDraglines")
envSetVal("layoutXL" "colorDraglines" 'boolean t)
envSetVal("layoutXL" "colorDraglines" 'boolean nil)
```

Additional Information

Displaying many draglines during interactive commands can impair the performance of Layout XL. To mitigate these effects, use the [maxDragFig](#) environment variable to limit the number of figures that can be in a dragset.

To set `maxDragFig` interactively so that it takes effect in the current session, type the following in the CIW, where *win* is the window to which the limit applies.

```
win~>maxDragFig = 1000
```

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

To set a default value for `maxDragFig`, which will be used in all future Layout XL sessions, put the following line in your `.cdsenv` file.

```
graphic maxDragFig int 1000 nil
```

Related Topics

[List of Layout XL Environment Variables](#)

constraintAwareEditing

```
layoutXL constraintAwareEditing 'boolean { t | nil }
```

Description

Controls Constraint-Aware Editing mode, which ensures that the *Copy*, *Move*, *Stretch*, and *Rotate* commands honor the following constraints: *Alignment*, *Fixed*, *Locked*, *Matched Parameters*, *Orientation*, *Matched Orientation*, and *Symmetry*.

Constraint-aware editing is on by default, meaning that Layout XL notifies you of any constraint violations in the design by placing a marker at the appropriate location in the design window. As you edit your design, Layout XL continuously monitors the changes you make and adds or removes violation markers appropriately as specific constraints are violated or satisfied.

Note: Constraint-aware editing is a session-wide setting. It applies to all existing and new windows opened in the current Virtuoso session.

Arguments

None.

GUI Equivalent

Command: *Options – Layout XL (General tab)*

Field: *Constraint-aware editing*

Examples

```
envGetVal("layoutXL" "constraintAwareEditing")
envSetVal("layoutXL" "constraintAwareEditing" 'boolean t)
envSetVal("layoutXL" "constraintAwareEditing" 'boolean nil)
```

Additional Information

- If you use design-rule-driven editing in Enforce mode in conjunction with constraint-aware editing, design rules are enforced not only for the object you are editing, but for all of its constraint partners as well.

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

For information on design-rule-driven editing, see the [*Virtuoso Design Rule Driven Editing User Guide*](#).

- The Layout GXL analog auto placer creates markers for constraint violations even when constraint-aware editing is not enabled. This lets you see the violations created by the placer regardless of the editing mode.

For information on the analog auto placer, see the [*Virtuoso Analog Placement User Guide*](#).

- Constraint-aware editing does not allow edits across pseudo-hierarchy levels. For example, if you have a symmetry constraint with one member inside a group and another member outside the group, the constraint cannot be satisfied.
- For detailed definitions of the supported constraints, see [*Default Constraint Types*](#) in the *Virtuoso Unified Custom Constraints User Guide*.
- For more information on toggling constraint-aware editing mode from the graphical user interface, see [*Enabling and Disabling Constraint-Aware Editing Mode*](#).

Related Topics

[Constraint-Aware Editing](#)

[List of Layout XL Environment Variables](#)

cphPromptConfigOrSchematic

```
layoutXL cphPromptConfigOrSchematic 'boolean { t | nil }
```

Description

Controls whether the user is prompted to open the physical configuration cellview or top cellview when descending into a physical configuration cellview in a schematic cellview.

When set to nil, the “Open Configuration or Top CellView” form will not be displayed. The schematic cellview will be opened in context of the physical configuration cellview.

The default is `t`.

Arguments

None.

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "cphPromptConfigOrSchematic")  
envSetVal("layoutXL" "cphPromptConfigOrSchematic" 'boolean t)  
envSetVal("layoutXL" "cphPromptConfigOrSchematic" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

cphStopLogicalElabAtPhysLeaf

```
layoutXL cphStopLogicalElabAtPhysLeaf 'boolean { t | nil }
```

Description

Stops the elaboration of the logical design hierarchy when a node is reached which maps to a physical leaf node. This lets you use the physical stop view list to limit the logical elaboration for large hierarchical designs that would otherwise take a long time to open or might not open at all in Layout XL.

The default is `t`, which means that the elaboration stops when it reaches a logical node that maps to one of the view names specified in the *Physical stop view list* field. Note that in this case, global nets below the leaf instance in the schematic are not considered during the elaboration. If your design relies on such nets, Cadence recommends that you switch off this environment variable.

When set to `nil`, the logical design is fully elaborated when you launch the *Configure Physical Hierarchy* command.

Note: A change to this environment variable takes effect only after you re-elaborate the design. To force a full re-elaboration, do one of the following.

- Change the *Logical switch view list* in the *Global Bindings* pane in the *Configure Physical Hierarchy* window.
- Close and re-open Layout XL and then choose *Launch –Configure Physical Hierarchy* again.

Arguments

None.

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "cphStopLogicalElabAtPhysLeaf")
envSetVal("layoutXL" "cphStopLogicalElabAtPhysLeaf" 'boolean t)
envSetVal("layoutXL" "cphStopLogicalElabAtPhysLeaf" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

createBoundaryLabel

```
layoutXL createBoundaryLabel 'boolean { t | nil }
```

Description

Specifies whether the *Generate All From Source* command automatically creates a label containing the library, cell, and view name of the design when creating a boundary from the Generate Layout form. The label is created on layer-purpose pair (`prBoundary` `boundary`).

The default is `nil`.

Arguments

None.

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "createBoundaryLabel")
envSetVal("layoutXL" "createBoundaryLabel" 'boolean t)
envSetVal("layoutXL" "createBoundaryLabel" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

createImplicitBusTerminals

```
layoutXL createImplicitBusTerminals 'boolean { t | nil }
```

Description

Creates, checks, and updates implicit bus terminals in the layout cellview, based upon the explicit bus terminals in the schematic cellview. This avoids the need for running VerilogAnnotate to annotate the bus terminal and bus bit order (ascending or descending) information into the layout view. In addition, it enhances the interoperability of layout cellviews with the SOC Encounter designs.

The bus connectivity is created in the layout view as "implicit" objects, so that they are not visible to other parts of Virtuoso. For example the implicit bus terminals generated by VXL do not show up in the list of terminals associated with the layout cellview when accessed through SKILL.

The `createImplicitBusTerminals` environment variable also controls the annotation of bus terminals into layout cellviews generated by the Generate Physical Hierarchy command in Virtuoso GXL".

The default is `nil`.

GUI Equivalent (Only supported by GFS)

Command: *Options – Layout XL (Generation tab)*

Field: *Create implicit bus terminals*

Examples

```
envGetVal("layoutXL" "createImplicitBusTerminals")
envSetVal("layoutXL" "createImplicitBusTerminals" 'boolean t)
envSetVal("layoutXL" "createImplicitBusTerminals" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

[Soft Block Mode](#)

[Check Against Source](#)

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

Generation Tab

crossSelect

```
layoutXL crossSelect 'boolean { t | nil }
```

Description

Specifies whether cross-selection is operational. Cross-selection means that when you select a component in the layout, the corresponding component is selected in the schematic and vice versa.

The default is `t`.

Arguments

None.

GUI Equivalent

Command: *Options – Layout XL (Display tab)*

Field: *Cross-selection*

Examples

```
envGetVal("layoutXL" "crossSelect")  
envSetVal("layoutXL" "crossSelect" 'boolean t)  
envSetVal("layoutXL" "crossSelect" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

deleteConfirmModeChange

```
ab deleteConfirmModeChange 'boolean { t | nil }
```

Description

Controls whether or not you must confirm explicitly that a read-only cellview is to be re-opened in edit mode before deleting a marker in the Annotation Browser assistant.

The default is `t`, which means that when you try to use the *Delete* command (or its browser pane equivalent) in a read-only cellview, the software prompts you to confirm that you want to re-open the cellview in edit mode before performing the requested operation. When set to `nil`, the software automatically re-opens the cellview in edit mode and performs the requested operation.



Tip

The [checkStateConfirmModeChange](#) environment variable performs the same function for the Annotation Browser's *Check/Uncheck* command.

Arguments

None.

GUI Equivalent

None.

Examples

```
envGetVal("ab" "deleteConfirmModeChange")  
envSetVal("ab" "deleteConfirmModeChange" 'boolean t)  
envSetVal("ab" "deleteConfirmModeChange" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

deleteUnmatchedInsts

```
layoutXL deleteUnmatchedInsts 'boolean { t | nil }
```

Description

Automatically deletes layout instances that are no longer present in the schematic during the *Update Components And Nets* command.

The default is `t`.

When set to `nil`, unmatched instances are not deleted when updating components and nets, but are instead indicated with a marker in the layout view.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Update – Components And Nets (Update tab)*

Field: *Delete Unmatched Instances*

None.

Examples

```
envGetVal("layoutXL" "deleteUnmatchedInsts")  
envSetVal("layoutXL" "deleteUnmatchedInsts" 'boolean t)  
envSetVal("layoutXL" "deleteUnmatchedInsts" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

deleteUnmatchedPins

```
layoutXL deleteUnmatchedPins 'boolean { t | nil }
```

Description

Automatically deletes layout pins that are no longer present in the schematic during the *Update Components And Nets* command. Redundant nets and terminals are deleted from the layout view at the same time.

The default is `t`.

When set to `nil`, unmatched pins are not deleted when updating components and nets, but are instead indicated with a marker in the layout view.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Update – Components And Nets (Update tab)*

Field: *Delete Unmatched Pins*

Examples

```
envGetVal("layoutXL" "deleteUnmatchedPins")
envSetVal("layoutXL" "deleteUnmatchedPins" 'boolean t)
envSetVal("layoutXL" "deleteUnmatchedPins" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

deviceExtractType

```
layoutXL deviceExtractType 'cyclic {"pcells" | "all" | "none"}
```

Description

Controls the Virtuoso Space-based Router's connectivity extractor, which ensures that connectivity is set correctly on all the shapes around the connection points in a cell loaded in the router. By default, only pcells are extracted. You can choose to extract all cells, or disable the spaced-based router extractor altogether.

This extraction is required mainly for pcells in device-level routing, where the router will otherwise consider a gate shape to be a blockage, even though the underlying gate pin has connectivity assigned. The extraction happens in the router's internal data structures; the original cell is not touched. You can avoid this step and its associated performance cost by updating your pcells to ensure that all the shapes around connection points have appropriate connectivity defined.

Arguments

pcells	Extracts pcells only.
all	Extracts all cells.
none	Disables the space-based router connectivity extractor.

GUI Equivalent

Command: *Options – Layout XL (Extraction tab)*
Field: *Spaced-based Router – Extract*

Examples

```
envGetVal("layoutXL" "deviceExtractType")  
envSetVal("layoutXL" "deviceExtractType" 'cyclic "pcells")  
envSetVal("layoutXL" "deviceExtractType" 'cyclic "all")  
envSetVal("layoutXL" "deviceExtractType" 'cyclic "none")
```

Related Topics

[Space-based Router Extraction](#)

[List of Layout XL Environment Variables](#)

disableCASOptionsPopUp

```
layoutXL disableCASOptionsPopUp 'boolean { t | nil }
```

Description

Prevents the Check Against Source dialog from being displayed every time you run the *Check Against Source* command.

You can switch this on from the dialog itself by clicking the *Do not show me this dialog again* check box. To see the dialog, unset this environment variable manually in the CIW.

The default is `nil`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Check – Against Source*

Field: *Do not show me this dialog again*

Examples

```
envGetVal("layoutXL" "disableCASOptionsPopUp")  
envSetVal("layoutXL" "disableCASOptionsPopUp" 'boolean t)  
envSetVal("layoutXL" "disableCASOptionsPopUp" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

distanceMFactorExpansion

```
constraint distanceMFactorExpansion 'string { firstToAll | pair }
```

Description

Maps constraints between layout and schematic and controls the transfer of distance constraints.

The default is `firstToAll`.

When schematic objects with constraints are transferred to the layout, it is possible that a single schematic object is mapped into multiple layout objects. Common occurrences of this are the use of MFactor, iterated instances or hierarchy.

In the cases where this occurs, the constraint system will expand the schematic constraint to preserve its intent in the layout. Infact, different constraints will expand in different manners to maintain this intent.

For example, if schematic device A is aligned with schematic device B; the constraint will expand into a single alignment between A.1 A.2 A.3 ... B.1 B.2 B.3 where A Symmetric to B would transfer as A.1 Symmetric B.1, A.2 Symmetric B.2.

Arguments

`firstToAll` *Transfers to the layout as one constraint for each expanded instance of the first member in the constraint. Each new constraint has the same parameters and all instances of all other expanded members.*

`firstToAll` is the default value for the constraint transfer.

By default, the Distance Constraints Transfer as:

```
M1 dist M2
```

transfers into

```
M1.1 dist M2.1 M2.2 ... M2.N
```

```
M1.2 dist M2.1 M2.2 ... M2.N
```

```
.
```

```
.
```

```
.
```

```
M1.P dist M2.1 M2.2 ... M2.N
```

`pair` *Requires the distance constraint to have only two members in the schematic where each member expands into the same number of layout objects.*

The schematic constraint transfers to the layout as a new constraint for each pair of expanded instances.

Example:

```
M1 dist M2
```

transfers into

```
M1.1 dist M2.1
```

```
M1.2 dist M2.2
```

```
M1.3 dist M2.3
```

```
.
```

```
.
```

```
M1.P dist M2.P
```

GUI Equivalent

None

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

Examples

```
envGetVal("constraint" "distanceMFactorExpansion")  
envSetVal("constraint" "distanceMFactorExpansion" 'string firstToAll)  
envSetVal("constraint" "distanceMFactorExpansion" 'string pair)
```

Related Topics

[List of Layout XL Environment Variables](#)

drdUseNetName

```
layoutXL drdUseNetName 'boolean { t | nil }
```

Description

Controls whether an object that is being moved retains the net name of the pin or instance terminal to which it was connected prior to the move. This in turn lets the design-rule-driven (DRD) editing functionality use the name to flag short violations during the move operation.

For example, when using DRD editing, if you move an object on net A towards an object on net B, the system displays a message warning you about the short violation. If the move leads to the object becoming disconnected from an I/O pin or instance terminal, the `drdUseNetName` environment variable controls the behavior as follows.

- When set to `t`, the object retains the net name of the pin or instance terminal to which it was connected prior to the move and DRD editing uses this name to flag short violations.
- When set to `nil`, the object does not retain the net name, Layout XL is unaware of any connectivity associated with the object, and therefore no short message is displayed.

The original net assignment remains intact in both cases. The default value is `nil`.

Arguments

None.

GUI Equivalent

None.

Examples

```
envGetVal ("layoutXL" "drdUseNetName")
envSetVal ("layoutXL" "drdUseNetName" 'boolean t)
envSetVal ("layoutXL" "drdUseNetName" 'boolean nil)
```

Additional Information

- The `lxStickyNet` property overrides any setting of `drdUseNetName`.
- For information on design-rule-driven editing, see the [*Design Rule Driven Editing User Guide*](#).

Important

`drdUseNetName` has no effect when you stretch an object because assigned connectivity is always maintained during the stretch operation. However, if all the points of the object are selected prior to stretching, the stretch is in fact a move and the behavior is as described above.

Related Topics

[List of Layout XL Environment Variables](#)

extractShowMustConnectMarkers

This environment variable has been replaced in the current release by [extractVerifyMustConnectMarkers](#).

Related Topics

[List of Layout XL Environment Variables](#)

extractShowUnimplementedInstTerms

This environment variable has been replaced in the current release by [extractVerifyUnimplementedInstTerms](#).

Related Topics

[List of Layout XL Environment Variables](#)

extractShowWeakMarkers

This environment variable has been replaced in the current release by [extractVerifyWeakMarkers](#).

Related Topics

[List of Layout XL Environment Variables](#)

flightLineEnable

```
layoutXL flightLineEnable 'boolean { t | nil }
```

Description

Controls whether draglines are displayed during the *Move*, *Stretch*, *Generate Selected From Source*, and *Generate Clone* commands. The draglines connect the pins of the object you are moving to pins of the nearest objects.

The default is `t`.



Tip

For information on how to change the display of the draglines, see [colorDraglines](#), [hideDraglinesForGlobalNets](#), and [showDraglinesForDistantConns](#).

Arguments

None.

GUI Equivalents

Command: *Edit – Basic – Move*
 Edit – Basic – Stretch

Field: *Display Draglines*

Command: *Connectivity – Generate – Selected From Source*
 Connectivity – Generate – Clones

Field: *Draglines*

Examples

```
envGetVal("layoutXL" "flightLineEnable")
envSetVal("layoutXL" "flightLineEnable" 'boolean t)
envSetVal("layoutXL" "flightLineEnable" 'boolean nil)
```

Additional Information

Displaying many draglines during interactive commands can impair the performance of Layout XL. To mitigate these effects, use the `maxDragFig` environment variable to limit the number of figures that can be in a dragset.

To set `maxDragFig` interactively so that it takes effect in the current session, type the following in the CIW, where `win` is the window to which the limit applies.

```
win~>maxDragFig = 1000
```

To set a default value for `maxDragFig`, which will be used in all future Layout XL sessions, put the following line in your `.cdsenv` file.

```
graphic maxDragFig int 1000 nil
```

Related Topics

[List of Layout XL Environment Variables](#)

foldOptimizeSupplyAndGroundNets

```
layoutXL foldOptimizeSupplyAndGroundNets 'boolean { t | nil }
```

Description

Updates IFold to optimise any supply and ground nets to the outside of the folded chain when an even number of folds exists.

The default is `t`.



Optimizing the supply and ground nets to the outside of the folded chain can impact the original orientation and the left net. If the priority is to maintain these, the environment variable must be set to `nil`.

Arguments

None.

GUI Equivalents

None.

Examples

```
envGetVal ("layoutXL" "foldOptimizeSupplyAndGroundNets")
envSetVal ("layoutXL" "foldOptimizeSupplyAndGroundNets" 'boolean t)
envSetVal ("layoutXL" "foldOptimizeSupplyAndGroundNets" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

hideDraglinesForGlobalNets

```
layoutXL hideDraglinesForGlobalNets 'boolean { t | nil }
```

Description

Hides draglines for global nets and power and ground nets specified using the [lxGroundNetNames](#) and [lxSupplyNetNames](#) environment variables.

This environment variable is honored by the *Generate Selected From Source*, *Generate Clones*, *Move*, and *Stretch* commands.



Tip

For more information on displaying draglines, see also [colorDraglines](#) and [showDraglinesForDistantConns](#).

Arguments

None.

GUI Equivalent

Command: *Options – Layout XL (Display tab)*

Field: *Initial Display – Draglines – Hide global nets*

Examples

```
envGetVal("layoutXL" "hideDraglinesForGlobalNets")
envSetVal("layoutXL" "hideDraglinesForGlobalNets" 'boolean t)
envSetVal("layoutXL" "hideDraglinesForGlobalNets" 'boolean nil)
```

Additional Information

Displaying many draglines during interactive commands can impair the performance of Layout XL. To mitigate these effects, use the [maxDragFig](#) environment variable to limit the number of figures that can be in a dragset.

To set `maxDragFig` interactively so that it takes effect in the current session, type the following in the CIW, where *win* is the window to which the limit applies.

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

```
win~>maxDragFig = 1000
```

To set a default value for `maxDragFig`, which will be used in all future Layout XL sessions, put the following line in your `.cdsenv` file.

```
graphic maxDragFig int 1000 nil
```

Related Topics

[List of Layout XL Environment Variables](#)

hideMarkersWhenBrowserHidden

```
ab hideMarkersWhenBrowserHidden 'boolean { t | nil }
```

Description

Hides any highlighted markers in the canvas when you close the Annotation Browser assistant.

The default is `nil`, which means the markers remain highlighted in the canvas when you close the browser.

Arguments

None.

GUI Equivalent

Command: *Windows – Assistants – Annotation Browser – Options*
 (RMB)

Field: *Remove highlighting when browser is closed*

Examples

```
envGetVal("ab" "hideMarkersWhenBrowserHidden")  
envSetVal("ab" "hideMarkersWhenBrowserHidden" 'boolean t)  
envSetVal("ab" "hideMarkersWhenBrowserHidden" 'boolean nil)
```

Related Topics

[Annotation Browser](#)

[List of Layout XL Environment Variables](#)

hierarchyDepth

```
ab hierarchyDepth 'int { 0 | 1 | 2 ... 32 }
```

Description

Specifies the number of levels of hierarchy searched for markers. The value must lie in the range from 0 through 32

The default is 3.

Arguments

None.

GUI Equivalent

Command: *Windows – Assistants – Annotation Browser – Options (RMB)*

Field: *Hierarchy depth*

Examples

```
envGetVal("ab" "hierarchyDepth")  
envSetVal("ab" "hierarchyDepth" 'int 3)
```

Related Topics

[Annotation Browser](#)

[List of Layout XL Environment Variables](#)

highlightedIsSelectable

```
ab highlightedIsSelectable 'boolean {t | nil}
```

Description

Specifies that markers that are highlighted in the canvas can be selected even if marker selection has been switched off in the LSW.

The default is `t`.

Arguments

None.

GUI Equivalent

Command: *Windows – Assistants – Annotation Browser*

Field: *Make Highlighted Markers Selectable* (toolbar button)

Examples

```
envGetVal("ab" "highlightedIsSelectable")  
envSetVal("ab" "highlightedIsSelectable" 'boolean t)  
envSetVal("ab" "highlightedIsSelectable" 'boolean nil)
```

Related Topics

[Annotation Browser](#)

[List of Layout XL Environment Variables](#)

incNetHiliteLayer

```
layoutXL incNetHiliteLayer 'string { "y1 drawing" | ... | "y9 drawing" }
```

Description

Specifies which layer-purpose pair is used to display incomplete nets.

The default is "y0 drawing".

Note: If you do not specify a purpose, the software assumes the purpose to be `drawing`.

Arguments

None.

GUI Equivalent

Command	<i>Windows – Assistants – Annotation Browser</i>
Field	<i>Cycle</i>

Examples

```
envGetVal("layoutXL" "incNetHiliteLayer")  
envSetVal("layoutXL" "incNetHiliteLayer" 'string "y0 drawing")
```

Related Topics

[List of Layout XL Environment Variables](#)

inductanceParamNames

```
layoutXL inductanceParamNames 'string "list_of_parameters"
```

Description

Lists the names of the schematic parameters used to specify inductance.

The default is "l L".

The inductance value must be split among the generated series-connected devices during the following *Connectivity* commands.

- *Generate All From Source*
- *Check Against Source*
- *Update Layout Parameters.*

Layout XL checks each schematic instance for one of the listed parameter names and updates the inductance values of the matching layout parameters accordingly.

Arguments

list_of_parameters

A list of parameter names each separated by a space. The list must be enclosed in quotation marks; for example,

"l L"

GUI Equivalent

Command: *Options – Layout XL (Parameters tab)s*

Field: *Inductance*

Examples

```
envGetVal("layoutXL" "inductanceParamNames")  
envSetVal("layoutXL" "inductanceParamNames" 'string "l L")
```

Related Topics

List of Layout XL Environment Variables

infoWindow

```
layoutXL infoWindow 'boolean { t | nil }
```

Description

Controls where messages issued by the *Check Against Source*, *Update Layout Parameters*, and *Update Schematic Parameters* commands are displayed. When set to `t`, messages are displayed in a separate Info window. When set to `nil`, messages are sent to the CIW.

The default is `t`.

Arguments

None.

GUI Equivalent

Command: *Options – Layout XL (Display tab)*

Field: *Redirect Layout XL messages to separate Info window*

Examples

```
envGetVal("layoutXL" "infoWindow")  
envSetVal("layoutXL" "infoWindow" 'boolean t)  
envSetVal("layoutXL" "infoWindow" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

infoWindowPos

```
layoutXL infoWindowPos 'string "pair_of_coordinates"
```

Description

Specifies the position and size of the information window on the screen. It is used by the autoArrange environment variable when rearranging the Layout XL windows on startup.

By default, the information window is positioned at the top left corner of the screen.

Arguments

`pair_of_coordinates` A pair of coordinates enclosed in quotation marks specifying the lower left and upper right corners of the information window

For example, `"((0.0 0.0) (600.0 300.0))"`

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "infoWindowPos")  
envSetVal("layoutXL" "infoWindowPos" 'string "((1.0 4.0) (480.0 275.0))")
```

Related Topics

[List of Layout XL Environment Variables](#)

initAspectRatio

```
layoutXL initAspectRatio 'float floating_point_number
```

Description

Specifies the width-to-height aspect ratio of the design to be generated or updated by the *Generate All From Source* and *Update Components And Nets* commands, or when defining soft blocks in the Configure Physical Hierarchy window.

The default is 1.0, which specifies a square boundary. An aspect ratio of 0.5 specifies a boundary twice as high as it is wide. A value of 2 specifies a boundary twice as wide as it is high.

Arguments

floating_point_number

A float specifying the width-to-height ratio for the design boundary; for example,

1.0

GUI Equivalent

Command: *Connectivity – Generate – All From Source (PR Boundary tab)*

Connectivity – Update – Components And Nets

Launch – Configure Physical Hierarchy (in Soft Block mode)

Field: *Aspect Ratio (W/H)*

Examples

```
envGetVal("layoutXL" "initAspectRatio")  
envSetVal("layoutXL" "initAspectRatio" 'float 1.0)
```

Related Topics

[List of Layout XL Environment Variables](#)

initAspectRatioOption

```
layoutXL initAspectRatioOption 'string { "Aspect Ratio W/H" | "Boundary Width"  
    | "Boundary Height" }
```

Description

Specifies which boundary option is used by the [Generate Layout](#) and [Update Components and Nets](#) forms, or when defining soft blocks in the [Configure Physical Hierarchy](#) window.

The default is "Aspect Ratio W/H".

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – All From Source (PR Boundary tab)*
 Connectivity – Update – Components And Nets (PR Boundary tab)
 Launch – Configure Physical Hierarchy (in Soft Block mode)

Field: *Aspect Ratio (W/H)*

Examples

```
envGetVal("layoutXL" "initAspectRatioOption")  
envSetVal("layoutXL" "initAspectRatioOption" 'string "Aspect Ratio W/H")  
envSetVal("layoutXL" "initAspectRatioOption" 'string "Boundary Width")  
envSetVal("layoutXL" "initAspectRatioOption" 'string "Boundary Height")
```

Related Topics

[List of Layout XL Environment Variables](#)

initCreateBoundary

```
layoutXL initCreateBoundary 'boolean { t | nil }
```

Description

Controls whether Layout XL creates a place and route boundary in the layout canvas.

The default is `nil`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – All From Source (Generate tab)*

Field: *PR Boundary*

Examples

```
envGetVal("layoutXL" "initCreateBoundary")  
envSetVal("layoutXL" "initCreateBoundary" 'boolean t)  
envSetVal("layoutXL" "initCreateBoundary" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

initCreateInstances

```
layoutXL initCreateInstances 'boolean { t | nil }
```

Description

Specifies that Layout XL layout editor is to create instances during the *Generate All From Source* command.

The default is *t*.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – All From Source (Generate tab)*

Field: *Instances*

Examples

```
envGetVal("layoutXL" "initCreateInstances")  
envSetVal("layoutXL" "initCreateInstances" 'boolean t)  
envSetVal("layoutXL" "initCreateInstances" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

initCreateMTM

```
layoutXL initCreateMTM 'boolean { t | nil }
```

Description

Specifies that Layout XL is to preserve user-defined mappings during *Generate All From Source* and *Update Components And Nets*.

The default is `nil`.



This option preserves only user-defined one-to-one, many-to-many, many-to-one, and one-to-many device correspondence defined in the Define Device Correspondence form. It does not report missing devices or shapes within a mapped group. For more information, see Updating Device Correspondence.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – All From Source (Generate tab)*
 Connectivity – Update – Components And Nets (Update tab)

Field: *Preserve User-Defined Mappings*

Examples

```
envGetVal("layoutXL" "initCreateMTM")  
envSetVal("layoutXL" "initCreateMTM" 'boolean t)  
envSetVal("layoutXL" "initCreateMTM" 'boolean nil)
```

Additional Information

The Generate Layout form uses this environment variable to set the value for its *Preserve User-Defined Mappings* option. If you change the setting for *Preserve user-Defined Mappings* in the form, the value of the environment variable changes as well. Conversely, the

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

Update Components and Nets form always initializes with *Preserve User-Defined Mappings* switched on and never sets the value of `initCreateMTM`.

Related Topics

List of Layout XL Environment Variables

initCreatePadPins

```
layoutXL initCreatePadPins 'boolean { t | nil }
```

Description

Specifies that Layout XL generates both pins and pads for schematic pins that are connected to I/O pads (cells of type `pad`, `padSpacer`, or `padAreaIO`).

The default is `t`.



Tip

To generate only pads but no pins, set this environment variable to `nil`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – All From Source (I/O Pins tab)*
 Connectivity – Update – Components And Nets (I/O Pins tab)

Field: *Except Pad Pins*

Examples

```
envGetVal("layoutXL" "initCreatePadPins")  
envSetVal("layoutXL" "initCreatePadPins" 'boolean t)  
envSetVal("layoutXL" "initCreatePadPins" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

initCreatePins

```
layoutXL initCreatePins 'boolean { t | nil }
```

Description

Specifies that Layout XL is to create pins.

The default is `t`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – All From Source (Generate tab)*

Field: *I/O Pins*

Examples

```
envGetVal("layoutXL" "initCreatePins")  
envSetVal("layoutXL" "initCreatePins" 'boolean t)  
envSetVal("layoutXL" "initCreatePins" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

initCreateSnapBoundary

```
layoutXL initCreateSnapBoundary 'boolean { t | nil }
```

Description

Controls whether Layout XL creates a rectangular snap boundary enclosing the generated PR boundary. (You can generate a snap boundary only if you also generate a PR boundary.)

The default is `nil`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – All From Source (Generate tab)*

Field: *Snap Boundary*

Examples

```
envGetVal("layoutXL" "initCreateSnapBoundary")  
envSetVal("layoutXL" "initCreateSnapBoundary" 'boolean t)  
envSetVal("layoutXL" "initCreateSnapBoundary" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

initDoFolding

```
layoutXL initDoFolding 'boolean { t | nil }
```

Description

Specifies that Layout XL is to divide prepared transistors into two or more folds.

The default is `nil`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – All From Source (Generate tab)*

Field: *Folding*

Examples

```
envGetVal("layoutXL" "initDoFolding")
envSetVal("layoutXL" "initDoFolding" 'boolean t)
envSetVal("layoutXL" "initDoFolding" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

initDoStacking

```
layoutXL initDoStacking 'boolean { t | nil }
```

Description

Specifies that Layout XL is to abut MOS transistors in a specific order.

The default is `nil`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – All From Source (Generate tab)*

Field: *Chaining*

Examples

```
envGetVal("layoutXL" "initDoStacking")  
envSetVal("layoutXL" "initDoStacking" 'boolean t)  
envSetVal("layoutXL" "initDoStacking" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

initEstimateArea

```
layoutXL initEstimateArea 'boolean { t | nil }
```

Description

Determines whether Layout XL estimates the size of the design boundary based on the boundary area estimation options during the *Generate All From Source* and *Update Components And Nets* commands, and when defining soft blocks in the Configure Physical Hierarchy window. To support defining soft blocks in the Configure Physical Hierarchy window, the environment variable feeds the *Boundary* tab with the Area Estimator values.

The default is `t`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – All From Source (PR Boundary tab)*
 Connectivity – Update – Components And Nets (PR Boundary tab)
 Launch – Configure Physical Hierarchy (in Soft Block mode)

Field: *Aspect Ratio (W/H)*

Examples

```
envGetVal("layoutXL" "initEstimateArea")  
envSetVal("layoutXL" "initEstimateArea" 'boolean t)  
envSetVal("layoutXL" "initEstimateArea" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

initGlobalNetPins

```
layoutXL initGlobalNetPins 'boolean { t | nil }
```

Description

Determines whether the Layout XL layout editor *Generate All From Source* command creates layout pins for the global nets in the schematic.

The default is `t`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – All From Source* (Generate tab)
 Connectivity – Update – Components And Nets (Update tab)

Field: *Create Label As*

Examples

```
envGetVal("layoutXL" "initGlobalNetPins")  
envSetVal("layoutXL" "initGlobalNetPins" 'boolean t)  
envSetVal("layoutXL" "initGlobalNetPins" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

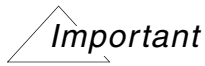
initIOLabelType

```
layoutXL initIOLabelType 'string { "Label" | "Text Display" | "None" }
```

Description

Specifies the type of label that Layout XL generates for I/O pins. When set to `None`, no pin label is created. This setting is honored by the *Generate All From Source*, *Generated Selected From Source*, and *Update Components And Nets* commands.

The default is "Label".



For this environment variable to take effect, you must also set the Layout L environment variable createPinLabel to `t`.

Arguments

Label	Creates pin labels on the layer specified by <u>initIOPinLayer</u> .
Text Display	Creates text displays on the layer specified by <u>initIOPinLayer</u> .
None	No pin labels or text displays are created.

GUI Equivalent

Command:	<i>Connectivity – Generate – All From Source (I/O Pins tab)</i> <i>Connectivity – Generate – Selected From Source</i> <i>Connectivity – Update – Components And Nets (I/O Pins tab)</i>
Field:	<i>Create Label As</i>

Examples

```
envGetVal("layoutXL" "initIOLabelType")
envSetVal("layoutXL" "initIOLabelType" 'string "Label")
envSetVal("layoutXL" "initIOLabelType" 'string "Text Display")
envSetVal("layoutXL" "initIOLabelType" 'string "None")
```

Related Topics

List of Layout XL Environment Variables

initIOPinLayer

```
layoutXL initIOPinLayer 'string "layer_purpose_pair"
```

Description

Specifies which of the layers and purposes defined in the technology file is used as the pin layer-purpose pair when using the *Generate All From Source* and *Update Components And Nets* commands to generate and update pins, or when defining soft block pins using the Configure Physical Hierarchy window in Soft Block mode.

The default value is the current entry layer and purpose from the LSW. For CPH soft block mode, the default value of the pin layers on the I/O Pins tab is read from the environment variable, `initIOPinLayer`.

Arguments

layer_purpose_pair

Specifies the layer and purpose on which pins are created. The string must be enclosed in quotation marks; for example,

`"metall drawing"`

GUI Equivalent

Command: *Connectivity – Generate – All From Source (I/O Pins tab)*
 Connectivity – Update – Components And Nets (I/O Pins tab)
 Launch – Configure Physical Hierarchy (in Soft Block mode)

Field: *Layer*

Examples

```
envGetVal("layoutXL" "initIOPinLayer")  
envSetVal("layoutXL" "initIOPinLayer" 'string "Metall drawing")
```

Related Topics

[List of Layout XL Environment Variables](#)

initIOPinName

The `initIOPinName` environment variable is not required in OpenAccess designs and is no longer supported. You can remove references to it from your setup files.

Related Topics

[List of Layout XL Environment Variables](#)

initPinHeight

layoutXL initPinHeight 'float *floating_point_number*

Description

Specifies the height of pins generated by the *Generate All From Source*, *Generated Selected From Source*, and *Update Components And Nets* commands, and when defining soft blocks in the Configure Physical Hierarchy (CPH) window. For CPH soft block, the environment variable feeds the I/O Pins tab with the default pin height values.

Note: This value is honored only if it is greater than the minimum width value obtained from the constraint lookup.

Arguments

floating_point_number

A float specifying the default height of pins generated by Layout XL; for example,

0.25

GUI Equivalent

Command: *Connectivity – Generate – All From Source (I/O Pins tab)*
 Connectivity – Generate – Selected From Source
 Connectivity – Update – Components And Nets (I/O Pins tab)
 Launch – Configure Physical Hierarchy (in Soft Block mode)

Field: *Height*

Examples

```
envGetVal("layoutXL" "initPinHeight")  
envSetVal("layoutXL" "initPinHeight" 'float 1.0)
```

Related Topics

List of Layout XL Environment Variables

initPinMultiplicity

```
layoutXL initPinMultiplicity 'int integer
```

Description

Specifies the number of pins of each type to be generated by the *Generate All From Source* and *Update Components And Nets* commands, and when defining soft blocks in the Configure Physical Hierarchy window. For CPH soft block, the environment variable feeds the I/O Pins tab with the default number of pins of each type to be generated.

The default is 1.

Arguments

<i>integer</i>	Non-zero positive integer specifying a number of pins to be generated; for example, 1

GUI Equivalent

Command:	<i>Connectivity – Generate – All From Source (I/O Pins tab)</i> <i>Connectivity – Update – Components And Nets (I/O Pins tab)</i> <i>Launch – Configure Physical Hierarchy (in Soft Block mode)</i>
Field:	<i>Num, Number</i>

Examples

```
envGetVal("layoutXL" "initPinMultiplicity")  
envSetVal("layoutXL" "initPinMultiplicity" 'int 2)
```

Related Topics

[List of Layout XL Environment Variables](#)

initPinWidth

```
layoutXL initPinWidth 'float floating_point_number
```

Description

Specifies the width of pins generated by the *Generate All From Source*, *Generated Selected From Source*, and *Update Components And Nets* commands, and when defining soft blocks in the Configure Physical Hierarchy window. For CPH soft block, the environment variable feeds the I/O Pins tab with the default pin width values.

Note: This value is honored only if it is greater than the minimum width value obtained from the constraint lookup.

Arguments

floating_point_number

A float specifying the default width of pins generated by Layout XL; for example,

0.25

GUI Equivalent

Command: *Connectivity – Generate – All From Source (I/O Pins tab)*
 Connectivity – Generate – Selected From Source
 Connectivity – Update – Components And Nets (I/O Pins tab)
 Launch – Configure Physical Hierarchy (in Soft Block mode)

Field: *Width*

Examples

```
envGetVal("layoutXL" "initPinWidth")  
envSetVal("layoutXL" "initPinWidth" 'float 1.0)
```

Related Topics

List of Layout XL Environment Variables

initPrBoundaryH

layoutXL initPrBoundaryH 'float *floating_point_number*

Description

Specifies the height of the design boundary created by the *Generate All From Source* and *Update Components And Nets* commands, and when defining soft blocks in the Configure Physical Hierarchy window. For CPH soft block mode, the environment variable feeds the Boundary tab with the default design boundary height values.

Arguments

floating_point_number

A float specifying the default height of the design boundary generated by Layout XL; for example,

8.0

GUI Equivalent

Command: *Connectivity – Generate – All From Source (PR Boundary tab)*
 Connectivity – Update – Components And Nets (PR Boundary tab)
 Launch – Configure Physical Hierarchy (in Soft Block mode)

Field: *Height*

Examples

```
envGetVal("layoutXL" "initPrBoundaryH")  
envSetVal("layoutXL" "initPrBoundaryH" 'float 471.31)
```

Related Topics

[List of Layout XL Environment Variables](#)

initPrBoundaryW

```
layoutXL initPrBoundaryW 'float floating_point_number
```

Description

Specifies the width of the design boundary created by the *Generate All From Source* and *Update Components And Nets* commands, and when defining soft blocks in the Configure Physical Hierarchy (CPH) window. For CPH soft block mode, the environment variable feeds the Boundary tab with the default design boundary width values.

Arguments

floating_point_number

A float specifying the default width of the design boundary generated by Layout XL; for example,

8.0

GUI Equivalent

Command: *Connectivity – Generate – All From Source (PR Boundary tab)*
 Connectivity – Update – Components And Nets (PR Boundary tab)
 Launch – Configure Physical Hierarchy (in Soft Block mode)

Field: *Width*

Examples

```
envGetVal("layoutXL" "initPrBoundaryW")  
envSetVal("layoutXL" "initPrBoundaryW" 'float 186.075)
```

Related Topics

[List of Layout XL Environment Variables](#)

initUtilization

```
layoutXL initUtilization 'float floating_point_number
```

Description

Specifies the percentage area utilization to be used by the *Estimate Area* function in the *Generate All From Source* and *Update Components And Nets* commands, and when defining soft blocks in the Configure Physical Hierarchy window. For CPH soft block mode, the environment variable feeds the Boundary tab with the utilization percentage of the design area.

The default is 25.0.

Arguments

floating_point_number

A float specifying the default area utilization for the design; for example,

25.0

GUI Equivalent

Command: *Connectivity – Generate – All From Source (PR Boundary tab)*
 Connectivity – Update – Components And Nets (PR Boundary tab)
 Launch – Configure Physical Hierarchy (in Soft Block mode)

Field: *Utilization (%)*

Examples

```
envGetVal("layoutXL" "initUtilization")  
envSetVal("layoutXL" "initUtilization" 'float 25.2)
```

Related Topics

[List of Layout XL Environment Variables](#)

labelOrient

```
layoutXL labelOrient 'cyclic { "R0" | "R90" | "R180" | "270" | "MY" | "MYR90" |  
    "MX" | "MXR90" }
```

Description

Controls the rotation of a pin label during generation.

The default is R90.

Arguments

R0

No rotation.

R90

Rotates pin labels 90 degrees clockwise.

R180

Rotates pin labels 180 degrees clockwise.

R270

Rotates pin labels 270 degrees clockwise.

MY

Mirrors pin labels over the Y axis.

MYR90

Mirrors pin labels over the Y axis and rotates 90 degrees clockwise.

MX

Mirrors pin labels over the X axis.

MXR90

Mirrors pin labels over the X axis and rotates 90 degrees clockwise.

GUI Equivalent

Command: *Connectivity – Generate – All From Source (I/O Pins tab)*
 Connectivity – Generate – Selected From Source
 Connectivity – Update – Components And Nets (I/O Pins tab)
 Launch – Configure Physical Hierarchy (in Soft Block mode)

Field: *Orientation*

Examples

```
envGetVal("layoutXL" "labelOrient")  
envSetVal("layoutXL" "labelOrient" 'cyclic "R0")  
envSetVal("layoutXL" "labelOrient" 'cyclic "MY")  
envSetVal("layoutXL" "labelOrient" 'cyclic "MXR90")
```

Related Topics

[List of Layout XL Environment Variables](#)

layoutWindow

layoutXL layoutWindow 'string "*pair_of_coordinates*"

Description

Specifies the position and size of the layout window on the screen. It is used by the autoArrange environment variable when rearranging the Layout XL windows on startup.

By default, the layout window is positioned in the bottom left corner of the screen.

Arguments

pair_of_coordinates

A pair of coordinates enclosed in quotation marks specifying the lower left and upper right corners of the layout window; for example,

"((577.0 89.0) (1268.0 992.0))"

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "layoutWindow")
```

```
envSetVal("layoutXL" "layoutWindow" 'string "((435.0 52.0) (1588.0 1165.0))")
```

Related Topics

List of Layout XL Environment Variables

lswWindow

```
layoutXL lswWindow 'string "pair_of_coordinates"
```

Description

Specifies the position and size of the layer selection window (LSW) on the screen. It is used by the autoArrange environment variable when rearranging the Layout XL windows on startup.

Arguments

pair_of_coordinates

A pair of coordinates enclosed in quotation marks specifying the lower left and upper right corners of the layout window; for example,

```
"((577.0 89.0) (1268.0 992.0))"
```

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "lswWindow")  
envSetVal("layoutXL" "lswWindow" 'string "((0 0) (0 0))")
```

Related Topics

[List of Layout XL Environment Variables](#)

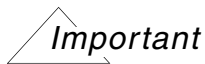
lxAllowPseudoParallelNets

```
layoutXL lxAllowPseudoParallelNets 'boolean { t | nil }
```

Description

Automatically identifies nodes that qualify as pseudoparallel connections and defines them as such during chaining and abutment. A pseudoparallel net connects nodes that are always the same voltage, so the current does not pass through the net. Contacts are dropped automatically if it is appropriate to do so, even if preserveTerminalContacts is switched on.

The default is `nil`.



This environment variable applies only when chaining is turned on. It does not control the automatic identification and creation of pseudoparallel nets during manual abutment. To do that, use the setPPConn environment variable.

Arguments

None.

GUI Equivalent

Command: *Options – Layout XL (Generation tab)*

Field: *Create interdigitated chains*

Command: *Connectivity – Generate Selected Components*

Field: *Interdigitate Chains*

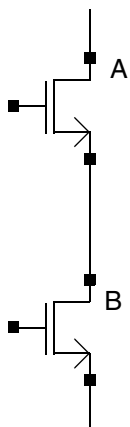
Examples

```
envGetVal("layoutXL" "lxAllowPseudoParallelNets")
envSetVal("layoutXL" "lxAllowPseudoParallelNets" 'boolean t)
envSetVal("layoutXL" "lxAllowPseudoParallelNets" 'boolean nil)
```

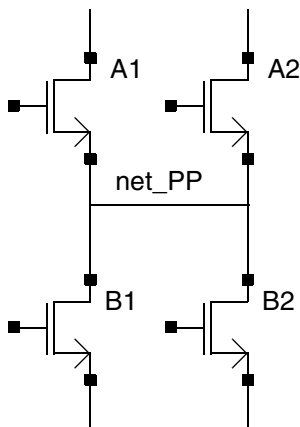

Additional Information

To see the significance of pseudoparallel nets, consider a symmetric series of N-transistors, A and B, tied in a series pull-down chain to ground, as shown in the following figure. Both A and B are folded into two legs, with A1, net_PP, and B1 connected in series to ground, and likewise A2 and B2 connected in series to ground, with both pairs in parallel. The nodes between A1 and B1 and between A2 and B2 are electrically equivalent. In reality no current passes through that connection and it need not physically be made.

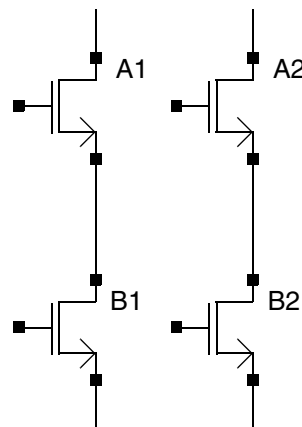
Original circuit



Folded circuit with pseudoparallel connection



Equivalent folded circuit with pseudoparallel net removed



The Assura™ physical verification tool understands pseudoparallel connections and does not report incomplete nets in such situations.

For more information, see [Pseudoparallel Connections](#).

Related Topics

[List of Layout XL Environment Variables](#)

lxChainAlignNMOS

```
layoutXL lxChainAlignNMOS 'string { "Top" | "Center" | "Bottom" }
```

Description

Controls the alignment of NMOS chains to default to bottom.

Arguments

Top

The NMOS chain alignment is set to “Top”.

Center

The NMOS chain alignment is set to “Center”.

Bottom

The NMOS chain alignment is set to “Bottom”.

GUI Equivalent

Command: *Options – Layout XL (Generation tab)*

Field: *Align NMOS*

Command: *Connectivity – Generate Chained Devices*

Field: *Align NMOS*

Command: *Connectivity – Generate Folded Devices*

Field: *Align*

Examples

```
envGetVal("layoutXL" "lxChainAlignNMOS")
envSetVal("layoutXL" "lxChainAlignNMOS" 'string "Top")
envSetVal("layoutXL" "lxChainAlignNMOS" 'string "Center")
envSetVal("layoutXL" "lxChainAlignNMOS" 'string "Bottom")
```

Related Topics

List of Layout XL Environment Variables

lxChainAlignPMOS

```
layoutXL lxChainAlignPMOS 'string { "Top" | "Center" | "Bottom" }
```

Description

Controls the alignment of PMOS chains to default to top.

Arguments

Top

The PMOS chain alignment is set to “Top”.

Center

The PMOS chain alignment is set to “Center”.

Bottom

The PMOS chain alignment is set to “Bottom”.

GUI Equivalent

Command: *Options – Layout XL (Generation tab)*

Field: *Align PMOS*

Command: *Connectivity – Generate Chained Devices*

Field: *Align PMOS*

Command: *Connectivity – Generate Folded Devices*

Field: *Align*

Examples

```
envGetVal("layoutXL" "lxChainAlignPMOS")
envSetVal("layoutXL" "lxChainAlignPMOS" 'string "Top")
envSetVal("layoutXL" "lxChainAlignPMOS" 'string "Center")
envSetVal("layoutXL" "lxChainAlignPMOS" 'string "Bottom")
```

Related Topics

[List of Layout XL Environment Variables](#)

lxDeltaWidth

layoutXL lxDeltaWidth 'float *floating_point_number*

Description

Determines the effective width of folded transistors.

The effective total width is defined as

$$W_{eff} = M (W + lxDeltaWidth)$$

Where M is the number of folds and W is the identical width of each transistor.

Arguments

floating_point_number

A float specifying the effective width of folded transistors; for example,

1.0

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "lxDeltaWidth")  
envSetVal("layoutXL" "lxDeltaWidth" 'float 1.0)
```

Related Topics

[List of Layout XL Environment Variables](#)

lxEvalCDFCallbacks

```
layoutXL lxEvalCDFCallbacks 'boolean { t | nil }
```

Description

Causes all SKILL callbacks defined on CDF parameters to be evaluated by default when you run the following Layout XL commands. Note that callbacks defined on buttons are never evaluated.

- *Generate All From Source*
- *Generate Folded Devices*
- *Update Components And Nets*
- *Update Layout Parameters*
- *Update Schematic Parameters*

The default is `nil`.

The callbacks are evaluated in the order in which they are listed in the CDF, so it is important that you set the appropriate callback to the appropriate place in the list. For more information, see [Defining Parameters](#) in the *Component Description Format User Guide*.



Enabling this environment variable can adversely impact performance, especially on large designs with many callbacks.

Arguments

None.

GUI Equivalent

Command: *Options – Layout XL (Parameters tab)*

Field: *Evaluate CDF callbacks during generation and update*

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

Examples

```
envGetVal("layoutXL" "lxEvalCDFCallbacks")  
envSetVal("layoutXL" "lxEvalCDFCallbacks" 'boolean t)  
envSetVal("layoutXL" "lxEvalCDFCallbacks" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

lxFingeringNames

```
layoutXL lxFingeringNames 'string "list_of_param_names"
```

Description

Specifies parameter names that define pcell gate fingering.

Folding and instance fingers are incompatible and this environment variable is used to filter out those instances when folding. If you do not set the names correctly, it could result in abutment failures for even fingered devices and folding.

The default is "fingers finger numFingers numFinger".

Note: This environment variable is also used by the analog placer when adding dummy devices to MOSFETs or resistors with the modgenMakeMinDummies environment variable set to `t`.

Arguments

list_of_param_names

List of parameter names each separated by a space. The list must be enclosed in quotation marks; for example,

```
"fingers finger numFingers numFinger"
```

GUI Equivalent

Command: *Options – Layout XL (Generation tab)*

Field: *Pcell fingering parameter names*

Examples

```
envGetVal("layoutXL" "lxFingeringNames")  
envSetVal("layoutXL" "lxFingeringNames" 'string "fingers finger numFingers  
numFinger")
```

Related Topics

List of Layout XL Environment Variables

lxGenerationOrientation

```
layoutXL lxGenerationOrientation 'string { "preserve" | "R0" | "R90" | "R180" |  
      "R270" | "MY" | "MYR90" | "MX" | "MXR90" }
```

Description

Controls the orientation of Layout XL layout devices generated from the schematic.

The default is "preserve", which means that devices are generated with the same orientation as in the schematic.

Note: Devices generated with a particular orientation may be further transformed if either the *Folding*, *Chaining*, or *Chain Folds* options are switched on.

Arguments

preserve	Devices are generated with the same orientation as in the schematic.
R0	Devices are generated with R0 orientation.
R90	Generated devices are rotated 90 degrees clockwise.
R180	Generated devices are rotated 180 degrees clockwise.
R270	Generated devices are rotated 270 degrees clockwise.
MY	Generated devices are mirrored over the Y axis.
MYR90	Generated devices are mirrored over the Y axis and rotated 90 degrees clockwise.
MX	Generated devices are mirrored over the X axis.
MXR90	Generated devices are mirrored over the X axis and rotated by 90 degrees clockwise.

GUI Equivalent

Command: *Options – Layout XL (Generation tab)*
Field: *Device orientation*

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

Examples

```
envGetVal("layoutXL" "lxGenerationOrientation")
envSetVal("layoutXL" "lxGenerationOrientation" 'string "preserve")
envSetVal("layoutXL" "lxGenerationOrientation" 'string "R0")
envSetVal("layoutXL" "lxGenerationOrientation" 'string "MXR90")
```

Related Topics

[List of Layout XL Environment Variables](#)

lxGenerationTopLevelOnly

```
layoutXL lxGenerationTopLevelOnly 'boolean { t | nil }
```

Description

Generates layout instances only from top-level schematic instances during the *Generate All From Source* and the *Generate Selected From Source* commands. Hierarchical schematic instances are not generated in the layout.

The default is `nil`.

Arguments

None.

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "lxGenerationTopLevelOnly")
envSetVal("layoutXL" "lxGenerationTopLevelOnly" 'boolean t)
envSetVal("layoutXL" "lxGenerationTopLevelOnly" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

lxGenerationTopLevelOnly

```
layoutXL lxGenerationTopLevelOnly 'boolean { t | nil }
```

Description

Generates layout instances only from top-level schematic instances during the *Generate All From Source* and the *Generate Selected From Source* commands. Hierarchical schematic instances are not generated in the layout.

The default is `nil`.

Arguments

None.

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "lxGenerationTopLevelOnly")  
envSetVal("layoutXL" "lxGenerationTopLevelOnly" 'boolean t)  
envSetVal("layoutXL" "lxGenerationTopLevelOnly" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

lxGenerationTopLevelOnly

```
layoutXL lxGenerationTopLevelOnly 'boolean { t | nil }
```

Description

Generates layout instances only from top-level schematic instances during the *Generate All From Source* and the *Generate Selected From Source* commands. Hierarchical schematic instances are not generated in the layout.

The default is `nil`.

Arguments

None.

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "lxGenerationTopLevelOnly")  
envSetVal("layoutXL" "lxGenerationTopLevelOnly" 'boolean t)  
envSetVal("layoutXL" "lxGenerationTopLevelOnly" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

```
layoutXL lxGenerationTopLevelOnly 'boolean { t | nil }
```

Description

Generates layout instances only from top-level schematic instances during the *Generate All From Source* and the *Generate Selected From Source* commands. Hierarchical schematic instances are not generated in the layout.

The default is `nil`.

Arguments

None.

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "lxGenerationTopLevelOnly")  
envSetVal("layoutXL" "lxGenerationTopLevelOnly" 'boolean t)  
envSetVal("layoutXL" "lxGenerationTopLevelOnly" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

lxGenerateInBoundary

```
layoutXL lxGenerateInBoundary 'boolean { t | nil }
```

Description

Generates layout representations within the design boundary when the *Generate All From Source* command is run.

The default is `nil`.

Arguments

None

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lxGenerateInBoundary")  
envSetVal("layoutXL" "lxGenerateInBoundary" 'boolean t)  
envSetVal("layoutXL" "lxGenerateInBoundary" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

lxGenerationTopLevelOnly

```
layoutXL lxGenerationTopLevelOnly 'boolean { t | nil }
```

Description

Generates layout instances only from top-level schematic instances during the *Generate All From Source* and the *Generate Selected From Source* commands. Hierarchical schematic instances are not generated in the layout.

The default is `nil`.

Arguments

None.

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "lxGenerationTopLevelOnly")
envSetVal("layoutXL" "lxGenerationTopLevelOnly" 'boolean t)
envSetVal("layoutXL" "lxGenerationTopLevelOnly" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

lxGetSignifDigits

```
layoutXL lxGetSignifDigits 'int { 0 | 3 ... 14 }
```

Description

Controls the number of significant digits used by Layout XL when calculating the value of layout CDF parameters.

When enabled `lxGetSignifDigits` overrides the setting of `aelGetSignifDigits` for calculations in the layout cellview. This lets you specify a different precision to be used in layout cellview calculations; for example, those involving parameterized cells.

Valid values are 0 and any integer between 3 and 14. The default is 0, which means that the environment variable is off. The higher the value you set, the more precise are the values used in the calculations.

Arguments

0	<code>lxGetSignifDigits</code> is disabled.
3 ... 14	An integer specifying the number of significant digits used by Layout XL. The minimum value is 3 and the maximum value 14. The higher the value, the more precise the values used in calculations.

GUI Equivalent

Command:	<i>Options – Layout XL (Parameters tab)</i>
Field:	<i>Use significant digits when calculating parameter values</i>

Examples

```
envGetVal("layoutXL" "lxGetSignifDigits")  
envSetVal("layoutXL" "lxGetSignifDigits" 'int 3)
```

Related Topics

List of Layout XL Environment Variables

lxGroundNetNames

```
layoutXL lxGroundNetNames 'string "list_of_ground_net_names"
```

Description

Specifies ground net names to help you achieve optimized chaining results in Layout XL device level schematics.

The list is also used by the Virtuoso custom digital placer to exclude ground nets during wire length optimization; and to determine the ground nets that are hidden when the hideDraglinesForGlobalNets environment variable is switched on.

The default values are "gnd gnd! gnd: vss vss! vss:".

Arguments

list_of_ground_net_names

A list of ground net names each separated by a space. The list must be enclosed in quotation marks; for example,

```
"gnd gnd! gnd: vss vss! vss:"
```

GUI Equivalent

Command: *Options – Layout XL (General tab)*

Field: *Ground net names*

Examples

```
envGetVal("layoutXL" "lxGroundNetNames")  
envSetVal("layoutXL" "lxGroundNetNames" 'string "gnd gnd! gnd: vss vss! vss:")
```

Related Topics

List of Layout XL Environment Variables

lxInitResetSource

```
layoutXL lxInitResetSource 'boolean { t | nil }
```

Description

Prompts you to reset the source before starting the Layout XL layout editor from the layout.

The default is `nil`.

Arguments

None.

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "lxInitResetSource")  
envSetVal("layoutXL" "lxInitResetSource" 'boolean t)  
envSetVal("layoutXL" "lxInitResetSource" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

lxLocalAbutment

```
layoutXL lxLocalAbutment 'boolean { t | nil }
```

Description

Controls device abutment on overlapping, unabutted pins when a full extraction is run.

The default is `nil`, which means that when you edit a parameter of an abutted instance, the instance is unabutted and then reabutted in order to take account of any differences in the diffusion sizes in wider or narrower transistors. To prevent this from happening, set `lxLocalAbutment` to `t`.

This environment variable also controls whether or not automatic abutment takes place when you launch Layout XL with a full connectivity extraction.

When set to `nil`, the tool automatically abuts any devices which overlap but have not been abutted previously. Any devices that are already abutted are not reabutted. To prevent devices from being abutted automatically when you open a cellview in Layout XL, set `lxLocalAbutment` to `t`. The software abuts only objects that are subsequently edited interactively.

Arguments

Command: *Options – Layout XL (Generation tab)*

Field: *Perform auto abutment only after a move or stretch*

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "lxLocalAbutment")
envSetVal("layoutXL" "lxLocalAbutment" 'boolean t)
envSetVal("layoutXL" "lxLocalAbutment" 'boolean nil)
```

Additional Information

If you change this environment variable, the change takes effect only after you restart Layout XL. To toggle local abutment and have the change take effect without restarting the software, use the lxToggleLocalAbutment SKILL command.

This environment variable has no effect on the autoSpace functionality.

Related Topics

List of Layout XL Environment Variables

lxPositionMinSep

```
layoutXL lxPositionMinSep [aFloat >= 0.0]
```

Description

Positions the instances in the layout at a minimum separation based on the user-defined value.

The environment variable can be used to spread out or compact the instances in a layout by increasing or decreasing the minimum separation value.

Used by the following commands:

- *Generate All From Source*
- *Update Components And Nets*
- *Pick From Schematic*
- *Place As In Schematic*
- *Generate Folded Devices*

Arguments

None

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lxPositionMinSep")  
envSetVal("layoutXL" "lxPositionMinSep" 'float 0.5)
```

Related Topics

[List of Layout XL Environment Variables](#)

lxPositionPinsOnBoundary

```
layoutXL lxPositionPinsOnBoundary 'boolean { t | nil }
```

Description

Positions the pins on the prBoundary.

The default is `t`.

Used by the following commands:

- *Generate All From Source, when* `lxGenerateInBoundary = t`
- *Place As In Schematic*

Arguments

<code>t</code>	Positions the pins on the boundary.
<code>nil</code>	Positions the pins inside the boundary.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lxPositionPinsOnBoundary")  
envSetVal("layoutXL" "lxPositionPinsOnBoundary" 'boolean t)  
envSetVal("layoutXL" "lxPositionPinsOnBoundary" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

lxRetainFoldOrient

```
layoutXL lxRetainFoldOrient 'boolean { t | nil }
```

Description

Preserves the original device orientation when running the *Generate Folded Devices* command or the *Generate All From Source* command with the *Folding* option switched on.

Note: If the *Chaining* option is also switched on, instance orientation is not preserved.

Arguments

None.

GUI Equivalent

Command: *Options – Layout XL (Generation tab)*

Field: *Retain instance orientation*

Examples

```
envGetVal("layoutXL" "lxRetainFoldOrient")  
envSetVal("layoutXL" "lxRetainFoldOrient" 'boolean t)  
envSetVal("layoutXL" "lxRetainFoldOrient" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

lxSchematicDefaultApp

```
layoutXL lxSchematicDefaultApp 'cyclic { "L" | "XL" }
```

Description

Specifies which tier of the Virtuoso Schematic Editor is started when you launch Layout XL or GXL.

The default is "XL".



The schematic editor tier level does not change when you launch Layout XL from the Virtuoso Schematic Editor XL nor when you switch between layout editor tier levels *after* Layout XL or GXL have been launched.

Arguments

L	Launches Virtuoso Schematic Editor L
XL	Launches Virtuoso Schematic Editor XL

GUI Equivalent

Command:	<i>Options – Layout XL</i> (<i>General</i> tab, Connectivity Reference groupbox)
Field:	<i>Open in</i>

Examples

```
envGetVal("layoutXL" "lxSchematicDefaultApp")  
envSetVal("layoutXL" "lxSchematicDefaultApp" 'cyclic "L")  
envSetVal("layoutXL" "lxSchematicDefaultApp" 'cyclic "XL")
```

Related Topics

[List of Layout XL Environment Variables](#)

lxSchExtractTopLevelOnly

```
layoutXL lxSchExtractTopLevelOnly 'boolean { t | nil }
```

Description

Specifies that only the top level of the schematic is to be extracted during the following commands in the *Connectivity* menu.

- *Generate All From Source*
- *Generate Clones*
- *Check Against Source*
- *Update Components And Nets*
- *Define Device Correspondence*

The lower level schematics are not extracted, even if they need to be extracted.

The default is `nil`.

Arguments

None.

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "lxSchExtractTopLevelOnly")
envSetVal("layoutXL" "lxSchExtractTopLevelOnly" 'boolean t)
envSetVal("layoutXL" "lxSchExtractTopLevelOnly" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

lxStackMinimalFolding

```
layoutXL lxStackMinimalFolding 'boolean { t | nil }
```

Description

Sets folding to yield the minimum number of legs for each folded device. It does not matter whether the number is even or odd.

The default is `t`.

Arguments

None.

GUI Equivalent

Command: *Options – Layout XL (Generation tab)*

Field: *Generate minimal folding*

Examples

```
envGetVal("layoutXL" "lxStackMinimalFolding")  
envSetVal("layoutXL" "lxStackMinimalFolding" 'boolean t)  
envSetVal("layoutXL" "lxStackMinimalFolding" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

lxStackPartitionParameters

```
layoutXL lxStackPartitionParameters 'string "(integer integer)"
```

Description

Sets the device chaining partition parameters.

The first parameter controls how far up the hierarchy Layout XL looks for opportunities to abut devices. A value of 0 means that chaining can look for abutments only in the same hierarchical level. A value of *N* means that chaining can look up *N* levels of hierarchy in order to find abutments.

The second parameter sets the maximum number of devices permitted in a chain.

The default is "(0 100)".



The first parameter is always considered by the *Generate All From Source* and *Generate Selected From Source* commands. However, the *Generate Chained Devices* command considers this parameter only if there are 100 instances or more in the selected set.

Arguments

```
(integer integer)
```

A list of two integers enclosed in parentheses and quotation marks; for example,

```
"(0 100)"
```

GUI Equivalent

Command:	<i>Options – Layout XL (Generation tab)</i>
Field:	<i>Search levels up for abutments</i> <i>Maximum number of devices in a chain</i>

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

Examples

```
envGetVal("layoutXL" "lxStackPartitionParameters")  
envSetVal("layoutXL" "lxStackPartitionParameters" 'string "(0 100)")
```

Related Topics

[List of Layout XL Environment Variables](#)

lxSupplyNetNames

```
layoutXL lxSupplyNetNames 'string "list_of_supply_net_names"
```

Description

Specifies the list of power net names used to achieve optimized chaining results in Layout XL device level schematics.

The list is also used by the Virtuoso custom digital placer to exclude power nets during wire length optimization; and to determine the power nets that are hidden when the hideDraglinesForGlobalNets environment variable is switched on.

The default values are "vcc vcc! vcc: vdd vdd! vdd:".

Arguments

list_of_supply_net_names

A list of power net names each separated by a space. The list must be enclosed in quotation marks; for example,

```
"vcc vcc! vcc: vdd vdd! vdd:"
```

GUI Equivalent

Command: *Options – Layout XL (General tab)*

Field: *Power net names*

Examples

```
envGetVal("layoutXL" "lxSupplyNetNames")  
envSetVal("layoutXL" "lxSupplyNetNames" 'string "vcc vcc! vcc: vdd vdd! vdd:")
```

Related Topics

List of Layout XL Environment Variables

lxUpdateFoldedWidth

```
layoutXL lxUpdateFoldedWidth 'string { "No Change" | "Equalize" | "Distribute" }
```

Description

Specifies how the *Update Layout Parameters* command updates folded devices when the width value is changed in the schematic.

The default is "No Change".

Arguments

No Change

Folded devices are not updated.

Equalize

Creates folds of equal width in the layout instance. The number of folds does not change, only (potentially) the width of each fold.

For example, the total width in the schematic is 12 and is distributed as follows in the layout.

```
layout widths I1.1 w = 2
               I1.2 w = 4
               I1.3 w = 6
```

If you change the width in the schematic to 15 and run *Update – Layout Parameters* with this option set, the new widths are as follows.

```
layout widths I1.1 w = 5
               I1.2 w = 5
               I1.3 w = 5
```

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

Distribute

Distributes the additional (or reduced) width equally amongst all the folds in the layout instance. Again, the number of folds does not change, only the width of each fold.

For example, the total width in the schematic is 12 and is distributed as follows in the layout.

```
layout widths I1.1 w = 2
               I1.2 w = 4
               I1.3 w = 6
```

If you change the width in the schematic to 18 and run *Update Layout Parameters* with this option set, the new widths are as follows.

```
layout widths I1.1 w = 4
               I1.2 w = 6
               I1.3 w = 8
```

An extra 2 units of width is added to each fold.

GUI Equivalent

Command: *Options – Layout XL (Generation tab)*

Field: *Update width parameter method*

Examples

```
envGetVal("layoutXL" "lxUpdateFoldedWidth")
envSetVal("layoutXL" "lxUpdateFoldedWidth" 'string "No Change")
envSetVal("layoutXL" "lxUpdateFoldedWidth" 'string "Equalize")
envSetVal("layoutXL" "lxUpdateFoldedWidth" 'string "Distribute")
```

Related Topics

[List of Layout XL Environment Variables](#)

lxUseLibList

```
layoutXL lxUseLibList 'string "list_of_libraries"
```

Description

Specifies a list of libraries for Layout XL to search for a layout view that corresponds to a particular schematic or symbol view.

When generating or updating layout components from a schematic or symbol view, Layout XL uses the layout cellview specified in the Configure Physical Hierarchy window for the schematic or symbol in question. If there is no cellview specified, Layout XL searches for a layout cellview of the same name stored in the same library as the schematic or symbol cellview. If there is no such cellview, the software searches any libraries specified by `lxUseLibList`.

The software searches the libraries in the sequence in which they are listed and uses the first layout cellview it finds with a name that matches the name of the schematic or symbol. If it is unable to find a corresponding layout cellview, the software issues a warning message.

The following commands in the *Connectivity* menu honor the `lxUseLibList` environment variable.

- *Generate All From Source*
- *Generate Clones*
- *Generate Selected From Source*
- *Generate Chained Devices*
- *Generate Folded Devices*
- *Check Against Source*
- *Update Components And Nets*
- *Update Layout Parameters*
- *Update Schematic Parameters*

Arguments

```
list_of_libraries
```

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

A list of library names each separated by a space. The list must be enclosed in quotation marks; for example,

```
"lib1 lib2 lib3"
```

GUI Equivalent

Command: *Launch – Configure Physical Hierarchy*

Field: *Physical library list*

Examples

```
envGetVal("layoutXL" "lxUseLibList")
```

```
envSetVal("layoutXL" "lxUseLibList" 'string "lib1 lib2 lib3")
```

Related Topics

[List of Layout XL Environment Variables](#)

lxValidateXLParameterEvaluation

```
layoutXL lxValidateXLParameterEvaluation 'string {"" | "Parallel(Warning)" |  
    "Parallel (Block Generation)"}
```

Description

Controls the behavior of the tool when the specified mfactor cannot be recognized as an integer.

The default is " ".

Arguments

-

" "

The mfactor value is not checked. No warning message is generated in this case.

"*Parallel(Warning)*"

A warning message is generated indicating that the specified mfactor value is invalid.

"*Parallel(Block Generation)*"

A warning message is generated indicating that the specified mfactor value is invalid. The following layout generation commands are blocked:

- *Generate All From Source*
- *Generate Selected From Source*
- *Update Components and Nets*

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "lxValidateXLParameterEvaluation")
```

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

```
envSetVal("layoutXL" "lxValidateXLParameterEvaluation" 'string "")
envSetVal("layoutXL" "lxValidateXLParameterEvaluation"
            'string "Parallel(Warning)")
envSetVal("layoutXL" "lxValidateXLParameterEvaluation"
            'string "Parallel(Block Generation)")
```

Related Topics

[List of Layout XL Environment Variables](#)

lxWidthTolerance

```
layoutXL lxWidthTolerance 'string "(value value)"
```

Description

Sets the allowed device width variation for folding.

The default is "(0.0 0.0)".

Arguments

```
(value value)
```

A list of two positive numbers separated by a space. The first is the absolute value of the negative tolerance; the second is the positive tolerance. The list must be enclosed in parentheses and quotation marks; for example,

```
"(0.0 0.0)"
```

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "lxWidthTolerance")  
envSetVal("layoutXL" "lxWidthTolerance" 'string "(0.25 0.25)"
```

Related Topics

[List of Layout XL Environment Variables](#)

mfactorNames

```
layoutXL mfactorNames 'string "list_of_names"
```

Description

Lists the names of properties used in the schematic to specify the multiplication factor (`mfactor`) for transistors.

The default is "m M".

Note: The `mfactor` property value can be expressed as an integer or an expression.

Arguments

list_of_names

A list of `mfactor` property names each separated by a space.
The list must be enclosed in quotation marks; for example,

"m M"

GUI Equivalent

Command: *Options – Layout XL (Parameters tab)*

Field: *Parallel-connected factor*

Examples

```
envGetVal("layoutXL" "mfactorNames")  
envSetVal("layoutXL" "mfactorNames" 'string "m M")
```

Related Topics

[List of Layout XL Environment Variables](#)

mfactorSplit

```
layoutXL mfactorSplit 'boolean { t | nil }
```

Description

Controls whether Layout XL places schematic devices with the mfactor property as multiple devices in the layout.

The default is `t`.

Note: You can override this environment variable for a given instance by setting the lxMfactorSplit property for the instance in question.

Arguments

None.

GUI Equivalent

Command: *Options – Layout XL (Generation tab)*

Field: *Split mfactored devices*

Examples

```
envGetVal("layoutXL" "mfactorSplit")  
envSetVal("layoutXL" "mfactorSplit" 'boolean t)  
envSetVal("layoutXL" "mfactorSplit" 'boolean nil)
```

Additional Information

This environment variable controls *only* whether Layout XL generates one device or several devices in the layout view. It does not change the widths of the generated devices.

Related Topics

List of Layout XL Environment Variables

moveAsGroup

```
layoutXL moveAsGroup 'boolean { t | nil }
```

Description

Specifies that the currently selected instances are treated as a single group and maintain their relative positions during move and stretch operations.

If two cells in the selected group can be abutted to different cells, only one abutment will occur because to abut both would upset the relative position of the set. The cell chosen to abut is the one that causes the least movement within the selected set.

Arguments

None.

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "moveAsGroup")  
envSetVal("layoutXL" "moveAsGroup" 'boolean t)  
envSetVal("layoutXL" "moveAsGroup" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

openConnRef

```
layoutXL openConnRef 'boolean { t | nil }
```

Description

Specifies whether or not the source schematic cellview is opened when you use the *Edit In Place* or *Descend* commands.

When switched off, the layout cellview is still opened in Layout XL mode, but with no connectivity reference.

Arguments

None.

GUI Equivalent

Command: *Options – Layout XL (General tab)*

Field: *Open connectivity reference during Edit
 In Place/Descend*

Examples

```
envGetVal("layoutXL" "openConnRef")  
envSetVal("layoutXL" "openConnRef" 'boolean t)  
envSetVal("layoutXL" "openConnRef" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

openConnRefTab

```
layoutXL openConnRefTab 'boolean { t | nil }
```

Description

Specifies whether or not the source cellview is opened in a new tab when you use the *Edit In Place* or *Descend* commands.

The default is `nil`.

Arguments

None.

GUI Equivalent

Command: *Options – Layout XL (General tab)*

Field: *Open connectivity reference during Edit In Place/Descend:*
 - *Open in new window*
 - *Open in new tab*

Examples

```
envGetVal("layoutXL" "openConnRefTab")  
envSetVal("layoutXL" "openConnRefTab" 'boolean t)  
envSetVal("layoutXL" "openConnRefTab" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

openLocation

```
ab openLocation 'string { "current tab " | "new tab" | "new window" }
```

Description

Specifies whether a cellview descended into directly from the Annotation Browser assistant is opened in a new tab, the current tab, or a new window.

The default is "current tab".

Arguments

current tab	The descended cellview is opened in the current tab.
new tab	The descended cellview is opened in a new tab.
new window	The descended cellview is opened in a new layout window.

GUI Equivalent

Command:	<i>Windows – Assistants – Annotation Browser – Options (RMB)</i>
Field:	<i>Default Open – Open in</i>

Examples

```
envGetVal("ab" "openLocation")  
envSetVal("ab" "openLocation" 'string "current tab")  
envSetVal("ab" "openLocation" 'string "new tab")  
envSetVal("ab" "openLocation" 'string "new window")
```

Related Topics

[Annotation Browser](#)

[List of Layout XL Environment Variables](#)

openMode

```
ab openMode 'string { "edit" | "read" }
```

Description

Specifies whether a cellview descended into directly from the Annotation Browser assistant is opened in edit mode or read mode.

The default is "edit".

Arguments

edit	The descended cellview is opened in edit mode.
read	The descended cellview is opened in read mode.

GUI Equivalent

Command: *Windows – Assistants – Annotation Browser – Options (RMB)*

Field: *Default Open – Mode*

Examples

```
envGetVal("ab" "openMode")  
envSetVal("ab" "openMode" 'string "edit")  
envSetVal("ab" "openMode" 'string "read")
```

Related Topics

[Annotation Browser](#)

[List of Layout XL Environment Variables](#)

paramsToIgnore

```
layoutXL paramsToIgnore 'string "list_of_parameters"
```

Description

Specifies the parameters that are ignored by the following commands.

- *Generate All From Source*
- *Generate Selected From Source*
- *Generate Clones*
- *Check Against Source*
- *Update Components And Nets*
- *Update Layout Parameters*
- *Update Schematic Parameters*

The following parameters are always ignored.

instancesLastChanged	lxPlacementStatus
instNamePrefix	lxRounding
lxIgnoreParamForCAS	lxStopList
lxIgnoredParams	lxTimeStamp
lxMFactorNum	lxUseCell
lxParamsToIgnore	pin#
lxParamsToIgnoreForCheck	posi

Arguments

list_of_parameters

A list of parameter names each separated by a space. The list must be enclosed in quotation marks.

GUI Equivalent

Command: *Options – Layout XL (Parameters tab)*

Field: *Parameters to ignore during generation*

Examples

```
envGetVal("layoutXL" "paramsToIgnore")  
envSetVal("layoutXL" "paramsToIgnore" 'string "effW w myParam")
```

Related Topics

[List of Layout XL Environment Variables](#)

paramsToIgnoreForCheck

```
layoutXL paramsToIgnoreForCheck 'string "list_of_parameters"
```

Description

Lists the parameters to be ignored during the *Check Against Source* command. Mismatches for any of the listed parameters are not reported by the check.

The list automatically includes all the parameters specified by the paramsToIgnore environment variable.

Arguments

list_of_parameters

A list of parameter names each separated by a space. The list must be enclosed in quotation marks.

GUI Equivalent

Command: *Options – Layout XL (Parameters tab)*

Field: *Parameters to ignore during check*

Examples

```
envGetVal("layoutXL" "paramsToIgnoreForCheck")  
envSetVal("layoutXL" "paramsToIgnoreForCheck" 'string "w l myParam")
```

Related Topics

List of Layout XL Environment Variables

paramTolerance

```
layoutXL paramTolerance 'float floating_point_number
```

Description

Specifies the relative tolerance allowed when comparing values between the layout and the schematic.

This tolerance is applied by any function that deals with parameters, including the following commands in the *Connectivity* menu.

- *Generate All From Source*
- *Generate Folded Devices*
- *Check Against Source*
- *Update Layout Parameters*
- *Update Schematic Parameters*

The default is 1e-6.

Arguments

floating_point_number

A float specifying the relative tolerance applied when comparing parameter values; for example,

1e-6

GUI Equivalent

Command: *Options – Layout XL (Parameters tab)*

Field: *Tolerance*

Examples

```
envGetVal("layoutXL" "paramTolerance")  
envSetVal("layoutXL" "paramTolerance" 'float 1e-06)
```

Related Topics

List of Layout XL Environment Variables

pinTextSamePurpose

```
layoutXL pinTextSamePurpose 'boolean { t | nil }
```

Description

Creates the pin name label on the same layer purpose as the pin it corresponds to. When set to `nil`, the pin name is created on the purpose defined by `pinTextPurpose`.

The default is `nil`.

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

Arguments

None.

GUI Equivalent

Command: *Set Pin Label Text Style*

Field: *Layer Purpose – Same As Pin*

Examples

```
envGetVal("layoutXL" "pinTextSamePurpose")
envSetVal("layoutXL" "pinTextSamePurpose" 'boolean t)
envSetVal("layoutXL" "pinTextSamePurpose" 'boolean nil)
```

Related Topics

[pinTextSameLayer](#)

[List of Layout XL Environment Variables](#)

prefixLayoutInstNamesWithPipe

```
layoutXL prefixLayoutInstNamesWithPipe 'boolean { t | nil }
```

Description

Controls the names of instances generated by Layout XL.

- When set to `t`, Layout XL prefixes all instance names with an OR bar: | P0
- When set to `nil`, Layout XL generates instance names without an OR bar: P0

The default is `t`.

Arguments

None.

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "prefixLayoutInstNamesWithPipe")
envSetVal("layoutXL" "prefixLayoutInstNamesWithPipe" 'boolean t)
envSetVal("layoutXL" "prefixLayoutInstNamesWithPipe" 'boolean nil)
```

Additional Information

Set this environment variable to `nil` to facilitate better interoperability with the OpenAccess database and the Cadence® Encounter™ Digital Implementation System. You will need to update any SKILL code that relies on the OR bar in the names.

Related Topics

[List of Layout XL Environment Variables](#)

preserveAreaBoundary

```
layoutXL preserveAreaBoundary 'boolean { t | nil }
```

Description

Specifies that any existing area boundaries in the layout are preserved when you run the *Generate All From Source* command.

The default is `nil`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – All From Source (Floorplan tab)*

Field: *Preserve Floorplanning Objects – Area Boundaries*

Examples

```
envGetVal("layoutXL" "preserveAreaBoundary")  
envSetVal("layoutXL" "preserveAreaBoundary" 'boolean t)  
envSetVal("layoutXL" "preserveAreaBoundary" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

preserveBlockages

```
layoutXL preserveBlockages 'boolean { t | nil }
```

Description

Specifies that any existing blockage objects in the layout are preserved when you run the *Generate All From Source* command.

The default is `nil`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – All From Source (Floorplan tab)*

Field: *Preserve Floorplanning Objects – Blockages*

Examples

```
envGetVal("layoutXL" "preserveBlockages")  
envSetVal("layoutXL" "preserveBlockages" 'boolean t)  
envSetVal("layoutXL" "preserveBlockages" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

preserveClusters

```
layoutXL preserveClusters 'boolean { t | nil }
```

Description

Specifies that any existing clusters in the layout are preserved when you run the *Generate All From Source* command.

The default is `nil`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – All From Source (Floorplan tab)*

Field: *Preserve Floorplanning Objects – Clusters*

Examples

```
envGetVal("layoutXL" "preserveClusters")  
envSetVal("layoutXL" "preserveClusters" 'boolean t)  
envSetVal("layoutXL" "preserveClusters" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

preserveClusterBoundaries

```
layoutXL preserveClusterBoundaries 'boolean { t | nil }
```

Description

Specifies that any existing cluster boundaries in the layout are preserved when you run the *Generate All From Source* command.

The default is `nil`.

Note: Cluster boundaries are preserved only if clusters are preserved.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – All From Source (Floorplan tab)*

Field: *Preserve Floorplanning Objects – Cluster Boundaries*

Examples

```
envGetVal("layoutXL" "preserveClusterBoundaries")
envSetVal("layoutXL" "preserveClusterBoundaries" 'boolean t)
envSetVal("layoutXL" "preserveClusterBoundaries" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

preserveRows

```
layoutXL preserveRows 'boolean { t | nil }
```

Description

Specifies that any existing rows and custom placement areas in the layout are preserved when you run the *Generate All From Source* command.

The default is `nil`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – All From Source (Floorplan tab)*

Field: *Preserve Floorplanning Objects – Rows and Custom Placement Areas*

Examples

```
envGetVal("layoutXL" "preserveRows")  
envSetVal("layoutXL" "preserveRows" 'boolean t)  
envSetVal("layoutXL" "preserveRows" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

preserveTrackPattern

```
layoutXL preserveTrackPattern 'boolean { t | nil }
```

Description

Specifies that any existing track patterns in the layout are preserved when you run the *Generate All From Source* command.

The default is `nil`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Generate – All From Source (Floorplan tab)*

Field: *Preserve Floorplanning Objects – Track Patterns*

Examples

```
envGetVal("layoutXL" "preserveTrackPattern")  
envSetVal("layoutXL" "preserveTrackPattern" 'boolean t)  
envSetVal("layoutXL" "preserveTrackPattern" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

probeCycleHilite

```
layoutXL probeCycleHilite 'boolean { t | nil }
```

Description

Specifies whether the *XL Probe* command cycles through the `hilite` drawing entry layers used for showing probes.

The default is `t`.

Arguments

None.

GUI Equivalent

Command: *Options – Highlight*

Field: *Display Layer – Cycle*

Examples

```
envGetVal("layoutXL" "probeCycleHilite")  
envSetVal("layoutXL" "probeCycleHilite" 'boolean t)  
envSetVal("layoutXL" "probeCycleHilite" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

probeDevice

```
layoutXL probeDevice 'boolean { t | nil }
```

Description

Controls whether the *Probe* command highlights devices on the schematic and layout.

The default is `t`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – XL Probe*

Field: *Devices*

Examples

```
envGetVal("layoutXL" "probeDevice")  
envSetVal("layoutXL" "probeDevice" 'boolean t)  
envSetVal("layoutXL" "probeDevice" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

probeDuringCreate

```
layoutXL probeDuringCreate 'boolean { t | nil }
```

Description

Controls whether or not nets that are tapped during interactive editing are automatically highlighted in the layout canvas.

Arguments

None.

GUI Equivalent

Command: *Create – Shape – Rectangle*
 Create – Shape – Polygon
 Create – Shape – Path
 Create – Wire
 Options – Layout XL (Display tab)

Field: *Probe nets during object creation*

Examples

```
envGetVal ("layoutXL" "probeDuringCreate")  
envSetVal ("layoutXL" "probeDuringCreate" 'boolean t)  
envSetVal ("layoutXL" "probeDuringCreate" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

probeHiliteLayer

```
layoutXL probeHiliteLayer 'string "layer_purpose_pair"
```

Description

Specifies the layer used to highlight probed components in the canvas.

The default is "hilite drawing".

Arguments

None.

GUI Equivalent

Command: *Connectivity – XL Probe – Probing Options*

Field: *Display Layer*

Examples

```
envGetVal("layoutXL" "probeHiliteLayer")  
envSetVal("layoutXL" "probeHiliteLayer" 'string "hilite drawing")
```

Related Topics

[List of Layout XL Environment Variables](#)

probeInfoInCIW

```
layoutXL probeInfoInCIW 'boolean { t | nil }
```

Description

Sends the output of the *Probe* command to the CIW as well as the XL Probe form.

The default is `nil`, meaning that the output is visible only in the XL Probe form.

Arguments

None.

GUI Equivalent

Command: *Connectivity – XL Probe*

Field: *Send Messages to CIW*

Examples

```
envGetVal("layoutXL" "probeInfoInCIW")  
envSetVal("layoutXL" "probeInfoInCIW" 'boolean t)  
envSetVal("layoutXL" "probeInfoInCIW" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

probeNet

```
layoutXL probeNet 'boolean { t | nil }
```

Description

Enables the *Probe* command to highlight nets in the schematic and layout.

The default is `t`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – XL Probe*

Field: *Nets*

Examples

```
envGetVal("layoutXL" "probeNet")  
envSetVal("layoutXL" "probeNet" 'boolean t)  
envSetVal("layoutXL" "probeNet" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

probePin

```
layoutXL probePin 'boolean { t | nil }
```

Description

Enables the *Probe* command to highlight pins in the schematic and layout.

The default is `t`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – XL Probe*

Field: *Pins*

Examples

```
envGetVal("layoutXL" "probePin")  
envSetVal("layoutXL" "probePin" 'boolean t)  
envSetVal("layoutXL" "probePin" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

propsUsedToIgnoreObjs

```
layoutXL propsUsedToIgnoreObjs 'string "list_of_properties"
```

Description

Lists the properties that cause pins and instances to be ignored by the following *Connectivity* commands. Any object with one of the listed properties set to `t` is ignored during these operations.

- *Generate All From Source*
- *Generate Selected From Source*
- *Generate Clones*
- *Check Against Source*
- *Update Components And Nets*
- *Update Layout Parameters*
- *Update Schematic Parameters*

By default, the list the contains the following properties: `"lxRemoveDevice ignore nlAction"`

Note: For information on the `nlAction` property, see [Netlisting](#) in the *Virtuoso NC Verilog Environment User Guide*.

Arguments

list_of_properties

A list of property names each separated by a space. The list must be enclosed in quotation marks; for example,

```
"lxRemoveDevice ignore nlAction"
```

GUI Equivalent

Command: *Options – Layout XL (Parameters tab)*

Field: *Properties used to ignore objects during generation*

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

Examples

```
envGetVal("layoutXL" "propsUsedToIgnoreObjs")  
envSetVal("layoutXL" "propsUsedToIgnoreObjs" 'string "lxRemoveDevice ignore  
nlAction")
```

Related Topics

[List of Layout XL Environment Variables](#)

propsUsedToIgnoreObjsForCheck

```
layoutXL propsUsedToIgnoreObjsForCheck 'string "list_of_properties"
```

Description

Lists the properties used to ignore objects during the *Check Against Source* command. Any object with one of the listed properties set to `t` is ignored during this operation.

The default is "lvsIgnore", but the list also inherits all the properties specified in the propsUsedToIgnoreObjs environment variable.

Arguments

list_of_properties

A list of property names each separated by a space. The list must be enclosed in quotation marks.

GUI Equivalent

Command: *Options – Layout XL (Parameters tab)*

Field: *Properties used to ignore objects during check*

Examples

```
envGetVal ("layoutXL" "propsUsedToIgnoreObjsForCheck")  
envSetVal ("layoutXL" "propsUsedToIgnoreObjsForCheck" 'string "lvsIgnore")
```

Related Topics

List of Layout XL Environment Variables

resistanceParamNames

```
layoutXL resistanceParamNames 'string "list_of_parameters"
```

Description

Lists the names of the schematic parameters used to specify resistance.

The default is "r R".

The resistance value must be split among the generated series-connected devices during the following *Connectivity* commands.

- *Generate All From Source*
- *Check Against Source*
- *Update Layout Parameters.*

Layout XL checks each schematic instance for one of the listed parameter names and updates the resistance values of the matching layout parameters accordingly.

Arguments

list_of_parameters

A list of parameter names each separated by a space. The list must be enclosed in quotation marks; for example,

"r R"

GUI Equivalent

Command: *Options – Layout XL (Parameters tab)*

Field: *Resistance*

Examples

```
envGetVal("layoutXL" "resistanceParamNames")  
envSetVal("layoutXL" "resistanceParamNames" 'string "r R")
```

Related Topics

List of Layout XL Environment Variables

schematicWindow

```
layoutXL schematicWindow 'string "pairs_of_coordinates"
```

Description

Specifies the position and size of the schematic window on the screen. It is used by the autoArrange environment variable when rearranging the Layout XL windows on startup.

By default, the schematic window is positioned in the bottom left corner of the screen.

Arguments

pair_of_coordinates

A pair of coordinates enclosed in quotation marks specifying the lower left and upper right corners of the schematic window; for example,

```
"((132.0 292.0) (565.0 992.0))"
```

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "schematicWindow")  
envSetVal("layoutXL" "schematicWindow" 'string "((545.0 319.0) (1172.0 1160.0))")
```

Related Topics

[List of Layout XL Environment Variables](#)

scopeLevel

```
ab scopeLevel 'string "Current Cellview Only"
```

Description

Controls the default scope level when the Annotation Browser is first opened in a window.

The default is "Current Cellview Only".

Arguments

Current Cellview Only

Controls the default scope level within the current cellview only.

Current Cellview to Depth

Controls the default scope level up to the specified depth in the current cellview.

Top Cellview Only

Controls the default scope level within the top cellview only.

Top Cellview to Depth

Controls the default scope level up to the specified depth in the top cellview.

GUI Equivalent

None.

Examples

When setting the environment variable in .cdsenv:

```
ab scopeLevel string "Top Cellview To Depth"
```

When setting the environment variable in CIW:

```
envSetVal("ab" "scopeLevel" 'string "Top Cellview To Depth")
```

Related Topics

List of Layout XL Environment Variables

setPPConn

```
layoutXL setPPConn 'boolean { t | nil }
```

Description

Automatically identifies nodes that qualify as pseudoparallel connections and defines them as such during chaining and abutment. Contacts are dropped automatically if it is appropriate to do so, even if the preserveTerminalContacts environment variable is switched on.

Note: A pseudoparallel net connects nodes that are always the same voltage, so the current does not pass through the net.

The default is `t`.



Tip

If set to `nil`, you must define pseudoparallel nets explicitly using the *Define Pseudo Parallel Connect Pins* command.

Argument

None.

GUI Equivalent

Command: *Options – Layout XL (Generation tab)*

Field: *Create pseudoparallel connections*

Examples

```
envGetVal("layoutXL" "setPPConn")  
envSetVal("layoutXL" "setPPConn" 'boolean t)  
envSetVal("layoutXL" "setPPConn" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

sfactorNames

```
layoutXL sfactorNames 'string "list_of_names"
```

Description

Note: Lists the names of properties used in the schematic to specify the number of series-connected (*sfactor*) devices to be generated in the layout.

Layout XL checks each instance for a property matching one of the names on the list and uses the value of that property to generate the appropriate number of series-connected devices. The *sfactor* property value can be expressed as an integer or an expression.

The default is "s S".

Note: This environment variable is also used by the analog placer when adding dummy devices to MOSFETs or resistors with the modgenMakeMinDummies environment variable set to t.

Arguments

list_of_names

A list of *sfactor* property names each separated by a space. The list must be enclosed in quotation marks; for example,

"s S"

GUI Equivalent

Command: *Options – Layout XL (Parameters tab)*

Field: *Series-connected factor*

Examples

```
envGetVal("layoutXL" "sfactorNames")  
envSetVal("layoutXL" "sfactorNames" 'string "s S")
```

Related Topics

List of Layout XL Environment Variables

showDraglinesForDistantConns

```
layoutXL showDraglinesForDistantConns 'boolean { t | nil }
```

Description

Displays draglines for all the connections to the instance that is being manipulated. When switched off, draglines are shown only for the connections closest to the current instance.

This environment variable is honored by the *Generate Selected From Source*, *Generate Clones*, *Move*, and *Stretch* commands.



Tip

For more information on displaying draglines, see also [colorDraglines](#) and [hideDraglinesForGlobalNets](#).

Arguments

None.

GUI Equivalent

Command: *Options – Layout XL (Display tab)*

Field: *Initial Display – Draglines – Show distant connections*

Examples

```
envGetVal("layoutXL" "showDraglinesForDistantConns")  
envSetVal("layoutXL" "showDraglinesForDistantConns" 'boolean t)  
envSetVal("layoutXL" "showDraglinesForDistantConns" 'boolean nil)
```

Additional Information

Displaying many draglines during interactive commands can impair the performance of Layout XL. To mitigate these effects, use the [maxDragFig](#) environment variable to limit the number of figures that can be in a dragset.

To set `maxDragFig` interactively so that it takes effect in the current session, type the following in the CIW, where *win* is the window to which the limit applies.

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

```
win~>maxDragFig = 1000
```

To set a default value for `maxDragFig`, which will be used in all future Layout XL sessions, put the following line in your `.cdsenv` file.

```
graphic maxDragFig int 1000 nil
```

Related Topics

[List of Layout XL Environment Variables](#)

snapCpaToPlacementGrid

```
layoutXL snapCpaToPlacementGrid 'boolean { t | nil }
```

Description

Snaps the Custom Placement Area to the placement grid if the placement grid is defined.

The default is `nil`.

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "snapCpaToPlacementGrid")  
envSetVal("layoutXL" "snapCpaToPlacementGrid" 'boolean t)  
envSetVal("layoutXL" "snapCpaToPlacementGrid" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

stopList

```
layoutXL stopList 'string "list_of_viewNames"
```

Description

Specifies the default value for the *Physical stop view list* field when a new physical configuration is created.

The physical stop view list is used to determine the view at which the hierarchical traversal of a design stops; i.e., when traversing a hierarchy, the system stops when it encounters a view with one of the specified names.

The default is "layout compacted symbolic".



This environment variable is used only when creating a new physical configuration view. Changing the value has no effect on an existing physical configuration view.

Arguments

list_of_viewNames

A list of view names each separated by a space. The list must be enclosed in quotation marks; for example,

```
"layout compacted symbolic"
```

GUI Equivalent

Command: *Launch – Configure Physical Hierarchy*

Field: *Physical stop view list* (in the *Global Bindings* pane)

Examples

```
envGetVal("layoutXL" "stopList")  
envSetVal("layoutXL" "stopList" 'string "layout compacted symbolic")
```

Related Topics

List of Layout XL Environment Variables

suppressExpansion

```
ab suppressExpansion 'int integer
```

Description

Specifies in 1000s the number of violations which prevents a node from being expanded in the Annotation Browser assistant. If the number of entries under a particular node exceeds the limit, the node is not expanded by default.

The default is 1, which means that any node with more than 1000 violations under it is not expanded by default. The value you enter must lie in the range 0 through 1000.

Arguments

<i>integer</i>	Non-negative integer in the range 0 through 1000. For example, to suppress the expansion of nodes containing more than 5000 violations, set this environment variable to 5.

GUI Equivalent

Command: *Windows – Assistants – Annotation Browser – Options (RMB)*

Field: *Suppress expansion at*

Examples

```
envGetVal("ab" "suppressExpansion")  
envSetVal("ab" "suppressExpansion" 'int 2)
```

Additional Information

If you select markers in the canvas and their parent category is not expandable in the browser pane, you can still use the *Go To The Next Selected Marker* and *Go To The Previous Selected Marker* buttons in the Annotation Browser toolbar to cycle through the selected markers.

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

Related Topics

[Annotation Browser](#)

[List of Layout XL Environment Variables](#)

transistorWidthParamNames

```
layoutXL transistorWidthParamNames 'string "list_of_parameters"
```

Description

Lists the names of the schematic parameters used to specify transistor width.

The default is "w".

The transistor width value must be split among the generated parallel-connected devices during the following *Connectivity* commands.

- *Generate All From Source*
- *Check Against Source*
- *Update Layout Parameters.*

Layout XL checks each schematic instance for one of the listed parameter names and updates the width values of the matching layout parameters such that

```
number of mfactored instances * mfactorSplit value = source parameter value
```

Note: This environment variable is also used by the analog placer when adding dummy devices to MOSFETs or resistors with the modgenMakeMinDummies environment variable set to t.

Arguments

list_of_parameters

A list of parameter names each separated by a space. The list must be enclosed in quotation marks; for example,

"w"

GUI Equivalent

Command: *Options – Layout XL (Parameters tab)*

Field: *Transistor width*

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

Examples

```
envGetVal("layoutXL" "transistorWidthParamNames")  
envSetVal("layoutXL" "transistorWidthParamNames" 'string "w")
```

Related Topics

[List of Layout XL Environment Variables](#)

turnInfixOffWhenSmartSnapping

```
layoutXL turnInfixOffWhenSmartSnapping 'boolean { t | nil }
```

Description

Switches the Infix mode OFF when creating a ruler using the smart snapping options or when creating a wire.

The default is `t`.

Arguments

None.

Examples

```
envGetVal("layoutXL" "turnInfixOffWhenSmartSnapping")  
envSetVal("layoutXL" "turnInfixOffWhenSmartSnapping" 'boolean t)  
envSetVal("layoutXL" "turnInfixOffWhenSmartSnapping" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

updateLayoutParameters

```
layoutXL updateLayoutParameters 'boolean { t | nil }
```

Description

Specifies that the *Update Components And Nets* command updates the parameters and parameter values on layout instances to match those on their schematic counterparts.

Note: Parameters that are set in layout instances but are not present on their schematic counterparts are not removed.

The default is `nil`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Update – Components And Nets (Update tab)*

Field: *Update Layout Parameters*

Examples

```
envGetVal("layoutXL" "updateLayoutParameters")
envSetVal("layoutXL" "updateLayoutParameters" 'boolean t)
envSetVal("layoutXL" "updateLayoutParameters" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

updateNetSigType

```
layoutXL updateNetSigType 'boolean { t | nil }
```

Description

Specifies that the signal types assigned to nets in the schematic are transferred to the layout view by the *Update Components And Nets* command.

The default is `t`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Update – Components And Nets (Update tab)*

Field: *Update Net Signal Type*

Examples

```
envGetVal("layoutXL" "updateNetSigType")  
envSetVal("layoutXL" "updateNetSigType" 'boolean t)  
envSetVal("layoutXL" "updateNetSigType" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

updateNetsOnly

```
layoutXL updateNetsOnly 'boolean { t | nil }
```

Description

Specifies that only net assignments and instance, terminal, and net names are updated by the *Update Components And Nets* command.

Note: User-defined mappings are also preserved when this environment variable is enabled.

The default is `nil`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Update – Components And Nets (Update tab)*

Field: *Update Nets and Instance Name Mismatches Only*

Examples

```
envGetVal("layoutXL" "updateNetsOnly")  
envSetVal("layoutXL" "updateNetsOnly" 'boolean t)  
envSetVal("layoutXL" "updateNetsOnly" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

updateOneToOneMappings

```
layoutXL updateOneToOneMappings 'boolean { t | nil }
```

Description

Updates and reports any connectivity and parameter differences for user-defined one-to-one mappings of devices between the schematic and the layout.

The default is `t`.

Arguments

None.

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "updateOneToOneMappings")
envSetVal("layoutXL" "updateOneToOneMappings" 'boolean t)
envSetVal("layoutXL" "updateOneToOneMappings" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

updatePlacementStatus

```
layoutXL updatePlacementStatus 'boolean { t | nil }
```

Description

Automatically updates the placement status of instances and pins that are moved into or out of the PR boundary, including situations where the PR boundary is moved or stretched to enclose or exclude an instance or pin.

Instances and pins with status `unknown` that are wholly inside the PR boundary after the edit are updated to status `placed`. Instances and pins with status `placed` that are wholly outside the PR boundary after the edit are updated to `unknown`.

The default is `nil`.

Arguments

None.

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "updatePlacementStatus")
envSetVal("layoutXL" "updatePlacementStatus" 'boolean t)
envSetVal("layoutXL" "updatePlacementStatus" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

updatePlacementStatusInBoundary

```
layoutXL updatePlacementStatusInBoundary 'cyclic {"placed" | "firm" | "locked"}
```

Description

Sets the placement status of instances and pins that are moved into the prBoundary to the value set for `updatePlacementStatusInBoundary` unless the current status of the instances and pins is a higher constrained status.

Note:

- In terms of the constraint status, locked > firm > placed.
- The environment variable `updatePlacementStatusInBoundary` is only used if `updatePlacementStatus = t`.

The default is `placed`.

Arguments

None.

GUI Equivalent

None.

Examples

```
envGetVal("layoutXL" "updatePlacementStatusInBoundary")
envSetVal("layoutXL" "updatePlacementStatusInBoundary" 'cyclic "placed")
envSetVal("layoutXL" "updatePlacementStatusInBoundary" 'cyclic "firm")
envSetVal("layoutXL" "updatePlacementStatusInBoundary" 'cyclic "locked")
```

Related Topics

[List of Layout XL Environment Variables](#)

updateReplacesMasters

```
layoutXL updateReplacesMasters 'boolean { t | nil }
```

Description

Updates any existing instances that use an incorrect master with instances that use the correct master during the *Update Components And Nets* command.

When set to `t`,

- With `updateWithMarkers` also set to `t`, the system puts a marker on the incorrect instance in the layout canvas and renames it `name_old`. It then creates a new instance with the correct master and places it below the design boundary.
- With `updateWithMarkers` set to `nil`, the system updates the instance in place to use the correct master.

When set to `nil`, the binding to the incorrect master is accepted with a warning message in the log file.

The default is `t`.

Arguments

None.

GUI Equivalent

Command:	<i>Connectivity – Update – Components And Nets (Update tab)</i>
Field:	<i>Update Instance Masters by (Replacing the Old or Creating a New) Instance</i>

Examples

```
envGetVal("layoutXL" "updateReplacesMasters")
envSetVal("layoutXL" "updateReplacesMasters" 'boolean t)
envSetVal("layoutXL" "updateReplacesMasters" 'boolean nil)
```

Related Topics

[List of Layout XL Environment Variables](#)

updateSelectedComponents

layoutXL updateSelectedComponents 'boolean { t | nil }

Description

Specifies that the *Update Components And Nets* command updates only the instances and pins currently selected in the layout window.

The default is `nil`.

Note: This environment variable automatically disables the generation of new components. However, if [updateWithMarkers](#) is set to `t`, Layout XL may generate new instances to replace existing instances with incorrect masters; you have the option to switch on automatic chaining and folding for these new instances.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Update – Components And Nets (Update tab)*

Field: *Update Selected Layout Components Only*

Examples

```
envGetVal("layoutXL" "updateSelectedComponents")
envSetVal("layoutXL" "updateSelectedComponents" 'boolean t)
envSetVal("layoutXL" "updateSelectedComponents" 'boolean nil)
```

Related Topics

[Updating Selected Layout Components](#)

[List of Layout XL Environment Variables](#)

updateWithMarkers

```
layoutXL updateWithMarkers 'boolean { t | nil }
```

Description

Specifies how the *Update Components And Nets* command updates an instance with an incorrect master.

- The default is `nil`, which means the command automatically removes the incorrect instance and replaces it with an instance of the correct master in the same location.
- When set to `t`, the command puts a marker on the instance with an incorrect master and renames it `name_old` during the *Update Components And Nets* command. It then creates a new instance with the correct master and places it below the PR boundary.



This environment variable is honored only when updateReplacesMasters is set to `t`.

Arguments

None.

GUI Equivalent

Command: *Connectivity – Update – Components And Nets (Update tab)*

Field: *Update Instance Masters by (Replacing the Old or Creating a New) Instance*

Examples

```
envGetVal("layoutXL" "updateWithMarkers")  
envSetVal("layoutXL" "updateWithMarkers" 'boolean t)  
envSetVal("layoutXL" "updateWithMarkers" 'boolean nil)
```

Related Topics

List of Layout XL Environment Variables

viewList

```
layoutXL viewList 'string "list_of_viewNames"
```

Description

Specifies the default value for the *Logical switch view list* field when a new physical configuration view is created.

The logical switch view list specifies the design views that are used to descend into a hierarchical design to find stop views (see [stopList](#)).

The default is "schematic netlist symbol layout compacted symbolic".



This environment variable is used only when creating a new physical configuration view. Changing the value has no effect on an existing physical configuration view.

Arguments

list_of_viewNames

A list of view names each separated by a space. The list must be enclosed in quotation marks; for example,

```
"schematic netlist symbol layout compacted symbolic"
```

GUI Equivalent

Command: *Launch – Configure Physical Hierarchy*

Field: *Logical switch view list* (in the *Global Bindings* pane)

Examples

```
envGetVal("layoutXL" "viewList")  
envSetVal("layoutXL" "viewList" 'schematic netlist symbol layout compacted  
symbolic')
```

Related Topics

List of Layout XL Environment Variables

GUI Options and Corresponding Environment Variables

Form	Option	Environment Variable
A		
<u>Annotation Browser Options</u>	<i>Cycle</i>	<u>incNetHiliteLayer</u>
	<i>Default open in</i>	<u>openLocation</u>
	<i>Default open mode</i>	<u>openMode</u>
	<i>Remove highlighting when browser is closed</i>	<u>hideMarkersWhenBrowserHidden</u>
	<i>Hierarchy depth</i>	<u>hierarchyDepth</u>
	<i>Highlighted markers can be selected in canvas</i>	<u>highlightedIsSelectable</u>
	<i>Suppress expansion at</i>	<u>suppressExpansion</u>
C		
<u>Check Against Source</u>	<i>Do not show me this dialog again</i>	<u>disableCASOptionsPopUp</u>
<u>Configure Physical Hierarchy</u>	<i>Logical switch view list</i>	<u>viewList</u>
	<i>Physical library list</i>	<u>lxUseLibList</u>
	<i>Physical stop view list</i>	<u>stopList</u>
<u>Create Path</u>	<i>Probe nets</i>	<u>probeDuringCreate</u>
<u>Create Polygon</u>	<i>Probe nets</i>	<u>probeDuringCreate</u>
<u>Create Rectangle</u>	<i>Probe nets</i>	<u>probeDuringCreate</u>
<u>Create Wire</u>	<i>Probe nets</i>	<u>probeDuringCreate</u>
G		
<u>Generate Clones</u>	<i>Draglines</i>	<u>flightLineEnable</u>
	<i>Exact Parameters</i>	<u>cloningDoExactMatch</u>

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

Form	Option	Environment Variable
	<i>Update Layout Parameters to Match Schematic</i>	<u>cloningAutomaticUpdateLayoutPara meters</u>

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

Form	Option	Environment Variable
<u>Generate Layout</u>		<u>initIOPinName</u>
	<i>Aspect Ratio W/H</i>	<u>initAspectRatio</u>
	<i>Boundary Area Estimation</i>	<u>initEstimateArea</u>
	<i>Boundary Height</i>	<u>initPrBoundaryH</u>
	<i>Boundary Width</i>	<u>initPrBoundaryW</u>
	<i>Create</i>	<u>initCreatePins</u>
	<i>Height</i>	<u>initPinHeight</u>
	<i>Except Global Pins</i>	<u>initGlobalNetPins</u>
	<i>Instances</i>	<u>initCreateInstances</u>
	<i>Layer</i>	<u>initIOPinLayer</u>
	<i>Num</i>	<u>initPinMultiplicity</u>
	<i>Create Label As</i>	<u>initIOLabelType</u>
	<i>Pin Width</i>	<u>initPinWidth</u>
	<i>PR Boundary</i>	<u>initCreateBoundary</u>
	<i>Preserve Floorplanning Objects – Area Boundaries</i>	<u>preserveAreaBoundary</u>
	<i>Preserve Floorplanning Objects – Blockages</i>	<u>preserveBlockages</u>
	<i>Preserve Floorplanning Objects – Clusters</i>	<u>preserveClusters</u>
	<i>Preserve Floorplanning Objects – Cluster Boundaries</i>	<u>preserveClusterBoundaries</u>
	<i>Preserve Floorplanning Objects – Rows and custom Placement Areas</i>	<u>preserveRows</u>
	<i>Preserve Floorplanning Objects – Track Patterns</i>	<u>preserveTrackPattern</u>
	<i>Preserve User-Defined Mappings</i>	<u>initCreateMTM</u>

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

Form	Option	Environment Variable
<u>Generate Selected Components</u>	<i>Draglines</i>	<u>flightLineEnable</u>
<u>Generate Chained Devices</u>	<i>Extend Selection</i>	<u>chainExtendSelection</u>
	<i>Preserve Existing Chains</i>	<u>chainPreserveExistingChains</u>
	<i>Use Device Order</i>	<u>chainUseDeviceOrder</u>
<u>Generate Folded Devices</u>	<i>Chain Folds</i>	<u>chainFolds</u>
L		
<u>Layout XL Options – General Tab</u>	<i>Constraint-aware editing</i>	<u>constraintAwareEditing</u>
	<i>Open connectivity reference</i>	<u>openConnRef</u>
	<i>Open connectivity reference in</i>	<u>lxSchematicDefaultApp</u>
<u>Layout XL Options – Display Tab</u>	<i>Auto arrange windows</i>	<u>autoArrange</u>
	<i>Cross-selection</i>	<u>crossSelect</u>
	<i>Hide global nets</i>	<u>hideDraglinesForGlobalNets</u>
	<i>Probe nets during object creation</i>	<u>probeDuringCreate</u>
	<i>Show distant connections</i>	<u>showDraglinesForDistantConns</u>
	<i>Use colors</i>	<u>colorDraglines</u>
	<i>Redirect Layout XL messages to separate Info window</i>	<u>infoWindow</u>
<u>Layout XL Options – Extraction Tab</u>	<i>Spaced-based Router – Extract</i>	<u>deviceExtractType</u>

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

Form	Option	Environment Variable
Layout XL Options – <u>Generation Tab</u>	<i>Abut transistors</i>	<u>autoAbutment</u>
	<i>Generate minimal folding</i>	<u>lxStackMinimalFolding</u>
	<i>Mirror transistors</i>	<u>autoMirror</u>
	<i>Pcell fingering parameter names</i>	<u>lxFingeringNames</u>
	<i>Retain instance orientation</i>	<u>lxRetainFoldOrient</u>
	<i>Space components</i>	<u>autoSpace</u>
	<i>Split mfactored devices</i>	<u>mfactorSplit</u>
	<i>Update layout instances</i>	<u>updateReplacesMasters</u>
	<i>Update width parameter method</i>	<u>lxUpdateFoldedWidth</u>
	<i>Create implicit bus terminals</i>	<u>createImplicitBusTerminals</u>

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

Form	Option	Environment Variable
<u>Layout XL Options – Parameters Tab</u>	<i>Capacitance</i>	<u>capacitanceParamNames</u>
	<i>Consider parameters only and Consider parameters and properties</i>	<u>checkParamsOnly</u>
	<i>Ignore missing parameters or properties</i>	<u>checkMissingParamsOrProps</u>
	<i>Inductance</i>	<u>inductanceParamNames</u>
	<i>Parallel-connected factor</i>	<u>mfactorNames</u>
	<i>Parameters to ignore during check</i>	<u>paramsToIgnoreForCheck</u>
	<i>Parameters to ignore during generation</i>	<u>paramsToIgnore</u>
	<i>Properties used to ignore objects during check</i>	<u>propsUsedToIgnoreObjsForCheck</u>
	<i>Properties used to ignore objects during generation</i>	<u>propsUsedToIgnoreObjs</u>
	<i>Resistance</i>	<u>resistanceParamNames</u>
	<i>Series-connected factor</i>	<u>sfactorNames</u>
	<i>Tolerance</i>	<u>paramTolerance</u>
	<i>Transistor width</i>	<u>transistorWidthParamNames</u>
M		
<u>Move</u>	<i>Display Draglines</i>	<u>flightLineEnable</u>
P		
<u>Highlight Options</u>	<i>Display Layer</i>	<u>probeHiliteLayer</u>
	<i>Display Layer – Cycle</i>	<u>probeCycleHilite</u>

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

Form	Option	Environment Variable
S		
<u>Set Pin Label Text Style</u>	<i>Layer Name – Same As Pin</i>	<u>pinTextSameLayer</u>
	<i>Layer Purpose – Same As Pin</i>	<u>pinTextSamePurpose</u>
<u>Stretch</u>	<i>Display Draglines</i>	<u>flightLineEnable</u>
U		
<u>Update Components and Nets – Update tab</u>	<i>Delete Unmatched Instances</i>	<u>deleteUnmatchedInsts</u>
	<i>Delete Unmatched Pins</i>	<u>deleteUnmatchedPins</u>
X		
<u>XL Probe</u>	<i>Instances</i>	<u>probeDevice</u>
	<i>Nets</i>	<u>probeNet</u>
	<i>Pins</i>	<u>probePin</u>
	<i>Send Messages to CIW</i>	<u>probeInfoInCIW</u>

Hidden Environment Variables with No Corresponding GUI Option

Environment Variable	Comments
<u>autoMirrorChains</u>	Advanced option. Cadence recommends that you do not change the value of this option from its default value.
<u>checkStateConfirmModeChange</u>	Advanced option which suppresses system prompts when re-opening cellviews from the Annotation Browser.
<u>ciwWindow</u>	Defines the size and position of the CIW window on the desktop. This is most easily set by resizing the window in the graphical user interface and saving the session.
<u>cphStopLogicalElabAtPhysLeaf</u>	Advanced option required only for specific design types. See <u>Global or Inherited Nets Beyond Physical Leaf Cells are not Detected</u> .
<u>createBoundaryLabel</u>	
<u>deleteConfirmModeChange</u>	Advanced option which suppresses system prompts when using the Annotation Browser's <i>Delete</i> function.
<u>drdUseNetName</u>	
<u>layoutWindow</u>	Defines the size and position of the layout window on the desktop. This is most easily set by resizing the window in the graphical user interface and saving the session.
<u>lswWindow</u>	Defines the size and position of the layer selection window on the desktop. This is most easily set by resizing the window in the graphical user interface and saving the session.
<u>lxDeltaWidth</u>	Advanced folding option in limited use.
<u>lxGenerationTopLevelOnly</u>	Advanced option in limited use. Cadence recommends that you use Configure Physical Hierarchy instead.

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

<u>lxInitResetSource</u>	Advanced option which prompts you to set the connectivity source every time you launch Layout XL.
<u>lxSchExtractTopLevelOnly</u>	Advanced option used when you do not have permission to write to the schematic view. Cadence recommends you do not use this option because there is no guarantee that the generated layout will be correct.
<u>lxWidthTolerance</u>	Advanced folding option in limited use.
<u>moveAsGroup</u>	
<u>schematicWindow</u>	Defines the size and position of the schematic window on the desktop. This is most easily set by resizing the window in the graphical user interface and saving the session.

Virtuoso Layout Suite XL User Guide

Layout XL Environment Variables

Command Quick Reference

Listed below are all the commands you can use in the Virtuoso® Layout Suite XL layout editor (Layout XL) with a brief summary of what each command does and a link to more complete documentation of the command.

Command	Function	Documented
Launch Menu		
<i>Configure Physical Hierarchy</i>	Lets you define how the physical implementation is generated from the schematic, define and edit component types, and define and edit soft blocks.	<u>Configuring the Physical Hierarchy</u>
File Menu		
<i>Load Physical View</i>	Loads an OpenAccess template cellview containing pin, boundary, and placement information into the current cellview.	<u>Load Physical View</u>
Edit Menu		
<i>Advanced – Swap Components</i>	Exchanges the position of two selected components.	<u>Swapping Components</u>
<i>Soft Blocks</i>	Lets you modify the attributes of individual soft blocks in your design.	<u>Editing Soft Blocks</u>

Virtuoso Layout Suite XL User Guide

Command Quick Reference

Command	Function	Documented
<i>Component Types</i>	Controls how components are placed in rows and defines their chaining and folding parameters.	<u>Component Types Mode</u>
Connectivity Menu		
<i>Pins – Permute</i>	Switches the position of pins set up for permutation within an instance.	<u>Pin Permutation</u>
<i>Nets – Assign</i>	Assigns instance pins with or without a connectivity source or schematic to nets in the layout.	<u>Assigning Instance Pins to a Net</u>
<i>Nets – Edit Attributes</i>	Lets you modify the criticality and signal type of existing nets.	<u>Editing Net Attributes and Properties</u>
<i>Nets – Set Order</i>	Lets you create an ordered net constraint on a group of nets.	<u>Creating and Editing Net Constraints</u>
<i>Nets – Show/Hide All Incomplete Nets</i>	Shows and hides all the incomplete nets in the design.	<u>Showing and Hiding All Incomplete Nets</u>
<i>Nets – Show/Hide Selected Incomplete Nets</i>	Shows and hides any incomplete nets associated with selected objects in the specified layout window.	<u>Showing and Hiding Selected Incomplete Nets</u>
<i>Nets – Show/Hide Current Incomplete Nets</i>	Toggles the visibility of the current set of incomplete nets displayed in the design canvas.	<u>Showing and Hiding Current Incomplete Nets</u>
<i>XL Probe</i>	Highlights the design element in the layout or schematic when you click on the corresponding one in the other window.	<u>Probing</u>

Virtuoso Layout Suite XL User Guide

Command Quick Reference

Command	Function	Documented
<i>Generate – All From Source</i>	Creates and places layout cells below the boundary in the layout canvas.	Generating All Components from Source
<i>Generate – Selected From Source</i>	Creates and places interactively selected schematic elements in the layout.	Generating Selected Components from Source
<i>Generate – Place As In Schematic</i>	Moves components generated in the layout into the design boundary, maintaining the relative positions of the components from the schematic.	Moving Generated Components into the Design Boundary
<i>Generate – Clones</i>	Copies parts of schematic or layout including connectivity info to use as other parts.	Generating Clones
<i>Generate – Chained Devices</i>	Abuts prepared MOS transistors into chains.	Manual Device Abutment Automatic Device Abutment
<i>Generate – Folded Devices</i>	Divides prepared MOS transistors or transistor chains into individual folds (smaller devices connected in parallel).	Folding a Transistor
<i>Check – Against Source</i>	Reports the devices and parameters in the connectivity source that are missing in the schematic.	Check Against Source
<i>Update – Components And Nets</i>	Updates the layout to add devices, pins, or connectivity changes added to the schematic.	Updating Components and Nets
<i>Update – Layout Constraints</i>	Transfers constraints from the schematic view to the layout view.	Updating Layout Constraints

Virtuoso Layout Suite XL User Guide

Command Quick Reference

Command	Function	Documented
<i>Update – Layout Parameters</i>	Updates parameters of instances in the layout to match those in the schematic.	Updating Layout Parameters
<i>Update – Schematic Parameters</i>	Updates parameters of devices in the schematic to match those in the layout.	Updating Schematic Parameters
<i>Update – Connectivity Reference</i>	Updates the connectivity source and physical configuration for a given layout view.	Updating the Connectivity Reference
<i>Update – Clone Families</i>	Lets you manage the synchronized clone families in your design.	Updating Clone Families
<i>Update – Extract Layout</i>	Verifies the connectivity of the layout design and generates markers to highlight open and short circuits and illegal overlaps.	Connectivity Extraction
<i>Define Device Correspondence</i>	Creates user-defined device correspondence between selected devices in the schematic and layout views; allows implementation of one or many to many or one mappings.	Updating Device Correspondence
Options Menu		
<i>Layout XL</i>	Sets values of Layout XL environment variables.	Getting Started with Layout XL
Place Menu		
<i>Pin Placement</i>	Sets constraints on and places the pins in your design.	Pin Placement

Virtuoso Layout Suite XL User Guide

Command Quick Reference

Command	Function	Documented
<i>Update Placement Status</i>	Updates the placement status of instances and pins in the current cellview.	<u>Updating the Placement Status Automatically</u>
<i>Custom Digital – Placement Planning</i>	Lets you create, change, and delete rows	<u>Placement Planning</u>
<i>Custom Digital – Placer</i>	Runs the automatic placer	<u>Auto Placer</u>
<i>Analog</i>	Provides a set of analog placement options	<u>Virtuoso Analog Placer User Guide</u>

Virtuoso Layout Suite XL User Guide

Command Quick Reference

Layout XL Properties

This section provides property names, descriptions, and graphical user interface (GUI) equivalents for the Virtuoso[®] Layout Suite XL layout editor (Layout XL).

Note: Only the properties documented in this chapter are supported for public use. All other Layout XL properties, regardless of their name or prefix, and undocumented aspects of the properties described below, are private and are subject to change at any time.

[abutAccessDir](#) on page 813

[abutClass](#) on page 814

[abutCondInclusion](#) on page 816

[abutFunction](#) on page 817

[abutGateNet](#) on page 820

[abutMosStretchMat](#) on page 821

[abutStretchMat](#) on page 825

[abutOffset](#) on page 827

[ignore](#) on page 829

[lvslgnore](#) on page 830

[lxAutoAbut](#) on page 832

[lxAutoSpace](#) on page 833

[lxBlockExtractCheck](#) on page 834

[lxBlockOverlapCheck](#) on page 836

[lxCombination](#) on page 838

Virtuoso Layout Suite XL User Guide

Layout XL Properties

[lxMfactorSplit](#) on page 841

[lxNetNamePrefix](#) on page 842

[lxParamsTolgnore](#) on page 845

[lxParamsTolgnoreForCheck](#) on page 847

[lxRemoveDevice](#) on page 848

[lxRounding](#) on page 849

[lxSeriesTerms](#) on page 850

[lxStickyNet](#) on page 851

[lxStopList](#) on page 853

[lxUseCell](#) on page 854

[lxViewList](#) on page 855

[mfactor](#) on page 856

[permuteRule](#) on page 859

[sfactor](#) on page 862

[vxllnstSpacingDir](#) on page 865

[vxllnstSpacingRule](#) on page 868

Setting Layout XL Properties

You can set some of the properties described in this chapter in the graphical user interface. This section tells you how to access the appropriate form.

Accessing the Symbol Master Add Property Form

To add a user property to a symbol master,

1. From the Virtuoso Symbol Editing window, choose *Edit – Properties – Cellview*.

The Edit Cellview Properties form appears.

2. In the *User Property* section, click *Add*.

The Add Property form appears.

Accessing the Add CDF Parameter Form

To add a CDF parameter to a layout or schematic master,

1. From the CIW, choose *Tools – CDF – Edit*.

The Edit Component CDF form is displayed.

2. Specify the library and cell names and click *Apply*.

The form is populated with the options you can set.

3. In the *Component Parameters* section, click *Add*.

The Add CDF Parameter form is displayed.

Accessing the Layout Instance Add Property Form

To add a user property to an instance in the layout,

1. In the layout canvas, select the instance to which you want to add a property.

2. From the menu bar, choose *Edit – Basic – Properties*.

The Edit Instance Properties form is displayed.

3. Choose the *Property* radio button and click *Add*.

The Add Property form is displayed.

Related Topics

[Layout XL Properties](#)

abutAccessDir

`abutAccessDir list (left right top bottom)`

Defines the direction in which pins are able to abut, ensuring that the correct edges are abutting; for example, a left side to a right side, or a bottom to a top.

Cells with rail pins need to add access directions for both sides because there is one pin for both right and left sides of the cell. MOS cells have just one pin per side, so they need only one direction.

Examples

```
dbReplaceProp(objId "abutAccessDir" "list" list("left"))
dbReplaceProp(objId "abutAccessDir" "list" list("left right"))
```

Related Topics

[abutClass](#)

[abutCondInclusion](#)

[abutFunction](#)

[abutGateNet](#)

[abutMosStretchMat](#)

[abutStretchMat](#)

[abutOffset](#)

[Device Abutment](#)

[Creating CMOS Pcells to Use with Abutment](#)

[Setting Layout XL Properties](#)

[Layout XL Properties](#)

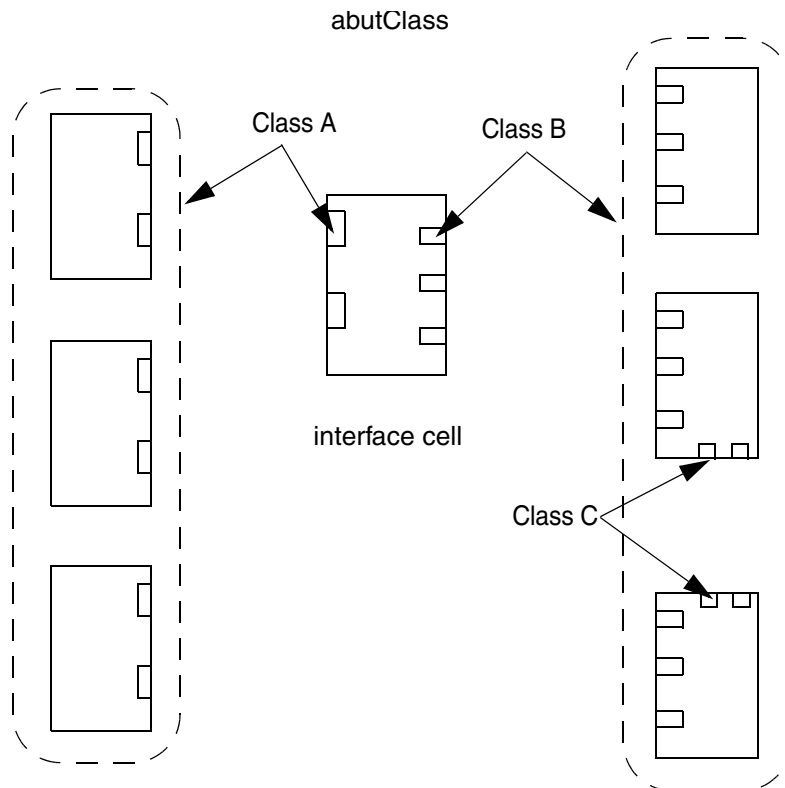
abutClass

`abutClass` string *t_user_defined_name*

Specifies that two cells can abut even if their master cellviews are not the same.

You can place this property on an individual pin or on a cellview in order to implement a global abutment class for all the pins in the cellview. The abutment function checks for the property first on the pin and then on the cellview.

When you place this property on a pin, remember that each pin targeted for abutment can belong to only one abutment class. Pins on different edges of a cell can belong to different classes.



Example

```
dbReplaceProp(leftObj~>dbId "abutClass" "string" d_layer)
```

Related Topics

[abutAccessDir](#)

Virtuoso Layout Suite XL User Guide

Layout XL Properties

[abutCondInclusion](#)

[abutFunction](#)

[abutGateNet](#)

[abutMosStretchMat](#)

[abutStretchMat](#)

[abutOffset](#)

[Device Abutment](#)

[Creating CMOS Pcells to Use with Abutment](#)

[Setting Layout XL Properties](#)

[Layout XL Properties](#)

abutCondInclusion

`abutCondInclusion string t_pcell_param_name`

Defines the name of the boolean property that adds or removes the source/drain contact.

You can conditionally include or exclude the contacts and the connecting metal tab. You must conditionally exclude these features if you are creating a graphical pcell. The pcell parameter must remove all features that are required for metal hookup and leave only the diffusion material to be stretched. For CMOS device abutment, you add this property to both drain and source sides of the device.

Example

```
dbReplaceProp(rightObj~>dbId "abutCondInclusion" "string" "rightCnt")
```

Related Topics

[abutAccessDir](#)

[abutClass](#)

[abutFunction](#)

[abutGateNet](#)

[abutMosStretchMat](#)

[abutStretchMat](#)

[abutOffset](#)

[Device Abutment](#)

[Creating CMOS Pcells to Use with Abutment](#)

[Setting Layout XL Properties](#)

[Layout XL Properties](#)

abutFunction

`abutFunction` string *t_function_name*

Specifies a user-defined callback to process abutment and unabutment.

The value of this property is the name of a user-defined Cadence® SKILL function to be executed before abutment takes place. Auto-abutment passes this function the following eight arguments.

1. The database ID of the cell to be abutted
 2. The database ID of the cell to be abutted to
 3. The database ID of the pin figure to be abutted
 4. The database ID of the pin to be abutted to
- Note:** If an instance is deleted, the arguments 3 and 4 above are set to `nil`.
5. An integer to indicate the **abutment access direction** of the pin being abutted where
 - ☐ 1 = top
 - ☐ 2 = bottom
 - ☐ 4 = left
 - ☐ 8 = right
 6. An integer to indicate the **connection condition** where
 - ☐ 1 = pins are connected to the same net and do not connect to any other pin
 - ☐ 2 = pins are connected to the same net and the net connects to other pins
 - ☐ 3 = pins are connected to different nets. Used for nets that are not connected, so that structures like guard rings and wells can be adjusted appropriately.
 7. An integer specifying the **auto-abutment event** where
 - ☐ 1 = `abutFunction` must compute and return abutment offset (in the direction of abutment)
 - ☐ 2 = `abutFunction` must adjust pcell parameters for abutment. This is called before the offset event.
 - ☐ 3 = `abutFunction` must adjust pcell parameters for unabutment
 - ☐ 4 = `abutFunction` must compute and return spacing offset for non abutment.

8. A database ID that is the **abutment group pointer** available to events 2 and 3. It is used to store information so that unabutment can return pins to their original state. It is a generally-accepted method to use the group attribute of a cell to hold original parameter values prior to modification by abutment.

`abutFunction` returns different values for different auto-abutment events.

- If the event is 1, it returns a floating-point number representing the offset needed to abut the cells in the direction of abutment.

If it returns two floating-point numbers, the second number is the distance in user units that the abutting cell is moved in the direction perpendicular to the abutting edge.

- If the event is 2, it must return a value of `true` to allow abutment and `nil` to block abutment. When it returns `nil`, it must not change any parameters in any way, including setting them to new values and then resetting them to the original values.
- If the event is 3, it can return anything; the value returned is not used.

In the case of non-abutment and if `vxllInstSpacingDir` is set to the string “`abutFunction`”, the abutment function is called first with event 2 and connection condition 3, and then with event 4.

Below is an example of an `abutFunction`.

```
procedure( abutFunction(iA iB pA pB pASide connection event @optional
  (group nil)) prog((result)
  case(event
    (1 ; Compute offset
      result = getAbutmentOffset(iA iB pA pB pASide connection)
    )
    (2 ; Adjust pcell parameters
      result = setAbutmentParams(iA iB pA pB pASide connection
        group)
    )
    (3 ; Adjust pcell parameters back to default
      resetAbutmentParams(group iA iB)
      result = t
    )
    (4 ; To get the spacing offset when pins are on different nets
      if(instA~>w > 3.0 && instB~>w > 3.0
        then
          result = 3
        else
          result = 1
      )
  )
  );; end of case 4
  (t ; Anything else return a nil
    result = nil
  )
  )
  return(result)
)
```

)

Note: If you are using pcells, you can use the default abutment functions.

If each pin being abutted has a different `abutFunction` defined, auto-abutment calls only the function defined by the pin in the cell being moved.

Example

```
dbReplaceProp(obj "abutFunction" "string" "mosAbutFunc")
```

Related Topics

[abutAccessDir](#)

[abutClass](#)

[abutCondInclusion](#)

[abutGateNet](#)

[abutMosStretchMat](#)

[abutStretchMat](#)

[abutOffset](#)

[Device Abutment](#)

[Creating CMOS Pcells to Use with Abutment](#)

[Setting Layout XL Properties](#)

[Layout XL Properties](#)

abutGateNet

`abutGateNet string t_pcell_param_name`

Specifies the name of the net the gate shapes are on, so that the abutment engine knows where the edge of the gate is.

Example

The value "G" is the name of the gate net inside the pcell.

```
dbReplaceProp(leftObj~>dbId "abutGateNet" "string" "G")
```

Related Topics

[abutAccessDir](#)

[abutClass](#)

[abutCondInclusion](#)

[abutFunction](#)

[abutMosStretchMat](#)

[abutStretchMat](#)

[abutOffset](#)

[Device Abutment](#)

[Creating CMOS Pcells to Use with Abutment](#)

[Setting Layout XL Properties](#)

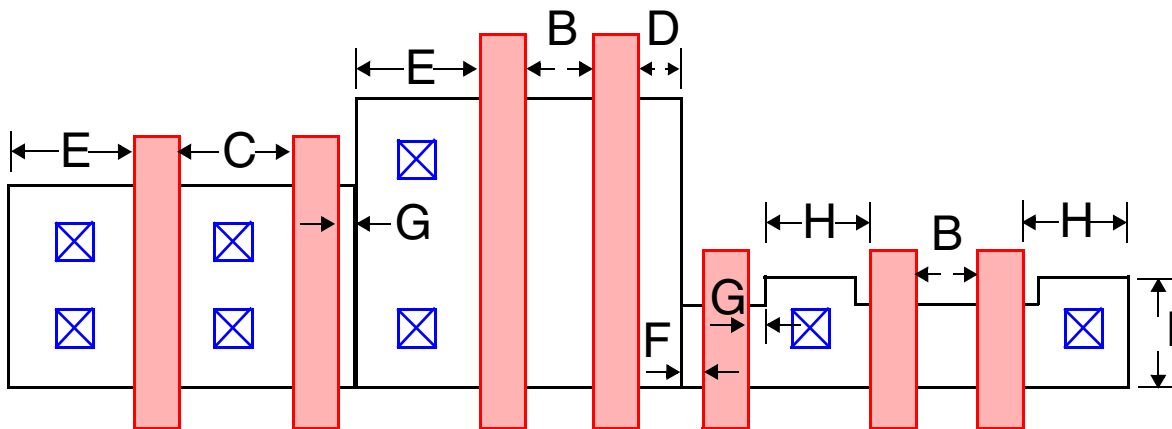
[Layout XL Properties](#)

abutMosStretchMat

```
abutMosStretchMat list abutFlush f_value
    abutGateSpaceSeries f_value
    abutGateSpaceParallel f_value
    abutEncloseGate f_value
    abutEncloseContact f_value
    abutSmallerSeries f_value
    abutSmallerParallel f_value
    abutEncloseDogBoneContact f_value
    abutMinContactWidth f_value
```

Governs how a material is stretched towards or away from a gate when an instance is altered for abutment.

The picture below illustrates the MOS diffusion extension rules for the different `abutMosStretchMat` property values.



A = `abutFlush` (Not shown). Applied to the pin which does not move when abutting two pins of equal width.

B = `abutGateSpaceSeries`. Applied to the moving pin when abutting two pins of equal width with no contact required between the gates. The `abutFlush` rule is applied to the non-moving pin.

C = `abutGateSpaceParallel`. Applied to the moving pin when abutting pins of equal width where a contact is required between the gates and the pin widths are greater than or equal to the `abutMinContactWidth` rule. The `abutFlush` rule is applied to the non-moving pin.

D = `abutEncloseGate`. Applied to the larger of two abutting pins when a contact is not required between the gates. The `abutSmallerSeries` rule is applied to the smaller of the two pins being abutted.

E = abutEncloseContact. Applied to the larger of two different-width abutting pins when a contact is required between the gates but only if the width of the larger pin is greater than or equal to the `abutMinContactWidth` rule. The `abutSmallerParallel` rule is applied to the smaller of the two pins being abutted.

F = abutSmallerSeries. Applied to the smaller of two abutting pins when a contact is not required between the gates.

G = abutSmallerParallel. Applied to the smaller of two abutting pins when a contact is required between the gates.

H = abutEncloseDogBoneContact. Applied when contacts are required and the width of the device is less than the `abutMinContactWidth` rule. It is applied to the larger of two different-width pins or to the moving pin when the two pins are of equal width.

I = abutMinContactWidth. Determines whether to use the `abutEncloseContact` rule or the `abutEncloseDogBoneContact` rule when a contact is required between gates.

Example

This example illustrates the properties being applied to pins on the left and right edges of a MOS pcell.

- The pin rectangles were created by the `rodCreateRect` command and are referenced by two variables called `leftObj` and `rightObj` pointing to their rod object IDs.
- The pcell parameters that control the stretching of the left and right pin extensions past the gate are called `leftSt` and `rightSt`.
- All abutment extension rule values in the `abutMosStretchMat` property list must be numeric.
- The pcell parameters that control whether or not the left or right pins have contacts over them are `leftCnt` and `rightCnt`.
- The contact parameters must be of type boolean.

```
;; Definition of variables used in the abutRuleList calculations for
;; the abutMosStretchMat property.

;; cw = contact width
;; doc = diffusion overlap of contact
;; pps = poly to poly space
;; pcs = poly to contact space
;; pds = poly to diffusion space
```

Virtuoso Layout Suite XL User Guide

Layout XL Properties

```
abutRuleList = list(  
  list("abutFlush" 0.0)  
  list("abutGateSpaceSeries" pps)  
  list("abutGateSpaceParallel" max((cw + (pcs * 2)) pps))  
  list("abutEncloseGate" doc)  
  list("abutEncloseContact" doc + cw + pcs)  
  list("abutSmallerSeries" max((pps - doc) pds))  
  list("abutSmallerParallel" max((pcs - doc) (pps - pcs - cw - doc) pds))  
  list("abutEncloseDogBoneContact" cw + doc + max(pds+doc pcs))  
  list("abutMinContactWidth" cw + (2 * doc))  
)  
  
;; Definition of variables used for abutClass and autoSpacer property values.  
;; d layer = diffusion layer (used for abutClass property value)  
;; dds = diffusion to diffusion space (used by autoSpacer)  
  
;; Add the abutment and spacer properties to the left diffusion pin.  
  
leftObj~>dbId~>pin~>name = "leftContact"  
dbReplaceProp(leftObj~>dbId "abutCondInclusion" "string" "leftCnt")  
dbReplaceProp(leftObj~>dbId "abutMosStretchMat" "list"  
  cons("leftSt" abutRuleList))  
dbReplaceProp(leftObj~>dbId "abutClass" "string" d_layer)  
dbReplaceProp(leftObj~>dbId "abutGateNet" "string" "G")  
dbReplaceProp(leftObj~>dbId "abutAccessDir" "list" list("left"))  
dbReplaceProp(leftObj~>dbId "vxlInstSpacingDir" "list" list("left"))  
dbReplaceProp(leftObj~>dbId "vxlInstSpacingRule" "float" dds)  
  
;; Add the abutment and spacer properties to the right diffusion pin.  
  
rightObj~>dbId~>pin~>name = "rightContact"  
dbReplaceProp(rightObj~>dbId "abutCondInclusion" "string" "rightCnt")  
dbReplaceProp(rightObj~>dbId "abutMosStretchMat" "list"  
  cons("rightSt" abutRuleList))  
dbReplaceProp(rightObj~>dbId "abutClass" "string" d_layer)  
dbReplaceProp(rightObj~>dbId "abutGateNet" "string" "G")  
dbReplaceProp(rightObj~>dbId "abutAccessDir" "list" list("right"))  
dbReplaceProp(rightObj~>dbId "vxlInstSpacingDir" "list" list("right"))  
dbReplaceProp(rightObj~>dbId "vxlInstSpacingRule" "float" dds)
```

Related Topics

[abutAccessDir](#)

[abutClass](#)

[abutCondInclusion](#)

[abutFunction](#)

[abutGateNet](#)

[abutStretchMat](#)

[abutOffset](#)

Virtuoso Layout Suite XL User Guide

Layout XL Properties

[Device Abutment](#)

[Creating CMOS Pcells to Use with Abutment](#)

[Setting Layout XL Properties](#)

[Layout XL Properties](#)

abutStretchMat

```
abutStretchMat list drainStretch t_material_name
    abutMinExt f_value
    abutRule1Ext f_value
    abutRule2Ext f_value
    abutContactExt f_value
```

Stretches material towards or away from a gate when an instance is altered for abutment.

The default value of the stretch parameter must be the distance from the edge of the material to the edge of the gate. Numeric values are in user units.

- `drainStretch` is the name of the stretchable material parameter. You name this parameter yourself to identify the material to be stretched.
- `abutMinExt` is the rule for minimum diffusion overhang from the edge of the poly gate. The argument to this element is the rule value.
- `abutRule1Ext` is the poly-layer-to-poly-layer rule. This rule is used when the net connecting the two instances does not share the net with any other pin.
- `abutRule2Ext` is the poly-layer-to-diffusion-layer rule. This rule is used when the transistor width is different for the two instances being abutted.



The Virtuoso custom digital placer does not take the `abutRule2Ext` into consideration, resulting in illegal overlap markers.

- `abutContactExt` is the diffusion extension value used when one of the contacts needs to be added during the abutment process. With the contacts off by default, the abutment program needs to know how close to move the selected cell to abut properly.

Example

```
abutStretchMat list("drainStretch" ilList
    list("abutMinExt" 0.5)
    list("abutRule1Ext" 0.75)
    list("abutRule2Ext" 1.0)
    list("abutContactExt" 3.0)
)
```

Related Topics

[abutAccessDir](#)

Virtuoso Layout Suite XL User Guide

Layout XL Properties

[abutClass](#)

[abutCondInclusion](#)

[abutFunction](#)

[abutGateNet](#)

[abutMosStretchMat](#)

[abutOffset](#)

[Device Abutment](#)

[Creating CMOS Pcells to Use with Abutment](#)

[Setting Layout XL Properties](#)

[Layout XL Properties](#)

abutOffset

`abutOffset float f_value`

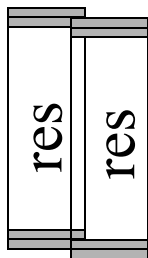
Specifies an offset based on the reference edge. The reference edge is the outside edge of the pin that triggered the abutment on each cell in the direction defined by the abutment direction.

The value number can be positive or negative and indicates a distance in user units to offset the reference edge from the outside direction of the cell. A negative number causes the pins to overlap.

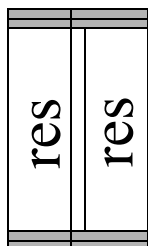
If the `abutFunction` parameter adjustment event returns `nil`, auto-abutment looks for a property on the pin called `abutOffset` and applies the value of this property for offset. If the property `abutOffset` does not exist, auto-abutment uses the outside edge of the pin with no offset.

If the pin of the device being abutted overlaps the other pin on the bottom or top, the pins are aligned by the bottom or top edges, respectively. This also applies from side to side when abutment is vertical.

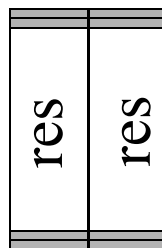
Interactive move places one cell



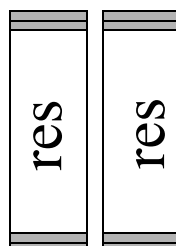
`abutOffset = -5`



Cells are first aligned and then the offset is applied.



`abutOffset = 5`



Example

`abutOffset 5`

Related Topics

[abutAccessDir](#)

[abutClass](#)

[abutCondInclusion](#)

[abutFunction](#)

[abutGateNet](#)

[abutMosStretchMat](#)

[abutStretchMat](#)

[Device Abutment](#)

[Creating CMOS Pcells to Use with Abutment](#)

[Setting Layout XL Properties](#)

[Layout XL Properties](#)

ignore

```
ignore boolean { t | nil }
```

Prevents a device symbol from being generated in the layout.



Tip

You can also set the `lvslgnore` property for layout devices if you do not want Layout XL to check them.

You can set the `ignore` property in either the symbol master Add Property form or the Add CDF Parameter form.

Symbol master Add Property form

<i>Name</i>	<code>ignore</code>
<i>Type</i>	<code>boolean</code>
<i>Value</i>	<code>t nil</code>

Add CDF Parameter form

<i>Add After Parameter</i>	<i>As First Parameter</i>
<i>paramType</i>	<code>boolean</code>
<i>name</i>	<code>ignore</code>
<i>defValue</i>	<code>t nil</code>

Related Topics

[Setting Layout XL Properties](#)

[Layout XL Properties](#)

lvIgnore

```
lvIgnore boolean { t | nil }
```

Defines an instance as physical-only, meaning that when you move the instance in the design canvas, Layout XL automatically reassigns the connectivity based on any new overlap. If there is no new overlap after the move, the existing connectivity is deleted.



Tip

You can make the connectivity of a physical-only instance sticky by setting the [lxStickyNet](#) property on the instance.

You set the `lvIgnore` property in the Add Property form.

Layout instance Add Property form

<i>Name</i>	<code>lvIgnore</code>
<i>Type</i>	<i>Boolean</i>
<i>Value</i>	<code>t nil</code>

Additional Information

By default, `lvIgnore` is included in the list of [propsUsedToIgnoreObjsForCheck](#).

Related Topics

[Instance Terminal Connectivity](#)

[Ignoring Components](#)

[Additional Pins on Ignored Instances](#)

[Updating Layout Parameters](#)

[Updating Schematic Parameters](#)

[Parameters Tab](#)

[Setting Layout XL Properties](#)

Virtuoso Layout Suite XL User Guide

Layout XL Properties

Layout XL Properties

lxAutoAbut

```
lxAutoAbut boolean { t | nil }
```

Prevents abutment if either instance in the abutment pair has this property set to `nil`.

Note: `lxAutoAbut` is a CDF-only property.

Add CDF Parameter form

<i>Add After Parameter</i>	<i>As First Parameter</i>
<i>paramType</i>	<i>boolean</i>
<i>name</i>	<code>lxAutoAbut</code>
<i>defValue</i>	<code>t nil</code>

Related Topics

[Device Abutment](#)

[Setting Layout XL Properties](#)

[Layout XL Properties](#)

lxAutoSpace

`lxAutoSpace` `boolean { t | nil }`

Prevents spacing if either instance in the abutment pair has this property set to `nil`.

Note: `lxAutoSpace` is a CDF-only property.

Add CDF Parameter form

<i>Add After Parameter</i>	<i>As First Parameter</i>
<i>paramType</i>	<i>boolean</i>
<i>name</i>	<code>lxAutoSpace</code>
<i>defValue</i>	<code>t nil</code>

Related Topics

[Device Abutment](#)

[Setting Layout XL Properties](#)

[Layout XL Properties](#)

lxBLOCKExtractCheck

```
lxBLOCKExtractCheck boolean { t | nil }
```

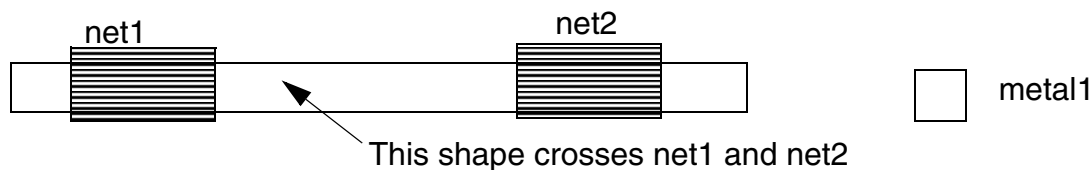
Prevents the extractor from checking the connectivity between a shape or instance and other shapes or instances it touches.

You can attach the `lxBLOCKExtractCheck` property on a shape or instance master. Use this property to prevent extraction of instances of masters that are not compatible with Layout XL or which would adversely affect the performance of the extractor.

Layout instance Add Property form

<i>Name</i>	<code>lxBLOCKExtractCheck</code>
<i>Type</i>	<i>Boolean</i>
<i>Value</i>	<code>t nil</code>

Example



In the illustration above, the layer that crosses net1 and net2 has the property `lxBLOCKExtractCheck`, so there is no connectivity checking between this shape and all others it touches. If the shape does not already have connectivity, the software assigns a net name to it.

If you assign connectivity to the shape manually or using Cadence® SKILL code, it will keep the connectivity, but the shape will always be considered incomplete because the software will not check it when checking the connectivity.

For hierarchical connectivity to work for pcells, the connectivity must be defined on the shapes within the pcell. You can assign connectivity interactively using the *Assign Nets* command or using SKILL functions.

Related Topics

[Connectivity Extraction](#)

[Setting Layout XL Properties](#)

[Layout XL Properties](#)

lxBLOCKOverlapCheck

```
lxBLOCKOverlapCheck boolean { t | nil }
```

Prevents the extractor from checking whether a nonoverlap layer of a shape, instance, or instance marker is touching a nonoverlap layer at any level in the current cellview.

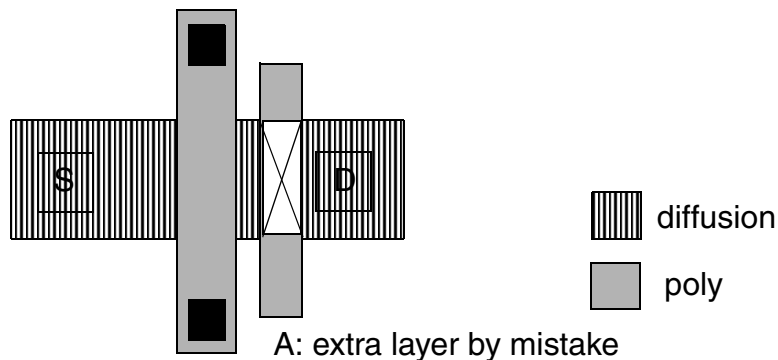
You attach the `lxBLOCKOverlapCheck` property on a shape, instance, or instance master.

Layout instance Add Property form

<i>Name</i>	<code>lxBLOCKOverlapCheck</code>
<i>Type</i>	<i>Boolean</i>
<i>Value</i>	<code>t nil</code>

Example

In the illustration below, the `lxBLOCKOverlapCheck` property can be defined on the diffusion layer so that no short marker is created.



For example, if you define the `lxBLOCKOverlapCheck` property on a shape in a cellview A, you will see no markers when overlapping the shape with an instance. However, if cellview A is placed inside cellview B, and a `noOverlap` layer is drawn over the shape in cellview A, then a marker appears. This happens even though the shape in cellview A has the `lxBLOCKOverlapCheck` property defined.

Related Topics

Connectivity Extraction

Virtuoso Layout Suite XL User Guide

Layout XL Properties

Setting Layout XL Properties

Layout XL Properties

lxCombination

`lxCombination` *string* *t_expression*

Builds a complex set of devices comprising series (*sfactor*) and parallel (*mfactor*) connections.

You typically use `lxCombination` to split a single resistor into multiple smaller devices in order to achieve maximum density in the layout by placing devices in spaces otherwise not utilized. This lets you approximate the resistance value defined in the schematic without violating the manufacturing grid.

Note: The `lxSeriesTerms` property must be defined on the instance in the schematic in order for Layout XL to understand the connectivity of devices with *sfactor* > 1 and instance *terminals* > 2.

Symbol master Add Property form

<i>Name</i>	<code>lxCombination</code>
<i>Type</i>	<i>string</i>
<i>Value</i>	<i>t_expression</i>

Add CDF Parameter form

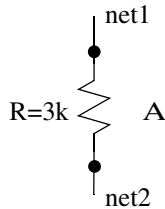
<i>Add After Parameter</i>	<i>As First Parameter</i>
<i>paramType</i>	<i>string</i>
<i>name</i>	<code>lxCombination</code>
<i>defValue</i>	<i>t_expression</i>

Example

In a schematic cellview, a resistor called `A` with resistance `3k` is connected to `net1` and `net2`, and has the following `lxCombination` expression defined.

$(2k \cdot 2) / 3 + 1500 / 4$

Schematic



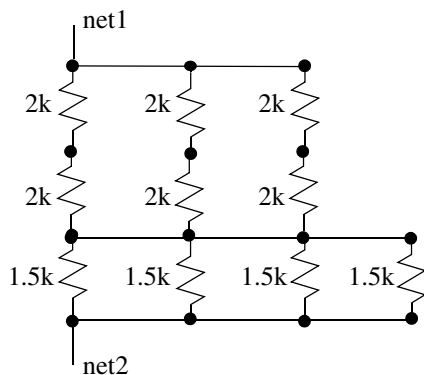
$\text{lxCombination} = (2k * 2) / 3 + 1500 / 4$

Where

- 2k, 2, 3, 1500, and 4 are device parameter values
- * and + are series connections (+ can be followed by an expression; * must be followed by an integer)
- / is a parallel connection (and must be followed by an integer)

During layout generation, Layout XL reads the `lxCombination` property, ignores the 3k value, and instead generates a total of 10 resistors: two 2k resistors connected in series, connected in parallel three times; and four parallel-connected 1.5k resistors connected in series to the 2k resistors.

Layout



Additional Information

- To ensure that `lxCombination` expressions are evaluated correctly, you must set the `lxEvalCDFCallbacks` environment variable to `t` before generating your resistors.
- You can also use `lxCombination` to define the topology for a network of capacitors. However, in this case, the property describes *only* the topology of the network; the values entered do not equate to the total capacitance of the network.
- If the `lxCombination` property is defined on the schematic symbol, the `r` parameter is ignored by the *Generate All From Source* and *Update Components And Nets* commands. The layout instances will have the correct parameters based on the `lxCombination` factor setting.
- Devices defined using `lxCombination` are named `instName.msinteger` in the layout. In our example above, if the schematic instance is called `R0`, the layout instances generated are named as follows.

Schematic	Layout
R0	R0.ms1
	R0.ms2
	R0.ms3
	R0.ms4
	R0.ms5
	R0.ms6
	R0.ms7
	R0.ms8
	R0.ms9
	R0.ms10

Nets associated with `lxCombination` devices are named `instName.sinteger.expandedNetinteger` in the layout.

For example,

`R0.s1.expandedNet1`

where “expandedNet” signifies that the layout net has no real match in the schematic.

Related Topics

[Setting Layout XL Properties](#)

[Layout XL Properties](#)

IxMfactorSplit

This property is obsolete in the current release. It is automatically converted to the new Layout XL schema underlying physical hierarchy configuration.

To control whether Layout XL places a schematic device with the mfactor property as multiple devices in the layout,

1. From the layout window menu bar, choose *Launch – Configure Physical Hierarchy*.
2. In the *Instances* table, select the logical instance for which you want to generate multiple devices.
3. In the *Generation* tab, switch on the *Split mfactored devices* option.

This overrides the global setting on the Generation Tab of the Layout XL Options form.

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

Related Topics

[Generating a Layout](#)

[Converting Data to Use the IC 6.1 Layout XL Schema](#)

[Configuring the Physical Hierarchy](#)

[Setting Layout XL Properties](#)

[Layout XL Properties](#)

lxNetNamePrefix

`lxNetNamePrefix` *string* `t_prefix`

Adds a unique prefix to internal nets generated in the layout using Layout XL conventions.

Note: `lxNetNamePrefix` can be used only with devices which have the `lxCombination` property or `sfactor` parameter.

Symbol master Add Property form

<i>Name</i>	<code>lxNetNamePrefix</code>
<i>Type</i>	<i>string</i>
<i>Value</i>	<code>t_prefix</code>

Add CDF Parameter form

<i>Add After Parameter</i>	<i>As First Parameter</i>
<i>paramType</i>	<i>string</i>
<i>name</i>	<code>lxNetNamePrefix</code>
<i>defValue</i>	<code>t_prefix</code>

Example

Using the `lxCombination` property to generate layout devices from the `R` device in the schematic below generates three internal nets with the following names.

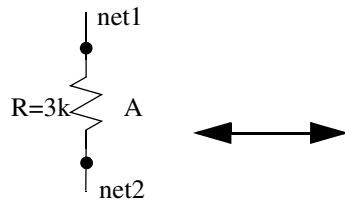
- `|R|net.ms1`
- `|R|net.ms2`

Virtuoso Layout Suite XL User Guide

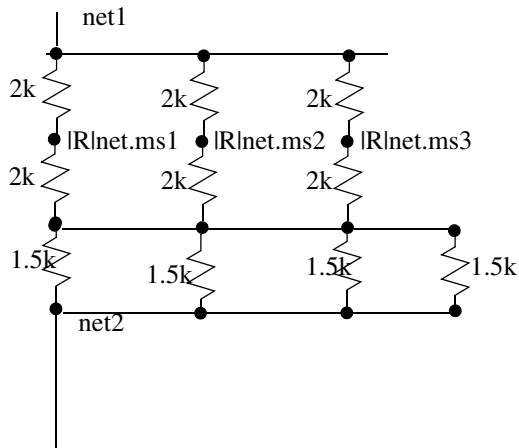
Layout XL Properties

■ |R|net.ms3

Schematic



Layout



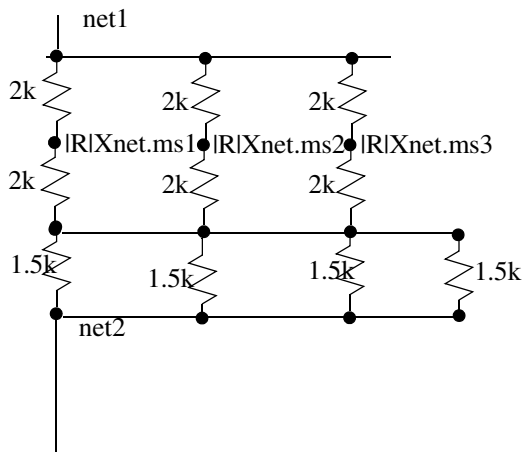
Using the `lxNetNamePrefix` property, you can add the prefix `x` to the internal net names. The names change to

■ |R|Xnet.ms1

■ |R|Xnet.ms2

■ |R|Xnet.ms3

Layout



Related Topics

Generating a Layout

Virtuoso Layout Suite XL User Guide

Layout XL Properties

Setting Layout XL Properties

Layout XL Properties

lxParamsToIgnore

`lxParamsToIgnore` string *list_of_parameter_names*

Specifies the layout device parameters to be ignored when using the following Layout XL commands.

- *Generate All From Source*
- *Generated Selected From Source*
- *Generate Clones*
- *Check Against Source*
- *Update Components And Nets*
- *Update Layout Parameters*



Cadence recommends that you use `lxParamsToIgnore` instead of the `lxIgnoredParams` property.

You attach the `lxParamsToIgnore` property on a layout instance.

Layout instance Add Property form

<i>Name</i>	<code>lxParamsToIgnore</code>
<i>Type</i>	<i>string</i>
<i>Value</i>	<i>list_of_parameter_names</i>

Additional Information

You can also use the Configure Physical Hierarchy window to specify the parameters to be ignored by the *Generate*, *Update*, and *Check* commands. To do this

1. From the layout window menu bar, choose *Launch – Configure Physical Hierarchy*.
2. In the *Instances* table, select the schematic instance for which you want to ignore parameters.
3. In the *Parameters* tab, list the parameters to be ignored in the *Ignore for generation* text field.

Virtuoso Layout Suite XL User Guide

Layout XL Properties

This overrides the global setting on the [Parameters Tab](#) form.

Related Topics

[Converting Data to Use the IC 6.1 Layout XL Schema](#)

[Configuring the Physical Hierarchy](#)

[Layout XL Properties](#)

lxParamsToIgnoreForCheck

`lxParamsToIgnoreForCheck` string *list_of_parameter_names*

Specifies the layout device parameters to be ignored when using the *Check Against Source* command.



Cadence recommends that you use `lxParamsToIgnore` instead of the `lxIgnoredParamsForCAS` property.

You attach the `lxParamsToIgnoreForCheck` property on a layout instance.

Layout instance Add Property form

<i>Name</i>	<code>lxParamsToIgnoreForCheck</code>
<i>Type</i>	<i>string</i>
<i>Value</i>	<i>list_of_parameter_names</i>

Additional Information

You can also use the Configure Physical Hierarchy window to specify the parameters to be ignored by the *Check Against Source* command. To do this,

1. From the layout window menu bar, choose *Launch – Configure Physical Hierarchy*.
2. In the *Instances* table, select the logical instance for which you want parameter differences to be ignored.
3. In the *Parameters* tab, list the parameters to be ignored in the *Ignore for check* text field.

This overrides the global setting on the Parameters Tab of the Layout XL Options form.

Related Topics

[Converting Data to Use the IC 6.1 Layout XL Schema](#)

[Configuring the Physical Hierarchy](#)

[Layout XL Properties](#)

IxRemoveDevice

This property is obsolete in the current release. It is automatically converted to the new Layout XL schema underlying physical hierarchy configuration.

To cause parasitic devices to be merged,

1. From the layout window menu bar, choose *Launch – Configure Physical Hierarchy*.
2. In the *Instances* table, select the logical instance in which you want to merge nets.
3. In the *Generation* tab, type the argument into the *Remove device* text field.

For more information, see

Related Topics

[Ignoring Parasitic Devices](#)

[Converting Data to Use the IC 6.1 Layout XL Schema](#)

[Configuring the Physical Hierarchy](#)

[Layout XL Properties](#)

IxRounding

This property is obsolete in the current release. It is automatically converted to the new Layout XL schema underlying physical hierarchy configuration.

To define how the value of a specified parameter on a cell master or schematic instance is rounded when it is evaluated by Layout XL,

1. From the layout window menu bar, choose *Launch – Configure Physical Hierarchy*.
2. In the table view, select the instance or cell in which you want to round values.
3. In the *Parameters* tab, type the required argument into the *Rounding* text field.

For more information, see

Related Topics

[Rounding Parameter Values](#)

[Converting Data to Use the IC 6.1 Layout XL Schema](#)

[Configuring the Physical Hierarchy](#)

[Layout XL Properties](#)

lxSeriesTerms

`lxSeriesTerms` *string* `t_terminal_name`

Specifies the connectivity of schematic and lxCombination devices with `sfactor` greater than 1 and instance terminals greater than 2. Set this property on the schematic instance.

The *Split mfactored devices* option must be turned on in the Generation Tab of the Layout XL Options form.

Symbol master Add Property form

<i>Name</i>	<code>lxSeriesTerms</code>
<i>Type</i>	<i>string</i>
<i>Value</i>	<code>t_terminal_names</code>

Add CDF Parameter form

<i>Add After Parameter</i>	<i>As First Parameter</i>
<i>paramType</i>	<i>string</i>
<i>name</i>	<code>lxSeriesTerms</code>
<i>defValue</i>	<code>t_terminal_names</code>

Example

The description below shows setting the `lxSeriesTerms` PLUS and MINUS on two instance terminals.

```
lxSeriesTerms = PLUS MINUS
```

Related Topics

[Setting Layout XL Properties](#)

[Layout XL Properties](#)

lxStickyNet

```
lxStickyNet boolean { t | nil }
```

Allows a shape to retain its assigned connectivity even after it has been modified in some way.

Layout XL adds the `lxStickyNet` property automatically to a shape or via when a net is assigned to that shape using the *Add Shape To Net* command. The property remains in effect until you

- Override it with a new net assignment
- Turn it off by setting it to `nil`
- Remove it using the *Edit – Basic – Properties* command.

Note: You can also manually add the property using the *Edit – Basic – Properties* command.

For more information on `lxStickyNet`, see [Interactive Wire Editing](#) in the *Virtuoso Space-based Router User Guide*.

Layout instance Add Property form

<i>Name</i>	<code>lxStickyNet</code>
<i>Type</i>	<i>Boolean</i>
<i>Value</i>	<code>t nil</code>

Notes on Legacy Contact Devices Converted from CDB

When a contact is converted from CDB to an `oaVia` object (standard or custom), its “stickiness” is derived from the `lxStickyNet` property (i.e., it behaves in the same way as a regular shape). If the via is part of a route, the `lxStickyNet` property is checked on the route too.

In some circumstances, a CDB contact device can be converted to a regular `OpenAccess` instance. Layout XL recognizes these instances as vias using the `function` property, which can be set on the instance, its master or in CDF.

- If the function is set to `contact`, the stickiness of the instance is derived from `lxStickyNet`.

- If `function` is set to `via` or `substrateContact`, the contact is sticky by construction, meaning that the connectivity of the contact is the net attached to the instance terminal. When the contact is not sticky, it can be the case that an instance terminal is reassigned dynamically based on the overlaps (exactly like a non-sticky shape).

Related Topics

[Connectivity Extraction](#)

[Setting Layout XL Properties](#)

[Layout XL Properties](#)

IxStopList

This property is obsolete in the current release. It is automatically converted to the new Layout XL schema underlying physical hierarchy configuration.

To define where Layout XL should stop when it is traversing the design hierarchy,

1. From the layout window menu bar, choose *Launch – Configure Physical Hierarchy*.
2. In the *Instances* or *Cells* table, type the view names you require into the *Inherited Stop List* field for the instance or cell in question.

For more information, see

Related Topics

[Converting Data to Use the IC 6.1 Layout XL Schema](#)

[Configuring the Physical Hierarchy](#)

[Layout XL Properties](#)

IxUseCell

This property is obsolete in the current release. It is automatically converted to the new Layout XL schema underlying physical hierarchy configuration.

To specify which cell to use when generating a layout for a device symbol,

1. From the layout window menu bar, choose *Launch – Configure Physical Hierarchy*.
2. Do one of the following.
 - ☐ Use the *Cells* table to specify which layout cell to use for a particular logical cell.
 - ☐ Use the *Instances* table to specify which layout cell to use for a specific logical instance.

Related Topics

[Converting Data to Use the IC 6.1 Layout XL Schema](#)

[Configuring the Physical Hierarchy](#)

[Layout XL Properties](#)

IxViewList

This property is obsolete in the current release. It is automatically converted to the new Layout XL schema underlying physical hierarchy configuration.

To tell Layout XL where to descend into the design hierarchy to find cells at a lower level,

1. From the layout window menu bar, choose *Launch – Configure Physical Hierarchy*.
2. In the *Instances* or *Cells* table, type the view names you require into the *Inherited View List* field for the instance or cell in question.

Related Topics

[Converting Data to Use the IC 6.1 Layout XL Schema](#)

[Configuring the Physical Hierarchy](#)

[Layout XL Properties](#)

mfactor

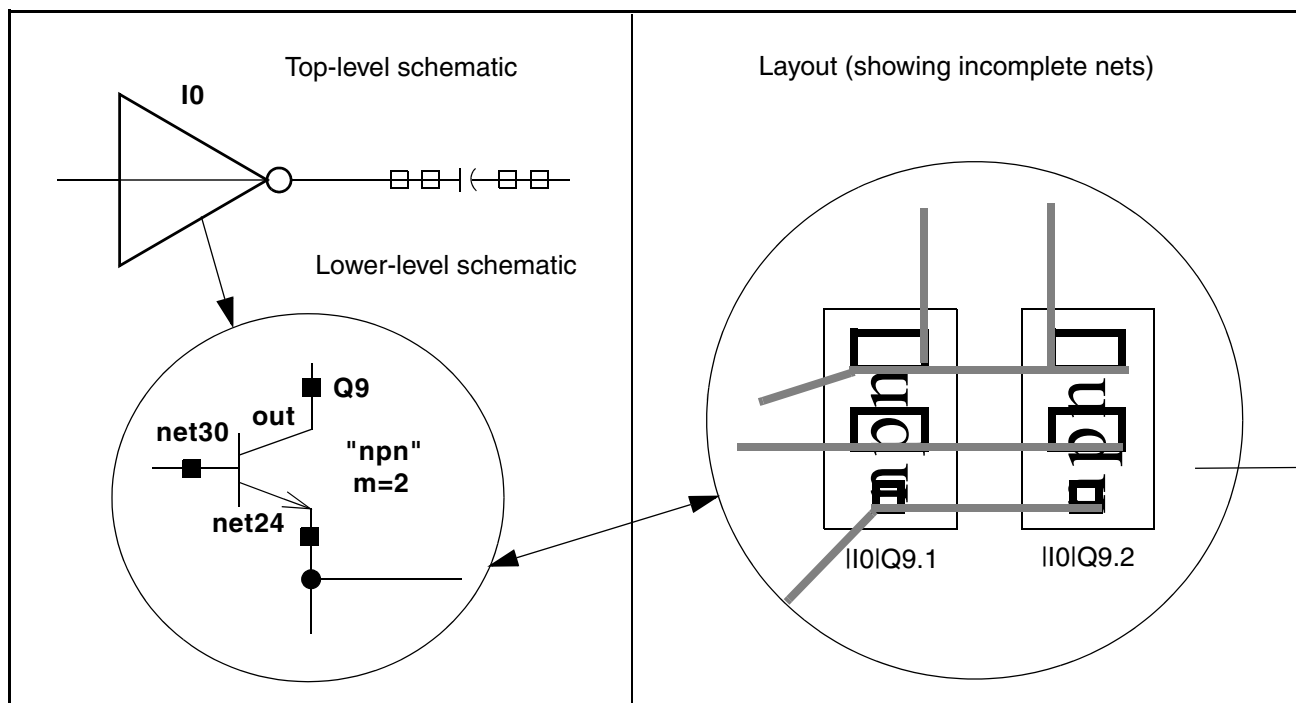
`mfactor integer x_integer`

Defines a one-to-many parallel relationship between a device in a schematic and multiple instances of the device in the layout.



You cannot use multiplication factors with components that cannot be used in parallel, such as voltage sources.

The `mfactor` property and `mfactorSplit` environment variable both work hierarchically. Any instance in the connectivity source hierarchy can have these properties, and the multiplicity is computed from the hierarchical path.



You can use the `mfactorSplit` environment variable to control whether the `mfactor` produces multiple layout devices or not. You can also use the `!xMfactorSplit` property on a given instance to override the global value given by the `mfactorSplit` environment variable.

Symbol master Add Property form

<i>Name</i>	<code>mfactor</code>

Virtuoso Layout Suite XL User Guide

Layout XL Properties

Symbol master Add Property form

Type	<i>int</i>
Value	<i>x_integer</i>

Add CDF Parameter form

Add After Parameter	As First Parameter
paramType	<i>int</i>
name	<i>mfactor</i>
defValue	<i>x_integer</i>

Naming Convention

Devices specified using the `mfactor` property in the schematic (and those that are folded in the layout) are named *instName.integer* in the layout.

For example, if you have a schematic instance named `P0` with

```
mfactor = 3
```

the layout instances generated are named as follows.

Schematic	Layout
P0	P0.1
	P0.2
	P0.3

Or, if the `prefixLayoutInstNamesWithPipe` environment variable is set to `nil`.

Schematic	Layout
P0	P0.1
	P0.2
	P0.3

Related Topics

[Generating a Layout](#)

[Setting Layout XL Properties](#)

[Layout XL Properties](#)

permuteRule

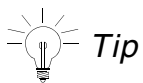
`permuteRule string t_permuteRule`

Makes the instance pins or terminals of a device permutable. The syntax is the same as is used by the Assura verification tools.

- `(p E1 E2)` specifies that E1 and E2 are permutable.
- `(f E1 E2)` specifies that E1 and E2 are fixed (i.e., not permutable).

All pins that are not listed in the `permuteRule` property for a device are considered fixed. If you define the `permuteRule` property on a schematic symbol, that setting overrides any other rule defined in the layout for that instance.

You define the `permuteRule` in the Edit Component CDF form in either the *Component Parameters* section or the *Simulation Information* section. If it is defined in both sections, Layout XL uses the definition in the *Component Parameters* section.



Tip

If you want Layout XL to permute pins automatically, switch on the *Permute pins* option in the Generation Tab of the Layout XL Options form. If this option is switched off, you can only permute pins manually. Use the *Disable Permutation* constraint to prevent permutation for a selected instance. See Disabling Permutation for an Instance.

Symbol master Add Property form

<i>Name</i>	<code>permuteRule</code>
<i>Type</i>	<code>string</code>
<i>Value</i>	<code>t_permuteRule</code>

Add CDF Parameter form

<i>Add After Parameter</i>	<i>As First Parameter</i>
<i>paramType</i>	<code>string</code>
<i>name</i>	<code>permuteRule</code>

Add CDF Parameter form

<i>defValue</i>	<i>t_permuteRule</i>

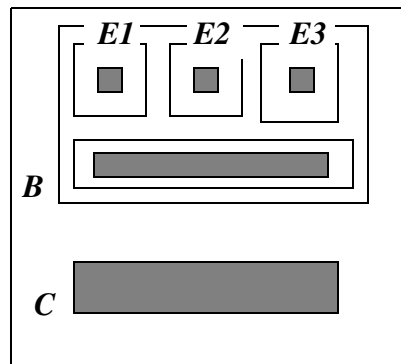
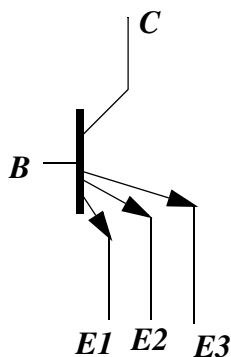
Examples

- To specify that pins E1 and E2 are permutable, use the following `permuteRule`.

```
(p E1 E2)
```

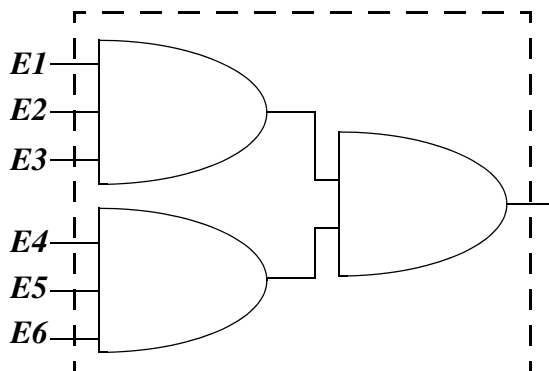
- To specify that pins E1, E2, and E3 are permutable, use the following `permuteRule`.

```
(p E1 E2 E3)
```



- You can also specify a hierarchical `permuteRule`. For example, the value below defines two sets of permutable pins (E1, E2, E3 and E4, E5, E6) and specifies that the two sets are permutable with each other.

```
(p (p E1 E2 E3) (p E4 E5 E6))
```



- The following `permuteRule` value defines two sets of permutable pins (E1, E2, E3 and E4, E5, E6) and specifies that the two sets are not permutable with each other.

```
(f (p E1 E2 E3) (p E4 E5 E6))
```


Macros

To avoid entering long lists of permutable pins for the `permuteRule` property, use the following macro notations.

- `(f ALL)`, which means that all the pins in the cell are fixed
- `(p ALL)`, which means that all the pins in the cell are permutable
- Range indications, where pins are numbered or sorted as in bus notations.

For example, the notation

`(p A<0:3> B<4:7>)`

is equivalent to

`(p A<0> A<1> A<2> A<3> B<4> B<5> B<6> B<7>)`

The notation

`(p (f A<0:3>) (f B<4:7>))`

defines two sets of fixed pins (`A<0:3>` and `B<4:7>`) which are permutable with each other

You can also express range indications in descending order

`(p A<3:0> B<7:4>)`

Related Topics

[Pin Permutation](#)

[Disabling Permutation for an Instance](#)

[Setting Layout XL Properties](#)

[Layout XL Properties](#)

Virtuoso Layout Suite XL User Guide

Layout XL Properties

sfactor

`sfactor { integer | float | string | IL expression } user_defined_value`

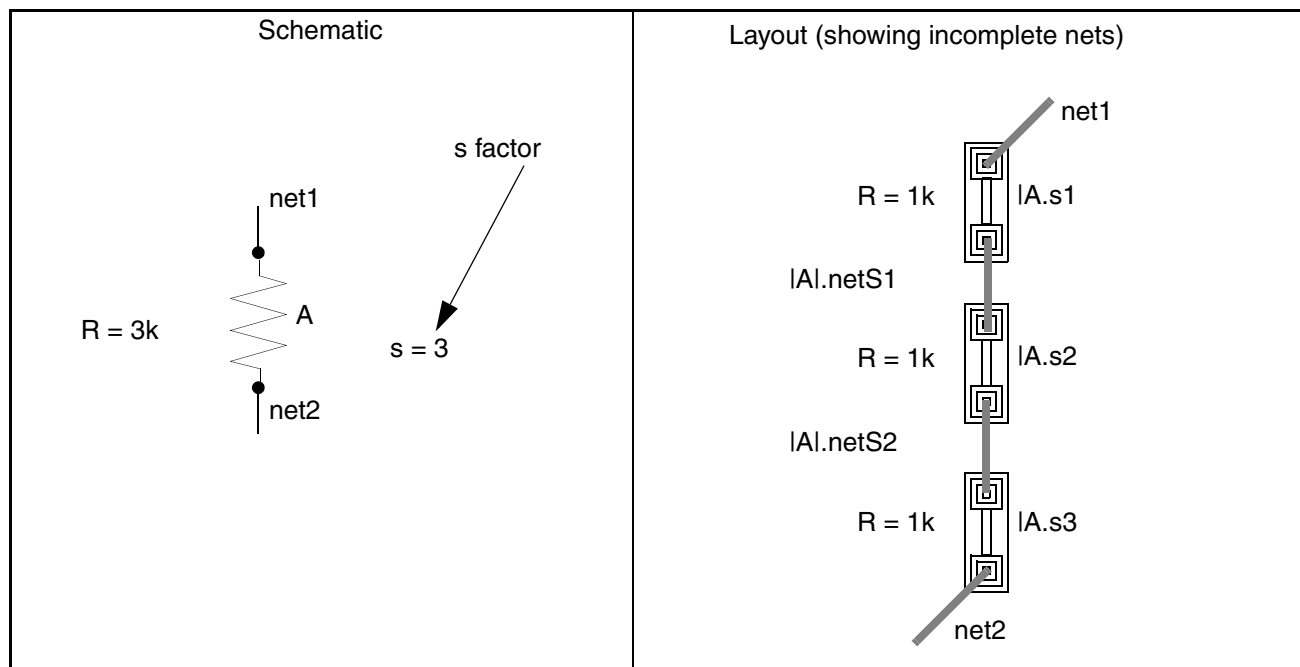
Defines a one-to-many relationship between a device in a schematic and multiple instances in a series connection in the layout.

Important

If both `mfactor` and `sfactor` are defined on the same instance, `sfactor` is ignored.

The `sfactor` property can be applied to two or three terminal resistors, capacitors, and inductors. You set the value of the `sfactor` property (called `s` or `S`, unless you change it on the General Tab of the Layout XL Options form) to the number of layout devices you want to generate. When you run the *Generate All From Source* or *Update Components And Nets* command, Layout XL generates the number of layout devices specified by the property value. It creates all the devices identical in size.

The `sfactor` must set at the layout stop level. Any property above or below that level is ignored.



You can use the `capacitanceParamNames`, `inductanceParamNames`, `resistanceParamNames`, and `transistorWidthParamNames` environment variables to tell Layout XL which parameters of the device are to be split.

Virtuoso Layout Suite XL User Guide

Layout XL Properties

The `lxSeriesTerms` property must be defined on the instance in the schematic in order for Layout XL to understand the connectivity of devices with `sfactor > 1` and instance terminals `> 2`.

Symbol master Add Property form

<i>Name</i>	<code>sfactor</code>
<i>Type</i>	<code>int</code> <code>float</code> <code>string</code> <code>ILExpr</code>
<i>Value</i>	<code>user_defined_value</code>

Add CDF Parameter form

<i>Add After Parameter</i>	<code>As First Parameter</code>
<i>paramType</i>	<code>string</code> <code>int</code> <code>float</code>
<i>name</i>	<code>sfactor</code>
<i>defValue</i>	<code>user_defined_value</code>

Naming Conventions

Devices specified using the `sfactor` property in the schematic are named `instName.sinteger` in the layout.

For example, if you have a schematic instance named `R0` with

```
sfactor = 3
```

the layout instances generated are named as follows.

Schematic	Layout
R0	R0.s1
	R0.s2
	R0.s3

Virtuoso Layout Suite XL User Guide

Layout XL Properties

Or, if the `prefixLayoutInstNamesWithPipe` environment variable is set to `nil`.

Schematic	Layout
P0	R0.s1
	R0.s2
	R0.s3

Nets associated with series-connected devices are named *instName.sinteger.expandedNetinteger* in the layout.

For example,

`R0.s1.expandedNet1`

where “expandedNet” signifies that the layout net has no real match in the schematic.

Related Topics

[Generating a Layout](#)

[Setting Layout XL Properties](#)

[Layout XL Properties](#)

vxInstSpacingDir

`vxInstSpacingDir list (left right top bottom)`

Assigns automatic spacing properties to pins of instances.

If these pins are on different nets or the pins cannot abut for any reason (for example, they are not assigned abutment properties), the software automatically separates the instances by the distance and in the direction you specify.

Note: If one of the devices is in any type of group (including a synchronous clone), the other device must be in the same group for automatic spacing to occur.

Symbol master Add Property form

<i>Name</i>	<code>vxInstSpacingDir</code>
<i>Type</i>	<code>ILList</code>
<i>Value</i>	<code>left right top bottom</code>

Add CDF Parameter form

<i>Add After Parameter</i>	<i>As First Parameter</i>
<i>paramType</i>	<code>list</code>
<i>name</i>	<code>vxInstSpacingDir</code>
<i>defValue</i>	<code>left right top bottom</code>

If you want automatic spacing to apply to Cadence® SKILL language or technology file parameterized cells (pcells), you can enter these properties in SKILL to add the spacing properties to the pins.

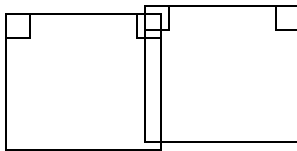
If you want automatic spacing to apply to standard cells, you can also use the Virtuoso Layout Suite L layout editor (Layout L) to enter properties on the pins.

Virtuoso Layout Suite XL User Guide

Layout XL Properties

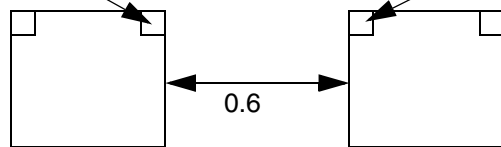
If the `vxInstSpacingRule` value is different for two pins affected by *Space components*, the larger of the two values is used; if the value is defined for only one of two pins affected by *Space components*, the defined value is used.

With *Space components* turned



With *Space components* turned

`vxInstSpacingDir`
value = right
`vxInstSpacingRule`
value = 0.6



`vxInstSpacingDir`
value = left
`vxInstSpacingRule`
value = 0.5

`vxInstSpacingDir`
value = right, left
`vxInstSpacingRule`
value = 0.6



`vxInstSpacingDir`
value = right, left
`vxInstSpacingRule`
value = 1.0

- To activate automatic spacing, turn on the *Space components* option in the Generation Tab form.

You can use the Layout L user interface to enter properties on the pins in the same way that you enter the `permuteRule` properties.

To add the access direction to the pin:

property	value type	valid values
<code>vxInstSpacingDir</code>	skill list	one or more of the following strings in the list: "left" "right" "top" "bottom"

Related Topics

[Setting Layout XL Properties](#)

[Layout XL Properties](#)

vxlInstSpacingRule

`vxlInstSpacingRule float f_value`

Defines a spacing offset when pins are on a different net.

Symbol master Add Property form

<i>Name</i>	<code>vxlInstSpacingRule</code>
<i>Type</i>	<code>float</code>
<i>Value</i>	<code><i>f_value</i></code>

Add CDF Parameter form

<i>Add After Parameter</i>	<i>As First Parameter</i>
<i>paramType</i>	<code>float</code>
<i>name</i>	<code>vxlInstSpacingRule</code>
<i>defValue</i>	<code><i>f_value</i></code>

A user-defined function, called `abutFunction`, can be used to define a spacing offset when pins are on a different net. When you set the `vxlInstSpacingRule` to float, two pins of the same abutment class and different nets are pushed apart a distance equal to the maximum values of the two `vxlInstSpacingRule` pins. To control the abutment spacing set the `vxlInstSpacingRule` to `abutFunction`.

Note: `abutFunction` is not the name of your custom abutment function, but is the property that is set to the name of your abutment function.

```
dbReplaceProp(sObj~>dbId "abutFunction" "string" "myCustomAbutmentFunction")
dbReplaceProp(sObj~>dbId "vxlInstSpacingRule" "string" "abutFunction")
```

You can write your own spacing function for the two instances involved. For example, `myCustomSpacingFunc ( )` is to be called in the body of the user defined abutment function (`abutFunction`). The `myCustomSpacingFunc ( )` must return a float value to be effective.

There are two numbers passed to the abutment function, the connection condition (1, 2, or 3) and the abutment event (1, 2, 3, or 4). The connection condition (3) is for different nets. The

Virtuoso Layout Suite XL User Guide

Layout XL Properties

sequence for nonabutable pins should be the same as for abutable. The calling of the abutment function for adjusting the pcells (event 2) and then one to calculate the spacing of the resultant reference edges (event 4). This allows you to adjust the well or isolation on the pcell and then space the devices based on the reference edges.

To get the spacing offset when pins are on different nets add the following case in your custom abutment function.

```
(4
  if(instA~>w > 3.0 && instB~>w > 3.0
    then
      result = 3
    else
      result = 1
  )
);; end of case 4
```

You can use the layout editor user interface to enter properties on the pins in the same way that you enter the `permuteRule` properties.

If you need to add a variable spacing rule to the pin, set the `vx1InstSpacingRule` to `abutFunction` as shown below;

```
dbReplaceProp(sObj~>dbId "vx1InstSpacingRule" "string" "abutFunction")
```

The properties to set automatic spacing are listed below.

To add the spacing rule to the pin:

property	value type	valid values
vx1InstSpacingRule	floating pt number	number greater than 0

Related Topics

[Setting Layout XL Properties](#)

[Layout XL Properties](#)

Virtuoso Layout Suite XL User Guide

Layout XL Properties

Layout XL Forms

This section lists the Virtuoso® Layout Suite XL layout editor (Layout XL) forms.

Note: Many of the options described in this section have a corresponding environment variable. For more information, see [GUI Options and Corresponding Environment Variables](#).

[Add A New Pin](#)

[Add Soft Pin](#)

[Assign Layout Instance Terminals](#)

[Assign Nets](#)

[Block Parameters using Physical View](#)

[Check Against Source](#)

[Chop](#)

[Configure Physical Hierarchy](#)

[Convert Libraries to Use Physical Configuration Views](#)

[Copy](#)

[Create Cluster Boundary](#)

[Create Custom Placement Area](#)

[Create Feed Through Terminal Block Pins](#)

[Create Instance](#)

[Create Path](#)

[Create Physical Configuration View](#)

Virtuoso Layout Suite XL User Guide

Layout XL Forms

[Create Polygon](#)

[Create Rectangle](#)

[Define Device Correspondence](#)

[Design Summary](#)

[Device List](#)

[Edit Net Attributes](#)

[Edit Soft Blocks](#)

[Extract Layout](#)

[Generate Chained Devices](#)

[Generate Clones](#)

[Generate Folded Devices](#)

[Generate Layout](#)

[Generate Selected Components](#)

[Generate Selected Components - Pin Options](#)

[Instance/Pin List](#)

[Layout XL Options](#)

[LDE Analysis](#)

[Litho Fixing](#)

[Load Physical View](#)

[Match And Fix](#)

[Modify Correspondence](#)

[Move](#)

[Move Cells](#)

[Open Physical Configuration View](#)

Virtuoso Layout Suite XL User Guide

Layout XL Forms

Permutation Information

Repeat Copy

Reshape

Selection Options

Set Net Order

Set Pin Label Text Style

Snap Pins

Soft Block Global Options

Startup Option

Stretch

Update Clone Families

Update Components and Nets

Update Connectivity Reference

XL Probe

Add A New Pin

Use the **Add a New Pin** form to specify a new pin to be generated in the layout view.

Enter a terminal name lets you type in the name of the terminal to which the new pin is connected.

Related Topics

[Generating All Components from Source](#)

[Specifying the I/O Pins to be Generated](#)

[Layout XL Forms](#)

Add Soft Pin

Use the **Add Soft Pin** form to create additional soft block pins in order to improve the connectivity of your design.

Soft block list lists all the soft blocks that were selected when you opened the form. You can create soft pins for any of the blocks that are listed this field. The *Lib*, *Cell*, and *View* fields confirm the currently selected block.

Choose the terminal you want from the drop-down list, which by default contains all the terminals in the currently selected block. You can filter the terminal names that are displayed by typing in the text box and filtering either **All Pins** or **Pins in Selected Set**. To assist you in defining constraints, the canvas shows flight lines to represent the nets between the terminals you select.

Terminal name specifies the name of the terminal on which you want to create a soft pin.

The pins are created as strong pins by default and are listed in the table underneath the *Terminal name* list. Specify the parameters using the controls described below and click *Create Strong Pin* to create a pin for the selected terminal.

Name specifies a name for the pin you want to create.

Layer specifies the layer purpose on which the pin is to be created. Choose the layer purpose you want from the list of extractable layers in the design.

Width and **Height** specify the width and height of the new pin. These fields are prepopulated with the `minWidth` spacing defined in the technology file for the selected pin layer.

Side specifies on which sides of the place and route boundary the pins can be created. If you specify more than one side for a pin, the pin optimizer can place the pins on any of the specified sides. The images indicate what each constraint means in the context of both a rectangular and polygonal boundary.

The *Update* button lets you update currently selected pins with the new parameter values; *Delete* removes the selected pins from the list.

Related Topics

[Soft Block Pins](#)

[Creating a Soft Pin](#)

[Layout XL Forms](#)

Assign Layout Instance Terminals

Use the **Assign Layout Instance Terminals** form to manually assign layout nets to unbound instance terminals in selected layout instances.

This might be necessary if the *Define Device Correspondence* command is unable to assign all the instance terminals automatically; for example, if the instance masters in a one-to-one correspondence do not match, or if you are creating a one-to-many, many-to-one, or many-to-many correspondence.

Select layout instance terminal to assign lists the instance terminals in the instances being mapped. Select one or more terminals you want to assign to a net. Each terminal name is appended with the name of the net to which is currently connected. Unconnected nets are labeled as “dangling”.

Select layout net lists global nets and the nets that connect to the instances being mapped. Choose the net you want to assign to the selected instance terminals and click *Assign instance terminal* to make the assignment.

Related Topics

[Updating Device Correspondence](#)

[Assigning Layout Instance Terminals to Nets](#)

[Layout XL Forms](#)

Assign Nets

If you place an instance in a layout that does not have a connectivity source, the software does not connect it to any net unless you specifically assign the instance pins to a net. Use the **Assign Nets** form to do this.

Click *Create* to add a new name to the list of net names. Type the net name you want to add in the entry field.

Click *Delete* to delete a net name from the list of net names. Type the net name to be deleted in the entry field or select it in the list.



Tip

The entry field above the list of net names has a search mechanism that highlights in the list box the net name you type in the field.

Related Topics

[Adding Components](#)

[Assigning Instance Pins to a Net](#)

[Layout XL Forms](#)

Block Parameters using Physical View

Use the **Block Parameters using Physical View** dialog to specify a source OpenAccess cellview containing predefined soft block parameters which you want to copy into the Configure Physical Hierarchy Attributes pane.

Type in the **Library**, **Cell** and **View Name** of the source cellview containing the information to import. When you click *OK*, the Attributes pane is updated with the settings from the specified cellview.

Related Topics

[Loading Soft Block Parameters from Another Cellview](#)

[Layout XL Forms](#)

Check Against Source

You check the differences between the schematic and layout using the *Check Against Source* command. Use the **Check Against Source** form to choose what types of differences you want to see in the report and what format the report should take.

Report Differences In

Instance masters reports instance masters that are missing or mismatched between layout and schematic.

Unbound instances reports layout instances that are not mapped to schematic instances.

Connectivity reports connectivity issues on top level pins and global nets; mismatched or missing terminals and instance terminals; and unbound nets in the layout.

Parameters/Properties reports issues with CDF parameters and with properties set on the schematic and layout views. You specify how the parameters and properties are compared on the Parameters Tab of the Layout XL Options form.

Output Control

Display report in a separate window prints the report in a separate Info window, otherwise the report is printed in the CIW.

Maximum differences reported limits the number of differences reported in the CIW for each of the four categories. For example, if you set the maximum number of differences to 100, the command reports up to a maximum of 100 cellview master differences, 100 unbound instance differences, and so on. Any messages over the specified limit are suppressed in the CIW but are still listed in the log file.

Log file name specifies the name of a file in which all the Check Against Source messages are written. You can use the browse button to locate an existing file.

Append to log file appends the results of the current run to the specified log file instead of overwriting the log file.

Do not show me this dialog again lets you re-run Check Against Source with the current options settings without the dialog being displayed.

Related Topics

[Check Against Source](#)

[Layout XL Forms](#)

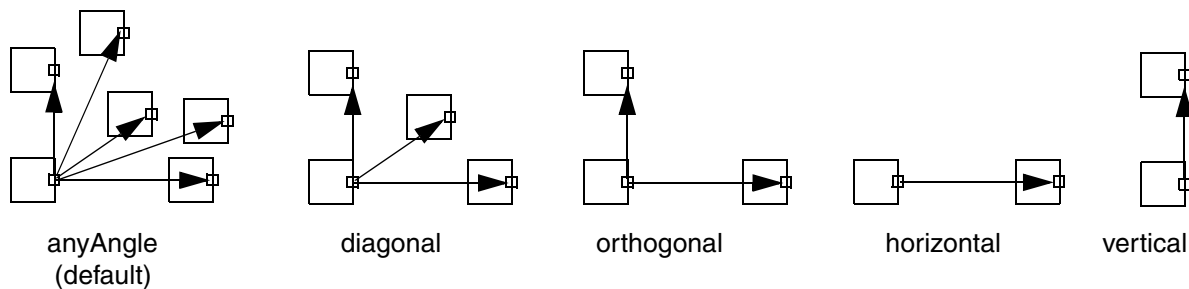
Chop

Use the **Chop** form to control the operation of the *Chop* command in Level-1 Editing mode. This command lets you cut away part of a place and route boundary without entering Edit In Place mode.

Chop Shape controls the shape you want to use as the cutter. If it is set to *line*, *Remove Chop* has no effect.

Remove Chop removes the part of the object enclosed by *Chop Shape*.

Snap Mode controls the shape of the polygon or line segments you can draw. It applies only when *Chop Shape* is set to *polygon* or *line*.



Snap To Grid snaps the chopped object to a grid depending on the block type. If it is an *analog* block, the place and route boundary and pin edges are snapped to the manufacturing grid. If it is a *digital* block, the place and route boundary is snapped to the placement grid and the pin centers are snapped to the routing grid. Pins with placement status *unplaced*, *fixed*, *locked*, or *unknown* are not snapped, even if they were touching the boundary before the chop.

Pin Snap Options

Snap Pins To Boundary automatically snaps pins to the place and route boundary.

Snap Soft Block Pins To automatically snaps soft block pins either to the *Modified Edge* or to the *Closest Edge* after the chop is complete.

Related Topics

[Editing Soft Blocks](#)

[Chopping an Object in Level-1 Editing Mode](#)

[Layout XL Forms](#)

Configure Physical Hierarchy

Use the **Configure Physical Hierarchy** window to specify how a layout implementation is generated from a specific schematic design.

Common Elements

[Top Cell Pane](#)

[Global Bindings Pane](#)

Hierarchy Configuration Mode

[Hierarchy Configuration Instances Table](#)

[Hierarchy Configuration Cells Table](#)

[Hierarchy Configuration Generation Attributes](#)

[Hierarchy Configuration Parameter Attributes](#)

[Hierarchy Configuration Terminal Attributes](#)

Component Types Mode

[Component Types Cells Table](#)

[Component Types Attributes](#)

Soft Block Mode

[Soft Block Instances Table](#)

[Soft Block Boundary Attributes](#)

[Soft Block I/O Pins Attributes](#)

[Soft Block Obstructions Attributes](#)

Related Topics

[Configuring the Physical Hierarchy](#)

[Layout XL Forms](#)

Top Cell Pane

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

Logical shows the library, cell, and view names of the logical view associated with the current physical configuration view. Use the *Open* button to open (or raise, if already open) the specified logical view in the context of the current physical configuration.

Physical shows the library, cell, and view names of the physical view associated with the current physical configuration view. Use the *Open* button to open (or raise, if already open) the specified physical view in the context of the current physical configuration.

Related Topics

[Configure Physical Hierarchy](#)

Global Bindings Pane

The **Global Bindings** pane lets you specify how the logical design is traversed and how the logical to physical correspondence for leaf cells (stop points) is made. It is a dockable window, which you can dock on either side of the main Configure Physical Hierarchy window, or leave undocked as a floating window.

Physical library list is the list of libraries that are searched to find the corresponding physical cell for a given logical cell. By default, the library containing the logical cell is always searched first. This option is equivalent to the `lxUseLibList` environment variable.

Logical switch view list specifies the view names that are used to descend into a hierarchical design to find layout views. This option is equivalent to the `viewList` environment variable.

Physical stop view list specifies the view names that are used to determine the corresponding physical view for a given logical view. When traversing a hierarchy, Configure Physical Hierarchy stops when it encounters a view with one of the specified names.

Virtuoso Layout Suite XL User Guide

Layout XL Forms

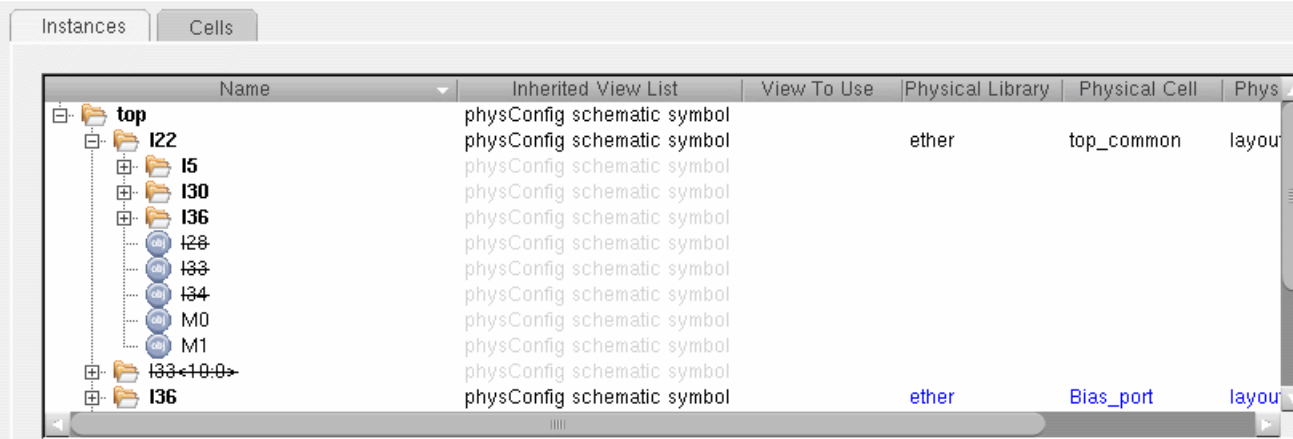
Constraint view list lists the names of the views containing constraint data. This list is passed to the Virtuoso Schematic Editor when it is opened in the context of a physical configuration, allowing it to determine whether the contents of the Constraint Manager must be updated.

Related Topics

[Configure Physical Hierarchy](#)

Hierarchy Configuration Instances Table

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



Name	Inherited View List	View To Use	Physical Library	Physical Cell	Phys
top	physConfig schematic symbol				
I22	physConfig schematic symbol		ether	top_common	layout
I5	physConfig schematic symbol				
I30	physConfig schematic symbol				
I36	physConfig schematic symbol				
I28	physConfig schematic symbol				
I33	physConfig schematic symbol				
I34	physConfig schematic symbol				
M0	physConfig schematic symbol				
M1	physConfig schematic symbol				
I33<10.0>	physConfig schematic symbol				
I36	physConfig schematic symbol		ether	Bias_port	layout

For each instance in the design, the table shows the view list used to reach the schematic instance and the corresponding layout cellview found.

Columns in the Hierarchy Configuration Instances Table

The individual fields show the following information.

Column Name	Description
<i>Name</i>	The name of the top cell, hierarchical node, or schematic leaf instance derived using either the <i>Inherited View List</i> or the <i>View To Use</i> field.

Virtuoso Layout Suite XL User Guide

Layout XL Forms

<i>Inherited View List</i>	Specifies the list of view names to switch into when traversing the hierarchy from this node onward. Note: This setting is inherited by all the subnodes and leaf cells under the current node, unless it is overridden by an <i>Inherited View List</i> or a specific <i>View To Use</i> setting on a subnode or leaf cell.
<i>View To Use</i>	Specifies the name of the view to use for a node or leaf cell in the hierarchy. If there is no <i>View To Use</i> specified, the leaf or node in question inherits the view list from its parent. Note: The <i>View To Use</i> setting is not inherited by the children of any node on which it is set. If your design contains two instances of the same cell and you specify different views to be used for each instance, then the <i>Cells</i> table lists two entries for that cell, one for each view to use.
<i>Physical Library</i>	The name of the physical library containing the corresponding layout cellview.
<i>Physical Cell</i>	The name of the physical cell to use.
<i>Physical View</i>	The name of the physical view to use.
<i>Inherited Stop List</i>	Specifies the view names that cause hierarchy traversal to stop its search for a layout cellview. If a view with one of the specified names is found, it is used as the physical counterpart of the schematic instance in question. Note: This setting is inherited by all the nodes and leaf cells under the current node, unless it is overridden by another <i>Inherited Stop List</i> or by a specific <i>Physical View</i> setting on a subnode or leaf cell.

Color Coding in the Hierarchy Configuration Instances Table

The entries in the table are color-coded as follows.

Text Color	Description

Virtuoso Layout Suite XL User Guide

Layout XL Forms

Gray	If the text label for a schematic instance has a strikethrough and the entries against it in the physical library, cell, and view fields are in gray text it means that the schematic instance is set to be ignored. Instance bindings under a valid physical stop point are also shown in gray text, indicating that although the binding has been set up, it is never reached because of the physical stop point at a higher level.
Black on pink	The layout fields in the table have a pink background if the schematic instance has no corresponding layout. For a leaf cell, this means that it has no corresponding layout; for a block, it means that at least one of its children does not have a corresponding layout.
Red	The physical binding points to a cellview that does not exist.
Blue	The value was typed in manually by the user, or a property was created while converting data to use the IC6.1 schema.

Context Menu for the Hierarchy Configuration Instances Table

Click the right mouse button to open a context menu with a number of other options. These are enabled and disabled dynamically depending on what is selected in the table view.

Command Name	Description
<i>Set/clear stop point</i>	Makes an instance a stop point in the hierarchy traversal. A 'STOP' icon against the schematic instance indicates that traversal stops at this instance. To clear the setting, select the instance and choose <i>Set/clear stop point</i> again.

Virtuoso Layout Suite XL User Guide

Layout XL Forms

<i>Set/remove force descend</i>	Forces hierarchy traversal to proceed beyond a leaf instance. A 'GO' icon indicates that the selected instance will now be traversed even if it would normally be considered a leaf instance. To remove the setting, select the instance and choose <i>Set/remove force descend</i> again. Note: You can set the force descend only on instances that have a physical binding.
<i>Create Physical Binding</i>	Lets you set a physical binding for the selected cell.
<i>Clear Physical Binding</i>	Lets you clear a physical binding for the selected cell.
<i>Edit Multiple <fieldName></i>	Lets you change the value of the specified field in multiple table entries.
<i>Options</i>	Specifies how the data is presented in the tree. For information on the Options form, see The Navigator Options Form.

Related Topics

[Hierarchy Configuration Cells Table](#)

[Hierarchy Configuration Generation Attributes](#)

[Hierarchy Configuration Parameter Attributes](#)

[Hierarchy Configuration Terminal Attributes](#)

[Configure Physical Hierarchy](#)

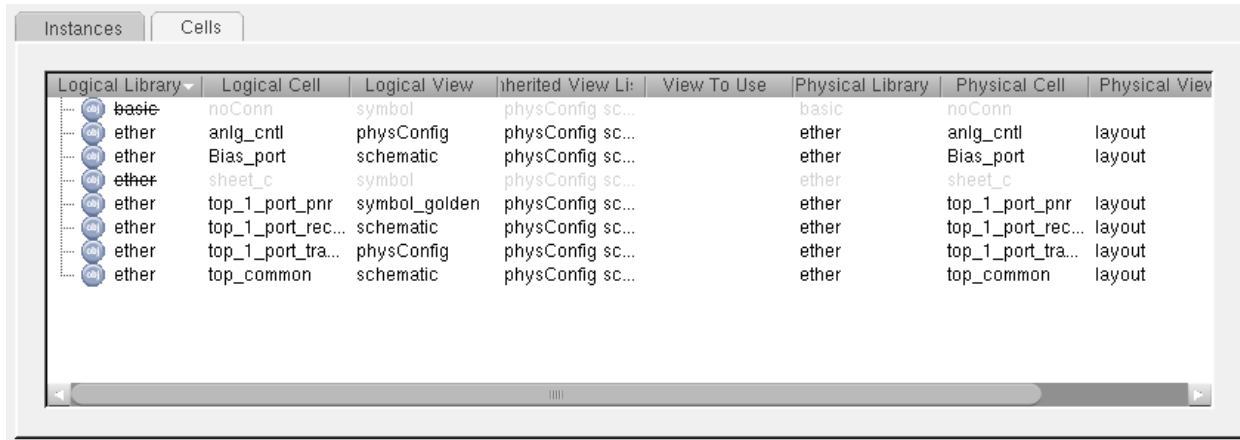
Hierarchy Configuration Cells Table

The *Cells* table view shows a flat view of the cellviews used in the schematic design and lists the corresponding layout cellview. The table lists each cellview explicitly, so if your design

Virtuoso Layout Suite XL User Guide

Layout XL Forms

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



Logical Library	Logical Cell	Logical View	Inherited View List	View To Use	Physical Library	Physical Cell	Physical View
basic	noConn	symbol	physConfig sc...		basic	noConn	
ether	anlg_cntl	physConfig	physConfig sc...		ether	anlg_cntl	layout
ether	Bias_port	schematic	physConfig sc...		ether	Bias_port	layout
ether	sheet_c	symbol	physConfig sc...		ether	sheet_c	
ether	top_1_port_pnr	symbol_golden	physConfig sc...		ether	top_1_port_pnr	layout
ether	top_1_port_rec...	schematic	physConfig sc...		ether	top_1_port_rec...	layout
ether	top_1_port_tra...	physConfig	physConfig sc...		ether	top_1_port_tra...	layout
ether	top_common	schematic	physConfig sc...		ether	top_common	layout

Columns in the Hierarchy Configuration Cells Table

The individual fields show the following information.

Column Name	Description
<i>Logical Library</i>	The name of the logical library.
<i>Logical Cell</i>	The name of the logical cell.
<i>Logical View</i>	The name of the logical view.
<i>Inherited View List</i>	Specifies the list of view names to switch into when traversing the hierarchy from this node onward. Note: This setting is inherited by all the subnodes and leaf cells under the current node, unless it is overridden by an <i>Inherited View List</i> or a specific <i>View To Use</i> setting on a subnode or leaf cell.

Virtuoso Layout Suite XL User Guide

Layout XL Forms

<i>View To Use</i>	Specifies the name of the view to use for a node or leaf cell in the hierarchy. If there is no <i>View To Use</i> specified, the leaf or node in question inherits the view list from its parent. Note: The <i>View To Use</i> setting is not inherited by the children of any node on which it is set. If your design contains two instances of the same cell and you specify different views to be used for each instance, then the <i>Cells</i> table lists two entries for that cell, one for each view to use.
<i>Physical Library</i>	The name of the physical library containing the corresponding layout cellview.
<i>Physical Cell</i>	The name of the physical cell to use.
<i>Physical View</i>	The name of the physical view to use.
<i>Inherited Stop List</i>	Specifies the view names that cause hierarchy traversal to stop its search for a layout cellview. If a view with one of the specified names is found, it is used as the physical counterpart of the schematic instance in question. Note: This setting is inherited by all the nodes and leaf cells under the current node, unless it is overridden by another <i>Inherited Stop List</i> or by a specific <i>Physical View</i> setting on a subnode or leaf cell.

Color Coding in the Hierarchy Configuration Cells Table

The entries in the table are color-coded as follows.

Text Color	Description
Gray	If the text label for a schematic instance has a strikethrough and the entries against it in the physical library, cell, and view fields are in gray text it means that the schematic instance is set to be ignored. Instance bindings under a valid physical stop point are also shown in gray text, indicating that although the binding has been set up, it is never reached because of the physical stop point at a higher level.
Red	The physical binding points to a cellview that does not exist.

Virtuoso Layout Suite XL User Guide

Layout XL Forms

Blue	The value was typed in manually by the user, or a property was created while converting data to use the IC6.1 schema.

Context Menu for the Hierarchy Configuration Cells Table

Click the right mouse button to reveal a context menu with a number of other options. These are enabled and disabled dynamically depending on what is selected in the table view.

Command Name	Description
<i>Create Physical Binding</i>	Lets you set a physical binding for the selected cell.
<i>Clear Physical Binding</i>	Lets you clear a physical binding for the selected cell.
<i>Edit Multiple <fieldName></i>	Lets you change the value of the specified field in multiple table entries.
<i>Options</i>	Specifies how the data is presented in the tree. For information on the Options form, see The Navigator Options Form in the <i>Virtuoso Schematic Editor XL User Guide</i> .

Related Topics

[Hierarchy Configuration Instances Table](#)

[Hierarchy Configuration Generation Attributes](#)

[Hierarchy Configuration Parameter Attributes](#)

[Hierarchy Configuration Terminal Attributes](#)

[Configure Physical Hierarchy](#)

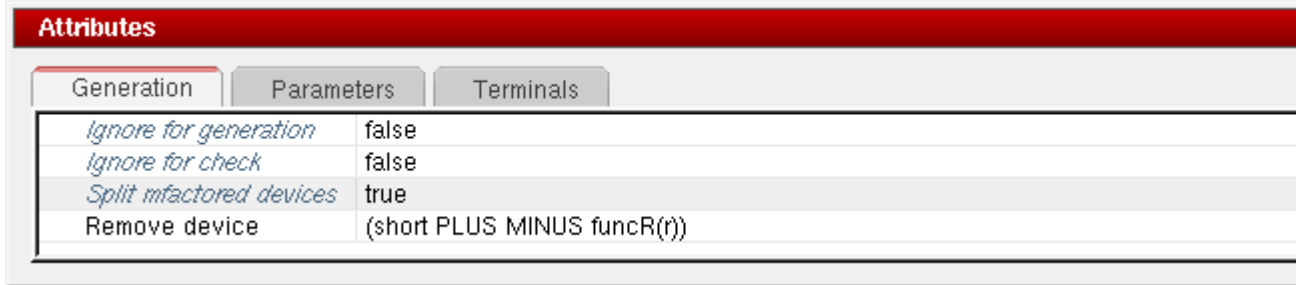
Hierarchy Configuration Attributes Pane

The Attributes pane lets you specify a number of the cell and instance attributes that affect instance generation.

Virtuoso Layout Suite XL User Guide

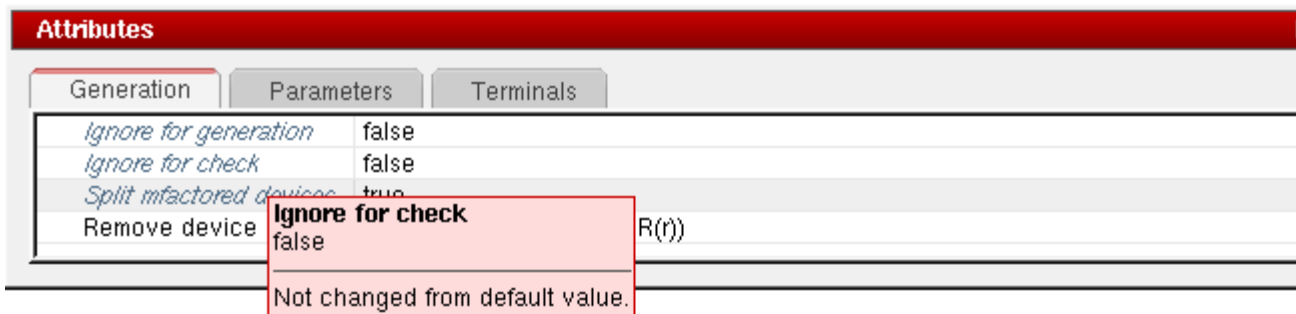
Layout XL Forms

- The attribute label is drawn in *italic* text when it is unchanged from its default value. On the *Generation* tab shown below, the first three attributes are the same as their default values, while the fourth has been changed.



Attributes		
Generation	Parameters	Terminals
<i>ignore for generation</i>	false	
<i>ignore for check</i>	false	
<i>Split mfactored devices</i>	true	
Remove device	(short PLUS MINUS funcR(r))	

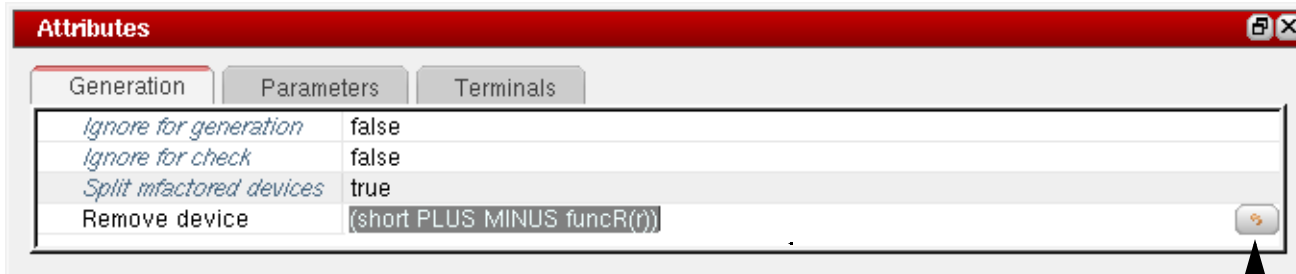
- Hover your cursor over a label to see more information on where the current value comes from.



Attributes		
Generation	Parameters	Terminals
<i>ignore for generation</i>	false	
<i>ignore for check</i>	false	
<i>Split mfactored devices</i>	true	
Remove device	false	R(r))

ignore for check
false
Not changed from default value.

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



Attributes		
Generation	Parameters	Terminals
<i>ignore for generation</i>	false	
<i>ignore for check</i>	false	
<i>Split mfactored devices</i>	true	
Remove device	(short PLUS MINUS funcR(r))	

Revert to default

Hierarchy Configuration Generation Attributes

The options on this tab let you override a number of the properties that affect layout generation, previously stored with the schematic.

Attributes		
Generation	Parameters	Terminals
<i>ignore for generation</i>	false	
<i>ignore for check</i>	false	
<i>Split mfactored devices</i>	true	
Remove device	(short PLUS MINUS funcR(r))	

Ignore for layout generation prevents the selected instance or cell from being generated in the layout by the *Generate All From Source* or *Generate Selected From Source* commands. This includes any instance or cell with the ignore property set and any instance or cell with a property listed in the propsUsedToIgnoreObjs environment variable.

Ignore for check prevents the selected instance or cell from being checked by the *Check Against Source* command. Any instance or cell that is ignored for generation is automatically ignored for check as well, along with any instance or cell which has a property listed in the propsUsedToIgnoreObjsForCheck environment variable.

Split mfactored devices controls whether Layout XL generates a schematic device with the mfactor property as a single device (*off*) or as multiple devices (*on*) in the layout. This option overrides the global option set on the Layout XL Options Generation Tab.

Note: *Split mfactored devices* controls *only* whether Layout XL generates one device or several devices in the layout view. It does not change the widths of the generated devices. It is grayed out if the selected schematic instances do not have the `mfactor` property.

Remove device causes parasitic devices to be ignored by merging nets connected to the terminals of a single instance. For more information, see Ignoring Parasitic Devices.

Related Topics

[Hierarchy Configuration Instances Table](#)

[Hierarchy Configuration Cells Table](#)

[Hierarchy Configuration Parameter Attributes](#)

[Hierarchy Configuration Terminal Attributes](#)

Configure Physical Hierarchy

Hierarchy Configuration Parameter Attributes

The options on this tab let you specify the names of parameters to be ignored during layout generation or by *Check Against Source*, how parameter names are mapped between schematic and layout, and how parameter values are rounded.

Attributes	
Generation	Parameters
Ignore for generation	param1 param2
Ignore for check	param0
Name mapping	l L ; w W ;
<i>Rounding</i>	

Ignore for generation lists the parameters that are ignored by the following commands; *Generate All From Source*, *Generate Selected From Source*, *Generate Clones*, *Check Against Source*, *Update Components And Nets*, *Update Layout Parameters*, *Update Schematic Parameters*.

The following parameters are always ignored.

instancesLastChanged	lxPlacementStatus
instNamePrefix	lxRounding
lxIgnoreParamForCAS	lxStopList
lxIgnoredParams	lxTimeStamp
lxMFactorNum	lxUseCell
lxParamsToIgnore	pin#
lxParamsToIgnoreForCheck	posi

Ignore for check lists the parameters that are ignored by the *Check Against Source* command. Mismatches for any of the listed parameters are not reported by the check. The list inherits all the parameter names from the *Ignore for generation* list.

Name mapping defines the mapping between parameter names in the schematic and layout cellviews; for example,

```
l L ; w W ;
```


Maps schematic parameters l and w to layout parameters L and W respectively. This field is not available as an instance attribute because you cannot override this setting for an individual instance of a cell.

Rounding defines how the value of a specified parameter on a schematic cell master or instance is rounded when it is evaluated by Layout XL. You can use this property in conjunction with the `lxDeviceWidth` parameter to prevent folded devices from becoming off-grid; for example, the following setting rounds the value of w to the closest multiple of 0.05 microns.

```
(w 0.05 round)
```

For more information on setting this option, see [Rounding Parameter Values](#).

Related Topics

[Hierarchy Configuration Instances Table](#)

[Hierarchy Configuration Cells Table](#)

[Hierarchy Configuration Generation Attributes](#)

[Hierarchy Configuration Terminal Attributes](#)

[Configure Physical Hierarchy](#)

Hierarchy Configuration Terminal Attributes

The options on this tab let you specify the names of terminals to be ignored during layout generation or by *Check Against Source*, and how terminal names are mapped between schematic and layout.



Attributes		
Generation	Parameters	Terminals
Ignore for generation		
Ignore for check		
Name mapping		B G ; X D ;

Ignore for generation lists the terminals that are ignored by the following commands; *Generate All From Source*, *Generate Selected From Source*, *Generate Clones*,

Check Against Source, Update Components And Nets, Update Layout Parameters, Update Schematic Parameters.

Ignore for check lists the terminals that are ignored by the *Check Against Source* command. Mismatches for any of the listed terminals are not reported by the check.

Name mapping defines the mapping between terminal names in the schematic and layout cellviews; for example,

`B G ; X D ;`

Maps schematic terminals `B` and `X` to layout terminals `G` and `D` respectively. This field is not available as an instance attribute because you cannot override this setting for an individual instance of a cell.

Related Topics

[Hierarchy Configuration Instances Table](#)

[Hierarchy Configuration Cells Table](#)

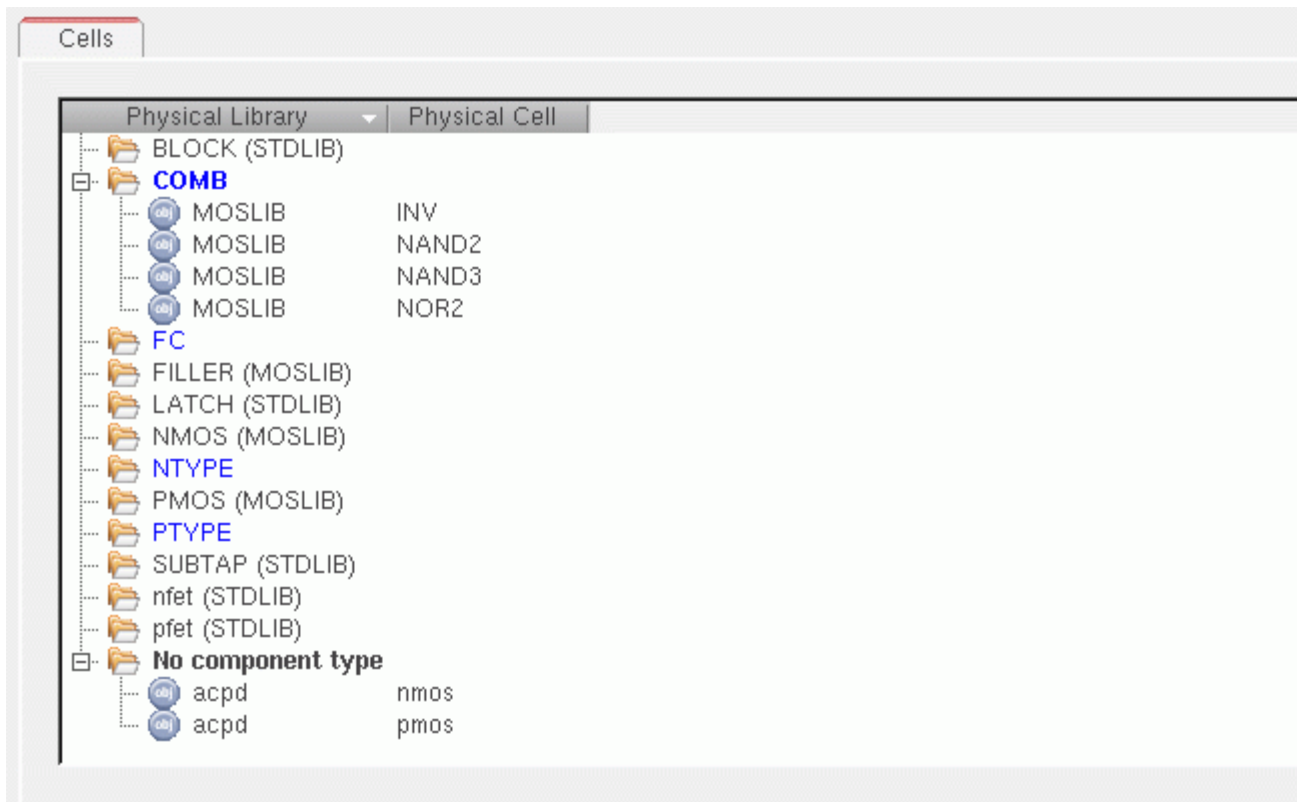
[Hierarchy Configuration Generation Attributes](#)

[Hierarchy Configuration Parameters Attributes](#)

[Configure Physical Hierarchy](#)

Component Types Cells Table

In the Component Types mode, you can create, edit, and remove component types, which identify NMOS and PMOS transistor cells, set the parameters for device chaining and folding, and control how components are placed in rows.



In this mode the cells table shows the cells in the design organized by component type. If a cell is not assigned to a component type, it is put in the *No component type* folder.

Related Topics

[Component Types Attributes](#)

[Configure Physical Hierarchy](#)

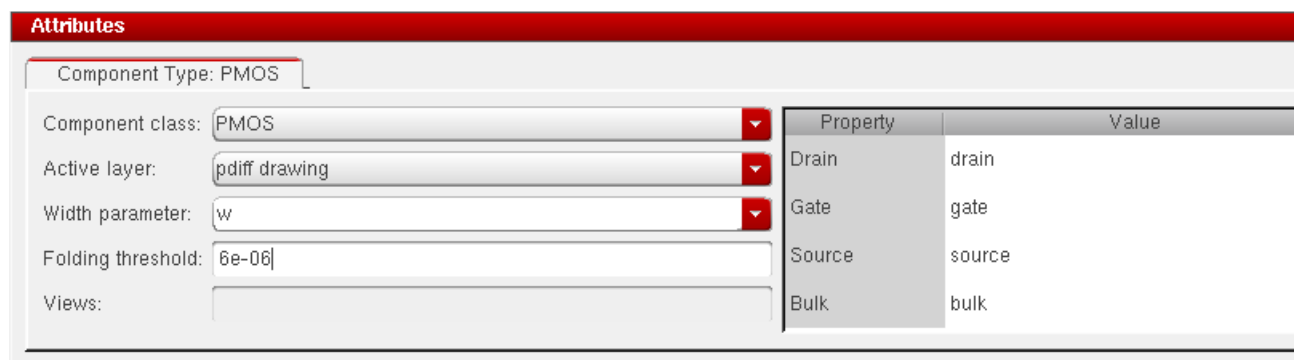
Context Menu for the Component Types Table

Click the right mouse button to display a context menu with a number of other options. These are enabled and disabled dynamically depending on what is selected in the table view.

Command Name	Description
<i>Add component type</i>	Opens the Create Component Type dialog which lets you create a new component type to be added to the table view. For more information, see Defining a Design-Level Component Type.
<i>Remove component type</i>	Removes the selected component type from the table view. You can remove only design-level component types. Evicted cellviews are put into the <i>No component type</i> folder. For more information, see Removing a Component Type.
<i>Move cells</i>	Opens the Move Cells dialog, which lets you move selected cells between component types. For more information, see Defining a Design-Level Component Type.
<i>Show cells</i>	Choose <i>In design</i> (the default) only to show the physical cells that will be generated in the layout based on the current physical configuration view. Choose one or more of the library names to toggle the display of cells from that library. The submenu lists all the libraries from the library definitions file, <code>cds.lib</code>. For more information, see Component Types Cells Table.
<i>Options</i>	Specifies how the data is presented in the tree, click the right mouse button and choose <i>Options</i>. For information on the Options form, see The Navigator Options Form in the <i>Virtuoso Schematic Editor XL User Guide</i>.

Component Types Attributes

The **Attributes** pane lets you edit the attributes for the selected component type. These controls are enabled only when you have a single component type selected in the *Cells* table.



Property	Value
Drain	drain
Gate	gate
Source	source
Bulk	bulk

Component class specifies the type of devices in the component type. Valid values are *PMOS*, *NMOS*, *STDCELL*, *STDSUBCONT*, *FILLER*. Leave blank to set to undefined. The component class is required to identify device types for folding, chaining, and assisted MOS and standard cell operations in Layout XL and the Custom Digital Placer.

- Choose *PMOS* or *NMOS* if you are using the placer's MOS component-assisted row generation. The *Views* field is grayed out.
- Choose *STDCELL* if you are using the placer's Standard Cell component-assisted row generation. All other settings are grayed out.
- Choose *STDSUBCONT* or *FILLER* for substrate contacts and filler cells respectively. Only the *Views* field is enabled; everything else is grayed out.

If the component class is defined, it is used to determine the device type for chaining. If the component class is not defined, the chaining engine looks for default terminals with names—s, S, g, G, d, D.

- If default terminals exist, the master cell name is used to determine the device type as follows:
 1. If the master cell name contains “pmos”, the device type is set to *PMOS*.
 2. If the master cell name contains “nmos”, the device type is set to *NMOS*.
 3. If the master cell name contains “P” or “p”, the device type is set to *PMOS*.
 4. Otherwise, the device type is set to *NMOS*.

Note: The search for “nmos” and “pmos” is case insensitive.

- If no default terminals exist, a warning message is generated.

Active layer specifies the diffusion layer-purpose pair for NMOS and PMOS devices. The drop-down lists all the layers from the technology file for which the function is set to one of `pdiff`, `ndiff`, `pwell`, `nwell`, `pplus`, `nplus`, or `diffusion`.

This parameter is required for NMOS and PMOS component types in order to control device chaining and folding during placement. The Custom Digital Placer also uses this layer to determine the spacing rules to be used when seeding the defined spacing type in the fields on the [Placement Planning](#) (Assisted CMOS) form. The *Calculate Estimates* command for Assisted CMOS placement uses the spacing rule between adjacent MOS device chains to achieve more accurate estimates.

Width parameter specifies the name of the transistor width parameter on the device master cell. The default name is `w`. If you use a different name for the transistor width parameter, type that name into the text field. You must set this parameter for *NMOS* and *PMOS* component types even if the devices in the component type will not be folded.

Folding threshold specifies the maximum width beyond which transistors are automatically folded. You must set this parameter for *NMOS* and *PMOS* component types even if the devices in the component type will not be folded. The value applies globally to all devices within the scope of the component type definition (i.e., the library containing the current cell, or just the current cell).

You can override this value using *Generate All From Source* and *Generate Selected From Source* commands, or by setting the *PMOS Width Threshold* and *NMOS Width Threshold* in the [Placement Planning](#) form.

Layout XL cannot enforce the *Folding threshold* unless all device widths and all values are specified in consistent units; for example, all in meters (10um is 1-e-5) or in all in user units (10um is 10).

Views specifies the view name to be used when defining a standard cell substrate contact or filler cell. This parameter is valid only when the *Component class* is set to *STDSUBCONT* or *FILLER*.

Property/Value lets you edit the default names used to identify *Drain*, *Gate*, *Source*, and *Bulk* terminals in the cells assigned to the component type. The names are not case-sensitive. The defaults are as follows: `drain` and `D`; `gate` and `G`; `source` and `S`; and `bulk` and `B` respectively.

Related Topics

[Component Types Cells Table](#)

[Configure Physical Hierarchy](#)

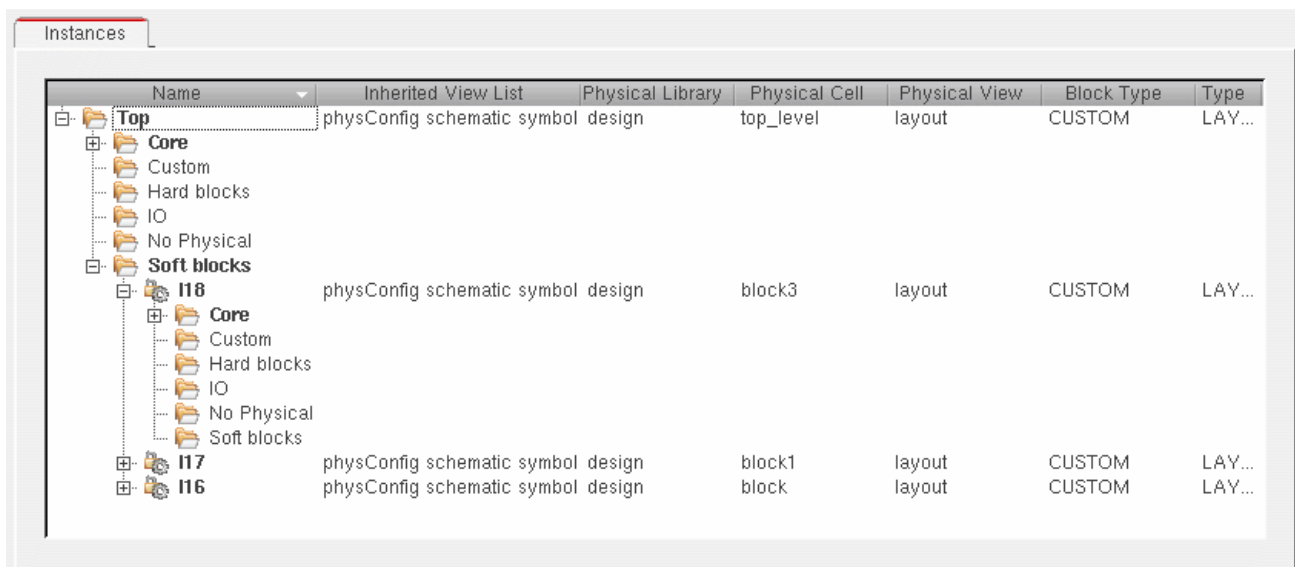
Virtuoso Layout Suite XL User Guide

Layout XL Forms

Soft Block Instances Table

In Soft Block mode you can partition your logical design by defining and editing parameters for the soft blocks that will be created when you generate the physical hierarchy.

The **Instances** table lists the instances in the design organized by cell type into *Core*, *Custom*, *Hard Blocks*, *I/Os*, and *Soft Blocks*. Instances with no physical view are also grouped together.



Name	Inherited View List	Physical Library	Physical Cell	Physical View	Block Type	Type
Top	physConfig schematic symbol design	top_level	top_level	layout	CUSTOM	LAY...
Core						
Custom						
Hard blocks						
IO						
No Physical						
Soft blocks						
I18	physConfig schematic symbol design	block3	block3	layout	CUSTOM	LAY...
Core						
Custom						
Hard blocks						
IO						
No Physical						
Soft blocks						
I17	physConfig schematic symbol design	block1	block1	layout	CUSTOM	LAY...
I16	physConfig schematic symbol design	block	block	layout	CUSTOM	LAY...

For information on the columns, table folders, and state indicator icons, see [Soft Block Instances Table](#).

Context Menu for the Soft Block Instances Table

Click the right mouse button to reveal a context menu with a number of other options. These are enabled and disabled dynamically depending on what is selected in the table view.

Command Name	Description

Virtuoso Layout Suite XL User Guide

Layout XL Forms

<i>Define Soft Block Parameters</i>	Marks the selected nodes of a schematic or netlist view so that the <i>Floorplan – Generate Physical Hierarchy</i> command can create soft blocks for them in the physical hierarchy.
<i>Define Top Level Parameters</i> (if the top-level block is selected)	For more information, see Defining Soft Block Parameters. Note: If the selected component already has soft block parameters defined, this menu item changes to <i>Edit Soft Block Parameters</i> (or <i>Edit Top Level Parameters</i> if the top-level block is selected).
<i>Initialize Soft Block Parameters Using Physical View</i>	Imports soft block parameters from another layout cellview. Note: This command uses the same functionality as the <i>Load Physical View</i> command. For more information, see Loading Soft Block Parameters from Another Cellview.
<i>Remove Soft Block Parameters</i>	Removes the soft block parameters from the selected nodes in the tree. For more information, see Removing Soft Block Parameters.
<i>Set Master Cell Type</i>	Sets the cell type for the selected instances. Cell types are used to categorize the blocks in your design. For more information, see Setting the Cell Type.
<i>Create Physical Binding</i>	Sets or clears a physical binding for the selected instance. For more information, see Creating a Physical Binding.
<i>Clear Physical Binding</i>	

Attributes lets you edit the attributes for the selected soft block. Click *Save Soft Block* to save all the attribute settings for the block; *Discard Changes to Soft Block* to discard any changes you have made since the last time the block was saved.

Related Topics

[Soft Block Boundary Attributes](#)

[Soft Block I/O Pins Attributes](#)

[Soft Block Obstructions Attributes](#)

[Configure Physical Hierarchy](#)

Soft Block Boundary Attributes

The **Boundary** attributes let you specify the shape and size of the boundary that encloses the soft block.

Rectangle specifies a rectangular boundary. Choose two of the available parameters to determine the dimensions of the boundary for a given value of **Area**.

- **Aspect Ratio (W/H)**
- **Boundary Width (W)**
- **Boundary Height (H)**
- **Utilization**

Choose one of the above from the first cyclic list and any one of the remaining valid choices from the second cyclic list.

If one of the parameters you choose is *Utilization*, you need to specify an area to which the utilization percentage can be applied. The **Area Calculation** group box offers you three ways to specify the area value.

Manual lets you type the value you want into the **Area** field at the bottom of the form.

Avg. Area Per Gate lets you specify the gate count and the average area per gate. The system calculates the area by multiplying the two values.

Use Estimator lets you choose between *PR Boundary Based* and *BBox Based* area estimator. The PR Boundary Based estimation sums up the polygonal area of the individual instances to derive the overall area estimation. The BBox Based estimation, on the other hand, sums up the minimum bounding box area of the individual instances to derive the overall area estimation.

You can also register your own area estimation functions and use them, as appropriate. For more information, see [Creating and Registering a User Defined Area Estimation Function](#).

Click **Estimate** to modify any user-defined parameters for the area estimation function before running the function.

Polygon specifies a polygonal boundary with no area estimation. Specify the coordinates required to define the boundary in the fields provided.

Related Topics

[Specifying the Boundary](#)

[Soft Block Instances Table](#)

[Soft Block I/O Pins Attributes](#)

[Soft Block Obstructions Attributes](#)

[Configure Physical Hierarchy](#)

Soft Block I/O Pins Attributes

The **I/O Pins** attributes pane lists the I/O pins in the design. All fields with a white background are editable. By default the table lists all the top-level pins detected in the schematic design, including schematic pins that are connected to I/O pads (cells of type `pad`, `padSpacer`, or `padAreaIO`). To generate only pads but no pins, set the `initCreatePadPins` environment variable to `nil`.

If the soft block has already been generated in the layout, the *Layer*, *Width*, and *Height* fields show the actual dimensions and layer purpose of the pin shape in the layout. If the soft block has not been generated, the default minimum width and height specified in the technology file are shown, while the default layer is specified by the Layout XL `initIOPinLayer` environment variable.

Use the **Add** button to add a new pin to the table and the **Delete** button to delete a selected pin.

Check the **Create Label** box to automatically generate a label for each of the pins in the current soft block.

Note: To set the style for the labels, choose *Options – Soft Block Options* from the Configure Physical Hierarchy menu bar and choose the *Create Label* check box to activate the options under the *Pin Label Text Style* group box.

Related Topics

[Specifying I/O Pins](#)

[Creating Pin Labels](#)

[Soft Block Instances Table](#)

[Soft Block Boundary Attributes](#)

[Soft Block Obstructions Attributes](#)

[Configure Physical Hierarchy](#)

Soft Block Obstructions Attributes

The **Obstructions** attributes pane lets you create routing and placement obstructions.

Blockage Type specifies the type of obstruction to be created. Choose from *placement, routing, fill, slot, pin, feedthru, or screen*.

Material filters the list of available layers based on the material defined for that layer in the technology file. This field is not available when defining a placement obstruction.

Layer specifies the layer on which the obstruction is created. This field is not available when defining a placement obstruction.

Offsets define the position of the obstruction.

Click *Add* to add the obstruction to the soft block definition, or *Delete* to delete a selected obstruction.

Note: You can create only one placement obstruction per soft block.

Related Topics

[Specifying Obstructions](#)

[Soft Block Instances Table](#)

[Soft Block Boundary Attributes](#)

[Soft Block I/O Pins Attributes](#)

[Configure Physical Hierarchy](#)

Convert Libraries to Use Physical Configuration Views

Use the **Convert Libraries to Use Physical Configuration Views** form to specify which schematic libraries and designs are converted to use the IC 6.1 Layout XL schema, and what base name is used for the physical configuration views that are created.

Libraries not to convert lists the libraries referenced in the current library definitions (`cds.lib`) file that will not be converted.

Libraries to convert lists the libraries that will be converted when you click *OK*. The software automatically moves all the valid libraries defined in the current library definitions (`cds.lib`) file into this field.

Note: Reference libraries shipped with Cadence software are not shown in either list. If you do not have permission to write to a library, that library is not converted.

Configuration view base name specifies the base name of the physical configuration views to be generated. This lets the system automatically add the physical configuration view names to the *Logical switch view list*. The default is `physConfig`.

If there are multiple schematic views to be converted, the base name is prepended to each schematic view name. For example, if your cell has the following views before conversion:

```
schematic
schematic1
schematic2
```

Then it will have the following views after conversion:

```
physConfig
physConfig_schematic1
physConfig_schematic2
```

Related Topics

[Configuring the Physical Hierarchy](#)

[Converting Schematic Libraries and Designs](#)

[Converting Data to Use the IC 6.1 Layout XL Schema](#)

[Layout XL Forms](#)

Copy

Snap To Grid snaps the pin edges to the manufacturing or the routing grid, depending on the selected block type:

- If the block type is custom, the pin edges are snapped to the manufacturing grid.
- If the block type is digital, the pin edges are snapped to the routing grid.

Create Synchronous Copy creates a synchronized copy of the selected layout objects. For more information, see [Generating a Synchronous Copy](#).

For information about the other options available in this form, see the [Copy Form](#) in the *Virtuoso Layout Suite L User Guide*.

Related Topics

[Repeat Copy](#)

[Generating a Synchronous Copy](#)

[Layout XL Forms](#)

Create Cluster Boundary

Use the **Create Cluster Boundary** form to specify a cluster boundary to be created around one or more cluster objects.

Cluster Name lists the clusters in the design. You create a cluster using the *Create – P&R Objects – Clusters* command. You can define clusters as *inclusive*, which means that you can place them in the cluster boundary, or *exclusive*, which means that they are excluded from and cannot be placed in the cluster boundary.

Cluster Boundary Name lets you specify a unique name for the cluster boundary. By default, the system assigns a name of the form *CB_integer*, where *integer* is incremented each time a cluster boundary is created.

Fit Member Instances Into Cluster Boundary places all the components in a cluster inside the specified cluster boundary. Components are placed according to size from left to right and bottom to top. That means that the largest object is placed in the bottom left corner and the smallest object in the top right corner. If there is not enough space, the components in the top row are overlapped.

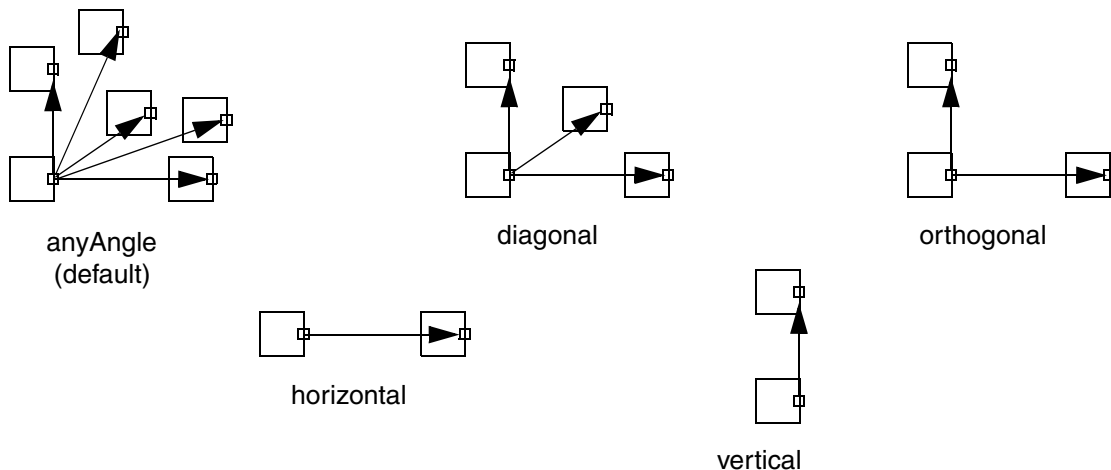
Cross-Highlight Boundaries highlights in the layout window the cluster boundary associated with the cluster selected in the form. When generating new cluster boundaries, use this option to help you associate existing clusters with their boundaries.

Create Method specifies whether boundary creation is **Manual** or **Automatic**.

If you choose manual, you can specify the cluster boundary by clicking with your mouse in the layout canvas.

- ☐ **Rectangle** lets you create a rectangular boundary by clicking twice in the canvas.
- ☐ **Polygon** lets you create a polygonal boundary by clicking once for each vertex in the canvas. The shape you draw must have at least three points and must not be self-intersecting. To close the polygon, double-click.

Use **Snap Mode** to select the types of shapes you can draw. Valid modes are:



If you choose automatic, the system creates a boundary for you based on the origin, aspect ratio and area information you specify.

Aspect Ratio (W/L) sets the width-to-length ratio of the cluster boundary. For example, a value of 1 specifies a square boundary, 2 specifies a boundary twice as high as it is long, and 0.5 specifies a boundary half as high as it is long. The default is 1.

Environment variable: `clusterBoundaryAspectRatio`

Origin X and **Origin Y** set the origin point.

Area Information specifies a target area for the cluster boundary or tells the system to estimate the area for you.

Area Estimator lets you estimate the target area based on the place and route boundary, the bounding box of the components contained in the cluster, or using a user-defined estimator you have registered in the system.

The *Estimate* button is grayed out unless there is a registered estimator. Use the `leRegClusterBdyEstimator` SKILL function to register an estimator.

Estimated Area For Cluster displays the estimated area required to accommodate the components in the cluster depending on the other settings in the form.

Utilization (%) specifies the percentage of area within the cluster boundary that you want to fill with components. The lower the figure, the more free space is available.

Target Area lets you specify a target area for the cluster boundary.

Related Topics

[Layout XL Forms](#)

Create Custom Placement Area

Use this form to create a custom placement area.

For information on the fields available in this form, see [Custom Placement Area Form](#) in the *Virtuoso Layout Suite L User Guide*.

Important

When using the form to manually create a custom placement area in the Layout XL or GXL environment, the custom placement area is automatically snapped to an appropriate grid depending on the state of the [snapCpaToPlacementGrid](#) environment variable.

- *Scenario 1: The [snapCpaToPlacementGrid](#) environment variable is set to `nil`.*

When the [snapCpaToPlacementGrid](#) environment variable is set to `nil`, which is the default state of the environment variable, the custom placement area is snapped to the X Snap and Y Snap spacing, provided the following conditions are fulfilled:

- a. X Snap and Y Snap spacing are multiples of the manufacturing grid, and
- b. Both, X Snap and Y Snap spacing are equal to or greater than the manufacturing grid.

If either of the snapping conditions is not met, the custom placement area is snapped to the manufacturing grid.

- *Scenario 2: The [snapCpaToPlacementGrid](#) environment variable is set to `t`.*

When the [snapCpaToPlacementGrid](#) environment variable is set to `t`, the custom placement area is snapped to the placement grid. If the placement grid is unavailable, the conditions for snapping to the X/Y spacing are verified. If both the conditions are fulfilled, the placement area is snapped to the X-snap and Y-snap spacing. Else, the placement area is snapped to the manufacturing grid.

Related Topics

[Layout XL Forms](#)

Create Feed Through Terminal Block Pins

Use the **Create Feed Through Terminal Block Pins** form to create additional feed through pins in order to improve the connectivity of your design.

Soft block list lists all the soft blocks that were selected when you opened the form. You can create soft pins for any of the blocks that are listed this field. The *Lib*, *Cell*, and *View* fields confirm the currently selected block.

Select Net/Bundle helps you select nets or net bundles. To assist you in defining constraints, the canvas shows flight lines to represent the nets between the terminals you select.

Search for filters the design objects that are searched. Choose one of *Nets*, *Bus*, or *Net Class*.

Name is where you type in the search expression. Click *Filter* to see the results.

Define TTB pin parameters lets you specify in detail the pins you want to create. Use the *Add*, *Update*, and *Delete* buttons to update the pin table with the pins and parameters you provide.

User defined terminal name specifies the name of the terminal on which the pins will be generated. If you do not specify a name, the system assigns one for you.

Layer specifies the layer purpose on which the pin is to be created. Choose the layer purpose you want from the list of extractable layers in the design.

Width and **Height** specify the width and height of the new pin. These fields are prepopulated with the `minWidth` spacing defined in the technology file for the selected pin layer.

Num specifies the number of pins to create on the specified terminal. The default is 1.

Side specifies on which sides of the place and route boundary the pins can be created. If you specify more than one side for a pin, the pin optimizer can place the pins on any of the specified sides. The images indicate what each constraint means in the context of both a rectangular and polygonal boundary.

Align Pin helps you to align the pins on a particular layer purpose.

Related Topics

[Soft Block Pins](#)

[Creating a Feedthrough Terminal Pin](#)

Virtuoso Layout Suite XL User Guide

Layout XL Forms

Layout XL Forms

Create Instance

Use the **Create Instance** form to specify a new instance to be created in your design.

The **Library**, **Cell**, and **View** fields set the library, cell, and view names of the master cell you want to place as an instance in the current. Use the *Browse* button to select the names by clicking on them in a file browser.

Names assigns a name to the instance. You can type any unique name or let the layout editor automatically assign a name beginning with the letter I, followed by a number. You can enter multiple names (separated by a space) to place several instances of the same cell. If the design has embedded module hierarchy (EMH), you can specify a hierarchical path name for creating the instance at the desired location in the EMH. See [Embedded Module Hierarchy](#).

Mosaic

Rows and **Columns** set the number of rows and columns in an array of instances.

Delta X and **Delta Y** set the spacing between rows and columns in an array of instances.

Associate To Cluster adds the instance being created to an existing cluster in the design.

Cluster Name specifies the name of the cluster to which the instance is added. Choose an existing cluster from the drop-down list or type the name of a new cluster into the field.

Cluster Type specifies the type of cluster; *inclusive*, *exclusive*, or *suggested*.

Halo turns on or off the ability to define halo blockages for the instance. Click the *Define Halo* button to open the Halo Editor form.

Physical Only creates an instance in the layout which has no equivalent in the schematic source. This option is enabled only for instances and is disabled for mosaics. Once you create an instance as a physical-only instance, you cannot change its domain visibility.

The orientation buttons at the bottom of the form let you change the orientation of the selected instance.

Rotate rotates the component 90 degrees counterclockwise.

Sideways mirrors the component on the y axis (flips it horizontally).

Upside Down mirrors the component on the x axis (flips it vertically).

Related Topics

[Layout XL Forms](#)

Create Path

Use the **Create Path** form to specify how a path is created in your design. This form is the same as the Create Path form available in Layout L; the only difference is the addition of the *Probe Nets* option.

Width specifies the path width in user units. The path will initially default to the minimum width defined for the layer on which you create the path unless you enter a new value in this field. Once you have entered a new value in the field, each time you create a path on the layer whose value you have changed, the value is reused until you enter a new value in the field. To return to the minimum width, click the *Defaults* button.

Fixed Width, when on, uses the width you specify in the *Width* field for all segments of the path. When off, the path width remains set at the last setting until you click the *Defaults* button. The *Defaults* setting uses the `minWidth` defined in the technology file for the current layer.

Justification controls which edge of the path you create: *left*, *center*, or *right*. If you set a path *Offset*, *Justification* controls which edge of the path is offset from the line you create.

Offset offsets the line you use to create the path from the final path. To control which edge of the path is offset, set the *Justification* cyclic field.

End Type controls how the path ends are created.

truncate means that the path ends and path points end at the same points.

extend means that the path ends extend from the path points by one half the path width.

round means that the path ends extend from the path points by one half the path width, symmetrically in four points at each end.

variable means that the path ends extend from the path points by whatever value you enter in the *Begin Extension* and *End Extension* fields.

Begin Extension and **End Extension**, when *End Type* is set to *variable*, let you enter a beginning and an ending extension in user units.

Net Name assigns the path to a specific net.

As ROD Object creates the path as a ROD object. **ROD Name** assigns a name to the new path in the current layout cellview if *As ROD Object* is on. The name must be unique in the cellview. If you do not edit this field, the system assigns a unique name consisting of the prefix `path`, followed by a number. For example, the first ROD path would be named `path0`, the second, `path1`, and so on.

Snap Mode controls how path segments snap to the grid.

anyAngle creates lines at any angle.

diagonal creates lines parallel to the X or Y axis or at a 45-degree angle to the axes.

orthogonal creates lines parallel to the X or Y axis.

L90XFirst creates orthogonal two-segment lines and creates the first line in the X direction.

L90YFirst creates orthogonal two-segment lines and creates the first line in the Y direction.

To toggle between *L90XFirst* and *L90YFirst* while you are creating a path, click right.

Acute Angle (available only when *Snap Mode* is set to *anyAngle*) When set to on, you can create acute angle paths. When set to off, you can create paths only from 90 degree to 180 degree angles.

Connectivity

Probe Nets highlights nets while you are creating interconnect.

Related Topics

[Creating Objects](#)

[Selecting a Layer Automatically when Creating a Path or Shape](#)

[Layout XL Forms](#)

Create Physical Configuration View

Use the **Create Physical Configuration View** form to create a new physical configuration view to be used in a Layout XL session.

Library Name is the name of the library in which the physical configuration view resides.

Cell Name is the name of the cell to which the physical configuration view relates.

View Name is the name of the physical configuration view.

Use Template specifies a template or file containing predefined values for the new physical configuration view.

Template lets you select a predefined template file from the drop-down list.

From File lets you specify a template file stored in another location.

If you do not specify a template or file, Layout XL uses the default template and any overriding values from the corresponding environment variables.

Open CPH opens the [Configure Physical Hierarchy](#) window automatically after you click *OK* in this form.

Related Topics

[Configuring the Physical Hierarchy](#)

[Starting Layout XL from a Schematic View](#)

[Starting Layout XL with a New Physical Configuration View](#)

[Layout XL Forms](#)

Create Polygon

Use the **Create Polygon** form to specify the polygons to be drawn in your layout design. This form is the same as the Create Polygon form available in Layout L; the only difference is the addition of the *Probe Nets* option.

Snap Mode limits how the cursor snaps when you create a polygon.

anyAngle creates lines at any angle.

diagonal creates lines parallel to the X or Y axis or at a 45-degree angle to the axes.

orthogonal creates lines parallel to the X or Y axis.

L90XFirst creates orthogonal two-segment lines and creates the first line in the X direction.

L90YFirst creates orthogonal two-segment lines and creates the first line in the Y direction.

Click *Create Arc* to create an arc in the polygon.

Net Name assigns the polygon to a specific net.

As ROD Object creates the polygon as a ROD object. **ROD Name** assigns a name to a new polygon in the current layout cellview if *As ROD Object* is on. The name must be unique in the cellview. If you do not edit this field, the system assigns a unique name consisting of the prefix `polygon`, followed by a number. For example, the first ROD polygon would be named `polygon0`, the second, `polygon1`, and so on.

Connectivity

Probe Nets highlights nets while you are creating interconnect.

Related Topics

[Creating Objects](#)

[Selecting a Layer Automatically when Creating a Path or Shape](#)

[Layout XL Forms](#)

Create Rectangle

Use the **Create Rectangle** form to specify the rectangles to be drawn in your layout design. This form is the same as the Create Rectangle form available in Layout L; the only difference is the addition of the *Probe Nets* option.

Net Name assigns the rectangle to a specific net.

As ROD Object creates the rectangle as a ROD object. **ROD Name** assigns a name to the new rectangle in the current layout cellview if *As ROD Object* is on. The name must be unique in the cellview. If you do not edit this field, the system assigns a unique name consisting of the prefix `rect`, followed by a number. For example, the first ROD rectangle would be named `rect0`, the second, `rect1`, and so on.

Connectivity

Probe Nets highlights nets while you are creating interconnect.

Related Topics

[Creating Objects](#)

[Selecting a Layer Automatically when Creating a Path or Shape](#)

[Layout XL Forms](#)

Define Device Correspondence

Use the *Define Device Correspondence* form to view and change the device correspondence for the instances and terminals in your design.

Note: The *Define Device Correspondence* form is also available in the Read mode in Layout XL. However, in this mode, the *Map* and *Unmap* buttons are inactive.

Device correspondence defined using the *Define Device Correspondence* form is persistent between Layout XL sessions, and can be preserved during *Generate All From Source* and *Update Components And Nets* using the *Preserve User-Defined Mappings* option in the Generate Tab of the *Generate Layout* form.

Important

Check Against Source does not report any issues found in user-defined bindings created using this form.

You can leave the form open on your desktop while you manipulate your design in the layout window; the form is dynamically updated to reflect the selections and changes you make. Note that for performance reasons, the form does not update dynamically if you select more than 100 objects from the design canvas. To define device correspondence for a large number of objects, select them in the form itself.

Filter lets you select what devices are listed in the *Schematic* and *Layout* panes. Choose between *All Instances*, *All Terminals*, *Unbound Instances*, and *Unbound Terminals*.

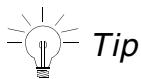
Important

The *All Terminals* and *Unbound Terminals* filters show the terminals in the design, not the pin figures that typically represent them in the layout window. If you delete a pin figure in the layout window, the corresponding terminal is removed from the *Define Device Correspondence* form only if the net connected to this terminal has no other connections. If the net has other connections, the terminal is not removed. Similarly, if you delete an instance, the terminals connected to the instance terminal's nets are deleted only if they are no longer connected in the design.

Schematic lists all the devices in the schematic view apart from those that are ignored for generation. An object or pin icon next to the name indicates that the device is mapped to a device in the layout. A red check mark indicates that it is not mapped. If you select a mapped device, the corresponding devices in the *Layout* pane are also selected.

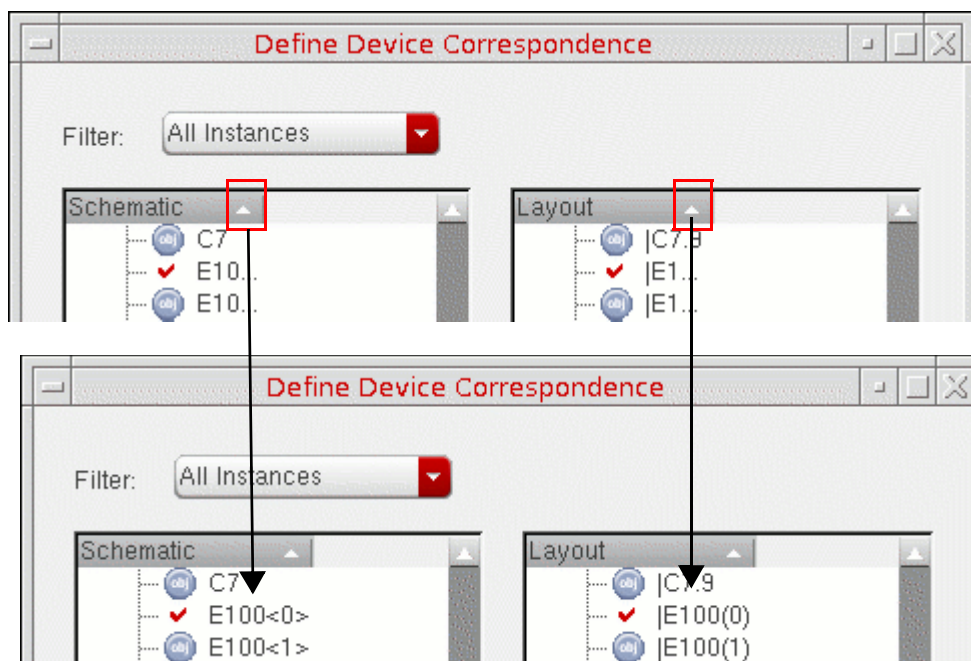
Layout lists all the devices in the layout view apart from those that are ignored for generation. An object or pin icon next to the name indicates that the device is mapped to a device in the

schematic. A red check mark indicates that it is not mapped. If you select a mapped device, the corresponding devices in the *Schematic* pane are also selected.



Tip

If you cannot see the entire instance or terminal name, hover your cursor over the column header in the areas indicated below to show the Resize cursor. When the Resize cursor is visible, you can drag to resize the column to the width you want, or double-click to expand the column to fit the longest name.



Update layout instance parameters to match correspondence updates the parameter values on the layout instance to match those of the corresponding schematic instance.

Update layout instance name to match correspondence updates the name of the layout instance name to match the name of the corresponding schematic instance.

When you have made your selections, use the buttons to perform the required action.

Map creates a correspondence between the devices selected in the *Schematic* pane and the devices selected in the *Layout* pane. This button is enabled only if all the selected devices are unmapped.

Unmap deletes the existing correspondence between mapped devices selected in the *Schematic* and *Layout* panes.

Deselect All clears the selected set in both panes.

Related Topics

[Updating Device Correspondence](#)

[Defining a One-to-One Device Correspondence](#)

[Defining a One-to-Many Device Correspondence](#)

[Defining a Many-to-One Device Correspondence](#)

[Defining a Many-to-Many Device Correspondence](#)

[Deleting a Device Correspondence](#)

[Define Device Correspondence SKILL API](#)

[Layout XL Forms](#)

Design Summary

The **Design Summary** form lets you customize the content of the design summary that is generated. Choose the statistics you want to see and click *OK*.

By default, the system generates the same *Basic Summary* as in Layout L. You can choose to add one or more of *Net Statistics*, *Connectivity Statistics*, and *Routing Statistics*.

Related Topics

[Layout XL Forms](#)

Device List

The **Device List** form shows all the devices in the design. Use it to select specific devices when using the *Update Layout Parameters* and *Update Schematic Parameters* commands.

Search finds and selects the names of devices you specify.

You can type the following in the field to the right of the *Search* button.

- ☐ A device name. The Device List form highlights the device and scrolls the list to make the device name visible.
- ☐ Part of a device name (for example, `IN`). The Device List form highlights all the devices with `IN` as an element in the name (`IN3`, `IN6`, and so on) and scrolls the list to make the first example visible.
- ☐ A regular SKILL expression using part of a device name (for example, `*mos`). The Device List form highlights all devices with the string `*mos` (`nmos`, `pmos`, for example) and scrolls the list to make the first example visible.

Related Topics

[Updating Design Data](#)

[Updating Layout Parameter Values in Postselection Mode](#)

[Updating Schematic Parameter Values in Postselection Mode](#)

[Layout XL Forms](#)

Edit Net Attributes

Use the **Edit Net Attributes** form to modify the criticality and signal type of existing nets in the design.

If you select multiple nets, use the *Previous* and *Next* buttons to cycle through the selected nets. The **Net Name** field shows you which net you are currently editing.

Common specifies that the current settings are to be applied to all the selected nets. When switched off, the settings are applied only to the net listed in the **Net Name** field.

Criticality sets the criticality of the specified net (or all nets). The default value is 10 and the specified value must be in the range -128 to 128.

Signal Type sets the signal type of the specified net. Choose one from *signal*, *supply*, *ground*, *clock*, *tieOff*, *tieLo*, *tieHi*, *analog*, *scan* and *reset*.

Related Topics

[Working With Nets](#)

[Editing Net Attributes and Properties](#)

[Layout XL Forms](#)

Edit Soft Blocks

Use the **Edit Soft Blocks** form to modify the type of soft block that will be created; the shape and size of the place and route boundary; the shape, size, and signal type of soft pins; and the routing and placement obstructions inside the block.

[Boundary Tab](#)

[I/O Pins Tab](#)

[Obstructions Tab](#)

Soft Block List lets you select the soft blocks you want to edit. Any soft blocks that were selected in the layout window when you opened the form are highlighted. If there were no soft blocks selected, the first block in the list is highlighted.

Select All selects all the soft blocks in the design.

Update Selected Soft Block Attributes lets you to modify the attributes of the selected soft blocks.

Note: This list represents only selected soft blocks unlike the *Soft Block List* which contains all the soft blocks present in the design. This list is affected by modifications to the *Soft Block List* section.

You can modify soft blocks individually. The *Library Name*, *Cell Name*, *View Name*, *Cell Type*, and *Block Type* fields display the soft block being modified.

You can also use Common mode to modify the attributes of multiple soft blocks simultaneously. In this case the *Library Name*, *Cell Name*, *View Name*, and *Cell Type* fields are grayed out.

Block Type specifies whether the block is *custom* or *digital*.

Related Topics

[Editing Soft Blocks](#)

[Changing the Block Type](#)

[Changing the Boundary](#)

[Changing I/O Pin Attributes](#)

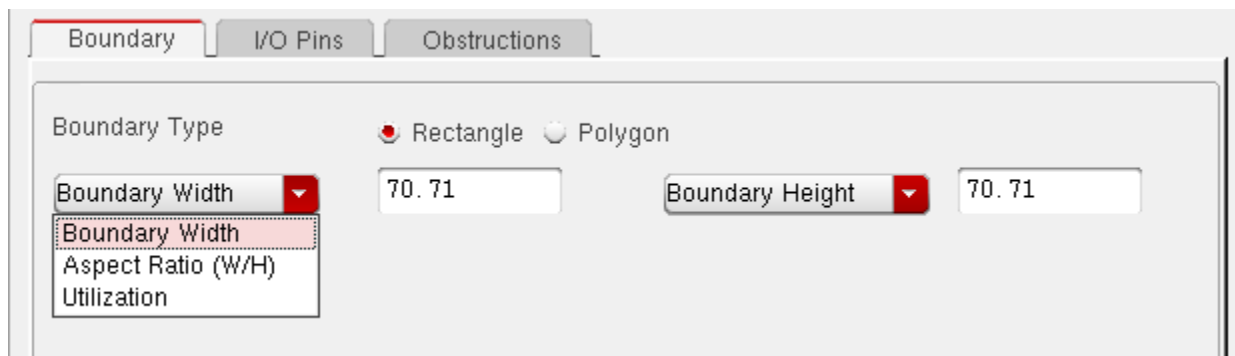
[Changing Obstructions](#)

Layout XL Forms

Boundary Tab

The *Boundary* tab lets you update the place and route boundary of the selected soft blocks.

Rectangle lets you create a rectangular boundary. Use the two cyclic fields below the *Rectangle* radio button to specify the dimensions of a soft block.



The table below lists the valid combinations of the options in fixed mode.

<i>Aspect ratio (W/H)</i>	<i>Boundary Height</i>
<i>Aspect ratio (W/H)</i>	<i>Boundary Width</i>
<i>Boundary Width</i>	<i>Boundary Height</i>

Non-fixed mode is where you have not specified the width and height of the blocks and you want to estimate the area. The table below lists the valid combinations in non-fixed mode.

<i>Utilization</i>	<i>Height</i>
<i>Utilization</i>	<i>Width</i>
<i>Utilization</i>	<i>Aspect ratio (W/H)</i>

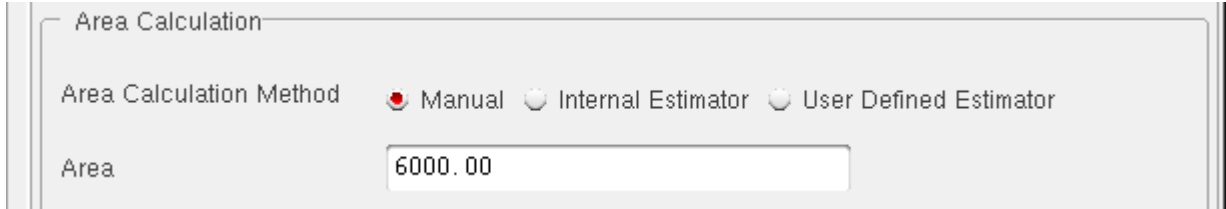
In non-fixed mode, the *Area Calculation* section becomes available, where you can select how the area of the place and route boundary is calculated. The methods available depend on the cell type of the soft blocks.

Virtuoso Layout Suite XL User Guide

Layout XL Forms

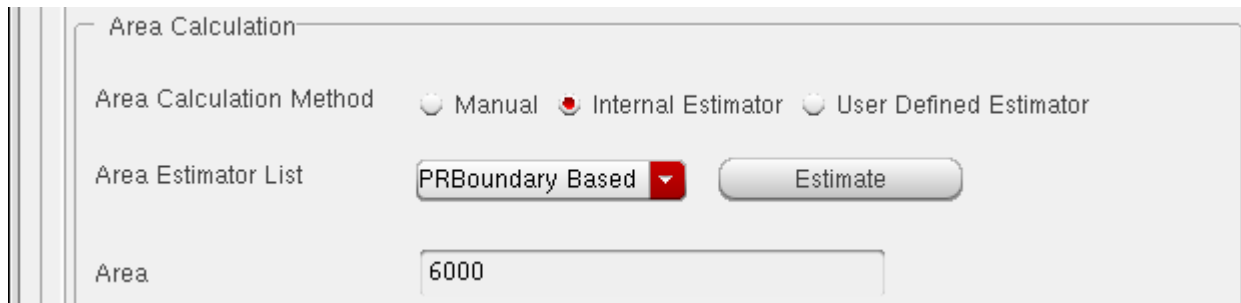
If the block has type *softMacro*, you can use the *Manual*, *Internal Estimator*, or *User Defined Estimator* methods; if the block has type *blockBlackBox*, you can use the *Manual*, *Area Per Gate*, or *User Defined Estimator* methods.

- **Manual** lets you specify the area in the field provided. Use this method if you already know the area of the soft block. Type the area into the field provided.



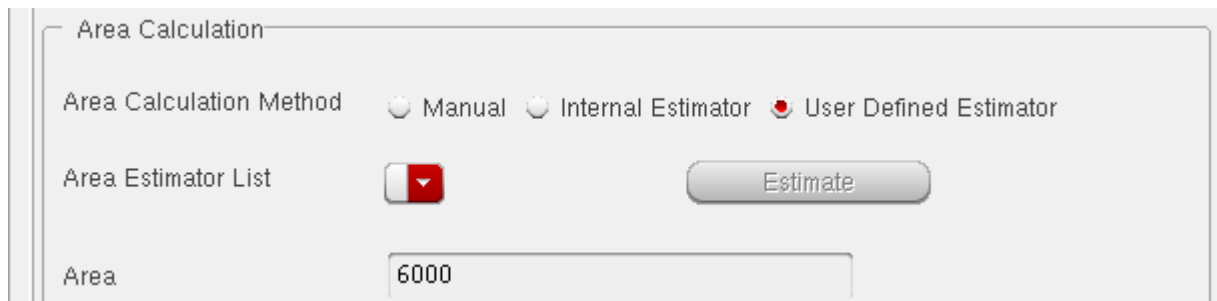
The screenshot shows the 'Area Calculation' dialog box. Under 'Area Calculation Method', the 'Manual' radio button is selected. The 'Area' text field contains the value '6000.00'.

- **Internal Estimator** calculates the area based on either the place and route boundary or the bounding box of the components contained in the soft block. Choose either *PRBoundary Based* or *BBox Based* from the cyclic field and click *Estimate* to calculate the area.



The screenshot shows the 'Area Calculation' dialog box. Under 'Area Calculation Method', the 'Internal Estimator' radio button is selected. The 'Area Estimator List' dropdown menu is set to 'PRBoundary Based'. The 'Estimate' button is visible. The 'Area' text field contains the value '6000'.

- **User Defined Estimator** lets you specify your own area calculation function using the framework described in [Creating and Registering a User Defined Area Estimation Function](#). The picture below shows no user-defined function registered.



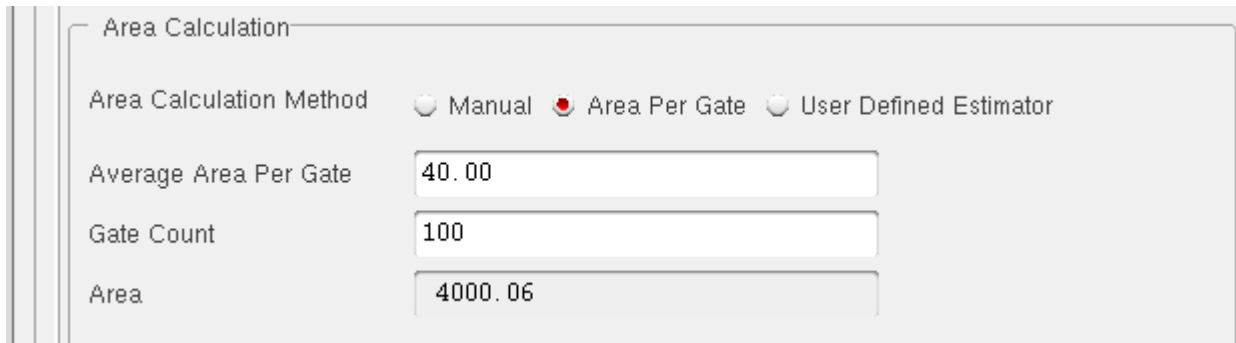
The screenshot shows the 'Area Calculation' dialog box. Under 'Area Calculation Method', the 'User Defined Estimator' radio button is selected. The 'Area Estimator List' dropdown menu is empty. The 'Estimate' button is visible. The 'Area' text field contains the value '6000'.

- **Area Per Gate** lets you enter the area per gate and the gate count for a digital block. The system calculates the area based on the following formula.

Virtuoso Layout Suite XL User Guide

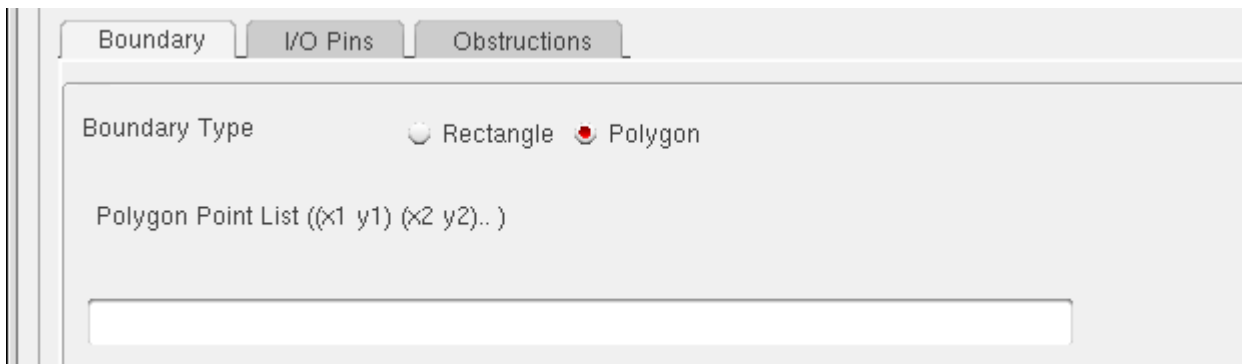
Layout XL Forms

$\text{Area} = \text{Area Per Gate} * \text{Gate Count}$



The screenshot shows a dialog box titled "Area Calculation". It contains three radio buttons for "Area Calculation Method": "Manual", "Area Per Gate" (which is selected), and "User Defined Estimator". Below these are three input fields: "Average Area Per Gate" with the value "40.00", "Gate Count" with the value "100", and "Area" with the calculated value "4000.06".

Polygon lets you specify the points of a polygonal boundary.



The screenshot shows a dialog box with three tabs: "Boundary", "I/O Pins", and "Obstructions". The "Boundary" tab is active. It contains a "Boundary Type" section with two radio buttons: "Rectangle" and "Polygon" (which is selected). Below this is a label "Polygon Point List ((x1 y1) (x2 y2)..)" and a large empty text input field for entering the coordinates.

Related Topics

[Changing the Boundary](#)

[Edit Soft Blocks](#)

Virtuoso Layout Suite XL User Guide

Layout XL Forms

I/O Pins Tab

The *I/O Pins* tab lets you update the layer, width, height, number, criticality, and signal type of existing pins. This table is initialized with *Term* and *Net* names from the symbol view of the selected soft block.

Term	Net	Layer	Width	Height	Num	Criticality	SigType
ADC<5>	ADC<5>	Metal3 drawing	0.15	0.15	1	10	signal
ADC<4>	ADC<4>	Metal3 drawing	0.15	0.15	1	10	signal
ADC<3>	ADC<3>	Metal3 drawing	0.15	0.15	1	10	signal
ADC<2>	ADC<2>	Metal3 drawing	0.15	0.15	1	10	signal

ADC<5> ADC<5> Metal3 dg 0.15 0.15 1 10 signal

Update Update All

Change the parameters as required and use the *Update* button to update the selected pins, or use the *Update All* button to update all the pins in the block with the current settings.

Virtuoso Layout Suite XL User Guide

Layout XL Forms

If you select multiple blocks the *I/O Pins* tab changes as shown below.

Update Selected Soft Block(s) Attributes

☒ Common

ether:top_1_port_pnr:layout
ether:Bias_port:layout
ether:anlg_cntl:layout

Library Name

Cell Name

View Name

Block Type Cell Type

Boundary I/O Pins Obstructions

I/O Pins

Term	Net	Layer	Width	Height	Num	Criticality	SigType
all	all	Metal3 drawing	1.00	1.00	1	10	signal

all all Metal3 dg 1.00 1.00 1 10 signal

Update Update All

Change the parameters as required and use the *Update* button to update all the pins in all the selected blocks.

Related Topics

[Changing I/O Pin Attributes](#)

[Edit Soft Blocks](#)

Obstructions Tab

The *Obstructions* tab lets you change routing and placement obstructions implemented using layer and placement halos.

The screenshot shows the 'Obstructions' tab in the Virtuoso Layout Suite XL. At the top, there are three tabs: 'Boundary', 'I/O Pins', and 'Obstructions' (which is selected). Below the tabs is a section titled 'Halos' containing a table with the following data:

Soft Block	Type	Layer	Top	Bottom	Left	Right
design:block3:layout	routing	metal1	5.000	5.000	5.000	5.000

Below the table, there are several controls:

- Blockage Type:** A list box with options: placement, routing (selected), fill, slot, pin, feedthru.
- Material:** A dropdown menu currently set to 'all'.
- Layer:** A list box with options: cont, metal1 (selected), metal2, metal3.
- Offset:** Four input fields for Top, Bottom, Left, and Right, each with the value '5'.
- Buttons:** 'Update' and 'Delete' buttons at the bottom.

Halos lists the halos or obstructions in the selected soft blocks.

Blockage Type lists the various types of halos that you can create.

Material filters the layers based on the functions defined in the technology file.

Layer lists the layers in the technology file that you can use to create halos.

Offset lets you specify the top, bottom, left, and right offsets for the halos.

To define a layer halo, you need to set the *Blockage Type*, the *Layer* and *Offsets*. There is also a *Material* filter which you can use to select layers based on the function defined in the technology file.

To define a placement halo, you need set only the *Offsets*.

Related Topics

Changing Obstructions

Virtuoso Layout Suite XL User Guide

Layout XL Forms

Edit Soft Blocks

Extract Layout

Use the **Extract Layout** form to run connectivity on your layout design.

- To extract connectivity for only the top level of your design, ensure the *Extract Hierarchical Cellviews* option is switched off and click *OK*.
- To run a hierarchical extraction, switch on *Extract Hierarchical Cellviews*, set the options described below as required, and click *OK*.

Use the *Options* button to open the Extraction Tab form where you can set the extraction parameters for the current design.

Hierarchical Cellviews lets you specify how a hierarchical extraction is performed.

Extract Hierarchical Cellviews enables the hierarchical extraction function. You can extract the whole design or select the cellviews you want to extract from the *Select Cellviews* pane.

Save Extracted Cellviews automatically saves the extracted cellviews.

Select Cellviews lists the lower-level cellviews that can be re-extracted. You can choose to list all the cellviews in the design or only those that have been edited since the last time the design was extracted. The content of the list also depends on the setting of the *Extract connectivity to level* option in the Extraction Tab form.

Select lets you filter the entries in the list using the library, cell, or view names. Type in the first characters of the names you want to select. The filter mechanism has no wildcard capability. To select a cell called 'resistor', you must type 'r', 're', 'res', and so on. If you type 'sis', the cell is not selected.

Alternatively, you can click *Select All* to select all the cellviews listed; click *Deselect All* to deselect all the cellviews listed.

Number Selected indicates the number of cellviews selected.

Related Topics

Connectivity Extraction

Extracting a Top-Level Design

Extracting Hierarchical Cellviews

Extraction Tab

Virtuoso Layout Suite XL User Guide

Layout XL Forms

Layout XL Forms

Generate Chained Devices

Use the **Generate Chained Devices** form to select and chain specific devices in your design. Select the devices you want to chain directly in the layout canvas and click *Apply* in the form to chain them. If the majority of the selected devices are oriented vertically, the generated chain is also oriented vertically. The devices to be chained need not be bound to a schematic instance.

Preserve Existing Chains ensures that any existing chains in the design are not broken during interactive chaining.

Use Device Order maintains the relative starting positions of the selected instances in the generated chain.

Extend Selection always selects the entire chain when you click on an individual instance, mfactored device, or folded device within the chain. When switched off, you can select individual devices, mfactors, or folded legs within a chain without selecting the entire chain.

Interdigitate Chains automatically identifies nodes that qualify as pseudoparallel connections and defines them as such during chaining and abutment. A pseudoparallel net connects nodes that are always the same voltage, so current does not pass through the net.

Align PMOS controls the alignment of PMOS chains. The default alignment for PMOS chains is “Top” but you can choose to align a chain to “Center” or “Bottom”.

Align NMOS controls the alignment of NMOS chains. The default alignment for NMOS chains is “Bottom” but you can choose to align a chain to “Top” or “Center”.

Show Device List expands the form to list the transistors available for chaining. If there is nothing preselected in the layout canvas, the form lists all the transistors in the design. If there were transistors selected in the layout canvas before you choose the *Generate Chained Devices* command, only the selected transistors are listed. Select the transistors to chain from the list and click *Apply* to chain them.

Related Topics

[Manual Device Abutment](#)

[Chaining Devices Interactively](#)

[Layout XL Forms](#)

Generate Clones

Use the **Generate Clones** form to replicate a section of the layout that is associated with a section of the schematic in such a way that the new piece of layout material can be placed at more than one location, with each part preserving the hierarchical structure of the design.

Clone Source shows the schematic and layout devices to be used as the clone source. You must specify a clone source in order to enable the other sections of the form.

Schematic Instances shows the schematic clone source; that is, the interconnected schematic devices bound to the instances in the layout clone source.

Layout Instances shows the layout clone source; that is, the layout instances and routing shapes that are to be cloned.

If there are layout instances and shapes selected when you open the form, and if they form a valid clone source, the fields are already filled in when the form opens.

Alternatively, you can choose another set of instances in the layout canvas and click *Update From Selected* to update the clone source.

Target Designs lets you choose which of the current open schematic and layout cellviews are used by the cloning engine.

Schematic Cellview specifies the schematic cellview containing the circuit to be cloned. By default this is the schematic cellview from the cellview pair which launched the command. If there is more than one schematic open in the current session, use the pull-down to choose the one you want from the list.

Layout Cellview is the cellview where the generated clone is to be placed. By default this is the layout cellview from the cellview pair which launched the command. The pull-down lists all the layout cellviews bound to the specified schematic cellview.

Search lets you control the scope and criteria used when searching for target structures that match the clone source structure. Set the options you want and click *Search* to find potential clones.

In Entire Schematic searches the entire schematic to find matching target structures.

In Selected Set in Schematic searches only the selected set in the schematic to find matching target structures. Use this option to limit the search to a specific area of the design.

With Allowed Permutation considers pin permutability when searching for matching target structures. If an appropriate permuteRule is defined for the device in question, and if permuting the pins on a device results in a match, then that match is reported (but only if no matching target structures were found without permutation).

With Exact Parameters requires that the parameter values on the components in a target structure match exactly the parameter values on the components in the clone source. When switched off, Layout XL also reports target structures that have different parameters and values. If required, you can update the parameters and values when you generate a clone by checking the *Update Layout Parameters to Match Schematic* option.

With Exact Connectivity requires that the connectivity of the components in a target structure matches exactly the connectivity of the components in the clone source. When switched off, Layout XL also reports target structures where the set of instances is the same, but they are connected differently. Note that searching for non-exact connectivity matches takes significantly longer than searching for exact matches.

When *Exact Connectivity* is switched off, clicking the *More Options* button opens the **Non-Exact Connectivity Matches** form where you can control the quality and number of non-exact connectivity matches reported.

Maximum partial net matches sets the maximum number of source nets which can be unmatched in a target structure. If a target structure has more than the specified number of unmatched nets, it is not reported.

Note that if you set the value too high, it might prevent the software from finding better target structures with fewer partial nets. To avoid this, either lower the value or limit the search to a selected set of instances.

Allowed partial net names specifies the names of source nets that are permitted to be unmatched in the target structure. You can use regular expressions to specify multiple net names with common elements. If a source net name is not listed (or does not match one of the regular expressions specified), then the net must be matched exactly in the target structure.

Maximum found matches sets the maximum number of non-exact target matches reported.

Create Options specify the types of clones that are generated from the target structures found. This section is enabled only if there are target structures reported in the *Clones Found* pane.

Create Clones as

Synchronized Family generates the clone source and each individual clone as a group in the layout view and links all the groups together as members of the same synchronized family. Each group contains all the physical shapes, vias, instances, and groups in the layout implementation of the respective clone. If you make a change to one member of a synchronized family, all the other members are automatically changed in the same way. See [Generating Synchronous Clones](#).

Grouped Objects generates the clone source and each individual clone as a group in the layout view but does not link the groups to each other in a synchronized family. Each group contains all the physical shapes, vias, instances, and groups in the layout implementation of the respective clone. Clones that are generated as groups are constrained in the ways they can be moved and rotated. See [Cloning a Group of Components](#).

Free Objects generates the clones as ungrouped objects in the layout view. See [Generating a Clone using the Generate Clones Form](#).

Update Layout Parameters to Match Schematic automatically updates the parameters and parameter values on the devices in the generated clone to match those on their counterparts in the schematic clone source. Parameters that are set on layout devices but are not present on their schematic counterparts are not removed. When switched off, the parameters and values are taken from the layout clone source.

Display Draglines controls whether draglines are shown while generating clones. Draglines indicate connections from the pins of the object you are moving to pins of the nearest objects. For information on how to change the draglines, see the *Draglines* options on the [Display Tab](#) of the Layout XL Options form.

Clones Found lists the target structures that can be generated as clones in the layout cellview. Select one of the structures and move your cursor into the layout canvas to generate a clone for that structure.

Use the *Edit* button to open the [Modify Correspondence](#) form where you can change the correspondence between the instances in source and target structures before you generate a clone. Use *Rotate* to rotate the clone through 90 degrees counterclockwise; *Flip Horizontal* to mirror the clone about its y axis; and *Flip Vertical* to mirror the clone about its x axis.

Related Topics

[Generating Clones](#)

[Generating a Clone using the Generate Clones Form](#)

[Cloning a Group of Components](#)

[Cloning Mfactored Components](#)

[Generating a Family of Synchronous Clones](#)

[Layout XL Forms](#)

Generate Folded Devices

Use the **Generate Folded Devices** form to control interactive folding, which lets you divide an individual transistor or chain of transistors into two or more folds. Folding lets you change the aspect ratios of the transistors in your design while retaining the original orientation. The devices to be folded need not be bound to a schematic instance.

Transistor Name shows the name of the transistor that will be folded. If you have selected more than one device, use the *Next* and *Previous* buttons to move through the selected set.

Transistor Width shows the value of the width property of the selected device, indicating also whether the value is taken from the schematic or layout view. When Ignore mfactor is on, the value shown is the effective width in the schematic; i.e., the product of mfactor and schematic gate width.

Number of Folds lets you type in the number of folds into which you want to fold the selected device. Entering a value in this field displays the *Set Fold Widths* button, which adds new fields that let you specify the width of each fold.

If you specify multiple folds to be generated for the device, the width value displayed for each fold is controlled by the environment variable, lxGetSignifDigits. If the environment variable is not set, the width of each fold defaults to 6 significant digits.

For example, if the transistor width is 4u, and you specify that three folds be created, then depending on the state of the lxGetSignifDigits environment variable, the width of the folds varies as follows:

- If the lxGetSignifDigits environment variable is set to 4 significant digits, the width of each fold is set to 1.333u.
- If the lxGetSignifDigits environment variable is not set, the fold width defaults to 6 significant digits, resulting in a width of 1.33333u for each fold.

Note: If you specify more than 500 folds to be added and click *Set Fold Widths*, a warning message displays prompting for confirmation to proceed with the folding.

Ignore mfactor ignores the schematic multiplication factor when generating folded devices. The *Transistor Width* is set to the effective width in the schematic; i.e., the product of mfactor and schematic gate width.

Chain Folds tells Layout XL to chain the resultant folds together where possible.

Total Width shows the total value of all the fold widths displayed in the *Width* fields at the bottom of the form.

Add Fold adds an additional fold to the device and provides a field where you can specify the *Width* for the new fold.

Same Width divides the total width of the transistor by the number of folds and sets the resultant *Width* as the value for each fold.

Delete removes the corresponding fold from the device.

Note: The system automatically adds the *Transistor Width* unit identifier to any width value with no unit. If the *Transistor Width* has no unit identifier, no identifier is added to the *Width* fields. See [Unit Identifiers in Layout XL](#) for a complete list of identifiers.

As in the case of individual fold widths, the value displayed in the *Total Width* field is controlled by the `lxGetSignifDigits` environment variable. If, for example, the environment variable is set to 4 significant digits, the value in the *Total Width* field displays up to 4 significant digits. If the environment variable is not set, the *Total Width* field displays up to 6 significant digits.

Irrespective of the original width of the transistor, the *Total Width* field displays the actual sum of the width values of the various folds. For example, if the original width of the transistor is 4u and the sum of the fold widths is calculated at 3.999u, the *Total Width* field displays the width as 3.999u instead of rounding off the value to 4.

However, if the difference between the *Total Width* and the original transistor width exceeds the tolerance value set by using the `paramTolerance` environment variable, a warning message is displayed indicating that the sum of the folded widths is not equal to the original transistor width.

Align controls the alignment of the device selected for folding.

- If the selected device is a PMOS, the value of the *Align* field is the current value of the `lxChainAlignPMOS` environment variable.
- If the selected device is an NMOS, the value of the *Align* field is the current value of the `lxChainAlignNMOS` environment variable.

Note: You can, however, choose from the following values for the field: “Top”, “Center”, or “Bottom”.

Related Topics

[Adding Components](#)

[Folding a Transistor](#)

[Ignoring Schematic Mfactor during Folding](#)

Virtuoso Layout Suite XL User Guide

Layout XL Forms

Layout XL Forms

Generate Layout

Use the **Generate Layout** form to generate layout representations of schematic design components.

The form is split into four tabs.

- [Generate Tab](#)
- [I/O Pins Tab](#)
- [PR Boundary Tab](#)
- [Floorplan Tab](#)

Related Topics

[Generating a Layout](#)

[Generating All Components from Source](#)

[Layout XL Forms](#)

Generate Tab

The **Generate** group box lets you choose which design objects are generated in the layout view.

Instances generates all the instances in the schematic that do not have one of the ignore properties attached to them.

Chaining automatically abuts MOS transistors into chains during layout generation.

Folding automatically divides devices into folds to prevent the gate width from exceeding a specified size.

Chain Folds automatically chains the individual folds of a transistor.

Note: For the *Chain Folds* option to be available, only the *Folding* check box must be selected. If you select the *Chaining* check box as well, *Chain Folds* is deactivated.

I/O Pins generates all the pins listed on the *I/O Pins* tab. The generated pins are automatically snapped to the placement grid.

Except Global Pins stops Layout XL generating layout pins for the global nets in the schematic.

Except Pad Pins stops Layout XL generating layout pins for schematic pins that are connected to I/O pads (cells of type `pad`, `padSpacer`, or `padAreaIO`). When unchecked, the software generates both pads and pins.

PR Boundary generates a place and route boundary based on the settings on the *PR Boundary* tab. All placements and estimations are based on the generated PR boundary.

Snap Boundary generates a rectangular snap boundary that encloses the generated PR boundary. You can generate a snap boundary only if the *PR Boundary* option is switched on.

Device Correspondence

Preserve User-Defined Mappings preserves user-defined mappings of devices between the schematic and the layout. This option preserves only user-defined one-to-one, many-to-many, many-to-one, and one-to-many device correspondence defined in the Define Device Correspondence form. It does not report missing devices or shapes within a mapped group.

Connectivity Extraction

Extract Connectivity After Generation runs connectivity extraction as part of the layout generation process. Check this option to see the incomplete nets in the design immediately after layout generation has run.

Related Topics

[I/O Pins Tab](#)

[PR Boundary Tab](#)

[Floorplan Tab](#)

[Layout XL Forms](#)

I/O Pins Tab

Specify Default Values For All Pins lets you specify attribute values and *Apply* them to all the pins shown in the list box. Click apply to apply the *Layer*, *Width*, *Height*, *Num*, and *Create* settings for all the listed pins.

Layer specifies the layer on which the pin is generated. The cyclic field offers only the conducting layers. The default is the current drawing layer if it is defined as an extractable

layer in the technology file; otherwise it is the first extractable layer found. If there are no extractable layers, the cyclic field lists all valid layout layers.

Width specifies the width for each pin. The default is the `minWidth` value set for the current layer in the technology file. Any change to the value is applied only if the new value is greater than the default value.

Height specifies the height for each pin. The default is the `minWidth` value set for the current layer in the technology file. Any change to the value is applied only if the new value is greater than the default value.

Num specifies how many instances of this pin to generate. If you type 0, the pin is not generated.

Create specifies that pins are to be generated in the layout.

Specify Pins To Be Generated lets you select pins from the list box and update the attribute values used when those pins are generated in the layout. Click *Update* to update the *Layer*, *Width*, *Height*, *Num*, and *Create* settings for the currently selected pins.

Select lets you type in a complete or partial pin name to select one or more pins in the list box.

Number Of Matches displays the number of pins that match the string you typed into the *Select* field.

Add New Pin opens the Add A New Pin dialog where you can specify the name of a terminal for which to generate a new pin.

Term Name is the schematic terminal name. You cannot change this value.

Net Name is the net associated with the pin in the layout. You cannot change this value.

Note: If terminal and net names differ in the schematic, Layout XL generates a pin with the same name as the schematic terminal and a net with the same name as the net attached to the terminal in the schematic. If there is no explicit net label in the schematic, both the pin and the net name in the layout are the same as the schematic terminal. This is the default behavior.

Layer specifies the layer on which the selected pins are generated. The cyclic field offers only the conducting layers. The default is the current drawing layer if it is defined as an extractable layer in the technology file. Otherwise, it is the first extractable layer found. If there are no extractable layers, the cyclic field lists all valid layout layers.

Width specifies the width for each selected pin. The default is the `minWidth` value set for the current layer in the technology file. Any change to the value is applied only if the new value is greater than the default value.

Height specifies the height for each selected pin. The default is the `minWidth` value set for the current layer in the technology file. Any change to the value is applied only if the new value is greater than the default value.

Num specifies how many instances of each selected pin are generated. If you type 0, the selected pin is not generated.

Create specifies whether or not the selected pins are to be generated in the layout.

Pin Label specifies the type of label generated when you create a pin. This setting is honored by the *Generate All From Source* and *Generate Selected From Source* commands.

Create Label As specifies whether a pin label is created when the pin is generated.

You can choose to create either a **Label** object or a **Text Display**.

Click the *Options* button to access the Set Pin Label Text Style form, where you can set the size, font, style, justification and orientation of the label lettering, and the drawing or pin layer on which the labels are displayed.

Note: If your pin labels are not visible in the canvas, turn on the *Pin Names* option in the Display Options form.



For this setting to take effect, you must also set the Layout L environment variable createPinLabel to `t`.

Related Topics

[Generate Tab](#)

[PR Boundary Tab](#)

[Floorplan Tab](#)

[Layout XL Forms](#)

PR Boundary Tab

The **Shape** group box specifies whether the place and route boundary is a rectangle or a polygon.

Rectangle specifies a rectangular place and route boundary. Use the **Area Estimation** group box to specify how the size of the boundary is calculated.

Origin specifies the coordinates of the boundary's origin. The default is (0.0 0.0).

Polygon specifies a polygonal boundary. Use the **Points List** to specify the coordinates of each of the vertices of the polygon.

The **Area Estimation** group box specifies how the system calculates the shape and size of a rectangular boundary.

Area estimation comprises two parts: the first specifies the aspect ratio and utilization of the boundary; the second estimates the size of boundary required to accommodate the components to be generated.

To specify the aspect ratio and utilization, set two of the following four parameters. Choose one of the parameters from the first cyclic list and any one of the remaining valid choices from the second cyclic list.

Width specifies the width of the design boundary. The default is the size of the last boundary or 10.

Height specifies the height of the design boundary. The default is the size of the last boundary or 10.

Utilization (%) specifies the percentage of area within the cell boundary that you want to fill. The default is 25.

Note: When the *Chaining*, *Folding*, or *Chain Folds* options are switched on, the *Utilization* value is applied only after chaining and folding is complete so that the size of the boundary is calculated accurately.

Aspect Ratio (W/H) is the width-to-height ratio of the design boundary. A value of 1 specifies a square boundary; 0.5 specifies a boundary twice as high as it is wide; and 2 specifies a boundary twice as wide as it is high. The default is 1.

To estimate the area, choose a **Calculation Method**.

Internal lets you select one of the built-in estimators from the **Choose Estimator** cyclic field.

- ☐ *PR Boundary Based* calculates the area based on the sum of all the place and route boundaries of the components to be generated.
- ☐ *BBox Based* calculates the area based on the sum of all the bounding boxes of the components to be generated.

User Defined lets you choose your own estimator registered using the framework described in [Creating and Registering a User Defined Area Estimation Function](#).

Change Parameters lets you modify the parameters of a user-defined area estimation function without changing the function itself.

Related Topics

[Generate Tab](#)

[I/O Pins Tab](#)

[Floorplan Tab](#)

[Layout XL Forms](#)

Floorplan Tab

Preserve Floorplanning Objects lets you choose the floorplanning objects to be preserved in the generated layout. Click *All* to select all the items in the list; *None* to deselect them all; or select one or more of

Rows and Custom Placement Areas preserves any existing rows and custom placement areas.

Blockages applies only to standalone blockages which have no parent object; a blockage with a parent object is preserved automatically along with its parent.

Area Boundaries preserves any existing area boundaries.

Track Patterns preserves any existing track patterns.

Clusters preserves any existing clusters.

Cluster Boundaries but only if *Clusters* are also preserved.

Related Topics

[Generate Tab](#)

[I/O Pins Tab](#)

[PR Boundary Tab](#)

[Layout XL Forms](#)

Generate Selected Components

Use the **Generate Selected Components** form to generate selected schematic instances and pins directly in the layout view.

Placement Mode

Group As In Schematic generates the selected schematic instances and pins together in the layout view. By default, they are placed in the same relative positions as in the schematic. However, you can change the alignment, spacing, and orientation of the components using the options in the form. You cannot change instance properties or pin attributes in this mode.

Place Individually generates each of the selected schematic components one at a time in the layout view. When generating instances, the form shows the name and master of the instance currently being generated; when generating pins, the form shows the name of the pin currently being generated and lets you change the attributes of that pin. For information on what you can change, see [Generate Selected Components - Pin Options](#).



Tip

The *Chaining* option is not available in the *Place Individually* mode. To chain multiple devices, select the *Chaining* option in the *Group As In Schematic* mode or use the *Generate Chained Devices* command.

Draglines

Display Draglines controls whether draglines are displayed during the *Generate Selected From Source* command. By default, the draglines indicate connections from the pins of the component you are moving to pins of the nearest objects. For information on how to change the display of draglines, see [Changing the Appearance of Draglines](#).

Chaining and Folding

Chaining automatically abuts the selected set of MOS transistors to form a chain. The *Chaining* option is available only for instances in *Group As In Schematic* mode. The transistors to be abutted must be set up for abutment first. For more information, see [Setting Up Cells for Abutment](#).

Folding automatically divides MOS transistors into folds to optimize utilization.

Chain Folds automatically chains the individual folds of a transistor.

Note: For the *Chain Folds* option to be available, only the *Folding* check box must be selected. If you select the *Chaining* check box as well, *Chain Folds* is deactivated.

Interdigitate Chains automatically identifies nodes that qualify as pseudoparallel connections and defines them as such during chaining and abutment. A pseudoparallel net connects nodes that are always the same voltage, so current does not pass through the net.

This option does not control the automatic identification and creation of pseudoparallel nets during manual abutment. To do that, switch on *Create pseudoparallel connections*.

NMOS Fold Threshold is the maximum size of the folded NMOS transistor. This value defaults to the `lxMaxWidth` value set on the connectivity source component. Changing this value overrides any `lxMaxWidth` value defined in the component type. For more information, see [Component Types Cells Table](#).

Note: The *NMOS Fold Threshold* option is available only when the *Folding* check box is selected.

PMOS Fold Threshold is the maximum size of the folded PMOS transistor. This value defaults to the `lxMaxWidth` value set on the connectivity source component. Changing this value overrides any `lxMaxWidth` value defined in the component type. For more information, see [Component Types Cells Table](#).

Note: The *PMOS Fold Threshold* option is available only when the *Folding* check box is selected.

Click *Unplaced* to open the [Instance/Pin List](#), which lists each component in the schematic for which there is no corresponding instance or pin in the layout. When you select a device from the list, the form displays properties set for that device. When you select a pin, the form shows the options described in [Generate Selected Components - Pin Options](#).

Alignment lets you change the alignment and spacing between the selected components before you place them in the layout view. For more information, including examples of each type of alignment, see [Using the Align Toolbar](#).

Align Left aligns the left edges of the selected components.

Align Vertical aligns the vertical center lines of the selected components.

Align Right aligns the right edges of the selected components.

Align Top aligns the top edges of the selected objects.

Align Horizontal aligns the horizontal center lines of the selected objects.

Align Bottom aligns the bottom edges of the selected objects.

Align Spacing spaces the components evenly, separated by the distance specified in the text field. If you switch off this option, the components are aligned but retain their original spacing.

Orthogonal Spacing additionally spaces the components perpendicular to the align direction, with each successive component offset by the distance specified in the text field.

The orientation buttons at the bottom of the form let you change the orientation of the selected instance.

Rotate rotates the component 90 degrees counterclockwise.

Sideways mirrors the component on the y axis (flips it horizontally).

Upside Down mirrors the component on the x axis (flips it vertically).

Related Topics

[Generating a Layout](#)

[Generating Selected Components from Source](#)

[Generating Components Together](#)

[Generating Individual Instances](#)

[Generating Individual Pins](#)

[Listing Unplaced Components](#)

[Layout XL Forms](#)

Generate Selected Components - Pin Options

When you use the *Generate Selected From Source* command to generate a pin in Place Individually mode, the Generate Selected Components form expands to show the attributes of the pin currently being generated. Any changes you make to the attributes before clicking in the layout view are applied to the current pin and also become the new defaults for all subsequent pins in the selected set.

Terminal shows the name of the current pin.

Layer specifies the layer-purpose pair for the pin.

Width specifies the width of the pin.

Height specifies the height of the pin.

Create Label As specifies whether a pin label is created when the pin is generated. You can choose to create either a *Label* object or a *Text Display*. When creating a label object, click *Options* to access the Soft Block Global Options form, where you can set the size, font, style, justification and orientation of the label lettering, and the drawing or pin layer on which the labels are displayed.

I/O Type assigns a property used by routers to identify the direction of the signal into or out of the pin. The signal can be *input*, *output*, *inputOutput* (bidirectional), *switch* (carries data either in or out but not simultaneously), *jumper* (passes data through the pin), *tristate*, or *unused*.

Access Direction assigns a property used to identify the part of the pin to which routers can connect routing. The access direction can be *Top*, *Bottom*, *Left*, *Right*, *Any*, or *None*.

Related Topics

[Generating a Layout](#)

[Generating Selected Components from Source](#)

[Generating Individual Pins](#)

[Listing Unplaced Components](#)

[Layout XL Forms](#)

Instance/Pin List

The **Instance/Pin List** form lets you see all the components in the schematic that are not yet placed in the layout. Use it when generating selected components from source.

Select the instances and pin you want to generate and click *Apply*. The components are selected in the schematic.

Wires, labels, text, instances with an ignore property, and instances that have already been placed in the layout are not considered available components and are not shown in the list.

Related Topics

[Listing Unplaced Components](#)

[Generating Components Together](#)

[Generating Individual Instances](#)

[Generating Individual Pins](#)

[Generating Selected Components from Source](#)

[Layout XL Forms](#)

Layout XL Options

Use the **Layout XL Options** form to set Layout XL options either for the current cellview or the current Layout XL session.

The form is split into five tabs.

- General Tab where you define the scope of the options settings you make in the form and to load and save settings from and to an options file.
- Display Tab where you specify how the Layout XL windows are arranged on your desktop, how draglines are displayed in the layout window, and to enable and disable cross-selection and net probing.
- Extraction Tab where you set the Layout XL and shape-based router extraction parameters for the current design.
- Generation Tab where you specify how certain components are handled when they are generated in the layout view.
- Parameters Tab where you specify which parameters are to be ignored by the generation and check command and how parameters are compared during *Check Against Source*.

Related Topics

Setting Environment Variables in the Layout XL Options Form

Layout XL Forms

General Tab

Use the **General** options to define the scope of the options settings you make in the form and to load and save settings from and to an options file.

Scope specifies whether the option settings apply for the current cellview only or globally for the current cellview *and* all the cellviews opened subsequently in Layout XL.

Global applies the settings to the current cellview and to all cellviews opened subsequently in Layout XL.

Cellview applies the settings only for the current cellview. Choosing this option grays out the *Extraction*, *Generation*, and *Parameters* tabs, along with certain options on the *Display* tab.

Load/Save lets you load, save or delete options values to or from a cellview, library, technology library, or options file.

Cellview updates the options for the cellview from which you opened the form.

Library updates the library that contains the cellview from which you opened the form.

Tech library updates the technology library of the cellview from which you opened the form.

File updates your `.cdsenv` file (the default) or another file specified in the text field.

Constraint-Aware Editing

Constraint-aware editing ensures that the *Copy*, *Move*, *Stretch*, and *Rotate* commands honor the following constraints: *Alignment*, *Fixed*, *Locked*, *Matched Parameters*, *Orientation*, *Matched Orientation*, and *Symmetry*. This is a system-wide setting, which applies to all existing and new windows opened in the current Virtuoso session. For more information, see [Constraint-Aware Editing](#).

Connectivity Reference

Open in specifies which tier of the Virtuoso Schematic Editor is started when you launch Layout XL or GXL. The default is *Schematics XL*.

Note: The schematic editor tier level does not change when you launch Layout XL from the Virtuoso Schematic Editor XL nor when you switch between layout editor tier levels *after* Layout XL or GXL have been launched.

Open connectivity reference during Edit In Place/Descend specifies whether or not the source cellview is opened when you use the *Edit In Place* or *Descend* commands. When switched off, the layout cellview is still opened in Layout XL mode, but with no connectivity reference.

You can choose to open the schematic cellview in a new window or a new tab. By default, the schematic cellview opens in a new window. Alternatively, you can use the `openConnRefTab` environment variable to choose if the connectivity reference should open in a new window or a new tab.

Power/Ground Nets

Power net names specifies the list of power net names used to achieve optimized chaining results in Layout XL device level schematics.

Ground net names specifies the list of ground net names to help you achieve optimized chaining results in Layout XL device level schematics.

Both lists are also used by the Virtuoso custom digital placer to exclude power and ground nets during wire length optimization; and to determine the power nets that are hidden when the [hideDraglinesForGlobalNets](#) environment variable is switched on.

Related Topics

[Constraint-Aware Editing](#)

[Display Tab](#)

[Extraction Tab](#)

[Generation Tab](#)

[Parameters Tab](#)

[Layout XL Options](#)

[Layout XL Forms](#)

Display Tab

Use the **Display** options to specify how the Layout XL windows are arranged on your desktop, how draglines are displayed in the layout window, and to enable and disable cross-selection and net probing.

Initial Display

Auto arrange windows controls whether Layout XL automatically rearranges its four windows on your desktop when you launch the application. The default is on, meaning that the windows are positioned based on the values specified for the following environment variables: [ciwWindow](#), [layoutWindow](#), [lswWindow](#), and [schematicWindow](#). If those environment variables are not set, or they are set to the default value – "(0.0 0.0) (0.0 0.0)" – Layout XL tiles the four windows based on the height and width of your screen. Note that this option is not available in Cellview mode; see [General Tab](#) for more information.

Draglines controls the display of draglines when using the *Generate Selected From Source*, *Generate Clones*, *Move*, and *Stretch* commands. Note that this option is not available in Cellview mode; see [General Tab](#) for more information.

Show distant connections displays draglines for all the connections to the instance that is being manipulated. When switched off, draglines are shown only for the connections closest to the current instance.

Hide global nets hides draglines for global nets and power and ground nets specified using the `lxGroundNetNames` and `lxSupplyNetNames` environment variables.

Use colors shows each dragline in a different color.

Note: Displaying many draglines during interactive commands can impair the performance of Layout XL. To mitigate these effects, use the `maxDragFig` environment variable to limit the number of figures that can be in a dragset.

To set `maxDragFig` interactively so that it takes effect in the current session, type the following in the CIW, where *win* is the window to which the limit applies.

```
win~>maxDragFig = 1000
```

To set a default value for `maxDragFig`, which will be used in all future Layout XL sessions, put the following line in your `.cdsenv` file.

```
graphic maxDragFig int 1000 nil
```

Cross Selection

Cross-selection turns on cross-selection between layout and schematic. When you select a component in the layout, the corresponding component is selected in the schematic and vice versa, unless the instance is ignored in either view.

Connectivity

Probe nets during object creation controls whether nets that are tapped during interactive editing are highlighted in the layout canvas.

Messages

Redirect Layout XL messages to separate Info window shows the messages issued by the *Check Against Source*, *Update Layout Parameters*, and *Update Schematic Parameters* commands in a separate info window instead of in the CIW.

Related Topics

[General Tab](#)

[Extraction Tab](#)

[Generation Tab](#)

[Parameters Tab](#)

[Layout XL Options](#)

Layout XL Forms

Extraction Tab



This tab is not available in the Cellview mode. See the [General Tab](#) for more information.

Use the **Extraction** options to set the Layout XL and shape-based router extraction parameters for the current design. If you change any of the options on this tab, you should re-extract your layout to ensure that the connectivity model is up-to-date throughout the cellview. Otherwise, the new settings are taken into account by the incremental extractor (if it is enabled) or the next time you run a manual extraction using the `Extract Layout` command.

Virtuoso Layout Suite XL User Guide

Layout XL Forms

Layout XL Options

General Display **Extraction** Generation Parameters

Connectivity Extraction

☒ Update connectivity information when design is modified

Extract connectivity to level:

Derive extractable layers from constraint group:

☐ Keep shape assignment

☒ Connect unassigned instance terminals

☐ Assign shapes from attached labels

☐ Assign shapes from overlapping labels as defined by 'stampLabelLayers' rule

☐ Verify well and substrate connections

☒ Verify weak-connect violations

☒ Verify must-connect violations

☒ Verify open violations

☒ Verify unimplemented instance terminals

☒ Verify blockage violations

☒ Verify pins against routing blockages

	By net	By cellview
Maximum number of open violations:	100	1000
Maximum number of short violations:	100	1000
Maximum number of illegal connections:		1000

Space-based Router

Extract

Extracted Pin Style

☒ Labeled Shapes Only

☐ Connected Shapes on Same Layer

☐ Whole Net on Routing Layers

☒ Extract Connectivity through Poly

Connectivity Extraction groups the options that control the Layout XL connectivity extractor. For information, see [Connectivity Extraction](#).

Update connectivity information when design is modified enables incremental connectivity extraction, which updates the connectivity model whenever the design is modified. When switched off, the connectivity model is not updated automatically and no new short or open violation markers are generated in the layout window.

Note: Even when incremental extraction is switched off, you can still run connectivity extraction at any time using the *Connectivity – Update – Extract Layout* command.

Extract connectivity to level specifies how much of the design hierarchy the extractor considers when extracting the top-level design. For more information on this option, see [Extracting a Top-Level Design](#).

Note: Whatever value you set for *Extract connectivity to level*, the extractor only ever extracts the top level; i.e., it only ever changes the connectivity of objects or creates markers at the top level of the design. To extract hierarchical cellviews, set the value of this option to a value greater than 0 and use the *Extract Layout* command.

Derive extractable layers from constraint group sets the default constraint group from which Layout XL derives the extractable layers in the design. For more information on how extractable layers are derived, see [Specifying Information Required by the Layout XL Connectivity Extractor](#).

Diagnostics Report displays extraction-related information about the selected constraint group and appropriate warning messages to indicate if the technology file has been correctly set up for extraction. You can display the report in a separate information window or in the CIW.

Virtuoso Layout Suite XL User Guide

Layout XL Forms

```
*** Constraint Group Diagnostics ***

-----
Techfile library: gpdk090
Constraint group: virtuosoDefaultExtractorSetup
-----

WARNING (LCE-2011): Cannot determine the function of layers 'Oxide Bondpad' in constraint g

Extractable layers:
Layer      Function      Mask
(Oxide)    undefined     1
(Nwell)    n-type well   2
(Poly)     poly          3
(Cont)     cut           6
(Metal1)   metal         7
(Via1)     cut           8
(Metal2)   metal         9
(Via2)     cut           10
(Metal3)   metal         11
(Via3)     cut           30
(Metal4)   metal         31
(Via4)     cut           32
(Metal5)   metal         33
(Via5)     cut           34
(Metal6)   metal         35
(Bondpad)  undefined     36
(Via6)     cut           37
(Metal7)   metal         38
(Via7)     cut           39
(Metal8)   metal         40
(Via8)     cut           41
(Metal9)   metal         42

No-overlap layers:
(Oxide) -> (Poly)
```

The report, as displayed above, contains the following information:

- ☐ Details about the extractable layers, equivalent layers, and other similar extraction-related information generated by the `lcePrintExtractLayers()` SKILL function.
- ☐ Details about the standard via and custom via definitions generated by the `lcePrintExtractVias()` SKILL function.
- ☐ Warning messages generated by the extractor to indicate errors in the technology file setup for the selected constraint group.
- ☐ Details about the options selected on the *Extraction* tab that determine the way the selected constraint group is parsed for extraction.

Virtuoso Layout Suite XL User Guide

Layout XL Forms

For example, depending on the options selected on the *Extraction* tab, the extractor may parse the selected constraint group to verify connectivity from stamp labels and to substrate or well layers.

The details about the various sections in the Diagnostics Report and the information these sections provide about the design to be extracted is tabulated below:

Section	Purpose
Extractable Layers	Lists the physical layers that have been selected for extraction by using the <code>validLayers</code> constraint. For more information, see Valid Layers.
No-overlap Layers	Lists the pairs of layers that should not overlap. For more information, see Error Layers.
Equivalent Layers	Lists the layers that have been defined as electrically equivalent. For more information, see Equivalent Layers and Connectivity Rules.
Electrically-Connected Layers	Lists the layers that are electrically-connected. For more information, see Valid Layers.
Bulk Area Layers	Lists the layers that can be used to define the bulk area of shapeless terminals. For more information, see Bulk Area Layers and Bulk Area: Connectivity Extraction.
Stop Layers	Lists the layers that are defined as “cutting” other layers. For more information, see Stop Layers.
Stamp_Label Layers	Lists the layers that can be used to create labels that stamp other layers. For more information, see stampLabelLayers.
Standard Via Definitions	Lists the valid standard via definitions that have been parsed by the extractor. For more information, see Valid Vias and Via Definitions.
Custom Via Definitions	Lists the valid custom via definitions that have been parsed by the extractor. For more information, see Valid Vias and Via Definitions.

Derived Layers in
'validLayers'
constraint

Lists the derived layers that have been parsed by the extractor for the 'validLayers' constraint.

For more information, see [Composing Layers](#).

Derived Layers in
'errorLayer' constraint

Lists the derived layers that have been parsed by the extractor for the 'errorLayer' constraint.

For more information, see [Composing Layers](#).

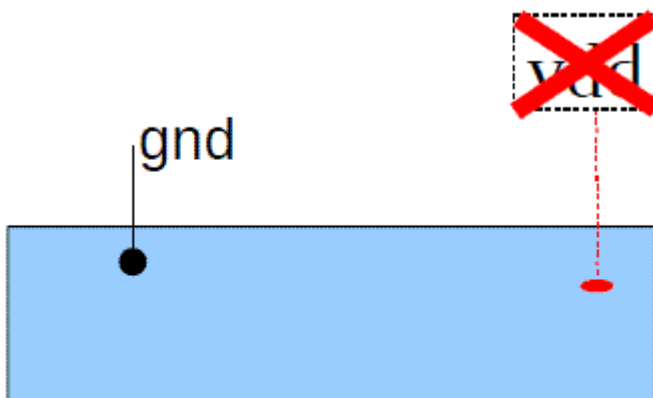
Keep shape assignment ensures that the connectivity of a design is preserved by retaining the assignment of shapes on a net. By default, the option is OFF.

Connect unassigned instance terminals ensures that an instance terminal is automatically connected to the net assigned to the overlapping shape. By default, the option is ON.

For example, consider an unassigned instance terminal that overlaps a shape connected to a net. If the option is ON, the instance terminal is assigned to the net. If the option is OFF, an illegal connection marker is created in the Annotation Browser and the instance terminal stays unassigned.

Assign shapes from attached labels ensures that a top-level shape with an attached label is automatically connected to the net referenced by the label text. By default, the option is OFF.

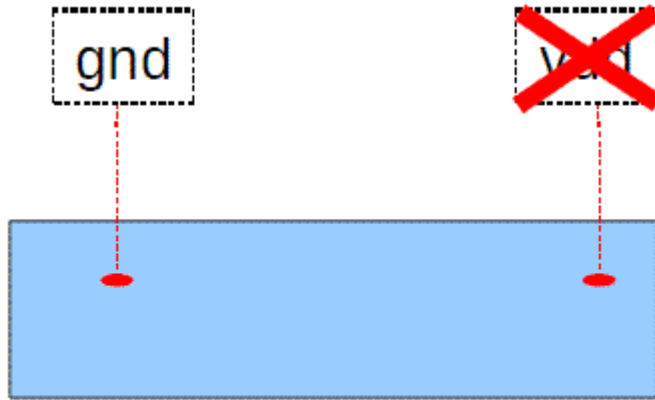
- ☐ If a label is attached to a shape that has sticky connectivity to another net, a short marker is created overlapping the label.



Short between:

- net "vdd" and attached label with text "vdd"
- net "gnd" and a shape on layer-purpose "Metal1 drawing"

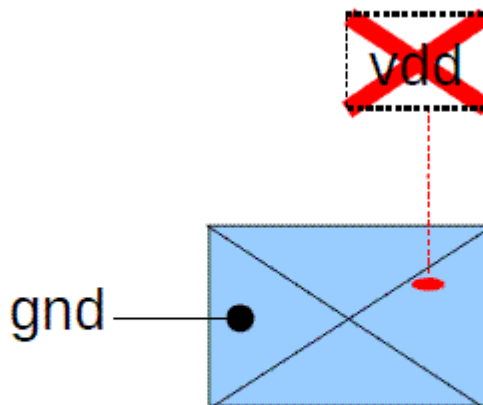
- ❑ If several labels with different text are attached to a single shape, the labels that create a short with the shape are overlapped by a short marker.



Short between:

- net "vdd" and an attached label with text "vdd"
- net "gnd" and a shape on layer-purpose "Metal1 drawing"

- ❑ If a label is attached to a pin shape that has a terminal name different from the label text, a short marker is created overlapping the label. If the label text and the terminal name are the same, no violations are reported.



Short between:

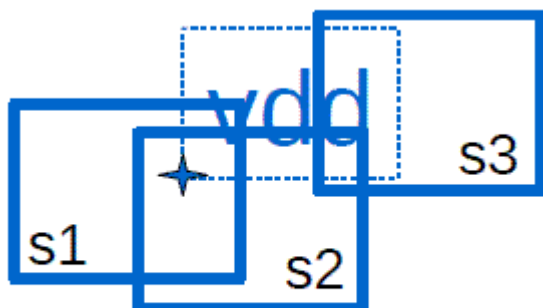
- net "vdd" and an attached label with text "vdd"
- net "gnd" and a pin figure (a rectangle on Metal1 pin) of terminal 'gnd'

For more information about attaching labels to shapes, see the [Attach](#) option under “Label Options” in the *Virtuoso Layout Suite L User Guide*.

Assign shapes from overlapping labels as defined by 'stampLabelLayers' rule ensures that a shape overlapping the origin of a label is assigned to the net referenced by the label text. For the shape to be connected to the label, a valid `stampLabelLayer` association should be defined in the technology file for the layer of the shape and that of the label.

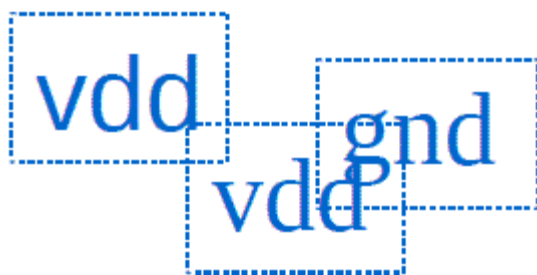
If the `stampLabelLayers` rule is not defined in the technology file, the option appears disabled. By default, the option is OFF.

- ☐ If a label is overlapped by several shapes, only those shapes that overlap the origin of the label are connected to the net referenced by the label text.



Only s1 and s2 are assigned to net "vdd"

- ☐ If a label is overlapped by other labels due to space constraints, no shorts are reported.



No shorts are reported

- ☐ If a shape overlaps a label that has sticky connectivity to another net, a short marker is created.

For more information, see [stampLabelLayers](#) in the *Virtuoso Technology Data ASCII Files Reference User Guide* and the [Auto](#) option under Label Layer/Purpose in the *Virtuoso Layout Suite L User Guide*.

Verify well and substrate connections toggles the verification of well and substrate connections in the current cellview.

Verify weak-connect violations toggles the display of weak-connect violation markers in the current cellview.

Verify must-connect violations toggles the display of must-connect violation markers in the current cellview.

Verify open violations toggles the verification of open violations in the design. When switched off, none of the dependent options is honored by Layout XL.

Verify unimplemented instance terminals specifies that the extractor should consider instance terminals with pin shapes that are not extractable (including instance terminals with no pin shapes at all).

Verify blockage violations toggles the verification of blockage violations (routing and placement blockage) in the design.

When switched on, the extractor creates a marker between the blockage and the overlapping object. For routing blockage, the overlapping object means a shape on the same layer and for placement blockage, the overlapping object means an instance.

Note: When the option is switched off, pin against routing blockage verification is not honored by Layout XL.

Verify pins against routing blockages toggles the verification of pins against routing blockages. When switched on, the extractor creates a marker between a routing blockage and a pin figure on the same layer. By default, the option is switched ON.

Maximum number of open violations specifies the highest number of open violation markers that can be generated by the extractor for a particular net or cellview. By default, the extractor can generate a maximum of 100 open violation markers per net and 1000 open violation markers per cellview. The option can also be controlled by the following environment variables:

- ☐ [extractCellviewOpenViolationLimit](#)
- ☐ [extractNetOpenViolationLimit](#)

Note: Switching on the option improves interactive performance when you are editing a large design or a net with a large number of instance terminals and terminals in an unrouted design.

Maximum number of short violations specifies the highest number of short violation markers that can be generated by the extractor for a particular net or cellview. By default,

the extractor can generate a maximum of 100 short violation markers per net and 1000 short violation markers per cellview.

Maximum number of illegal connections specifies the highest number of illegal connections that can be reported by the extractor for a particular design. By default, the extractor can report a maximum of 1000 illegal connections per design.

Space-based Router connectivity extraction ensures that connectivity is set correctly on all the shapes around the connection points in a cell loaded in the Virtuoso Space-based Router. This is required mainly for pcells in device-level routing, where the router will otherwise consider a gate shape to be a blockage, even though the underlying gate pin has connectivity assigned.

Extract lets you extract *all* cells, only *pcells*, or disable the extraction altogether. Note that the extraction happens in the router's internal data structures; the original pcell is not touched.

You can avoid this step and its associated cost by updating your pcells to ensure that all the shapes around connection points have appropriate connectivity defined.

For more information, see [Space-based router Extraction](#) in the *Virtuoso Space-based Router User Guide*.

Extracted Pin Style lets you control which shapes should become pin shapes.

Labeled Shapes Only allows only the shapes that are marked by text or property to become pin shapes. This option is selected by default.

Connected Shapes On Same Layer allows only the marked shapes and the shapes that are recursively connected to become pin shapes.

Whole Net on Routing Layers allows all the shapes in the net to become pin shapes.

Extract Connectivity through Poly extracts shapes on the poly layer. This option is selected by default.

Related Topics

[General Tab](#)

[Display Tab](#)

[Generation Tab](#)

[Parameters Tab](#)

[Layout XL Options](#)

[Layout XL Forms](#)

Generation Tab



This tab is not available in the Cellview mode. See [General Tab](#) for more information.

Use the **Generation** options to specify how certain components are handled when they are generated in the layout view.

Generation

Split mfactored devices controls whether Layout XL places schematic devices with the mfactor property as multiple devices in the layout.

Create pseudoparallel connections enables the detection of pseudoparallel nets during chaining. A pseudoparallel net connects nodes that are always the same voltage, so the current does not pass through the net.

Create implicit bus terminals creates, checks, and updates implicit bus terminals in the layout cellview, based upon the explicit bus terminals in the schematic cellview. This avoids the need to run `verilogAnnotate` on the layout cellview before referencing the cellview for:

- ☐ Running `verilog2oa`
- ☐ Importing another verilog design into SOC Encounter

The **Create implicit bus terminals** option is by default OFF.

Device orientation controls the orientation of Layout XL layout devices generated from the schematic. Devices generated with a particular orientation may be further transformed if either the *Chaining*, *Folding*, or *Chain Folds* options are switched on.

Auto Adjustment

Abut transistors turns on automatic abutment, which abuts prepared transistors so that they can share pins. Abutment is switched on for both automatic and interactive layout generation and during automatic placement using the Virtuoso custom digital placer.

Align abutted transistors snaps an instance in the direction perpendicular to the direction of abutment in order to align the instances.

Preserve terminal contacts specifies whether or not pin properties are preserved when an instance is flattened. Switching this option on means that the terminal contacts will be preserved when you abut devices in Layout XL.

Perform auto abutment only after a move or stretch restricts abutment to occur only after interactive edits using the *Move* and *Stretch* commands.

Space components turns on automatic spacing, which allows components with the properties [vxInstSpacingDir](#) and [vxInstSpacingRule](#) to be spaced automatically according to the values specified in the properties.

Note: If one of the devices is in any type of group (including a synchronous clone), the other device must be in the same group for automatic spacing to occur.

Permute pins turns on automatic pin permutation during manual routing or editing. For more information, see [permuteRule](#).

Mirror transistors specifies that if a short violation is created during abutment, then Layout XL first mirrors the device in question in order to resolve the short. If this is unsuccessful, Layout XL attempts to resolve the short by permuting the pins. If you switch the option off, Layout XL uses only pin permutation when attempting to resolve shorts.

Note: If one of the devices is in any type of group (including a synchronous clone), the other device must be in the same group for automatic mirroring to occur.

Device Folding

Generate minimal folding creates the minimum number of folded devices. When switched off, Layout XL generates an odd number of folded devices. For example, it adds one fold if the device width divided by number of folds yields an even number.

Retain instance orientation preserves the original device orientation when running the *Generate Folded Devices* command or the *Generate All From Source* command with the *Folding* option switched on. (Note that if the *Chaining* option is also switched on, instance orientation is not preserved.)

Pcell fingering parameter names specifies the parameter names that define pcell gate fingering. Folding and instance fingers are incompatible and this option is used to filter out those instances when folding. If you do not set the names correctly, it might result in abutment failures for even-fingered devices and folding.

Update width parameter method specifies how the *Update Layout Parameters* command updates folded devices when the width value is changed in the schematic.

No Change folded devices are not updated.

Equalize creates folds of equal width in the layout instance. The number of folds does not change, only (potentially) the width of each fold.

For example, the total width in the schematic is 12 and is distributed as follows in the layout.

```
layout widths I1.1 w = 2
               I1.2 w = 4
               I1.3 w = 6
```

If you change the width in the schematic to 15 and run *Update – Layout Parameters* with this option set, the new widths are as follows.

```
layout widths I1.1 w = 5
               I1.2 w = 5
               I1.3 w = 5
```

Distribute distributes the additional (or reduced) width equally amongst all the folds in the layout instance. Again, the number of folds does not change, only the width of each fold.

For example, the total width in the schematic is 12 and is distributed as follows in the layout.

```
layout widths I1.1 w = 2
               I1.2 w = 4
               I1.3 w = 6
```

If you change the width in the schematic to 18 and run *Update – Layout Parameters* with this option set, the new widths are as follows.

```
layout widths I1.1 w = 4
               I1.2 w = 6
               I1.3 w = 8
```

An extra 2 units of width is added to each layout instance.

Device Chaining

Create interdigitated chains automatically identifies nodes that qualify as pseudoparallel connections and defines them as such during chaining and abutment. A pseudoparallel net connects nodes that are always the same voltage, so the current does not pass through the net. Contacts are dropped automatically if it is appropriate to do so, even if the *Preserve terminal contacts* option is switched on.

Note: This option applies only when chaining is turned on. It does not control the automatic identification and creation of pseudoparallel nets during manual abutment. To do that, switch on *Create pseudoparallel connections*.

Maximum number of devices in a chain sets the maximum number of devices permitted in a chain.

Search levels up for abutments controls how far up the hierarchy Layout XL looks for opportunities to chain devices. A value of 0 means that chaining can look only in the same hierarchical level. A value of N means that chaining can look up N levels of hierarchy.

Note: This option is always considered by the *Generate All From Source* and *Generate Selected From Source* commands. However, the *Generate Chained Devices* command considers it only if there are 100 instances or more in the selected set.

Align PMOS controls the alignment of PMOS chains. The default alignment for PMOS chains is “Top” but you can choose to align a chain to “Center” or “Bottom”.

Align NMOS controls the alignment of NMOS chains. The default alignment for NMOS chains is “Bottom” but you can choose to align a chain to “Top” or “Center”.

Related Topics

[General Tab](#)

[Display Tab](#)

[Extraction Tab](#)

[Parameters Tab](#)

[Layout XL Options](#)

[Layout XL Forms](#)

Parameters Tab



This tab is not available in Cellview mode. See [General Tab](#) for more information.

Use the **Parameters** options to specify which parameters are to be ignored by the generation and check command and how parameters are compared during *Check Against Source*.

Ignore lets you specify the names of parameters to be ignored during update and check commands and the names of properties that cause objects to be ignored during generation

and check. Click *Add* to add to the list a parameter or property name you have typed into the text field. Click *Remove* to remove the selected parameters or properties from the list.

Parameters to ignore during check lists the parameters that are ignored by the *Check Against Source* command. Mismatches for any of the listed parameters are not reported by the check. The list inherits all the parameter names from *Parameters to ignore during generation*.

Parameters to ignore during generation lists the parameters that are ignored by the following *Connectivity* commands.

- ☐ *Generate – All From Source*
- ☐ *Generate – Selected From Source*
- ☐ *Generate – Clones*
- ☐ *Check – Against Source*
- ☐ *Update – Components And Nets*
- ☐ *Update – Layout Parameters*
- ☐ *Update – Schematic Parameters*

The following parameters are always ignored.

instancesLastChanged	lxPlacementStatus
instNamePrefix	lxRounding
lxIgnoreParamForCAS	lxStopList
lxIgnoredParams	lxTimeStamp
lxMFactorNum	lxUseCell
lxParamsToIgnore	pin#
lxParamsToIgnoreForCheck	posi

Properties used to ignore objects during check lists the properties that cause pins and instances to be ignored during the *Check Against Source* command. Any object with one of the listed properties set to `t` is ignored during these operations. The list inherits all the property names from *Properties used to ignore objects during generation* and also includes `lvIgnore` by default.

Properties used to ignore objects during generation lists the properties that cause pins and instances to be ignored during the generate, check, and update commands listed above. Any object with one of the listed properties set to `t` is ignored during these operations.

By default, the list the contains the following properties.

```
lxRemoveDevice ignore nlAction
```

(For information on the `nlActionS` property, see “Adding `nlAction` Properties” in Chapter 5, “Netlisting,” in the *Virtuoso NC Verilog Environment User Guide*.

Inherited from Generation lists the parameters or properties that are inherited from the list defined for generation. Parameters and properties ignored for generation are always ignored for check.

Schematic Parameter Names lets you specify the names of special parameters used in the schematic. Choose the parameter you want to edit from the cyclic field and type the names in the text field.

Parallel-connected factor lists the names of the schematic parameters used to specify the multiplication factor (`mfactor`) for transistors. The default is `m M`. Layout XL checks each schematic instance for a parameter matching one of the names on the list and uses the value of that parameter to generate the appropriate number of parallel-connected devices in the layout.

Series-connected factor lists the names of the schematic parameters used to specify the number of series-connected (`sfactor`) devices to be generated in the layout. The default is `s S`. Layout XL checks each schematic instance for a parameter matching one of the names on the list and uses the value of that parameter to generate the appropriate number of series-connected devices in the layout.

Transistor lists the names of schematic parameters used to specify transistor width. The default is `w` and the value must be split among the generated parallel-connected devices during the *Generate All From Source*, *Check Against Source*, or *Update Layout Parameters* commands. Layout XL checks each schematic instance for one of the listed parameter names and updates the width values of the matching layout parameters such that

```
number of mfactored instances * mfactorSplit value = source parameter value
```

Capacitance lists the names of the schematic parameters used to specify capacitance. The default is `c C`. The capacitance value must be split among the generated series-connected devices during the *Generate All From Source*, *Check Against Source*, or *Update Layout Parameters* commands. Layout XL checks each schematic instance for one of the listed parameter names and updates the capacitance values of the matching layout parameters accordingly.

Resistance lists the names of the schematic parameters used to specify resistance. The default is `r R`. The resistance value for a device must be split among generated series-connected devices during the *Generate All From Source*, *Check Against Source*, or *Update Layout Parameters* commands. Layout XL checks each schematic

instance for one of the listed names and updates the values of the matching layout parameters accordingly.

Inductance lists the names of the schematic parameters used to specify inductance. The default is `1 L`. The inductance value for a device must be split among generated series-connected devices during the *Generate All From Source*, *Check Against Source*, or *Update Layout Parameters* commands. Layout XL checks each schematic instance for one of the listed names and updates the values of the matching layout parameters accordingly.

Parameter/Property Comparison defines how CDF parameters and properties are compared during the *Check Against Source*, *Update Layout Parameters*, *Update Schematic Parameters*.

Consider parameters only compares only CDF parameters in the schematic against the parameters and properties in the layout and reports (or updates) values that do not match, parameters that cannot be checked, and parameters that are missing from the layout view.

Consider parameters and properties checks CDF parameters and cell and instance properties in the schematic against the parameters and properties in the layout and reports (or updates) values that do not match, parameters that cannot be checked, and parameters that are missing from the layout view. Use this option to check or update user-defined properties in the layout and schematic or when you know that a particular instance has certain properties that you are interested in comparing or updating.

Ignore missing parameters or properties ignores parameters (and properties, if *Consider parameters and properties* is checked) that are present in one view but missing from the other. Switch off this option if, for example, your layout has additional parameters or properties that you want to propagate to the schematic using the *Update Schematic Parameters* command.

Tolerance specifies the relative tolerance used when comparing values between the layout and the schematic.

If a is the source parameter value, b is the layout parameter value, and e is the *Tolerance* then parameter values are considered equal if

$$| (a/b) - 1.0 | < e$$

The default of e is $1e-6$.

Component Description Format

Evaluate CDF callbacks during generation and update causes all SKILL callbacks defined on CDF parameters to be evaluated by default when you run Layout XL's generate and update commands. Callbacks defined on buttons are never evaluated.

Use significant digits when calculating parameter values controls the number of significant digits used by Layout XL when calculating the value of layout CDF parameters. When the option is selected, the value it specifies overrides the setting of `aelGetSignifDigits` for calculations in the layout cellview. This lets you specify a different precision to be used in layout cellview calculations; for example, those involving pcells.

Note: The **Use significant digits when calculating parameter values** check box is available if the shell environment variable, `CDS_Netlisting_Mode` is set to `Analog`. For information on setting the environment variable, see [Setting the CDS_Netlisting_Mode Environment Variable](#) in the *Virtuoso Software Licensing and Configuration User Guide*.

Related Topics

[Generating All Components from Source](#)

[Generating Selected Components from Source](#)

[Updating Components and Nets](#)

[Check Against Source](#)

[General Tab](#)

[Display Tab](#)

[Extraction Tab](#)

[Generation Tab](#)

[Layout XL Options](#)

[Layout XL Forms](#)

LDE Analysis

Use the **LDE Analysis** form to run a Litho Electrical Analysis for evaluating the performance of a selected set of devices in the layout.

The form is split into two tabs.

- Setup Tab where you specify the options for setting up the LDE analysis.
- Region Tab where you specify the layout area to be analyzed.

Related Topics

Layout XL Forms

Setup Tab

Work Directory specifies the directory location of the layout design to be analyzed.

LEA Setup Script specifies the script to be used for running the analysis.

Analysis Mode specifies the type of analysis that needs to be run.

Constraint specifies that the analysis must be run on the basis of the applied constraints.

Excursion specifies that the analysis must be run in a statistical outlier detection mode.

View Stress Param when checked, displays a window detailing the stress analysis parameter resulting from the LDE analysis for each device.

Diff Layer allows you to select a layer from the cyclic field to be used as the diffusion layer.

Poly Layer allows you to select a layer from the cyclic field to be used as the poly layer.

Note: The tool automatically detects and selects the correct Diff and Poly layers. However, Cadence recommends that you should always check if the layers are indeed correct.

DP Mode specifies the distributed processing mode to be used for the analysis.

DP Sub Command specifies the LSF or SunGRID parameters to be used, such as queue and resource strings. This option is unavailable for Local, SSH, and RSH distributed processing mode.



Cadence recommends that you consult your administrator for the LSF or SunGRID parameters that are applicable to your environment.

DP # Hosts allows you to select a host from the cyclic field to be used for the analysis.

Note: This option is available only when the SunGRID, LSF, or Local distributed processing modes are selected.

RSH/SSH Hosts specifies the names of the remote machines to which you want to distribute the LDE analysis.

Note: The machine names should be separated by spaces. Depending on the number of machine names you specify, the number of CPUs to be used for the RSH/SSH distributed processing is determined.

INFO displays warning and error messages.

Region Tab

Specifies the layout areas on which the LDE analysis needs to be run. Use the various buttons on this tab to add, delete, clear, or zoom into the layout areas that you select.

To select a layout area for the analysis:

- ➔ Click the *Add* button on the Region tab and select a layout area by either clicking the area or by defining the area using a drag operation.

The co-ordinates of the area you selected display on the Region tab.

Note: To select another section of the layout, click the *Add* button again and specify the area to be selected for the analysis.

Related Topics

[Litho/LDE Analysis](#)

[Layout XL Forms](#)

Litho Fixing

Use the *Litho Fixing* form to read and fix lithography hotspots detected by Litho Physical Analyzer (LPA) or any other tool that exports a Hotspots Interchange Format (HIF) file.

This form has the following three tabs:

- General Tab

In this tab, you define the scope of the options to be used for checking and fixing the lithography hotspots.

- Read Tab

In this tab, you set one or more HIF files to be read to identify lithography hotspots and specify whether or not to limit the number and severity of hotspots.

- Fix Tab

In this tab, you specify the options for fixing the lithography hotspots.

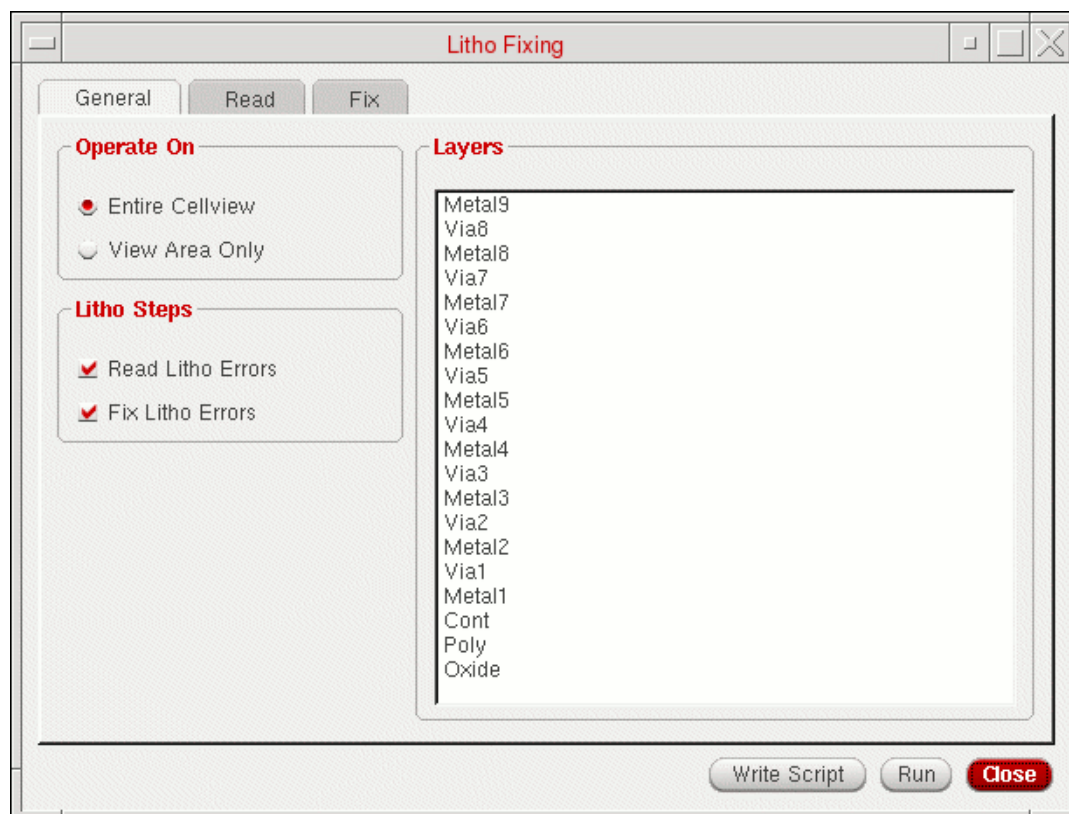
Related Topics

Litho Fixing Flow

Layout XL Forms

General Tab

In this tab, you define the scope of the options to be used for checking and fixing lithography.



Operate On controls whether the fixing of lithography be done on the entire cellview or the selected view area only.

Entire Cellview specifies the fixing of lithography be done on the entire cellview.

View Area Only specifies the fixing of lithography be done on the view area only.

Litho Steps controls whether to read the hotspots or to automatically fix these hotspots. Enabling either step further enables modification of the corresponding tabs in the form.

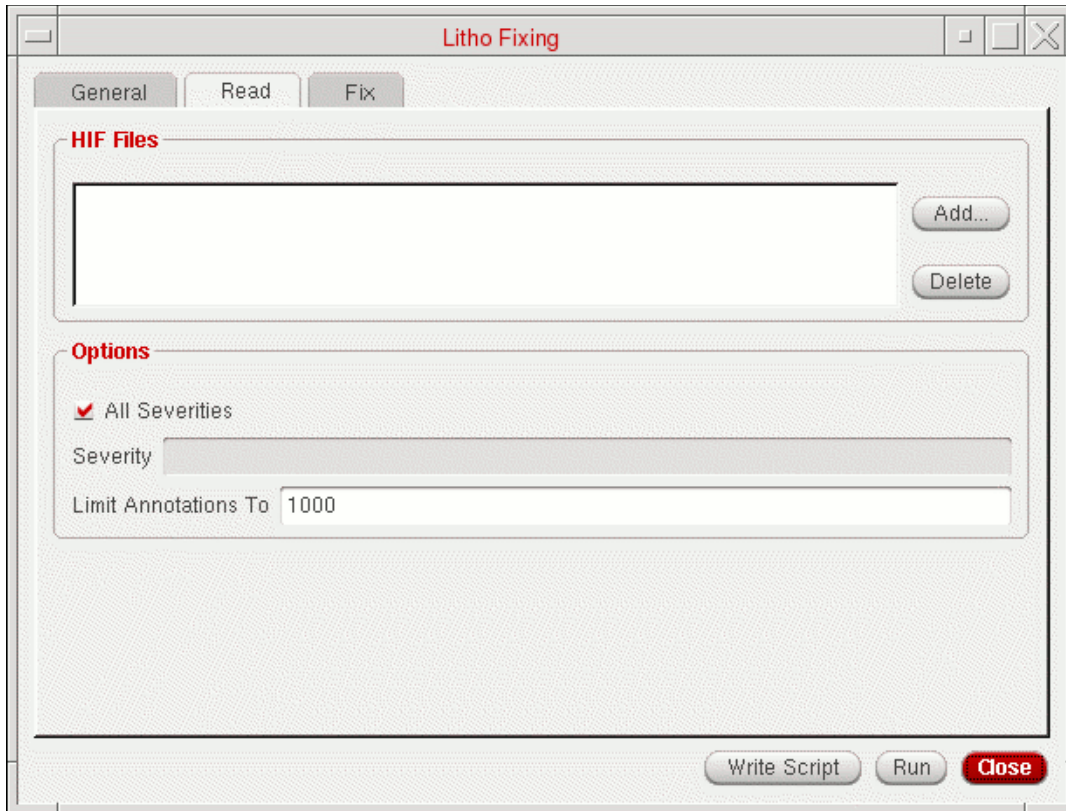
Read Litho Errors reads in lithography hotspots from one or more HIF files and enables modification of the options on the *Read* tab.

Fix Litho Errors applies automatic fixing to the violations that were read in and enables modification of the options on the *Fix* tab.

Layers lets you choose the layers on which the fixing of lithography hotspots should be applied.

Read Tab

In this tab, you set one or more HIF files to be read to identify hotspots and specify whether or not to limit the type and number of severities.



HIF Files enables you to load one or more HIF files to be read to identify hotspots. An HIF file is an American Standard Code for Information Interchange (ASCII) format that has marker information such as error types, location, severities, and hints on how to correct the hotspot. Use the *Add* button to add files. Use the *Delete* button to delete a file.

Options controls whether to limit the number or severity of lithography hotspots to be read in.

All Severities specifies that all severities are read.

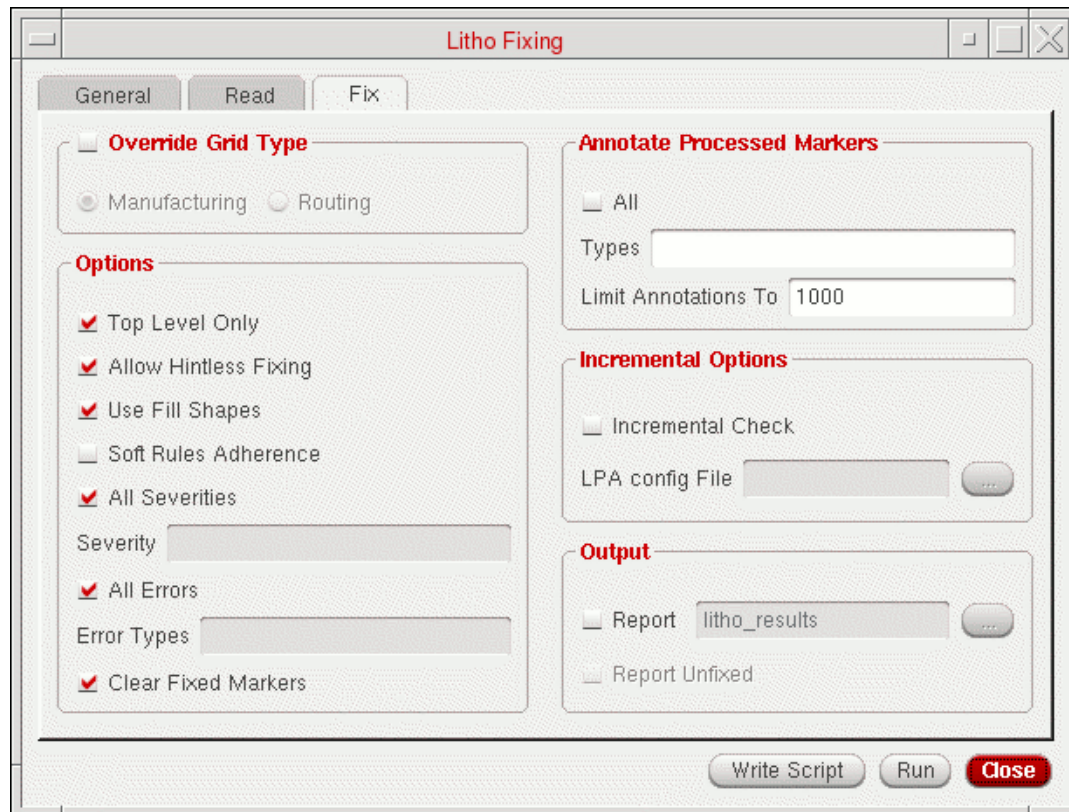
Severity specifies the type of severity to be read in.

Limit Annotations To specifies the limit of the total number of violations to be read in.

Note: By default, the total number of violations is limited to 1000.

Fix Tab

In this tab, you specify the options for fixing the lithography hotspots.



Override Grid Type enables you to override the default grid type being used in the layout.

Manufacturing specifies that the manufacturing grid be used and all the shapes in the layout be snapped to this grid.

Routing specifies that the routing grid be used and all the shapes in the layout be snapped to this grid.

Options specifies the parameters for fixing the lithography hotspots.

Top Level Only specifies fixing of the lithography hotspots be done only on the shapes at the top level of the open design.

Allow Hintless Fixing specifies that lithography errors that do not have hints be fixed. When this option is OFF, lithography errors that do not have hints are not fixed.

Use Fill Shapes specifies use of fill shapes for fixing the lithography hotspots.

Soft Rules Adherence controls whether soft or hard constraints be used for design rule checks.

All Severities specifies that all severity levels identified by lithography error annotations be fixed.

Severity specifies the type of lithography errors that will be fixed.

All Errors specifies that all lithography errors be fixed.

Error Types specifies the type of lithography errors to be fixed.

Clear Fixed Markers specifies deletion of fixed markers.

Annotate Processed Markers controls whether additional markers be added to the annotation browser. These additional markers show the region of the layout that has been modified while fixing the lithography.

All specifies that all fixes be annotated.

Types the type of fixes to be annotated.

Limit Annotations To specifies the maximum number of fixes to be annotated.

Note: By default, the number of annotations are limited to 1000.

Incremental Options specifies the additional checks to be done to fix the lithography errors.

Incremental Check specifies that LPA be run in the region after each lithography fix in order to check the results. If a lithography error still exists in the changed area, the change is reverted and the next hint is attempted. Once all hints have been attempted, a rip-up and re-route will be applied to the region to try to fix the error.

LPA config File specifies the name of the configuration file used by LPA to verify any layout changes caused by fixing.

Output specifies the name of the litho hotspot check file and controls if unfixed areas be reported in the hotspot check file.

Report controls whether or not to create the litho hotspot check file and the name of the check file.

Report Unfixed specifies that the unfixed areas be reported in the hotspot check file.

Related Topics

[Litho Fixing Flow](#)

[Layout XL Forms](#)

Load Physical View

Instead of defining a new cellview, you can import cellview information from an existing OpenAccess cellview, called the source physical cellview. To do this, use the **Load Physical View** form. To display the form, choose File - Load Physical View in the Virtuoso Layout Suite XL window.

Virtuoso Layout Suite XL User Guide

Layout XL Forms

Note: The Load Physical View menu item is available only in the edit mode.

Virtuoso Layout Suite XL User Guide

Layout XL Forms

Virtuoso Layout Suite XL User Guide

Layout XL Forms

Load Physical View

Choose Source Physical View

Library Name:

Cell Name:

View Name:

☒ Select All ☐ None

Update Instances

☒ Using Instance Name ☐ Using Master Name

☒ Standard Cells ☒ Custom Cells

☒ Macros ☒ IO Pads

☒ Add physical only instances

☒ Re-master parameterized instances

Load Pins

☒ Pins Mode: ☒ Replace ☐ Update

Nets

☐ Update Sig Type

Add Geometries

☐ Delete Shapes and Wires in Destination

☒ Shapes ☒ Wires

Update Boundaries

☒ PR Boundary ☒ Snap Boundary

Replace Rows

☒ Standard Rows ☒ Custom Rows

Replace Obstructions

☒ Blockages ☒ Halos

Type

☒ Placement ☒ Routing

☒ Slot ☒ Pin

☒ Fill ☒ Feedthru

☒ Screen

Transfer Constraints

☒ All Constraints and Process Overrides

Mode: ☒ Replace ☐ Update

Virtuoso Layout Suite XL User Guide

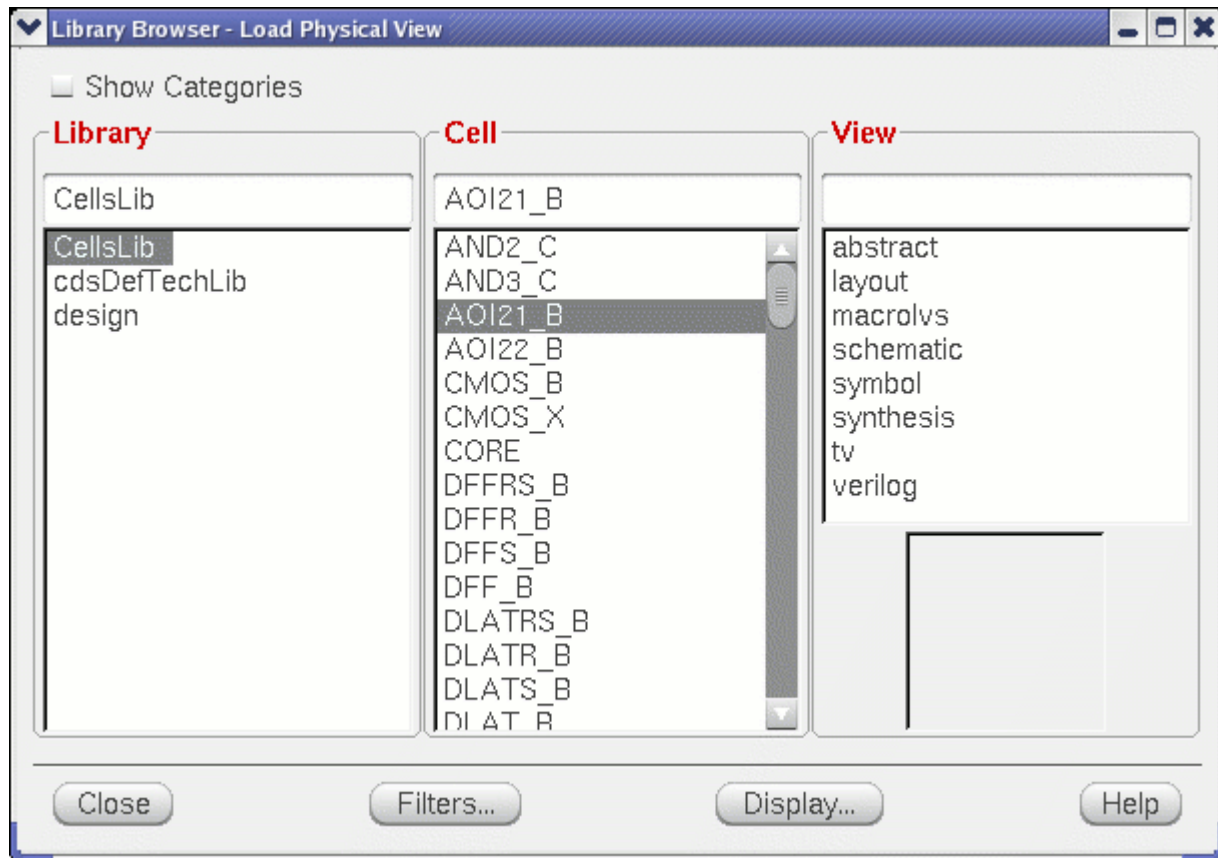
Layout XL Forms

To specify the Library, Cell, and View names of the source cellview you want to load; use the **Choose Source Physical View** section. If you are accessing the Load Physical View form for the first time, you will see the destination library and cell names appear in the Choose Source Physical View section. If you have accessed the form earlier and specified the library and cell name to be loaded from the source, you will see the same till you change the values.

1. To select an appropriate source cellview parameter to be loaded, click **Browse**.



2. This displays the **Library Browser** window. To list the available cell types within a library, select the library. Likewise, to list the available views for a cell, select the cell.



Note: You can also choose to view the categories in which the various library cells are classified. For this, select the **Show Categories** check box.

3. To apply your selections to the cellview you want to import, click **Close**. This takes you back to the Load Physical View form.



Tip

Alternatively, you can specify an appropriate library, cell, or view by typing in the exact name for each on the Load Physical View form. Remember each text field on the form is case-sensitive. After you select a source cellview that you want to load, use the remaining options on the Load Physical View form to select the properties to be imported from the source cellview into the target cellview. For information on how each property is handled during the cellview import, see [How Cellview Information is Handled](#).

To select all the check boxes on the form, click **Select All**. This selects all the properties for import into the target cellview. Selecting **None**, clears all the selections.

☒ Select All ☐ None

Section-wise details of the various sections on the Load Physical View form are provided below.

Update Instances

To specify the instances you want to load into the target cellview, use the **Update Instances** section. You can choose to update using an instance name or a master name by selecting the appropriate radio button.

Update Instances

☒ Using Instance Name ☐ Using Master Name

☒ Standard	☐ Custom Cells
☒ Macros	☐ IO Pads

☒ Add physical only instances

☒ Re-master parameterized instances

If you choose to update an instance using its name, select the **Using Instance Name** option; which is also the default update option on the form. This option is useful for updating specific instances for which the instance name at the source and that at the destination is exactly the same.

On the contrary, when you choose to update instances irrespective of their name, select the **Using Master Name** option. This allows the PAD type IO instances to be updated based on the labels they carry. If the source cellview carries IO instances displaying the same label text as that of instTerm associated with the destination IO instance, the destination IO instance is updated with the sourceIO instance. During this update, the label is also copied to the destination cellview instance.

Note:

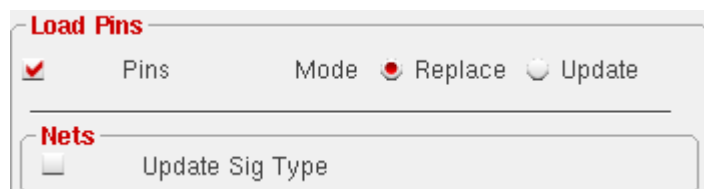
- When updating instances, only those instance attributes are loaded that do not impact the connectivity or the constraints of the target cellview.
- The Using Master Name option allows updating only the IO Pad type instances. For updating the other type of instances, select the **Using Instance Name** option.
- For the Standard, Custom Cell, and Macro type instances to be updated; the source and destination master for each instance should be the same.

Apart from loading a specific instance type, you can choose to add physical only instances. As the name suggests, adding physical-only instances allows you to add instances that exist only in the physical domain and lack a corresponding instance in the logical domain. For example, filler cells.

When updating instances, you can also choose to remaster parameterized cells by selecting the Re-master parameterized instances option. This remasters the parameterized source cell (p-cell) instances in the target cellview.

Load Pins

To load pins from the source cellview into the target cellview, select the **Pins** check box in the **Load Pins** section. You can choose to replace the existing pins in the target with the pins from the source, or update the pins. By default, the pins are replaced.



Nets

To update the Sig Type information on the nets, use the **Update Sig Type** option under the **Load Pins** section.

Note: For the sig type information to be updated, the net names in the source and destination cellviews should be the same.

Add Geometries

To ensure that the shapes and wires you import from the source cellview appear at exactly the same location in the target cellview, use the **Add Geometries** section. Using the Add Geometries feature not only loads the selected components (shapes and wires) from the source but it also establishes the same geometry in the target cellview as that in the source.



However, if a target cellview already has a shape or wire present at the same location as that in the source cellview, copying shapes and wires from the source view can lead to an overlap of the components in the target cellview. Therefore, it is recommended that you delete your existing shapes and wires (except modgens and pins) before adding the same from the source. To do this, select the **Delete Shapes and Wires in Destination** check box before loading an existing physical source view.

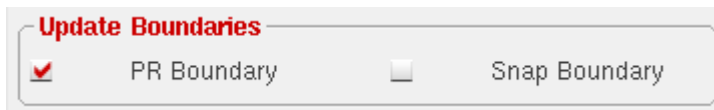
Note:

- The Delete Shapes and Wires in Destination option is available only for “updating” the existing geometries. Therefore, the option is enabled after you select a shape or wire to be updated. Else, the option remains disabled.
- The Add Geometries functionality checks out a Layout GXL license.

Update Boundaries

To import the boundary information from the source cellview to the target cellview, use the **Update Boundaries** section. You can choose to import the following types of boundaries into your destination cellview:

- Place and Route Boundary (PR Boundary)
- Snap Boundary



If a boundary you chose to import already exists in the source cellview, the points of the target boundary will get updated. Else, the boundary from the source cellview gets replicated at the target destination.

Replace Rows

To replace any standard or custom row information in the target design with the row information from the source view, select the appropriate option from the **Replace Rows** section.



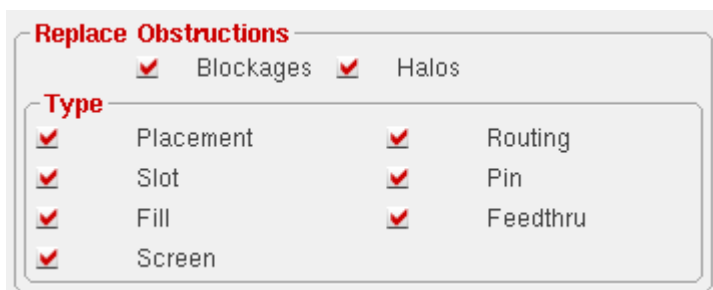
Note: The **Replace Rows** option first deletes existing rows from the destination and then replicates the same from the source cellview.

Replace Obstructions

To replace an existing blockage or halo from the target cellview with an appropriate obstruction from the source cellview, use the **Replace Obstructions** section. You can specify whether you want to replace an existing blockage or a halo by selecting appropriate check boxes.

In addition, you can select the type of obstructions you want to replace for blockages or halos.

Note:



- Before any new obstructions get loaded, the system looks for existing obstructions in the target cellview. Depending on the presence of the obstruction owner; which may be an instance, a cluster, or a PR boundary; an existing obstruction in the target cellview may or may not be updated with the one from the source cellview.
- Any existing obstructions in the target cellview automatically get deleted before any new obstructions get loaded into the target cellview.

Transfer Constraints

To transfer any relationship constraints in the schematic cellview and the process rule overrides from the source to the target view, you can use the **Transfer Constraints** section. Select the **All Constraints and Process Overrides** check box and specify if you would like the target cellview constraints to be replaced or updated.



The constraints are transferred from the source cellview to the target cellview depending on the default mapping rules applicable for each constraint.

Note: The **Transfer Constraints** functionality checks out a Layout GXL license.

Related Topics

[Layout XL Forms](#)

Match And Fix

Use the *Match And Fix* form to identify layout patterns in a design and to fix them according to the fixing rules or guidelines associated with each pattern.

This form has the following four tabs:

- *General Tab*

In this tab, you define the scope of the options to be used for matching and fixing the layout patterns.

- *Display Tab*

In this tab, you set one or more rule deck files and specify the set of rules to be applied to match and fix the layout pattern.

- *Extraction Tab*

In this tab, you specify the options for matching the layout pattern.

- *Fix Tab*

In this tab, you specify the options for fixing the layout pattern.

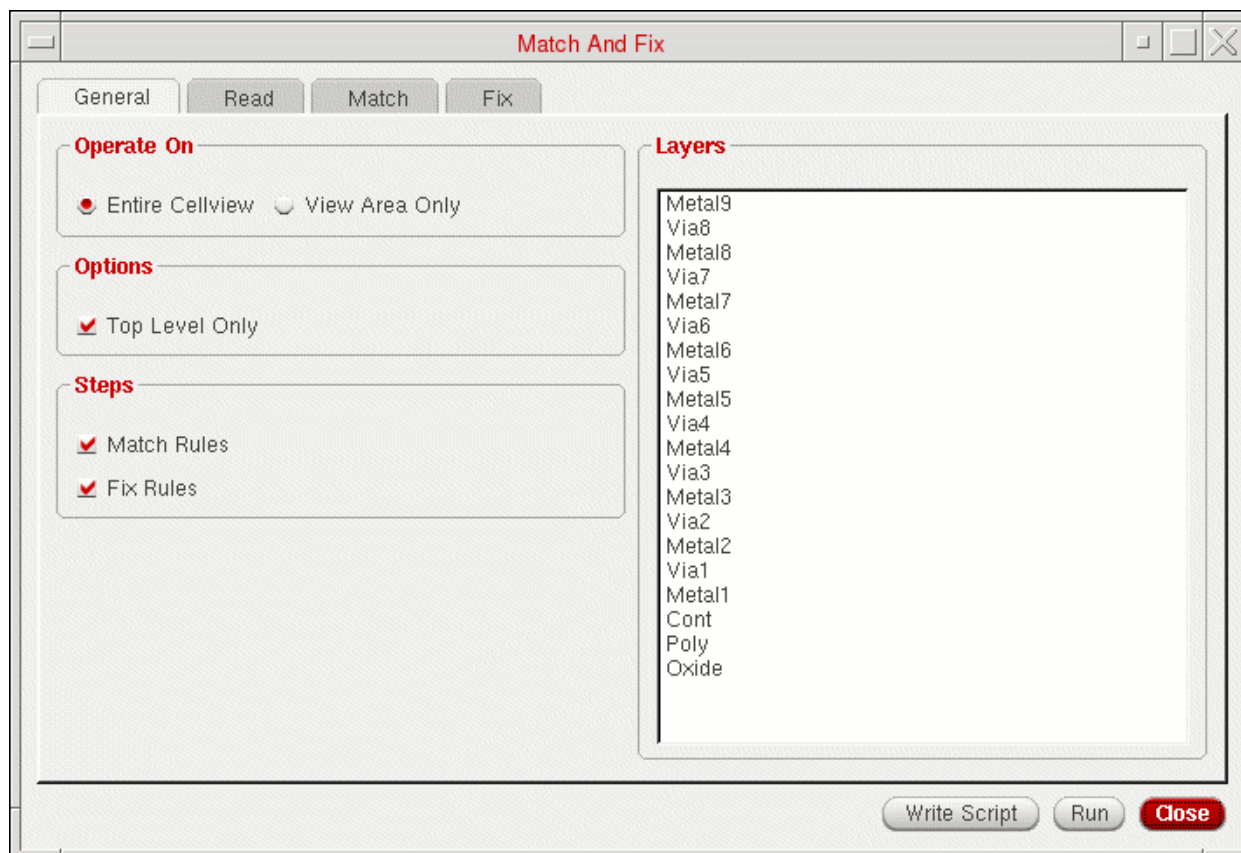
Related Topics

Match and Fix Flow

Layout XL Forms

General Tab

Use the *General* tab to define the scope of the options to be used for matching and fixing the layout patterns.



Operate On controls whether the matching and fixing of the layout patterns be done on the entire cellview or the selected view area only.

Entire Cellview specifies the matching and fixing of the layout patterns be done on the entire cellview.

View Area Only specifies the matching and fixing of the layout patterns be done on the view area only.

Options specifies whether or not the matching and fixing of the layout patterns be done only on the shapes at the top level of the open design.

Top Level Only specifies matching and fixing of the layout patterns be done only on the shapes at the top level of the open design.

Note: Fixing of layout patterns can only be done at the top level of the open design.

Steps controls whether to perform the match or to perform the automatic fixing of matched patterns or violations. Enabling either of these flows also enables modification of the corresponding tabs in the form.

Match Rules check box performs matching of layout patterns and enables modification of the options on the *Match* tab.

Fix Rules check box performs automatic fixing of the matched layout patterns and enables modification of the options on the *Fix* tab.

Layers lets you choose the layers on which the matching and fixing should be applied.

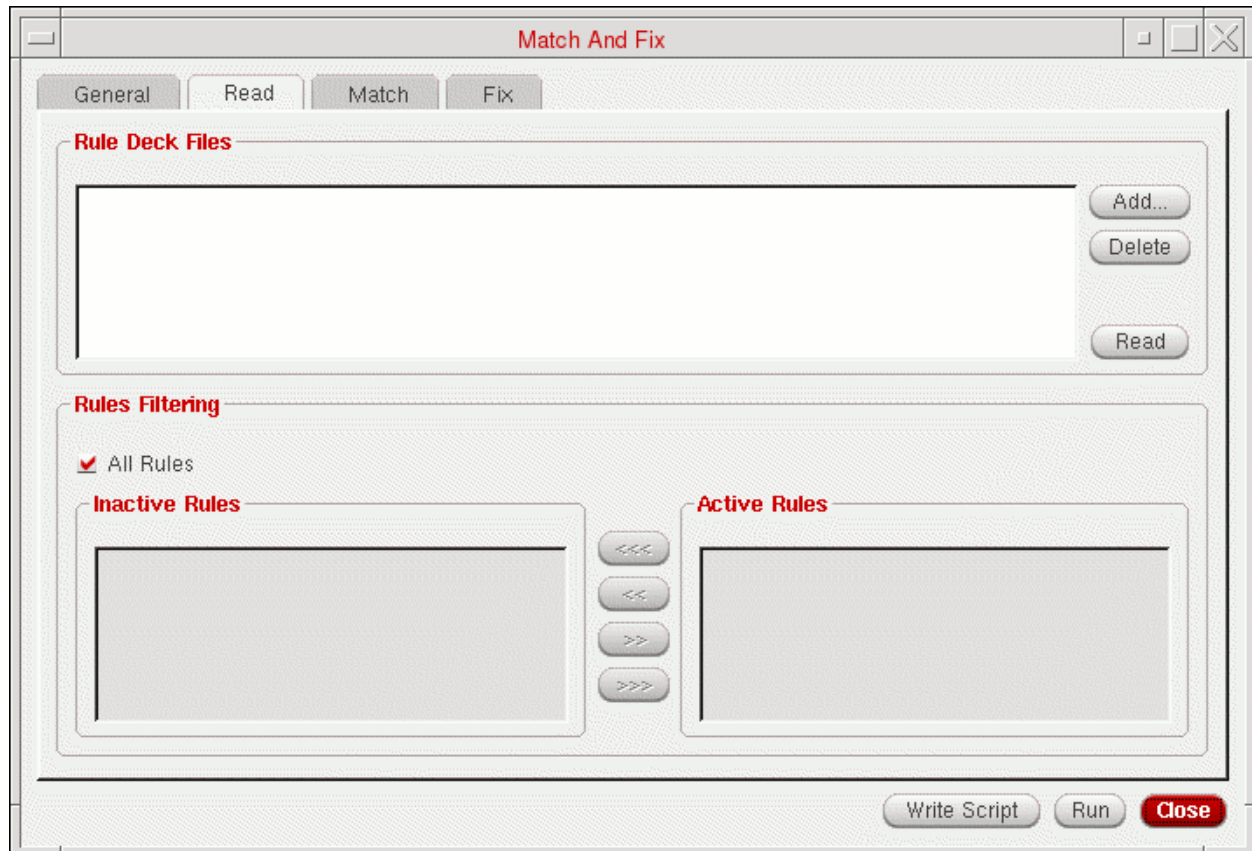
Read Tab

Use the *Read* tab to set one or more rule deck files and specify the active rules to be applied to match and fix the layout pattern.

Virtuoso Layout Suite XL User Guide

Layout XL Forms

A rule deck file defines pattern-based manufacturability checks. These match and fix rules are composed of a pattern definition and corresponding fixing rules. Consult with your foundry for a compatible rule deck file.



Rule Deck Files enables you to load the layout patterns from one or more rule deck files. Use the *Add* button to add files. Use the *Delete* button to delete a file. Use the *Read* button to read the rules specified in the selected rule deck files and import the active rules to be applied to match and fix the layout pattern.

Rules Filtering specifies the set of rules to be applied from the loaded rule deck files.

All Rules specifies that all rules from the selected rule deck files be applied to match and fix layout patterns.

By default, the option is ON.

Inactive Rules specifies the list of rules that should not be applied to match and fix layout patterns.

Active Rules specifies the list of rules that should be applied to match and fix the layout patterns.

Match Tab

Use the *Match* tab to specify the options for matching the layout pattern.

The screenshot shows the 'Match And Fix' dialog box with the 'Match' tab selected. The 'Output' section has 'Annotate' checked and 'Limit Annotations To' set to 1000. The 'Use Multithreading' section is unchecked, with 'Threads' set to 2, 'Available' set to 0, and 'Total' set to 2. The 'Close' button is highlighted in red.

Output controls whether or not to return an annotation for each match to the annotation browser and specifies the limit of the number of matches returned.

Annotate controls whether or not to return an annotation for each match to the annotation browser.

Limit Annotations To specifies the limit of the number of matches returned.

Note: By default, the number of matches returned are limited to 1000.

Use Multithreading specifies whether to enable multi-threaded search for additional efficiency while performing the match. By default, the *Use Multithreading* option is OFF.

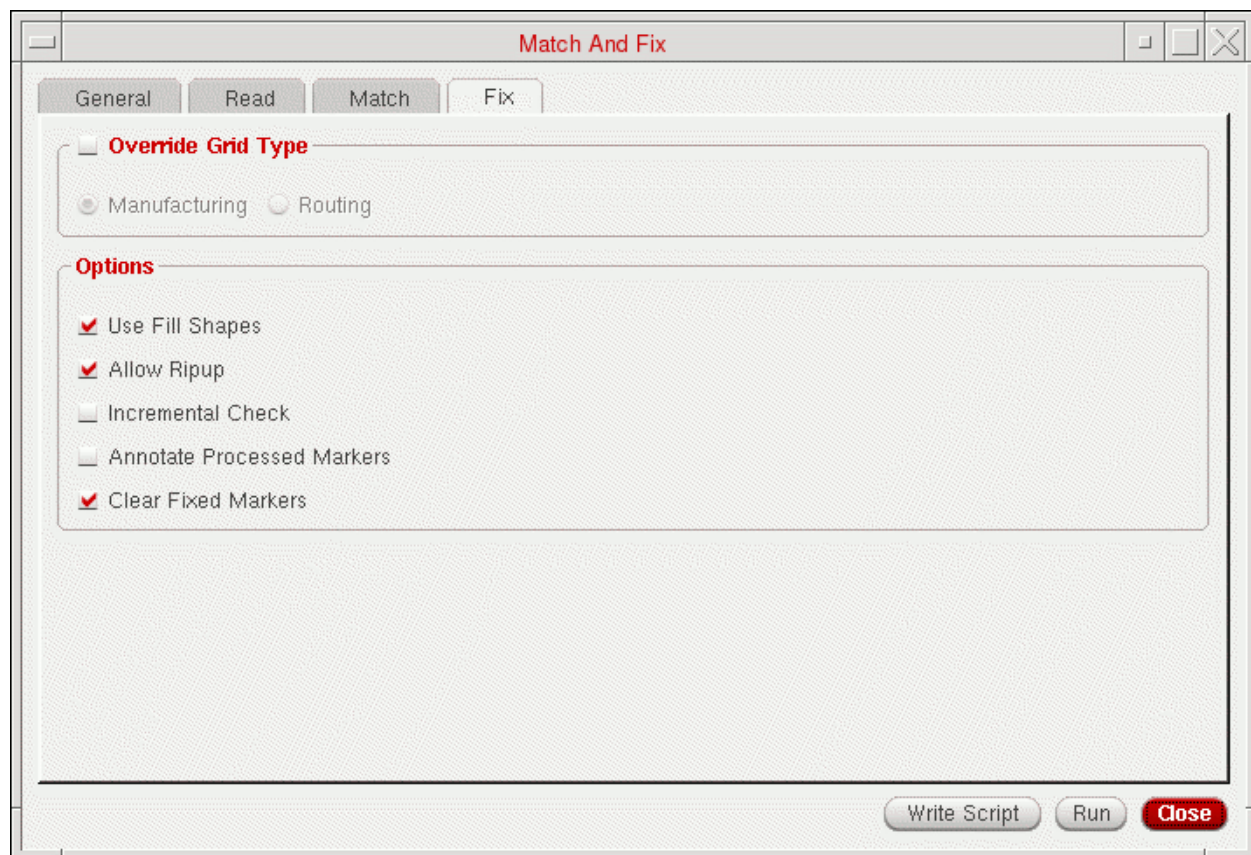
Threads specifies the number of threads to be used for matching the layout pattern.

Total specifies the number of processors available on the station.

Available specifies the difference between the total and used processors on the station.

Fix Tab

Use the *Fix* tab to specify the options for fixing the layout pattern.



Override Grid Type enables you to override the default grid type being used in the layout.

Manufacturing specifies that the manufacturing grid be used and all shapes in the layout be snapped to this grid.

Routing specifies that the routing grid be used and all shapes in the layout be snapped to this grid.

Options specifies the parameters for fixing the layout pattern.

Use Fill Shapes specifies use of fill shapes for fixing the layout pattern.

Allow Ripup specifies local ripup and reroute fixing be allowed for fixing the layout pattern.

Incremental Check controls the fixing engine to incrementally call the matching engine to check whether or not an active layout pattern has been introduced while fixing the layout pattern.

Annotate Processed Markers controls whether additional markers be added to the annotation browser to show the region of the layout that has been modified while fixing the layout pattern.

Clear Fixed Markers controls display of fixed matches or violations in the annotation browser. If this option is selected, all fixed markers are cleared from the annotation browser, otherwise, the fixing step does not clear any of the markers from the matching step.

Related Topics

[Match and Fix Flow](#)

[Layout XL Forms](#)

Modify Correspondence

Use the **Modify Correspondence** form to change the relative placement of instances in the target structure manually by changing the correspondence of the individual instances in the source and the target.

Connectivity Source lists the instances in the source structure. The list is read-only.

Connectivity Target lists the instances in the target structure. When you select an instance in the target structure, the corresponding instance in the source structure is highlighted in the *Connectivity Source* list.

You change the correspondence of an instance by selecting it in the *Connectivity Target* list and moving it up or down in the list using the arrow buttons. You can also swap the correspondence of a pair of target instances using the *Swap* button.

When you click *OK*, Layout XL checks that connectivity is still correct before accepting the new correspondence.

Related Topics

[Generating Clones](#)

[Generating a Clone using the Generate Clones Form](#)

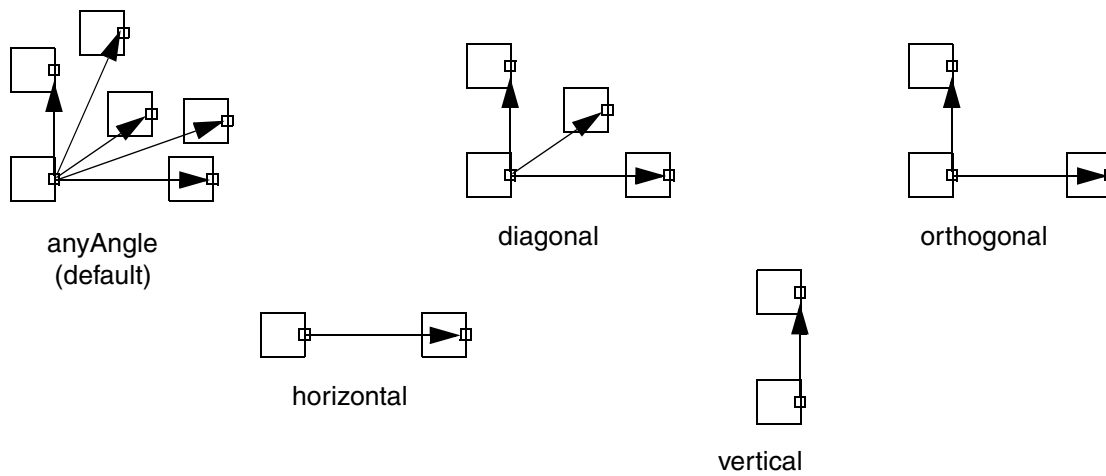
[Editing a Target Structure Before Generation](#)

[Layout XL Forms](#)

Move

Use the **Move** form to control how the move operation is performed, including the layer of the object you select, the angles at which you can move the object, the presence of draglines on the object as you move it, and the orientation of the object you select.

Snap Mode controls the direction in which you can move the object.



Change To Layer lets you move a shape (including level-1 pins) to another layer. Choose the new layer from the cyclic field, which lists all the available layers.

Chain Mode lets you specify how to move chained devices.

All moves the selected device and all the other devices in the same chain.

Selected moves the selected device out of the chain.

Selected Plus Left moves the selected device and all the devices to its left out of the chain.

Selected Plus Right moves the selected device and all the devices to its right out of the chain.

Delta X/Y specifies the distance to move the selected shape (including level-1 pins) in the X and Y direction. Click *Apply XY* to move the selected objects by the distances specified in the X and Y fields.

Display Draglines controls whether draglines are displayed during the *Move*, *Stretch* and *Generate Selected From Source* commands. By default, the draglines indicate connections from the pins of the object you are moving to pins of the nearest objects. For

information on how to change the display of draglines, see the *Draglines* options on the Display Tab form.



Tip

Displaying many draglines during interactive commands can impair the performance of Layout XL. To mitigate these effects, use the maxDragFig environment variable to limit the number of figures that can be in a dragset.

To set `maxDragFig` interactively so that it takes effect in the current session, type the following in the CIW, where *win* is the window to which the limit applies.

```
win~>maxDragFig = 1000
```

To set a default value for `maxDragFig`, which will be used in all future Layout XL sessions, put the following line in your `.cdsenv` file.

```
graphic maxDragFig int 1000 nil
```

Snap To Grid snaps the moved object to a grid depending on the block type. If it is a digital block, the place and route boundary is snapped to the placement grid and pins with placement status `placed` are snapped to the routing grid. If it is an analog block, both place and route boundary and `placed` pins are snapped to the manufacturing grid. Pins with placement status `unplaced`, `fixed`, `locked`, or `unknown` are not snapped, even if they were touching the boundary before the move.

Snap Pins To Boundary automatically snaps pins to the place and route boundary.

Click the buttons at the bottom to change the orientation of the selected instance.

Rotate rotates the component 90 degrees counterclockwise.

Sideways mirrors the component on the y axis (flips it horizontally).

Upside Down mirrors the component on the x axis (flips it vertically).



Tip

When constraint-aware editing is on, the *Move* command honors the following physical constraints: *Alignment*, *Fixed*, *Locked*, *Matched Parameters*, *Orientation*, *Matched Orientation*, and *Symmetry*.

- ❑ For more information on Constraint-Aware Editing mode, see Constraint-Aware Editing.
- ❑ For detailed definitions of the supported constraints, see Default Constraint Types in the *Virtuoso Unified Custom Constraints User Guide*.

Related Topics

[Moving Objects](#)

[Moving Objects Using the Move Form](#)

[Layout XL Forms](#)

Move Cells

Use the **Move Cells** form to move selected cells from one component type to another or to the *No component type* folder.

Select new component type lets you choose which component type should be set for the selected cells. Select the component type from the pull-down list and click *OK*.

No component type moves the selected cells to the *No component type* folder, which means that they are not assigned to any component type.

Related Topics

[Component Types Mode](#)

[Defining a Design-Level Component Type](#)

[Layout XL Forms](#)

Open Physical Configuration View

Use the **Open Physical Configuration View** form to open an existing physical configuration view to be used in a Layout XL session.

Library Name is the name of the library in which the physical configuration view resides.

Cell Name is the name of the cell to which the physical configuration view relates.

View Name is the name of the physical configuration view.

Open CPH opens the Configure Physical Hierarchy window automatically after you click *OK* in this form.

Related Topics

Configuring the Physical Hierarchy

Starting Layout XL from a Schematic View

Starting Layout XL with an Existing Physical Configuration View

Layout XL Forms

Permutation Information

The **Permutation Information** form tells you what changed as a result of using the *Permute Pins* command. Display it by pressing **F3** after you have run the command.

Related Topics

[Pin Permutation](#)

[Checking Permutation Information](#)

[Layout XL Forms](#)

Repeat Copy

Snap To Grid snaps the pin edges to the manufacturing or the routing grid, depending on the selected block type.

- If the block type is custom, the pin edges are snapped to the manufacturing grid.
- If the block type is digital, the pin edges are snapped to the routing grid.

Create Synchronous Copy creates a synchronized copy of the selected layout objects. For more information, see [Generating a Synchronous Copy](#).

For information about the other options available in this form, see the [Repeat Copy Form](#) in the *Virtuoso Layout Suite L User Guide*.

Related Topics

[Copy](#)

[Generating Synchronous Copy](#)

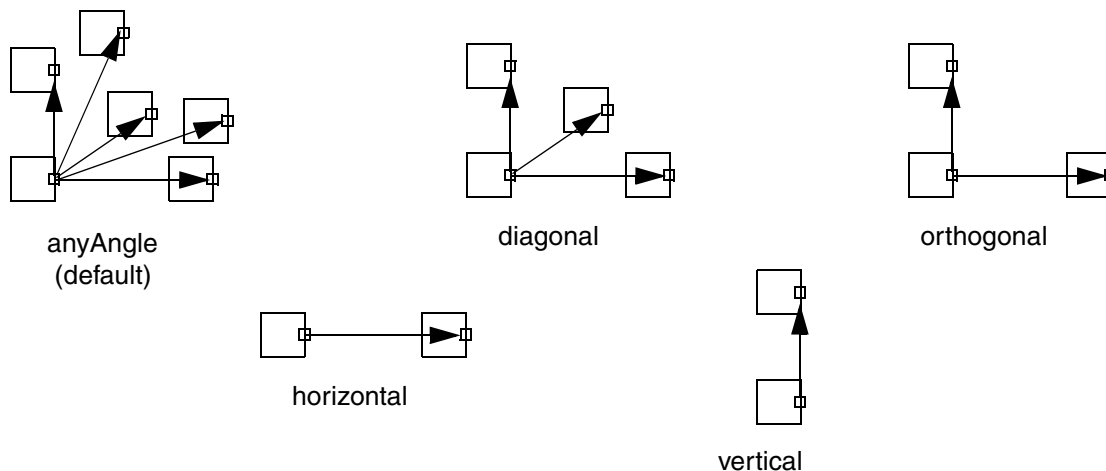
[Layout XL Forms](#)

Reshape

Use the **Reshape** form to control the operation of the *Reshape* command in Level-1 Editing mode. This command lets you reshape a place and route boundary without entering Edit In Place mode.

Reshape Type sets the geometry to use for reshaping a selected object. *rectangle* lets you add or remove a rectangle shape; *line* lets you add a polygon to a shape or reshape a path. Only *line* can be used to reshape a path.

Snap Mode controls the direction in which you can move the object.



Snap To Grid snaps the stretched object to a grid depending on the block type. If it is an *analog* block, the place and route boundary and pin edges are snapped to the manufacturing grid. If it is a *digital* block, the place and route boundary is snapped to the placement grid and the pin centers are snapped to the routing grid. Pins with placement status *unplaced*, *fixed*, *locked*, or *unknown* are not snapped, even if they were touching the boundary before the reshape.

Note: During the stretch operation, incase the soft block is stretched outside the toplevel prBoundary, then all the instance pins on the soft block, with the placement status as PLACED, are snapped to the manufacturing grid. This is irrespective of the block type because the routing grid is not initialized outside the prBoundary but the manufacturing grid exists outside the prBoundary.

Pin Snap Options

Snap Pins To Boundary automatically snaps pins to the place and route boundary.

Snap Soft Block Pins To automatically snaps soft block pins either to the *Modified Edge* or to the *Closest Edge* after the reshape is complete.

Related Topics

[Editing Soft Blocks](#)

[Reshaping an Object in Level-1 Editing Mode](#)

[Layout XL Forms](#)

Selection Options

Use the **Selection Options** form to control how selection is performed in the layout window.

You can choose between full and partial section modes, control the behavior of point-and-click and area selection functionality, specify automatic selection expansion for routing objects, and control the size and number of selected sets that can be saved in the system for later reuse.

Selection Controls

Mode defines whether selection is full or partial.

Full fully selects objects inside the area selection box.

Partial fully selects objects if you completely enclose them with the area selection box, and partially selects objects if you partially enclose them with the area selection box.

Note: In the *Partial* selection mode, the Routing Object Granularity cyclic field is not available.

Routing Object Granularity helps select routing objects by automatically expanding the selection based on the modes described below. You can also set these modes from the right mouse button context menu in the layout canvas.

Shapes or Vias selects a via or shape (either a path, pathSeg, rectangle, or polygon) used to make a wire. In Partial selection mode, this is the only granularity that is available.

When zoomed out, a single click selects an individual shape or via; two clicks extends the selection to include all the connected pathSegs in the wire until a T-junction or via is encountered.

When zoomed in, clicking in the middle of the shape selects only that shape; clicking on the boundary of the shape selects all the connected pathSegs in the wire until a T-junction or via is encountered.

Entire Wire automatically expands the selection to include all the objects forming the wire when you select a shape or via.

Connected Shapes automatically expands the selection to include all the objects with a physical connection to a selected shape or via. This mode does not consider net assignments on shapes and vias. The expansion is abandoned if more than 1000 objects are found – in this case, only the object under the mouse is selected.

All Shapes on Net automatically expands the selection to include all the objects assigned to the same net as a selected shape or via. If the selected shape or via has

no net assigned, then the selection is not expanded. The expansion is abandoned if there are more than 20,000 objects assigned to the net – in this case, only the object under the mouse is selected.

Net lets you select nets as well as instances from the canvas. When a net is selected on the canvas, the corresponding net in the Navigator Assistant is also selected. The selected net is probed in the layout window according to the options set in the Highlight Options Form.

The Net granularity behaves similar to *All shapes on Net* except that in this case the net object is selected instead of the shape. In fact, there is no limit to the number of shapes that can be highlighted. However, if a shape has no assigned net, no shapes are highlighted.

Important

For performance reasons only *Shapes or Vias*, *All Shapes on Net*, and *Net* modes are supported for area selection. If you perform an area selection when the routing granularity is set to either *Entire Wire* or *Connected Shapes*, the selection defaults to *Shapes or Vias* mode.

Spine automatically selects all contiguous pathsegs with the same width as the pathseg on which you click. It excludes pathsegs that branch off from the main spine at a T junction. This option is switched on by default; if you switch it off, only the pathSeg you click is selected.

Note: This option is enabled only when *Routing Object Granularity* is set to *Shapes or Vias*.

Point Selection Controls

Overlap Mode

Toggle mode lets you change selection between two objects.

Click using the left mouse button on one object and the software toggles back and forth between the smaller object and the larger object.

Add objects to the selected set by holding down the `Shift` key and selecting additional objects.

Delete selections by holding down the `Ctrl` key and clicking the left mouse button on the original selected object. Objects are deselected in the reverse order to which they were selected.

Once you have selected the objects you want in the set you can also click on the selected edge of one of the objects. This deletes all the selections except the one you have clicked on. This does not delete the objects themselves.

Cycle mode lets you select overlapped objects one by one by using the left mouse button to select any object and continuing to select them by clicking on the original object until you reach the largest overlapped object.

Use Cycle mode when three or more objects overlap at the same point.

Cycle mode always cycles from the smallest object to the largest object. If you continue to click on the original object after reaching the largest object, the cycle returns to the smallest object you first clicked on. The cycle repeats by re-selecting the smallest object you clicked on and selecting outwards again to the largest object. If you start by selecting the largest object in the overlapping stack the selection will not cycle back to the smaller objects. You must select a smaller object so that each selected point selects larger and larger objects.

Multiple objects are not retained in the selected set. The final selected object is the selection that is retained. You can deselect by clicking a point outside of all the overlapping objects.

Add objects to the selected set by holding down the `Shift` key and selecting additional objects.

Delete selections by holding down the `Ctrl` key and clicking the left mouse button on the original selected object. Objects are deselected in the reverse order to which they were selected.

Once you have selected the objects you want in the set you can also click on the selected edge of one of the objects. This deletes all the selections except the one you have clicked on. This does not delete the objects themselves.

The objects are selected according to the selection filter set for the Layer Selection Window and the Object Selection Window.

Cycle mode is window-based.

User Aperture Mode enables the point selection aperture function and lets you specify the *Size* of the aperture in user units. This helps you select objects that are not directly under the cursor but close to it (within the distance specified by the aperture).

off specifies that point selection does not use the specified aperture.

on specifies that point selection uses the specified aperture.

visible specifies that point selection uses the specified aperture and that the aperture is displayed on the canvas.

Size defines a box around the initial point. You can select multiple points within this aperture range and the software will treat it as one point.

Area Selection Controls

Full Mode controls area selection in Full selection mode.

Enclosed Figures selects the objects you completely enclose with the area selection box.

Enclosed and Crossed Figures selects the objects you completely or partially enclose with the area selection box.

Crossed Figures selects objects you partially enclose with the area selection box, but does not select any fully-enclosed objects.

Partial Mode controls area selection in Partial selection mode.

Vertex selects all the objects you completely enclose with the area selection box, vertices of any objects you partially select with the area selection box, and edges of any objects you enclose with the area selection box.

Enclosed Edges selects all the objects you completely enclose with the area selection box, and the edges of any objects you partially enclose with the area selection box.

Crossed Edges selects the edges of any objects you enclose with the area selection box, and does not select fully-enclosed objects.

Enclosed and Crossed Edges selects all the objects you completely enclose with the area selection box, the vertices of any objects you partially enclose with the area selection box, and the edges of any objects you enclose with the area selection box.

Auto Store Selection Controls stores multiple selected sets you have created. You can reload a previously-selected set into the cellview without having to manually reselect the objects in the canvas.

Auto Store enables the auto store functionality. The default is off.

- ☐ The selected set consists of full or partially selected figures.
- ☐ The selected set is a cellview property. The *Edit – Select – Save/Restore* is not cumulative. Once the cellview is closed the recorded information is not retained.
- ☐ You can iterate through the sets by selecting the sets in the in the Save/Restore Selection Set form or by using the *Edit – Select – Next* or *Edit – Select – Previous* commands.

- ❑ Deleted or modified objects are removed from a saved selection set. If all objects are removed from the selection set, the set is removed from the Save/Restore Selection Set form.
- ❑ Restore is independent of the object selectability of the Layer Selection Window and the Object Selection Window.
- ❑ The command is not compatible with the *Undo* or *Redo* commands.
- ❑ The command does not work with the *Edit In Place* or *Descend* commands if the lower-level cellview is already open.

Max number of Stored sets limits the number of sets saved in the Save/Restore Selection Set form. The default is 10, which is also the maximum number that can be stored. When the limit is reached, the oldest saved selection set is removed from the list.

Max number of selected elements within a set limits the number of objects you can save in a single selection set. The default is 100, which is also the maximum number that can be saved.

Related Topics

[Editing Objects](#)

[Layout XL Forms](#)

Set Net Order

Use the **Set Net Order** form to create an ordered net constraint on a group of nets. An ordered net constraint is honored by the pin optimizer engine, which optimizes the nets while maintaining the order specified by the constraint.

Net Order Name specifies a name for the ordered net constraint. If you do not specify a name, the system generates one for you.

Selected Nets lists the nets that are currently selected in the layout canvas. Use the *Add* button to add one or more selected nets to the constraint.

Nets Name helps you find nets in the list of selected nets. Type in a string and click *Filter* to see only the nets you specified.

Ordered Net Members lists the names of the nets in the constraint. Use the *Remove* button to move one or more selected nets out of the constraint.

Related Topics

[Creating and Editing Net Constraints](#)

[Layout XL Forms](#)

Set Pin Label Text Style

Use the **Set Pin Label Text Style** form to set the style of the labels used to annotate pins in the design window.

Height sets the height of the label.

Font sets the text style of the label. The choices are *euroStyle*, *gothic*, *math*, *roman*, *script*, *stick*, and *swedish*.

Text Options

Drafting prevents the label from being rotated more than 90 degrees.

Overbar determines how text strings containing underscore characters are displayed in a layout window.

The default is off, which means that the software displays underscore characters (`_`) as part of the text string. When switched on, the software interprets underscore characters in the text string name as toggle switches that control where overbars begin and end. Overbars appear above the text string, as shown in the examples below.

Text string	Appears in design window as
<code>_abcde</code>	abcde
<code>ab_cde</code>	ab $\overline{cde}$
<code>_abc_de</code>	$\overline{abc}de$
<code>ab_cd_ef_gh_ij</code>	ab $\overline{cd}ef\overline{gh}ij$

Layer Name sets the layer on which the labels are generated. Choose a layer from the cyclic field or select *Same As Pin* to create the label on the same layer as the pin with which it is associated.

Layer Purpose sets the layer purpose on which the labels appear. Choose a purpose from the cyclic field or select *Same As Pin* to create the label on the same layer purpose as the pin with which it is associated.

Justification sets the location of the label origin. The origin appears as a small square on the label when you place or select it. Choose from *lowerLeft*, *centerLeft*, *upperLeft*, *lowerCenter*, *centerCenter*, *upperCenter*, *lowerRight*, *centerRight*, and *upperRight*.

Virtuoso Layout Suite XL User Guide

Layout XL Forms

Orientation lets you specify the orientation of the labels. Choose from

Value	Meaning
R0	No rotation
R90	Rotate 90 degrees clockwise
R180	Rotate 180 degrees clockwise
R270	Rotate 270 degrees clockwise
MY	Mirror over the Y axis
MYR90	Mirror over the Y axis and rotate 90 degrees clockwise
MX	Mirror over the X axis
MXR90	Mirror over the X axis and rotate 90 degrees clockwise

Related Topics

[Generating All Components from Source](#)

[Specifying the I/O Pins to be Generated](#)

[Layout XL Forms](#)

Snap Pins

Use the **Snap Pins** form to snap top-level and level-1 pins to the grid appropriate to the block type you are editing.

Snapping Mode lets you select whether *All* pins or only *Selected* pins are snapped. Click *Snap Pins* to snap the pins to the grid appropriate to the block type you are editing.

Block Type	Snap Behavior
<i>Analog</i>	The place and route boundary and pin edges are snapped to the manufacturing grid.
<i>Digital</i>	The place and route boundary is snapped to the placement grid and pin centers are snapped to the routing grid defined for the metal layer on which the pins are defined. Note: If the pins are not on a metal layer, they are snapped to the manufacturing grid.

Related Topics

[Snapping Soft Block Pins to Grid](#)

[Layout XL Forms](#)

Soft Block Global Options

Use the **Soft Block Global Options** form to specify that labels are to be created for all the pins in all the soft blocks in your design.

Create Label specifies that pin labels are to be created for all the pins in all the soft blocks in the design.

Use the **Pin Label Text Style** group box to set the style of the labels used to annotate pins in the design window.

Height sets the height of the label.

Font sets the text style of the label. The choices are *euroStyle*, *gothic*, *math*, *roman*, *script*, *stick*, and *swedish*.

Text Options

Drafting prevents the label from being rotated more than 90 degrees.

Overbar determines how text strings containing underscore characters are displayed in a layout window.

The default is off, which means that the software displays underscore characters (`_`) as part of the text string. When switched on, the software interprets underscore characters in the text string name as toggle switches that control where overbars begin and end. Overbars appear above the text string, as shown in the examples below.

Text string	Appears in design window as
<code>_abcde</code>	<code>abcde</code>
<code>ab_cde</code>	<code>abcde</code>
<code>_abc_de</code>	<code>abcde</code>
<code>ab_cd_ef_gh_ij</code>	<code>abcdefghij</code>

Layer Name sets the layer on which the labels are generated. Choose a layer from the cyclic field or select *Same As Pin* to create the label on the same layer as the pin with which it is associated.

Layer Purpose sets the layer purpose on which the labels appear. Choose a purpose from the cyclic field or select *Same As Pin* to create the label on the same layer purpose as the pin with which it is associated.

Virtuoso Layout Suite XL User Guide

Layout XL Forms

Justification sets the location of the label origin. The origin appears as a small square on the label when you place or select it. Choose from *lowerLeft*, *centerLeft*, *upperLeft*, *lowerCenter*, *centerCenter*, *upperCenter*, *lowerRight*, *centerRight*, and *upperRight*.

Orientation lets you specify the orientation of the labels. Choose from

Value	Meaning
R0	No rotation
R90	Rotate 90 degrees clockwise
R180	Rotate 180 degrees clockwise
R270	Rotate 270 degrees clockwise
MY	Mirror over the Y axis
MYR90	Mirror over the Y axis and rotate 90 degrees clockwise
MX	Mirror over the X axis
MXR90	Mirror over the X axis and rotate 90 degrees clockwise

Related Topics

[Specifying I/O Pins](#)

[Creating Pin Labels](#)

[Layout XL Forms](#)

Virtuoso Layout Suite XL User Guide

Layout XL Forms

Startup Option

Use the Startup Option form to control the Layout XL launch operation. Depending on what you specify, you will be asked to complete different forms as Layout XL is launched.

Layout	Configuration	You see the following dialogs...
<i>Create New</i>	<i>Create New</i>	New File Create Physical Configuration View
<i>Create New</i>	<i>Use Existing</i>	New File Open Physical Configuration View
<i>Create New</i>	<i>Automatic</i>	New File
<i>Use Existing</i>	<i>New</i>	Open File Create Physical Configuration View
<i>Use Existing</i>	<i>Use Existing</i>	Open File Open Physical Configuration View
<i>Use Existing</i>	<i>Automatic</i>	New File

Layout controls whether the system creates a new layout view or opens an existing one for the schematic.

Create New opens the New File form where you type the name of a new layout view for your Layout XL design. For more information, see [Managing Libraries](#) in the *Cadence Library Manager User Guide*.

Open Existing opens the Open File form where you can choose an existing layout view. For more information on this form, see [Working with Cellviews](#) in the *Virtuoso Design Environment User Guide*.

Configuration controls the physical configuration view used for your design.

Create New opens the [Create Physical Configuration View](#) form where you can specify a new physical configuration view for the design. (This happens after you have specified the layout view.)

Open Existing opens the [Open Physical Configuration View](#) form where you can choose an existing physical configuration view. (This happens after you have specified the layout view.)

Automatic tells Layout XL to handle all operations related to the physical configuration view automatically.

The system creates a physical configuration view (or opens it, if one already exists) containing the minimum information required for Layout XL to operate correctly, and manages this information automatically throughout the session.

If you need to modify the hierarchy configuration, component type definitions, or soft blocks at a later time, use the *Launch – Configure Physical Hierarchy* command. For more information, see [Configuring the Physical Hierarchy](#).

Related Topics

[Starting Layout XL](#)

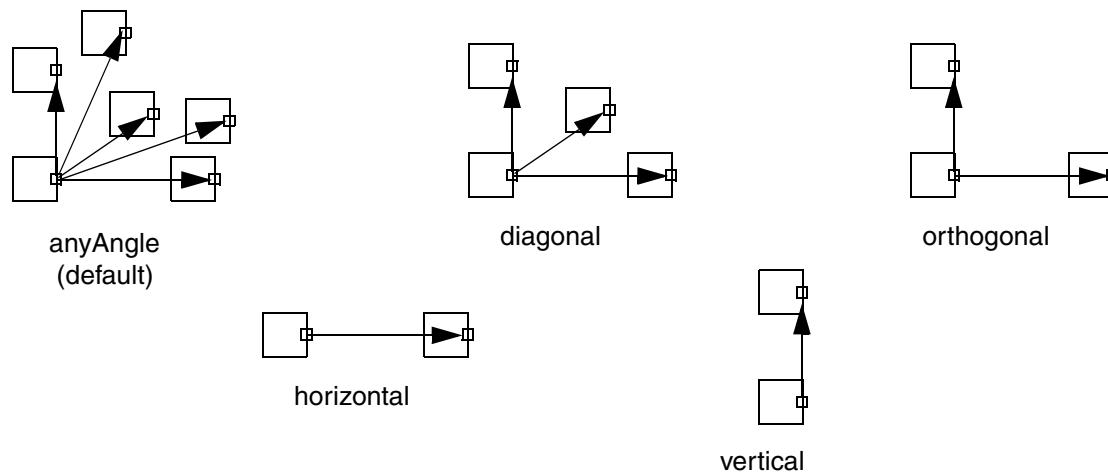
[Starting Layout XL from a Schematic View](#)

[Layout XL Forms](#)

Stretch

Use the **Stretch** form to control the operation of the *Stretch* command in Level-1 Editing mode. This command lets you stretch multiple shapes, including multiple boundaries both at level 1 and at the top level, in a single operation.

Snap Mode controls the direction in which you can stretch an object.



Lock Angles prevents you from changing the angle of a corner or edge as you stretch it.

Keep Wires Connected extends wires when required to maintain existing connectivity during the stretch.

Chain Mode specifies how to stretch chained devices.

All stretches the selected device and all the other devices in the same chain.

Selected stretches only the selected device

Selected Plus Left stretches the selected device and all the devices to its left in the chain

Selected Plus Right stretches the selected device and all the devices to its right in the chain.

Delta X/Y specifies the distance to stretch the selected objects in the X and Y direction. Click *Apply XY* to stretch the selected objects by the distances specified in the X and Y fields.

Display Draglines controls whether draglines are displayed during the *Stretch* command. By default, the draglines indicate connections from the pins of the object you are moving to pins of the nearest objects.

Virtuoso Layout Suite XL User Guide

Layout XL Forms

For information on how to change the display of draglines, see the *Draglines* options on the [Layout XL Display Options](#) form.



Tip

Displaying many draglines during interactive commands can impair the performance of Layout XL. To mitigate these effects, use the `maxDragFig` environment variable to limit the number of figures that can be in a dragset.

To set `maxDragFig` interactively so that it takes effect in the current session, type the following in the CIW, where *win* is the window to which the limit applies.

```
win~>maxDragFig = 1000
```

To set a default value for `maxDragFig`, which will be used in all future Layout XL sessions, put the following line in your `.cdsenv` file.

```
graphic maxDragFig int 1000 nil
```



Tip

When constraint-aware editing is on, the *Stretch* command honors the following physical constraints: *Alignment*, *Fixed*, *Locked*, *Matched Parameters*, *Orientation*, *Matched Orientation*, and *Symmetry*.

- ☐ For more information on Constraint-Aware Editing mode, see [Constraint-Aware Editing](#).
- ☐ For detailed definitions of the supported constraints, see [Default Constraint Types](#) in the *Virtuoso Unified Custom Constraints User Guide*.

Snap To Grid snaps the stretched object to a grid depending on the block type. If it is an *analog* block, the place and route boundary and pin edges are snapped to the manufacturing grid. If it is a *digital* block, the place and route boundary is snapped to the placement grid and the pin centers are snapped to the routing grid. Pins with placement status `unplaced`, `fixed`, `locked`, or `unknown` are not snapped, even if they were touching the boundary before the stretch.

Note: During the stretch operation, incase the soft block is stretched outside the top-level prboundary, then all the instance pins on the soft block, with the placement status as `PLACED`, are snapped to the manufacturing grid. This is irrespective of the block type because the routing grid is not initialized outside the prboundary but manufacturing grid exists outside the prboundary.

Pin Snap Options

Snap Pins To Boundary automatically snaps pins to the place and route boundary.

Snap Soft Block Pins To automatically snaps soft block pins either to the *Modified Edge* or to the *Closest Edge* after the reshape is complete.

Use Constant Area Stretch For Soft Block keeps the area of a soft block constant while its place and route boundary is stretched. Additional options let you specify the adjustable edges, i.e., the edges that can be moved automatically in order to maintain a constant area.

Related Topics

[Stretching Objects](#)

[Stretching an Object in Level-1 Editing Mode](#)

[Stretching Multipart Paths](#)

[Stretching Wires and Vias](#)

[Removing Loops from Wires](#)

[Layout XL Forms](#)

Update Clone Families

Use the **Update Clone Families** form to manage the families of synchronized clones in your design. You can update the status of a clone family, remove a clone from a family, create and remove clone families and add a group to an existing family.



You cannot undo the changes you make using this form.

Automatically zoom to selected zooms in on the clone or clone family that is currently selected in the *Clone Families* pane, or on the layout group selected in the *Layout groups* pane. Clicking on the top-level fits the whole cellview in the layout window. If you select more than one layout group, the software zooms in on the bounding box that encloses the selected groups.

Note: This option is considered only when you make the selection in the Update Clone Families form; selecting a synchronous clone in the canvas does not zoom the display to the selected clone or group.

Clone Families lists the clone families in the current design. You can expand each entry to display the clones it contains.

The buttons on the right hand side of the form operate on the families displayed in the pane on the left.

Verify Synchronization checks the consistency of the clones in the selected family (or in all the families if no family is selected) and makes any changes required to resynchronize the family.

- ☐ Any clone that is out of synch with the other clones in the family is removed from the family.
- ☐ If two or more clones are out of synch with the family but in synch with each other, they are moved into a new clone family.
- ☐ If all the clones in the family are different, the family is removed altogether.

The results are printed in the CIW.

Note: You cannot undo *Verify Synchronization*.

Remove Family removes the selected clone family from the design. This button is enabled only when you select a clone family. You cannot remove a family if one of its members is currently being edited in place. You need to exit the *Edit In Place* command first.

Virtuoso Layout Suite XL User Guide

Layout XL Forms

Remove Clone removes the selected clone from a family. This button is enabled only when you select a clone. You cannot remove a clone if one of its members is currently being edited in place. You need to exit the *Edit In Place* command first.

Create/Add To Family expands the form to let you create a new family of synchronous clones or add clones to an existing family.

The screenshot shows the 'Update Clone Families' dialog box. At the top, a note states: 'Note: Changes you make in this form cannot be canceled.' Below this is a checkbox labeled 'Automatically zoom to selected'. The main area is titled 'Clone Families' and contains a tree view showing a hierarchy: 'ether adc_cascode_opamp layout' > 'cloneFamily1' > 'SynchronousClone1 (JNM12)', 'SynchronousClone2 (JNM11)', 'SynchronousClone3 (JNM15)', and 'SynchronousClone5 (JNM17)'. To the right of the tree are five buttons: 'Verify Synchronization', 'Rename Family', 'Remove Family', 'Remove Clone', and '<< Create/Add To Family'. Below the tree, there are two paragraphs of instructions. The first paragraph says: 'To create a new family, select two or more groups, choose one of the groups as the reference for the synchronization, and click Create Family.' The second paragraph says: 'To add groups to an existing family, select a family and one or more groups, choose either the family or one of the groups as the reference for the synchronization, and click Add To Family.' Below these instructions is a section titled 'Layout groups' containing a list box with 'SynchronousClone4 group1'. To the right of this list box is a 'Synchronization reference' label and a dropdown menu currently showing 'SynchronousClone4'. To the right of the dropdown are two buttons: 'Create Family' and 'Add To Family'. At the bottom right of the dialog are 'Close' and 'Help' buttons.

Note: Changes you make in this form cannot be canceled.

☐ Automatically zoom to selected

Clone Families

- ether adc_cascode_opamp layout
 - cloneFamily1
 - SynchronousClone1 (JNM12)
 - SynchronousClone2 (JNM11)
 - SynchronousClone3 (JNM15)
 - SynchronousClone5 (JNM17)

Buttons:

- Verify Synchronization
- Rename Family
- Remove Family
- Remove Clone
- << Create/Add To Family

To create a new family, select two or more groups, choose one of the groups as the reference for the synchronization, and click Create Family.

To add groups to an existing family, select a family and one or more groups, choose either the family or one of the groups as the reference for the synchronization, and click Add To Family.

Layout groups

- SynchronousClone4 group1

Synchronization reference: SynchronousClone4

Buttons:

- Create Family
- Add To Family

Close Help

In this expanded form **Layout groups** lists all the regular layout groups not currently assigned to a clone family. It does not list any other type of group.

Important

When you remove a clone from an existing group, its type is change to `none` and it is added to the list of layout groups. However, its name is left unchanged, which means that the list might contain names like `SynchronousClone123`. Such groups are regular groups and **not** synchronous clones.

Synchronization reference lets you choose how the members of the new or updated family are synchronized. All routine geometry is synchronized, including shapes and vias. However, labels and text displays are not synchronized.

- ☐ If you are creating a new family, the drop-down list contains all the *Layout groups* selected to be included in the family. Choose the group against which you want to synchronize the other members of the family.
- ☐ If you are adding groups to an existing family, the drop-down list contains the family itself and all the groups you want to add to it. If you choose the family, the new members are synchronized against the existing members of the family. If you choose one of the groups, the new and existing members are synchronized against the group you selected.

Click *Create Family* to create a new family containing the members highlighted in the *Layout groups* list. This button is enabled only when you select two or more layout groups and there is no existing family selected in the *Clone Families* pane.

Click *Add To Family* to add one or more selected groups to an existing family. This button is enabled only when you select an existing family in the *Clone Families* pane and at least one group from the *Layout groups* pane.

Related Topics

[Updating Clone Families](#)

[Verifying a Clone Family](#)

[Removing a Clone from a Family](#)

[Removing a Clone Family](#)

[Creating a New Clone Family](#)

[Adding a Group to an Existing Family](#)

Virtuoso Layout Suite XL User Guide

Layout XL Forms

[Editing a Clone in a Synchronized Family](#)

[Generating a Family of Synchronous Clones](#)

[Layout XL Forms](#)

Update Components and Nets

Use the **Update Components and Nets** form to automatically update your layout to take account of instances, pins, and connectivity you have changed in the schematic.

While *Generate All From Source* deletes any existing components in the layout view and regenerates everything from scratch, *Update Components And Nets* updates only the components that have changed. It

- Adds new instances and pins.
- Removes old instances (including any unbound vector and mosaic instances) and pins, along with any empty nets left as a result of the removal.
- Updates instance masters, connectivity and instance, terminal, and net names to match those in the schematic.
- Removes any non-matching global terminals on the net, if a matching terminal already exists. Such non-matching global terminals may result if an original global terminal is modified, may be due to a name change, in an associated cell.

The form is split into three tabs.

- Update Tab
- I/O Pins Tab
- PR Boundary Tab

Related Topics

Updating Components and Nets

Layout XL Forms

Update Tab

The **Update** group box lets you specify how much of the design is updated and what happens to existing layout components that need to be changed or removed from the design.

Update Selected Layout Components Only updates only the instances and pins currently selected in the layout window.

Note: This option automatically disables the generation of new components. However, if the *Update Instance Masters* option is enabled and set to *Creating a New*, Layout XL may generate new instances to replace existing instances with

incorrect masters; you have the option to switch on *Chaining, Folding, and Chain Folds* for these new instances.

Update Nets and Instance Name Mismatches Only specifies that only net assignments and instance, terminal, and net names are updated. Enabling this option also automatically preserves user-defined mappings. All other options on the form are disabled, except *Update Selected Layout Components Only*.

Update Net Signal Types specifies that the signal types assigned to nets in the schematic are updated in the layout view.

Update Instance Masters by (Replacing the Old or Creating a New) Instance updates an existing instance that uses an incorrect master to use the correct master.

- ☐ The default is *Replacing the Old*, which means the command removes the incorrect instance and replaces it with an instance of the correct master in the same location.
- ☐ When set to *Creating a New*, the command puts a marker on the instance with the incorrect master and renames it *name_old*. It then creates a new instance with the correct master and places it below the PR boundary.

Note: Because you are creating a new instance, you can optionally enable *Chaining, Folding, and Chain Folds* in this mode.

Delete Unmatched Pins deletes layout pins that are no longer present in the schematic. Redundant nets and terminals are deleted from the layout view at the same time. When switched off, unmatched pins are not deleted but are instead indicated with a marker in the layout view.

Delete Unmatched Instances deletes layout instances that are no longer present in the schematic. When switched off, unmatched instances are not deleted but are instead indicated with a marker in the layout view.

Update Layout Parameters automatically updates the parameters and parameter values on layout instances to match those on their schematic counterparts. Parameters that are set in layout instances but are not present on their schematic counterparts are not removed.

Move Changed Overlapping Instances Below PR Boundary automatically places instances below the PR boundary, if they overlap other instances during the update. The overlapping instances are the ones that have changed in size, and if placed at their original location, may overlap other unchanged instances.

This data is then used by the Virtuoso Custom Digital Placer (VCP) for its Eco Mode placement.

The **Generate** group box lets you specify which missing design objects are generated in the layout view.

Instances generates instances that are in the schematic but are not currently in the layout.

Chaining automatically abuts transistors into chains during layout generation.

Folding automatically divides devices into folds to prevent gate width from exceeding a specified size.

Chain Folds automatically chains the individual folds of a transistor.

Note: For the *Chain Folds* option to be available, only the *Folding* check box must be selected. If you select the *Chaining* check box as well, *Chain Folds* is deactivated.

I/O Pins generates all the pins listed on the *I/O Pins* tab. The generated pins are automatically snapped to the placement grid.

Except Global Pins stops Layout XL generating layout pins for the global nets in the schematic.

Except Pad Pins stops Layout XL generating layout pins for schematic pins that are connected to I/O pads (cells of type `pad`, `padSpacer`, or `padAreaIO`). When unchecked, the software generates both pads and pins.

PR Boundary regenerates the new place and route boundary based on the settings on the *PR Boundary* tab. All placements and estimations are based on the generated PR boundary. When switched off, the existing boundary is retained.

Snap Boundary generates a rectangular snap boundary that encloses the generated PR boundary. You can generate a snap boundary only if the *PR Boundary* option is switched on.

Device Correspondence

Preserve User-Defined Mappings preserves user-defined mappings of devices between the schematic and the layout. This option preserves only user-defined one-to-one, many-to-many, many-to-one, and one-to-many device mappings defined in the Define Device Correspondence form. It does not report missing devices or shapes within a mapped group.

Related Topics

[I/O Pins Tab](#)

[PR Boundary Tab](#)

Layout XL Forms

I/O Pins Tab

Specify Default Values For All Pins lets you specify attribute values and *Apply* them to all the pins shown in the list box. Click *apply* to apply the *Layer*, *Width*, *Height*, *Num*, and *Create* settings for all the listed pins.

Layer specifies the layer on which the pin is generated. The cyclic field offers only the conducting layers. The default is the current drawing layer if it is defined as an extractable layer in the technology file; otherwise it is the first extractable layer found. If there are no extractable layers, the cyclic field lists all valid layout layers.

Width specifies the width for each pin. The default is the `minWidth` value set for the current layer in the technology file. Any change to the value is applied only if the new value is greater than the default value.

Height specifies the height for each pin. The default is the `minWidth` value set for the current layer in the technology file. Any change to the value is applied only if the new value is greater than the default value.

Num specifies how many instances of this pin to generate. If you type 0, the pin is not generated.

Create specifies that pins are to be generated in the layout.

Specify Pins To Be Generated lets you select pins from the list box and update the attribute values used when those pins are generated in the layout. Click *Update* to update the *Layer*, *Width*, *Height*, *Num*, and *Create* settings for the currently selected pins.

Select lets you type in a complete or partial pin name to select one or more pins in the list box.

Number Of Matches displays the number of pin with names that match the string you typed into the *Select* field.

Add New Pin opens the Add A New Pin dialog where you can specify the name of a terminal for which to generate a new pin.

Term Name is the schematic terminal name. You cannot change this value.

Net Name is the net associated with the pin in the layout. You cannot change this value.

Note: If terminal and net names differ in the schematic, Layout XL generates a pin with the same name as the schematic terminal and a net with the same name as the net attached to the terminal in the schematic. If there is no explicit net label in the schematic, both the pin and the net name in the layout are the same as the

schematic terminal. This is the default behavior.

Layer specifies the layer on which the selected pins are generated. The cyclic field offers only the conducting layers. The default is the current drawing layer if it is defined as an extractable layer in the technology file. Otherwise, it is the first extractable layer found. If there are no extractable layers, the cyclic field lists all valid layout layers.

Width specifies the width for each selected pin. The default is the `minWidth` value set for the current layer in the technology file. Any change to the value is applied only if the new value is greater than the default value.

Height specifies the height for each selected pin. The default is the `minWidth` value set for the current layer in the technology file. Any change to the value is applied only if the new value is greater than the default value.

Num specifies how many instances of each selected pin are generated. If you type 0, the selected pin is not generated.

Create specifies whether or not the selected pins are to be generated in the layout.

Pin Label specifies the type of label generated when you create a pin. This setting is honored by the *Generate All From Source* and *Generate Selected From Source* commands.

Create Label As specifies whether a pin label is created when the pin is generated.

You can choose to create either a **Label** object or a **Text Display**.

Click the *Options* button to access the Set Pin Label Text Style form, where you can set the size, font, style, justification and orientation of the label lettering, and the drawing or pin layer on which the labels are displayed.

Note: If your pin labels are not visible in the canvas, turn on the *Pin Names* option in the Display Options form.

Important

For this setting to take effect, you must set the Layout L environment variable `createPinLabel` to `t`.

Related Topics

Update Tab

PR Boundary Tab

Layout XL Forms

PR Boundary Tab

The **Shape** group box specifies whether the place and route boundary is a rectangle or a polygon.

Rectangle specifies a rectangular place and route boundary. Use the **Area Estimation** group box to specify how the size of the boundary is calculated.

Origin specifies the coordinates of the boundary's origin. The default is (0.0 0.0).

Polygon specifies a polygonal boundary. Use the **Points List** to specify the coordinates of each of the vertices of the polygon.

The **Area Estimation** group box specifies how the system calculates the shape and size of a rectangular boundary.

Area estimation comprises two parts: the first specifies the aspect ratio and utilization of the boundary; the second estimates the size of boundary required to accommodate the components to be generated.

To specify the aspect ratio and utilization, set two of the following four parameters. Choose one of the parameters from the first cyclic list and any one of the remaining valid choices from the second cyclic list.

Utilization (%) specifies the percentage of area within the cell boundary that you want to fill. The default is 25.

Note: When the *Chaining*, *Folding*, or *Chain Folds* options are switched on in the *Generate* tab, the *Utilization* value is applied only after chaining and folding is complete so that the size of the boundary is calculated accurately.

Aspect Ratio (W/H) is the width-to-height ratio of the design boundary. A value of 1 specifies a square boundary; 0.5 specifies a boundary twice as high as it is wide; and 2 specifies a boundary twice as wide as it is high. The default is 1.

Width specifies the width of the design boundary. The default is the size of the last boundary or 10.

Height specifies the height of the design boundary. The default is the size of the last boundary or 10.

To estimate the area, choose a **Calculation Method**.

Internal lets you select one of the built-in estimators from the **Choose Estimator** cyclic field.

- ❑ *PR Boundary Based* calculates the area based on the sum of all the place and route boundaries of the components to be generated.
- ❑ *BBox Based* calculates the area based on the sum of all the bounding boxes of the components to be generated.

User Defined lets you choose your own estimator registered using the framework described in [Creating and Registering a User Defined Area Estimation Function](#).

Change Parameters lets you modify the parameters of a user-defined area estimation function without changing the function itself.

Related Topics

[Update Tab](#)

[I/O Pins Tab](#)

[Layout XL Forms](#)

Update Connectivity Reference

Use the **Update Connectivity Reference** command to update the source for your layout view and to change the physical configuration which specifies how the layout is generated from the source.

Source View Definition lets you specify the view to use as the connectivity source.

Use schematic view tells Layout XL to use a schematic view as the connectivity reference. If you uncheck the box, it means there is no connectivity reference.

To specify the schematic to use, do one of the following.

- ☐ Type the *Library*, *Cell*, and *View* names of the schematic in the fields provided. You can type in the name of a different library. The default view name is `schematic`.
- ☐ Use *Sel by Cursor* and click on an open schematic window
- ☐ Use the *Browse* button to browse your file system for the schematic you want.

Update layout cellviews hierarchically descends into a hierarchical layout cellview and updates the library setting for any lower-level cellviews in accordance with the library name specified for the top-level cell, except

- ☐ Where the lower-level cellview is a leaf (i.e., there is no instance in the cellview)
- ☐ The current setting points to a non-existent cellview.

Lower-level cellviews updated in this way are not saved until you confirm that you want to do so at the end of the session.

Physical Configuration View Definition lets you specify the physical configuration view to use for the schematic-layout pair. If you deselect this check box, Layout XL runs without a physical configuration.

To specify a physical configuration to use, do one of the following.

- ☐ Type the *Library*, *Cell*, and *View* names of the view in the fields provided. You can type in the name of a different library. The default view name is `physConfig`.
- ☐ Use the *Browse* button to browse your file system for the view you want.

Related Topics

[Updating the Schematic View](#)

[Updating the Physical Configuration View](#)

Virtuoso Layout Suite XL User Guide

Layout XL Forms

Layout XL Forms

XL Probe

Object Filter specifies the types of objects that can be probed when you click in the layout or schematic window. For example, if you check only *Pins*, you can create probes only for pin objects by clicking in either window. Clicking on a net or instance has no effect.

The object filters are applied in the order in which they appear in the form. For example, if you check all three then click in the layout window, Layout XL first looks for a pin shape and creates a pin probe if it finds one. If not, it looks for a routing shape and creates a net probe if it finds one. If not, it looks for an instance and creates an instance probe if it finds one.

When you create a probe on a pin, Layout XL probes only the `pinFigs` associated with the pin. When you create a probe on a net, Layout XL probes all the shapes on the specified net, including routing shapes and `pinFigs`.

Important

The *Object Filter* option has no effect on probe creation using the object list in the XL Probe form, only on probe creation by clicking directly in the layout or schematic window.

Send Messages to CIW sends information about the objects probed to the CIW, as well as displaying it in the box at the bottom of the form.

Show lists the names of the nets, pins, net classes, or instances in the layout depending on which category you select in the cyclic field. When you click on one or more items in the list box, the items are highlighted in the schematic and the layout, and the name, window, and equivalency information about each item is shown in the display box. When you remove a probe, that information disappears from the display box.

Important

CMX legacy net classes are listed in the XL Probe form but are not shown under the *Net Class* category in the Constraint Manager assistant. This does not impact the individual constraints migrated.

Probing Options brings up the [Highlight Options](#) form where you can specify how nets, instances, and terminals are probed and which layer and colors are used to draw the probes.

Zoom to the bounding box of the currently selected objects.

Related Topics

[Probing](#)

Virtuoso Layout Suite XL User Guide

Layout XL Forms

[Probing Objects by Clicking in a Window](#)

[Probing Components in OTM, MTM, and MTO Relationships](#)

[Probing with Multiple Layout Cellviews](#)

[Probing a Hierarchical Design](#)

[Removing Probes](#)

[Layout XL Forms](#)

Virtuoso Layout Suite XL User Guide

Layout XL Forms

Library and Attributes Mapping File Syntax

This section describes the Virtuoso® Layout Suite XL library and attributes mapping (LAM) file, which was introduced in the IC6.1 release in order to support the new physical hierarchy configuration functionality. For more information on that functionality, see [Configuring the Physical Hierarchy](#).

The LAM file stores the symbol properties that drive layout generation, component type information for a given library, and library and cell-level mapping information. The file is called `cph.lam` and is created automatically when you convert your library data to use the 6.1 Layout XL schema using either

- *Convert Libraries to Use physConfigs* from the Conversion Tool Box. For more information, see [Converting Data to Use the IC 6.1 Layout XL Schema](#) on page 104.
- The `cphUprevLibrary` SKILL function. For more information, see [cphUprevLibrary](#) in the *Virtuoso Layout Suite SKILL Reference*.

You can override the settings in the library LAM file by creating your own version called `library.lam` and placing it in one of the locations defined in your `setup.loc` file. For more information, see [LAM File Locations](#) on page 1042.

Important

You cannot export a LAM file from the Configure Physical Hierarchy window, nor is there any graphical user interface provided to help you create or edit a LAM file. Cadence recommends that your CAD group generate the required LAM files for a given GPDK from your golden library sources, or that you use a text or XML editor to create or edit your own LAM files.

Related Topics

[LAM File Locations](#) on page 1042

Basic LAM File Structure on page 1044

LAM File Logical Section on page 1046

LAM File Physical Section on page 1051

LAM File Component Type Section on page 1052

LAM File Mappings Section on page 1057

Sample LAM File on page 1064

LAM File Locations

The LAM file generated as part of the library conversion process is stored in the following location.

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

During LAM initialization, this file is imported for each library defined in the library definitions file, `cds.lib`.

The system then searches the locations defined in the `setup.loc` file for any other `library.lam` file and loads the first one it finds. The settings in the `library.lam` file override the settings in the `library/.cadence/dfII/cph/lam/cph.lam` file.

Example

The default `setup.loc` file is

```
.                current working directory always searched first
@LIBRARY         look in the design libraries for the file
$CDS_WORKAREA    user workarea if defined
$CDS_SEARCHDIR   this is set by various tools during tool startup
$HOME
$CDS_PROJECT     project storage area, ignored if not defined
$CDS_SITE        Site Setup Info - default is $CDS_INST_DIR/share/local
$(compute:THIS_TOOL_INST_ROOT)/share Cadence Default Setup Info
```

For example, if a library called `acpd` is part of the library list, then during LAM initialization the system imports

■ `acpd/.cadence/dfII/cph/lam/cph.lam`

And then checks for the following files

Virtuoso Layout Suite XL User Guide

Library and Attributes Mapping File Syntax

- `./ .cadence/dfII/cph/acpd/lam/acpd.lam`
- `<alllibs- inc acpd>./ .cadence/dfII/cph/acpd/lam/acpd.lam`
- `$CDS_WORKAREA/.cadence/dfII/cph/acpd/lam/acpd.lam`
- ...

The system imports the first `acpd.lam` it finds and stops searching.

Inserting Comments in a LAM File

Use the following syntax to insert a comment in a LAM file.

```
<!-- This is a comment -->
```

Related Topics

[Library and Attributes Mapping File Syntax](#)

Basic LAM File Structure

```
<!DOCTYPE LAM_FILE>
<LIBRARY_DATA>
  <VERSION NUMBER="versionNumber" />
  <LOGICAL>
    <!-- logical section goes here -->
  </LOGICAL>
  <PHYSICAL>
    <COMPONENT_TYPE_GROUPS>
      <!-- component type definitions -->
    </COMPONENT_TYPE_GROUPS>
    <MAPPINGS>
      <!-- logical to physical mappings -->
    </MAPPINGS>
  </PHYSICAL>
</LIBRARY_DATA>
```

Description

Identifies the file as a library and attributes mapping (LAM) file, indicates the version of the data it contains, and differentiates between logical and physical information contained in the file. The physical section is further divided into a section that defines component type groups and a section that specifies logical-to-physical mappings for the design.

Statements

`<!DOCTYPE LAM_FILE>`

Identifies the file as a library and attributes mapping (LAM) file.

`<LIBRARY_DATA>`

Encloses the data in the current LAM file.

`<VERSION NUMBER="versionNumber" />`

Indicates the version of the attributes and mappings in the current file. For example,

`<VERSION NUMBER="0.2" />`

`<LOGICAL>`

Encloses the section of the LAM file that contains logical information. For more information, see [LAM File Logical Section](#) on page 1046.

`<PHYSICAL>`

Encloses the section of the LAM file that contains physical information. For more information, see [LAM File Physical Section](#) on page 1051.

<COMPONENT_TYPE_GROUPS>

Encloses the library-level component type definitions in the LAM file. For more information, see [LAM File Component Type Section](#) on page 1052.

<MAPPINGS>

Encloses all the statements used to map physical cellviews, parameters, and terminals to their counterparts in the logical design. For more information, see [LAM File Mappings Section](#) on page 1057.

Related Topics

[Library and Attributes Mapping File Syntax](#)

[LAM File Logical Section](#)

[LAM File Physical Section](#)

[LAM File Component Type Section](#)

[LAM File Mappings Section](#)

[Sample LAM File](#)

LAM File Logical Section

```
<LOGICAL>
  <LIB NAME="libName" >
    <CELL NAME="cellName" >
      <VIEW NAME="viewName" >
        <MFACTOR_SPLIT>t | nil</MFACTOR_SPLIT>
        <REMOVE_DEVICE>expression</REMOVE_DEVICE>
        <STOP_LIST>list_of_viewNames</STOP_LIST>
        <VIEW_LIST>list_of_viewNames</VIEW_LIST>
        <PARAMETERS>
          <IGNORE>
            <GEN>list_of_paramNames</GEN>
            <CHECK>list_of_paramNames</CHECK>
          </IGNORE>
        </PARAMETERS>
        <TERMINALS>
          <IGNORE>
            <GEN>list_of_termNames</GEN>
            <CHECK>list_of_termNames</CHECK>
          </IGNORE>
        </TERMINALS>
      </VIEW>
    </CELL>
  </LIB>
</LOGICAL>
```

Description

Specifies for each cellview in the logical design the view list that was used to find the cellview in the hierarchy, the stop list that will be used to find the corresponding physical view, and options that specify whether the device in question is to be generated or ignored in the physical view.

Statements

<LIB NAME="*libName*" >

Specifies the logical library name and encloses all the statements relating to that library. The name must be enclosed in quotation marks.

<CELL NAME="*cellName*" >

Specifies the logical cell name and encloses all the settings relating to that cell. The name must be enclosed in quotation marks.

<VIEW NAME="*viewName*" >

Specifies the logical view name and encloses all the settings relating to that cellview. The name must be enclosed in quotation marks.

`<MFACTOR_SPLIT>t | nil</MFACTOR_SPLIT>`

Controls whether Layout XL generates logical devices with the mfactor property as multiple devices in the physical view.

GUI: *Split mfactored devices*

`<REMOVE_DEVICE>expression</REMOVE_DEVICE>`

Causes parasitic devices to be ignored by merging nets connected to the terminals of a single instance. For example, to short terminals PLUS and MINUS on a resistor, type the following.

`<REMOVE_DEVICE>(short (PLUS MINUS))</REMOVE_DEVICE>`

Do not enclose the expression in quotation marks.

GUI: *Remove device*

`<STOP_LIST>list_of_viewNames</STOP_LIST>`

Lists the view names to be used to determine the corresponding physical view for a given logical view. When traversing a hierarchy, Configure Physical Hierarchy stops when it encounters a view with one of the specified names.

Separate each name with a single whitespace. Do not enclose the list in quotation marks.

GUI: *Physical stop view list*

`<VIEW_LIST>list_of_viewNames</VIEW_LIST>`

Specifies the view names used to descend into a hierarchical design to find logical views.

Separate each name with a single whitespace. Do not enclose the list in quotation marks.

GUI: *Logical switch view list*

`<PARAMETERS>`

Encloses the `<IGNORE>>` statements that specify which logical instance parameters are to be ignored for generation and check.

GUI: Options on the *Parameters* tab

<TERMINALS>

Encloses the <IGNORE> statements that specify which logical instance terminals are to be ignored for generation and check.

GUI: Options on the *Terminals* tab

<IGNORE>

Encloses the <GEN> and <CHECK> statements that specify which logical instance parameters (or terminals) are to be ignored for generation or check.

<GEN>*list_of_names*</GEN>

Lists the logical instance parameters (or terminals) that are ignored by the following commands; *Generate All From Source, Generate Selected From Source, Generate Clones, Check Against Source, Update Components And Nets, Update Layout Parameters, Update Schematic Parameters*.

Separate each name with a single whitespace. Do not enclose the list in quotation marks.

GUI: *Ignore for generation*

<CHECK>*list_of_names*</CHECK>

Lists the logical instance parameters (or terminals) that are ignored by the *Check Against Source* command. Mismatches for any of the listed parameters are not reported by the check. The list inherits all the parameter names from the *Ignore for generation* list.

Separate each name with a single whitespace. Do not enclose the list in quotation marks.

GUI: *Ignore for check*

Example

```
<LOGICAL>
  <LIB NAME="basic" >
    <CELL NAME="cellA" >
      <VIEW NAME="symbol" >
        <REMOVE_DEVICE>(short(src dst))</REMOVE_DEVICE>
      </VIEW>
    </CELL>
  </LIB>
  <LIB NAME="lib1" >
    <CELL NAME="cellB" >
```

Virtuoso Layout Suite XL User Guide

Library and Attributes Mapping File Syntax

```
<VIEW NAME="symbol" >
  <MFACTOR_SPLIT>t</MFACTOR_SPLIT>
  <REMOVE_DEVICE>(short(PLUS MINUS) funcR(r))</REMOVE_DEVICE>
  <STOP_LIST>abstract layout layoutS</STOP_LIST>
  <VIEW_LIST>schematic symbol layout compacted</VIEW_LIST>
  <PARAMETERS>
    <IGNORE>
      <GEN>w l</GEN>
      <CHECK>a b</CHECK>
    </IGNORE>
  </PARAMETERS>
</VIEW>
</CELL>
</LIB>
</LOGICAL>
```

In the example above,

For cellview	LAM specifies that
basic/cellA/symbol	The cellview is to be ignored by merging the nets connected to its terminals <code>src</code> and <code>dst</code> .
lib1/cellB/symbol	Multiple physical devices are to be generated for mfactored instances of this cellview.
	The cellview is to be ignored by merging the nets connected to terminals <code>PLUS</code> and <code>MINUS</code> but only if the function <code>funcR(r)</code> returns a value that is non-nil.
	The view to be used to generate the physical equivalent of this device is either <code>abstract</code> , <code>layout</code> , or <code>layoutS</code> .
	The <code>w</code> and <code>l</code> parameters are to be ignored during generation and that the <code>a</code> and <code>b</code> parameters are to be ignored during check.

Related Topics

[Library and Attributes Mapping File Syntax](#)

[Basic LAM File Structure](#)

[LAM File Physical Section](#)

[LAM File Component Type Section](#)

[LAM File Mappings Section](#)

Virtuoso Layout Suite XL User Guide

Library and Attributes Mapping File Syntax

Sample LAM File

LAM File Physical Section

```
<PHYSICAL>
  <COMPONENT_TYPE_GROUPS>
    <!-- component type definitions -->
  </COMPONENT_TYPE_GROUPS>
  <MAPPINGS>
    <!-- logical to physical mappings -->
  </MAPPINGS>
</PHYSICAL>
```

Description

The physical section of the LAM file is divided into two subsections. One of these defines the library-level component types, the other specifies the logical-to-physical mappings for the design.

Statements

`<COMPONENT_TYPE_GROUPS>`

Encloses the library-level component type definitions.

`<MAPPINGS>`

Encloses all the statements used to map physical cellviews, parameters, and terminals to their counterparts in the logical design.

Related Topics

[Library and Attributes Mapping File Syntax](#)

[LAM File Component Type Section](#)

[LAM File Mappings Section](#)

[Basic LAM File Structure](#)

[LAM File Logical Section](#)

[Sample LAM File](#)

LAM File Component Type Section

```
<COMPONENT_TYPE_GROUPS>
  <LIB NAME="libName" >
    <COMPONENT_TYPE_GROUP NAME="compTypeName" >
      <COMP_CLASS>PMOS | NMOS | STDCELL | STDSUBCONT | FILLER</COMP_CLASS>
      <WIDTH>widthParamName</WIDTH>
      <DRAIN>list_of_drainTermNames</DRAIN>
      <GATE>list_of_gateTermNames</GATE>
      <SOURCE>list_of_sourceTermNames</SOURCE>
      <BULK>list_of_bulkTermNames</BULK>
      <ACTIVE_LP>layerPurposePairName</ACTIVE_LP>
      <FOLD_THRESH>float</FOLD_THRESH>
      <VIEWS>list_of_viewNames</VIEWS>
    </COMPONENT_TYPE_GROUP>
  </LIB>
</COMPONENT_TYPE_GROUPS>
```

Description

Specifies the library-level component types, which identify NMOS and PMOS transistor cells and set the parameters for device chaining and folding. It also controls how components are assigned to rows when using the Virtuoso custom digital placer. LAM file component type settings are inherited by all the cells in the specified library. They can be overridden but not overwritten in the Configure Physical Hierarchy window.

The component type settings defined in the LAM file are created automatically during the logical library conversion. This takes the component types stored in CDF files or (if none exist in CDF) in the `data.dm` file in the library directory and converts them to LAM file format.

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

You can also add new component type groups using the syntax described in this section. for more information, see [How Data is Converted to the Layout XL IC 6.1 Schema](#) on page 106.



Tip

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



Once you have defined the component type group, you must specify the cells that are assigned to this group using the `<COMPONENT_TYPE>` statement. See [LAM File Mappings Section](#) for more information.

Statements

`<COMPONENT_TYPE_GROUPS>`

Encloses the library-level component type definitions in the LAM file.

`<LIB NAME="libName" >`

Specifies the name of the library to which the component type definitions apply.

`<COMP_CLASS>PMOS | NMOS | STDCELL | STDSUBCONT | FILLER</COMP_CLASS>`

Specifies the type of devices in the component type. Valid values are `PMOS`, `NMOS`, `STDCELL`, `STDSUBCONT`, `FILLER`. Leave blank to set to undefined.

- Choose `PMOS` or `NMOS` if you are using the placer's MOS component-assisted row generation. The `<VIEWS>` statement is ignored.
- Choose `STDCELL` if you are using the placer's Standard Cell component-assisted row generation. All other settings are ignored.
- Choose `STDSUBCONT` or `FILLER` for substrate contacts and filler cells respectively. Only the `<VIEWS>` statement is considered; everything else is ignored.

GUI: *Component class*

`<WIDTH>widthParamName</WIDTH>`

Specifies the name of the transistor width parameter on the device master cell. The default name is `w`. If you use a different name for the transistor width parameter, type that name here.

Note: You must set this parameter for `NMOS` and `PMOS` component types even if the devices in the component type will not be folded.

GUI: *Width parameter*

`<DRAIN>list_of_drainTermNames</DRAIN>`

Specifies the default names used to identify the *Drain* terminal in the cells assigned to the component type. The names are not case-sensitive. The default is `drain` and `D`.

Separate each name with a single whitespace. Do not enclose the list in quotation marks.

GUI: *Drain*

`<GATE>list_of_gateTermNames</GATE>`

Specifies the default names used to identify the *Gate* terminal in the cells assigned to the component type. The names are not case-sensitive. The default is `gate` and `G`.

Separate each name with a single whitespace. Do not enclose the list in quotation marks.

GUI: *Gate*

`<SOURCE>list_of_sourceTermNames</SOURCE>`

Specifies the default names used to identify the *Source* terminal in the cells assigned to the component type. The names are not case-sensitive. The default is `source` and `S`.

Separate each name with a single whitespace. Do not enclose the list in quotation marks.

GUI: *Source*

`<BULK>list_of_bulkTermNames</BULK>`

Specifies the default names used to identify the *Bulk* terminal in the cells assigned to the component type. The names are not case-sensitive. The default is `bulk` and `B`.

Separate each name with a single whitespace. Do not enclose the list in quotation marks.

GUI: *Bulk*

`<ACTIVE_LP>layerPurposePairName</ACTIVE_LP>`

Specifies the diffusion layer-purpose pair for NMOS and PMOS devices. This parameter is required for component types of component class `NMOS` and `PMOS` in order to control device chaining and folding during placement.

For example, to specify the `ndiff` drawing layer as the diffusion layer, type the following.

`<ACTIVE_LP>ndiff drawing</ACTIVE_LP>`

Separate the layer and purpose names with a single whitespace. Do not enclose the names in quotation marks.

GUI: *Active layer*

`<FOLD_THRESH>float</FOLD_THRESH>`

Specifies the maximum width beyond which MOS devices are automatically folded.

Note: You must set this parameter for `NMOS` and `PMOS` component types even if the devices in the component type will not be folded.

GUI: *Folding threshold*

`<VIEWS>list_of_viewNames</VIEWS>`

Specifies the view name to be used when defining a standard cell substrate contact or filler cell. This parameter is valid only when the component class is either `STDSUBCONT` or `FILLER`.

Separate each name with a single whitespace. Do not enclose the list in quotation marks.

GUI: *Views*

Example

```
<COMPONENT_TYPE_GROUPS>
  <LIB NAME="lib1" >
```

Virtuoso Layout Suite XL User Guide

Library and Attributes Mapping File Syntax

```
<COMPONENT_TYPE_GROUP NAME="nmos" >
  <COMP_CLASS>NMOS</COMP_CLASS>
  <WIDTH>w</WIDTH>
  <DRAIN>d</DRAIN>
  <GATE>g</GATE>
  <SOURCE>s</SOURCE>
  <BULK>b</BULK>
  <ACTIVE_LP>pdiff drawing</ACTIVE_LP>
  <FOLD_THRESH>0.92</FOLD_THRESH>
  <VIEWS>layout filler</VIEWS>
</COMPONENT_TYPE_GROUP>
<COMPONENT_TYPE_GROUP NAME="pmos" >
  <COMP_CLASS>PMOS</COMP_CLASS>
  <WIDTH>w</WIDTH>
  <DRAIN>d</DRAIN>
  <GATE>g</GATE>
  <SOURCE>s</SOURCE>
  <BULK>b</BULK>
  <ACTIVE_LP>ndiff drawing</ACTIVE_LP>
  <FOLD_THRESH>0.9</FOLD_THRESH>
  <VIEWS>layout</VIEWS>
</COMPONENT_TYPE_GROUP>
</LIB>
<LIB NAME="lib2" >
  <COMPONENT_TYPE_GROUP NAME="lib2" />
</LIB>
</COMPONENT_TYPE_GROUPS>
```

The example above defines three library-level component types.

- One called `nmos` and one called `pmos` for a library called `lib1`.
- One called `lib2` for a library called `lib2`.

Related Topics

[Library and Attributes Mapping File Syntax](#)

[LAM File Physical Section](#)

[LAM File Mappings Section](#)

[Basic LAM File Structure](#)

[LAM File Logical Section](#)

[Sample LAM File](#)

LAM File Mappings Section

```
<MAPPINGS>
  <LIB NAME="libName" >
    <CELL NAME="cellName" >
      <COMPONENT TYPE>
        <NAME>compTypName</NAME>
        <LIB>libName</LIB>
      </COMPONENT TYPE>
      <VIEW NAME="viewName" >
        <LOGICAL_CELL>
          <LIB NAME="libName" >
            <CELL NAME="cellName" >
              <VIEW NAME="viewName" >
                <ROUNDING>roundingExpr</ROUNDING>
                <PARAMETERS>
                  <MAP>p1 p2 ; p3 p4 ; </MAP>
                  <IGNORE>
                    <GEN>list_of_paramNames</GEN>
                    <CHECK>list_of_paramNames</CHECK>
                  </IGNORE>
                </PARAMETERS>
                <TERMINALS>
                  <MAP>t1 t2 ; t3 t4 ; </MAP>
                  <IGNORE>
                    <GEN>list_of_termNames</GEN>
                    <CHECK>list_of_termNames</CHECK>
                  </IGNORE>
                </TERMINALS>
              </VIEW>
            </CELL>
          </LIB>
        </LOGICAL_CELL>
      </VIEW>
    </CELL>
  </LIB>
</MAPPINGS>
```

Description

Maps physical cellviews and parameter and terminal names to their counterparts in the logical design, specifies which parameters and terminal names are to be ignored for generation and for check, and overrides the component type setting for a specific physical cellview.

Statements

```
<MAPPINGS>
```

Virtuoso Layout Suite XL User Guide

Library and Attributes Mapping File Syntax

Encloses all the statements used to map physical cellviews, parameters, and terminals to their counterparts in the logical design.

```
<LIB NAME="libName" >
```

Specifies the physical library name and encloses all the statements relating to that library. The name must be enclosed in quotation marks.

```
<CELL NAME="cellName" >
```

Specifies the physical cell name and encloses all the statements relating to that cell. The name must be enclosed in quotation marks.

```
<COMPONENT_TYPE>
```

Encloses the component type group to which the current cell is assigned.

For example, the syntax below assigns the current cell to component type `myPmos` defined for `lib1`.

```
<COMPONENT_TYPE>
  <NAME>myPmos</NAME>
  <LIB>lib1</LIB>
</COMPONENT_TYPE>
```

```
<VIEW NAME="viewName" >
```

Specifies the physical view name and encloses all the statements relating to that cellview. The name must be enclosed in quotation marks.

```
<LOGICAL_CELL>
```

Specifies the logical cellview that corresponds to the current physical cellview. For example,

```
<LOGICAL_CELL>
  <LIB NAME="lib1">
    <CELL NAME="cell1">
      <VIEW NAME="schematic">
        .
        .
        .
      </VIEW>
    </CELL>
  </LIB>
</LOGICAL_CELL>
```

```
<ROUNDING>expression</ROUNDING>
```

Defines how parameter values are rounded when evaluated by Layout XL; for example,

```
<SROUNDING>(w 0.05 round)</ROUNDING>
```

Rounds the value of *w* to the nearest multiple of 0.05 nanometers.

<PARAMETERS>

Encloses the <IGNORE> statements that specify which logical instance parameters are to be ignored for generation and check.

GUI: Options on the *Parameters* tab

<TERMINALS>

Encloses the <IGNORE> statements that specify which logical instance terminals are to be ignored for generation and check.

GUI: Options on the *Terminals* tab

```
<MAP>p1 p2 ; p3 p4 ; </MAP>
```

Maps a logical parameter (or terminal) name to the corresponding parameter (or terminal) in the physical view. For example, to map logical parameters *D* and *G* to physical parameters *d* and *g*, type the following.

```
<MAP>D d ; G g ; </MAP>
```

Do not enclose the mapping in quotation marks.

GUI: *Name mapping* (cell attributes only)

<IGNORE>

Encloses the <GEN> and <CHECK> statements that specify which logical instance parameters (or terminals) are to be ignored for generation or check.

```
<GEN>list_of_names</GEN>
```

Lists the names of logical instance parameters (or terminals) that are ignored by the following commands; *Generate All From Source, Generate Selected From Source, Generate Clones, Check Against Source, Update Components And Nets, Update Layout Parameters, Update Schematic Parameters*.

Separate each name with a single whitespace. Do not enclose the list in quotation marks.

GUI: *Ignore for generation*

```
<CHECK>list_of_names</CHECK>
```

Lists the names of logical instance parameters (or terminals) that are ignored by the *Check Against Source* command. Mismatches for any of the listed parameters are not reported by the check.

Separate each name with a single whitespace. Do not enclose the list in quotation marks. The list inherits all the parameter names from the *Ignore for generation* list.

GUI: *Ignore for check*

Example

```
<MAPPINGS>
  <LIB NAME="lib1" >
    <CELL NAME="cellB" >
      <VIEW NAME="layout" >
        <LOGICAL_CELL>
          <LIB NAME="lib2" >
            <CELL NAME="cellF" >
              <VIEW NAME="symbol" />
            </CELL>
          </LIB>
        </LOGICAL_CELL>
      </VIEW>
    </CELL>
  </LIB>
  <LIB NAME="lib2" >
    <CELL NAME="cellE" >
      <COMPONENT_TYPE>
        <NAME>myNmos</NAME>
        <LIB>lib2</LIB>
      </COMPONENT_TYPE>
      <VIEW NAME="layout" >
        <LOGICAL_CELL>
          <LIB NAME="lib1" >
            <CELL NAME="cellA" >
              <VIEW NAME="schematicB" >
                <ROUNDING>(w 0.01 round)</ROUNDING>
                <PARAMETERS>
```


Virtuoso Layout Suite XL User Guide

Library and Attributes Mapping File Syntax

```
<MAP>l L ; w W ; x X2 ; </MAP>
<IGNORE>
  <GEN>w l</GEN>
  <CHECK>a b c</CHECK>
</IGNORE>
</PARAMETERS>
<TERMINALS>
  <MAP>B G ; X D ; Y S ; </MAP>
</TERMINALS>
</VIEW>
<VIEW NAME="symbol" >
  <ROUNDING>(w 0.05 round)</ROUNDING>
  <PARAMETERS>
    <MAP>l L ; w W ; </MAP>
    <IGNORE>
      <GEN>x y z</GEN>
      <CHECK>d e f</CHECK>
    </IGNORE>
  </PARAMETERS>
  <TERMINALS>
    <MAP>D d ; G g ; S s ; </MAP>
    <IGNORE>
      <GEN>X</GEN>
    </IGNORE>
  </TERMINALS>
</VIEW>
</CELL>
</LIB>
</LOGICAL_CELL>
</VIEW>
</CELL>
<CELL NAME="cellG" >
  <VIEW NAME="layout" >
    <LOGICAL_CELL>
      <LIB NAME="lib1" >
        <CELL NAME="cellD" >
          <VIEW NAME="symbol" />
        </CELL>
      </LIB>
    </LOGICAL_CELL>
  </VIEW>
  <VIEW NAME="layoutS" >
    <LOGICAL_CELL>
      <LIB NAME="lib1" >
        <CELL NAME="cellD">
          <VIEW NAME="symbols" />
        </CELL>
      </LIB>
    </LOGICAL_CELL>
  </VIEW>
</CELL>
</LIB>
</MAPPINGS>
```

Virtuoso Layout Suite XL User Guide

Library and Attributes Mapping File Syntax

The example above

For cellview	LAM specifies that
lib1/cellB/layout	The cellview maps to logical cellview lib2/cellF/symbol.
lib2/cellE	The cell is assigned to component type myNmos defined in library lib2.
lib2/cellE/layout	The cellview maps to logical cellviews lib1/cellA/schematicB and lib1/cellA/symbol. When lib1/cellA/schematicB is used: ■ The value of the <i>w</i> parameter is rounded to the nearest multiple of 0.01 nanometers. ■ Logical instance parameters <i>l</i>, <i>w</i>, and <i>x</i> are mapped to physical parameters <i>L</i>, <i>W</i>, and <i>X2</i> respectively. ■ Logical instance parameters <i>w</i> and <i>l</i> are not copied during layout generation; parameters <i>a</i>, <i>b</i>, and <i>c</i> are not checked by the <i>Check Against Source</i> command ■ Logical terminals <i>B</i>, <i>X</i>, and <i>D</i> are mapped to physical terminals <i>G</i>, <i>D</i>, and <i>S</i> respectively. When lib1/cellA/symbol is used: ■ The value of the <i>w</i> parameter is rounded to the nearest multiple of 0.05 nanometers. ■ Logical instance parameters <i>l</i> and <i>w</i> are mapped to physical parameters <i>L</i> and <i>W</i> respectively. ■ Logical instance parameters <i>x</i>, <i>y</i>, and <i>z</i> are not copied during layout generation; parameters <i>d</i>, <i>e</i>, and <i>f</i> are not checked by the <i>Check Against Source</i> command ■ Logical terminals <i>D</i>, <i>G</i>, and <i>S</i> are mapped to physical terminals <i>d</i>, <i>g</i>, and <i>s</i> respectively. ■ Logical terminal <i>x</i> is ignored during layout generation.
lib2/cellG/layout	The cellview maps to logical cellview lib1/cellD/symbol.

Virtuoso Layout Suite XL User Guide

Library and Attributes Mapping File Syntax

For cellview	LAM specifies that
lib2/cellG/layoutS	The cellview maps to logical cellview lib1/cellD/symbols.

Related Topics

[Library and Attributes Mapping File Syntax](#)

[LAM File Physical Section](#)

[LAM File Component Type Section](#)

[Basic LAM File Structure](#)

[LAM File Logical Section](#)

[Sample LAM File](#)

Sample LAM File

```
<!DOCTYPE LAM FILE>
<LIBRARY_DATA>
  <VERSION NUMBER="0.2" />
  <LOGICAL>
    <LIB NAME="basic" >
      <CELL NAME="cellA" >
        <VIEW NAME="symbol" >
          <REMOVE_DEVICE>(short(src dst))</REMOVE_DEVICE>
        </VIEW>
      </CELL>
    </LIB>
    <LIB NAME="lib1" >
      <CELL NAME="cellB" >
        <VIEW NAME="symbol" >
          <MFACTOR_SPLIT>t</MFACTOR_SPLIT>
          <REMOVE_DEVICE>(short(PLUS MINUS) funcR(r))</REMOVE_DEVICE>
          <STOP_LIST>abstract layout layoutS</STOP_LIST>
          <VIEW_LIST>schematic symbol layout compacted</VIEW_LIST>
          <PARAMETERS>
            <IGNORE>
              <GEN>w 1</GEN>
              <CHECK>a b</CHECK>
            </IGNORE>
          </PARAMETERS>
        </VIEW>
      </CELL>
      <CELL NAME="cellC" >
        <VIEW NAME="symbol" >
          <MFACTOR_SPLIT>t</MFACTOR_SPLIT>
          <STOP_LIST>abstractX layoutX layoutSX</STOP_LIST>
          <VIEW_LIST>schematicX symbolX layoutX</VIEW_LIST>
          <TERMINALS>
            <IGNORE>
              <GEN>x y z</GEN>
              <CHECK>a b c</CHECK>
            </IGNORE>
          </TERMINALS>
        </VIEW>
      </CELL>
    </LIB>
  </LOGICAL>
  <PHYSICAL>
    <COMPONENT TYPE GROUPS>
      <LIB NAME="lib1" >
        <COMPONENT TYPE GROUP NAME="nmos" >
          <COMP_CLASS>NMOS</COMP_CLASS>
          <WIDTH>w</WIDTH>
          <DRAIN>d</DRAIN>
          <GATE>g</GATE>
          <SOURCE>s</SOURCE>
          <BULK>b</BULK>
          <ACTIVE_LP>pdiff drawing</ACTIVE_LP>
          <FOLD_THRESH>0.92</FOLD_THRESH>
          <VIEWS>layout filler</VIEWS>
        </COMPONENT TYPE GROUP>
        <COMPONENT TYPE GROUP NAME="pmos" >
          <COMP_CLASS>PMOS</COMP_CLASS>
          <WIDTH>w</WIDTH>
```

Virtuoso Layout Suite XL User Guide

Library and Attributes Mapping File Syntax

```
<DRAIN>d</DRAIN>
<GATE>q</GATE>
<SOURCE>s</SOURCE>
<BULK>b</BULK>
<ACTIVE_LP>ndiff drawing</ACTIVE_LP>
<FOLD_THRESH>0.9</FOLD_THRESH>
<VIEWS>layout</VIEWS>
</COMPONENT_TYPE_GROUP>
</LIB>
<LIB NAME="lib2" >
  <COMPONENT_TYPE_GROUP NAME="lib2" />
</LIB>
</COMPONENT_TYPE_GROUPS>
<MAPPINGS>
  <LIB NAME="lib1" >
    <CELL NAME="cellB" >
      <VIEW NAME="layout" >
        <LOGICAL_CELL>
          <LIB NAME="lib2" >
            <CELL NAME="cellF" >
              <VIEW NAME="symbol" />
            </CELL>
          </LIB>
        </LOGICAL_CELL>
      </VIEW>
    </CELL>
  </LIB>
  <LIB NAME="lib2" >
    <CELL NAME="cellE" >
      <COMPONENT_TYPE>
        <NAME>myNmos</NAME>
        <LIB>lib2</LIB>
      </COMPONENT_TYPE>
      <VIEW NAME="layout" >
        <LOGICAL_CELL>
          <LIB NAME="lib1" >
            <CELL NAME="cellA" >
              <VIEW NAME="schematicB" >
                <ROUNDING>(w 0.01 round)</ROUNDING>
                <PARAMETERS>
                  <MAP>l L ; w W ; x X2 ; </MAP>
                  <IGNORE>
                    <GEN>w l</GEN>
                    <CHECK>a b c</CHECK>
                  </IGNORE>
                </PARAMETERS>
                <TERMINALS>
                  <MAP>B G ; X D ; Y S ; </MAP>
                </TERMINALS>
              </VIEW>
            <VIEW NAME="symbol" >
              <ROUNDING>(w 0.05 round)</ROUNDING>
              <PARAMETERS>
                <MAP>l L ; w W ; </MAP>
                <IGNORE>
                  <GEN>x y z</GEN>
                  <CHECK>d e f</CHECK>
                </IGNORE>
              </PARAMETERS>
              <TERMINALS>
                <MAP>D d ; G g ; S s ; </MAP>
```

Virtuoso Layout Suite XL User Guide

Library and Attributes Mapping File Syntax

```

        <IGNORE>
        <GEN>X</GEN>
    </IGNORE>
</TERMINALS>
</VIEW>
</CELL>
</LIB>
</LOGICAL_CELL>
</VIEW>
</CELL>
<CELL NAME="cellG" >
    <VIEW NAME="layout" >
        <LOGICAL_CELL>
            <LIB NAME="lib1" >
                <CELL NAME="cellD" >
                    <VIEW NAME="symbol" />
                </CELL>
            </LIB>
        </LOGICAL_CELL>
    </VIEW>
    <VIEW NAME="layoutS" >
        <LOGICAL_CELL>
            <LIB NAME="lib1" >
                <CELL NAME="cellD">
                    <VIEW NAME="symbolS" />
                </CELL>
            </LIB>
        </LOGICAL_CELL>
    </VIEW>
</CELL>
</LIB>
</MAPPINGS>
</PHYSICAL>
</LIBRARY_DATA>
```

Related Topics

[Library and Attributes Mapping File Syntax](#)

[Basic LAM File Structure](#)

[LAM File Logical Section](#)

[LAM File Physical Section](#)

[LAM File Component Type Section](#)

[LAM File Mappings Section](#)

[Sample LAM File](#)

Layout XL Assistants

This section describes the dockable assistants available in the Virtuoso® Layout Suite XL layout editor (Layout XL).

Note: Some of the assistants described in this section are inherited from the Virtuoso Design Environment and are not described in full in this section. Instead, links are provided to the relevant documentation.

Displaying a Dockable Assistant

To display an assistant, do one of the following.

- From the layout window menu bar, choose *Windows – Assistants – <Assistant Name>*.
- Click the right mouse button on any toolbar/status bar/menu bar and choose the assistant you want to dock.

The assistant you chose opens and is docked in its default position in the session window.

Hiding a Dockable Assistant

To hide an assistant, do one of the following.

- Click the *Hide* button (X) in the assistant title bar.
- From the layout window menu bar, choose *Windows – Assistants – <Assistant Name>*.
- Click the right mouse button on any toolbar/status bar/menu bar and choose the assistant you want to hide.



Tip

You can also use the F11 key to hide a dockable assistant. Pressing F11 toggles all the active assistants On and Off, giving quick access to the assistants and making it possible to create more canvas space with a single click. However, it is important to note that the shortcut key works only with the “active” assistants.

Related Topics

[Annotation Browser](#) on page 1069

[Constraint Manager](#) on page 1110

[Navigator](#) on page 1119

[Property Editor](#) on page 1136

[Search](#) on page 1137

[World View](#) on page 1138

Annotation Browser

Use the **Annotation Browser** to view and manage the violation markers generated for the current design by different Virtuoso applications.

The Annotation Browser displays the violation markers due to the current environment of the design window. If you modify the environment by editing a cellview in place or descending into a hierarchical cellview, the Annotation Browser gets updated accordingly to reflect a different set of violation markers.

In addition to the generic methods for showing or hiding an assistant, as described in [Layout XL Assistants](#), you can use the Layout XL [Toolbar](#) button, , to show and hide the Annotation Browser.

[Annotation Browser Graphical User Interface](#)

[Annotation Browser Context Menus](#)

[Annotation Browser Forms](#)

[Grouping Markers in the Annotation Browser](#)

[Sorting Markers in the Annotation Browser](#)

[Filtering Markers in the Annotation Browser](#)

[Assigning Colors to Markers](#)

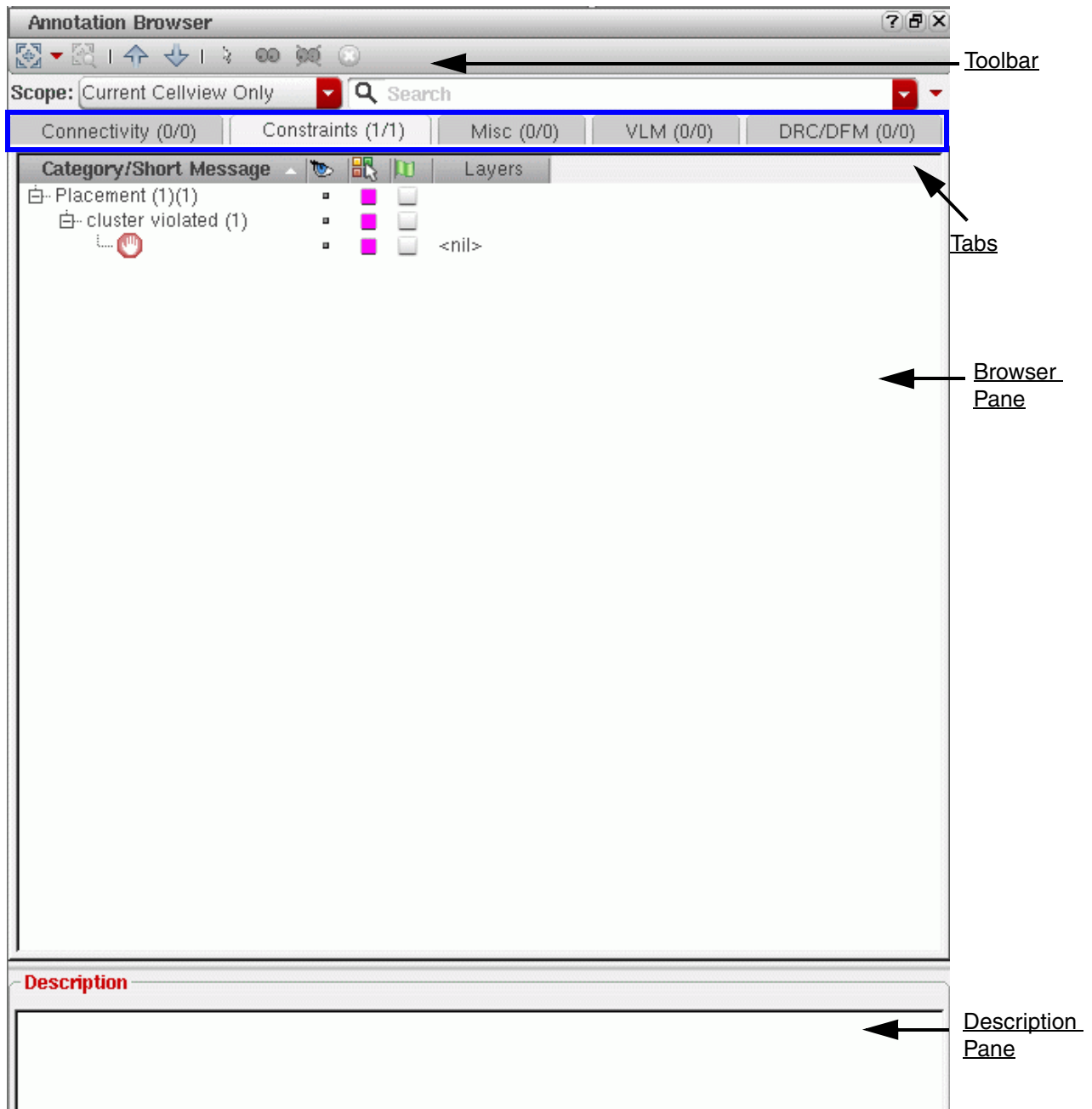
[Zooming In on Markers](#)

Related Topics

[Layout XL Assistants](#)

Annotation Browser Graphical User Interface

The Annotation Browser graphical user interface comprises the following elements.



Toolbar

The toolbar provides functions to filter and manage the violation markers displayed in the browser pane, and to define how the markers are displayed in the design window.

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

Icon	Name	Description
	<i>Auto Zoom</i>	Zooms and pans to the currently selected markers when using the Up and Down arrows in the toolbar, or to the merged bounding box of all the selected nodes when a selection is changed in the browser. For example, if you select a node in the tree; the tool zooms to the merged bounding box of all the markers under that node. So, in the example below <pre>SRC Incomplete Nets m1 m2 Shorts m3 m4</pre> If you select <code>Incomplete Nets</code>, the bounding boxes of <code>m1</code> and <code>m2</code> are merged and the tool zooms in on the resultant box. If you select <code>SRC</code>, the bounding boxes of <code>m1</code>, <code>m2</code>, <code>m3</code>, and <code>m4</code> are merged and the tool zooms on this box. The <i>Annotation Scale</i> slider defines a ratio specifying how much of the available canvas area is occupied by the merged bounding box of the selected markers after the zoom is performed. For example, if you set it to 4, the ratio is 1/4; if you set it to 10, the ratio is 1/10, and so on. In Fixed mode, the ratio is simply applied to the bounding box of all the selected markers. So, if set to 4, the merged bounding box of all the selected markers occupies 1/4 of the canvas area after the <i>Auto Zoom</i> is performed. In Minimal mode, the Annotation Scale defines a minimum zoom level that will ensure all the selected nodes are visible in the canvas. For example, assume the <i>Annotation Scale</i> is set to 4. ■ If the merged bounding box of the selected markers is greater than the canvas size, the display zooms out so that the bounding box fits in the canvas. ■ If the canvas size is greater than the merged bounding box of the selected nodes, which in turn is greater than 1/4 of the canvas size, the display pans to the bounding box of the selected markers.

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

Icon	Name	Description
	<i>Zoom To Selected</i>	Zooms to the selected set of markers in the design window.
	<i>Go To The Previous Selected Marker</i>	The behavior (and label) of this button changes depending on what is selected in the browser.
	<i>Go To The Previous Row</i>	■ For multiple selected markers, it makes the previous marker in the selected set the current marker. This works even if the markers were selected in the canvas and their parent category is not expandable in the browser pane. ■ For an individual selected marker or marker category, it deselects the currently selected entry and selects the entry immediately above it in the list. ■ For multiple selected marker categories, it deselects all the currently selected entries and selects the entry immediately above the original selected set. ■ If there is nothing selected in the browser pane, it selects the last entry in the list.
	<i>Go To The Last Row</i>	
		Note: You can use the Up arrow on your keyboard to perform the last three functions listed above. Only the toolbar button performs the first function.

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

Icon	Name	Description
	<i>Go To The Next Selected Marker</i>	The behavior (and label) of this button changes depending on what is selected in the browser.
	<i>Go To The Next Row</i>	■ For multiple selected markers, it makes the next marker in the selected set the current marker. This works even if the markers were selected in the canvas and their parent category is not expandable in the browser pane. ■ For an individual selected marker or marker category, it deselects the currently selected entry and selects the entry immediately below it in the list. ■ For multiple selected marker categories, it deselects all the currently selected entries and selects the entry immediately below the original selected set. ■ If there is nothing selected in the browser pane, it selects the first entry in the list. Note: You also can use the Down arrow on your keyboard to perform the last three functions listed above. Only the toolbar button performs the first function.
	<i>Go To The First Row</i>	
	<i>Make Highlighted Markers Selectable</i>	Specifies that markers that are highlighted in the canvas can be selected even if marker selection has been switched off in the LSW. Note: This option is off by default, meaning that you cannot select highlighted markers in the canvas.
	<i>Show Checked Markers</i>	Shows markers that are marked as <i>Checked</i> in the Annotation Browser. Note: This command has no effect on the marker display in the canvas, only in the Annotation Browser.

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

Icon	Name	Description
	<i>Hide Checked Markers</i>	Hides markers that are marked as <i>Checked</i> in the Annotation Browser. Note: This command has no effect on the marker display in the canvas, only in the Annotation Browser.
	<i>Delete Selected Markers</i>	Deletes the selected markers from the Annotation Browser and design window.
	<i>Scope</i>	Specifies the scope of the violations displayed. Choose from ■ Top cellview only ■ Current cellview only ■ Hierarchy from the current cellview to a depth set in the Annotation Browser Options form ■ Hierarchy from the top cellview to a depth set in the Annotation Browser Options form All markers that do not belong to the current cellview are dimmed. For example, when you descend into a hierarchical cellview, the markers not at the current level are dimmed. You can still delete or mark these as checked, but you cannot highlight them in the canvas. When you return to the higher level, the browser restores the highlighting that was set for that level prior to the descend.

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

Icon	Name	Description
	<i>Search</i>	Filters the results in the browser pane based on a user-specified string. When you type a string in the field, the browser updates automatically to show only those markers that feature the specified string in the marker description. For example, type in an instance or net name to show only the markers associated with that particular instance or net. Previous filters used in the current session are accessible from the drop-down list to the right of the entry field. There is also a further drop-down menu which you can use to specify the criteria to be used for the search. ■ Find With: <i>All Of The Words, Any Of The Words, The Exact Phrase, or None Of The Words</i> ■ Match Word: <i>Substring, Prefix, Exactly, or Suffix</i> ■ Using Case: <i>Insensitive Match or Sensitive Match</i> Note: The Virtuoso Search auto complete functionality is not supported in the Annotation Browser.

Tabs

The Annotation Browser assistant includes four tabs that carry violation markers of different types.

- The **Connectivity tab** carries the connectivity extractor markers that indicate connectivity violations in the design. The markers supported by this tab are:
 - ☐ *Opens*
 - ☐ *Shorts*
 - ☐ *Weak Connect Violations*
 - ☐ *Must Connect Violations*

- ❑ *Illegal Overlaps*

Related Topics

[Working with Incomplete Nets](#)

[Showing and Hiding Current Incomplete Nets](#)

[Showing and Hiding Selected Incomplete Nets](#)

[Searching for Specific Incomplete Nets](#)

[Automatically Showing Newly-Created Incomplete Nets](#)

[Hiding All Incomplete Nets](#)

[Checking Shorts, Incomplete Nets, and Illegal Overlaps](#)

- The **Constraints** tab carries the markers generated by various Virtuoso tools that verify the design for Placement, Speciality Routing, and Process Rule Overrides. The supported tools conducting these checks are:

- ❑ DiffPairCheck
- ❑ Symmetry Check
- ❑ checkRoutabilityTool
- ❑ VLS Constraint Aware Editing
- ❑ drdEdit

Note: For a marker to be supported by the *Constraints* tab, it must be associated with a tool listed above. Else, the marker will be automatically placed in the *Misc* tab, unless it corresponds to an application supported by the *Connectivity* or *VLM* tabs.

Depending on the tool with which the marker is associated, the markers on the *Constraints* tab are organized into the following categories:

Category Name	Tool Name
Specialty Routing	DiffPairCheck
	SymmetryCheck
	checkRoutabilityTool
Placement	VLS Constraint Aware Editing
Process Rule Overrides	drdEdit

- The **Misc tab** carries all those markers that are not associated with any of the applications supported by the remaining tabs. This tab also carries violation markers for any process rule violations in the design.
- The **VLM tab** carries violation markers generated during the Virtuoso Layout Migration.
- The **DRC/DFM tab** carries violation markers generated during a Match and Fix run and during a Litho Fixing or LDE Analysis.

For more information, see the [Match and Fix](#) form and the [Litho Fixing](#) form and [LDE Analysis](#).

The markers supported by each tab are displayed in the [Browser Pane](#). Each tab name in the Annotation Browser includes a value within parentheses. This value indicates the number of markers currently shown in the browser pane as opposed to the total number of markers in the tab.

For example, *Connectivity (23/120)* means that the design contains 120 markers generated by the connectivity extractor, of which only 23 are being displayed in the browser pane.

Browser Pane

The Browser Pane displays the markers corresponding to each application category supported by a tab. For example, on the *Constraints* tab, the browser pane displays markers for all the supported categories: Specialty Routing, Placement, and Process Rule Overrides.

Every category in the tree is followed by two numbers in parentheses. The first of these is the number of categories one level below the current node; the second is the total number of markers under the current node.

When you select a marker in the browser pane, it is automatically highlighted in the canvas and vice versa. The current marker is colored magenta in the browser and is haloed in the canvas. If you selected a marker in the canvas and that marker is listed in a tab that is currently hidden, that tab is automatically brought to the front and the tree is scrolled to show the selected marker. Where there are multiple selected markers on different tabs, the focus is given to the current (magenta) marker of the selected set.

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

The columns in the table show the following information for each marker.

Icon	Name	Description
	<i>Set Highlight State</i>	Shows and sets the highlight state of the corresponding marker in the design window. Clicking in this column for a particular node sets the highlight state for all the markers under that node. Note: A grayed-out version of the icon displayed against a node indicates that only some of the markers under that node are highlighted.
	<i>Set Highlight Color</i>	Shows and sets the color used to draw the corresponding marker in the design window. Clicking in this column for a node sets the highlight color for all the markers under that node. Choose <i>cycle</i> to let Layout XL select the colors automatically by cycling through a predefined list. ■ If the node has the dot icon in this column, it means that no single color is assigned for the markers in the node (i.e., the markers either have no color assigned or at least one marker has a different color from the rest). ■ If the node has a color in this column, it means that all the markers in the node have the same highlight color assigned. Note: This command does not change the color assigned to the marker in the database, only the color used to draw the marker in the canvas.

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

Icon	Name	Description
	<i>Set Marker Check State</i>	Shows and sets the <i>Checked</i> state of the corresponding marker. When a marker is set to <i>Checked</i>, you can use the <i>Hide Checked Markers</i> button to hide it in the Annotation Browser. Clicking in this column for a particular node sets the <i>Checked</i> state for all the markers under that node. Note: This command has no effect on the marker display in the canvas, only in the Annotation Browser.
	<i>Description</i>	Shows a message describing the marker in more detail. Similar information is presented in the <i>Description</i> pane at the bottom of the assistant. Note: This column is hidden by default.
	<i>Lib Name</i>	Shows the master library of the cell associated with the marker. Note: This column is hidden by default.
	<i>Cell Name</i>	Shows the master cell name of the cell associated with the marker. Note: This column is hidden by default.
	<i>View Name</i>	Shows the master view name of the cell associated with the marker. Note: This column is hidden by default.
	<i>Layers</i>	Shows the name of the layer associated with the marker. Note: This column is hidden by default in the layout window Annotation Browser. It is not available in the schematic window Annotation Browser.

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

Icon	Name	Description
	<i>Purposes</i>	Shows the purpose of the layer associated with the marker. Note: This column is hidden by default in the layout window Annotation Browser. It is not available in the schematic window Annotation Browser.
	<i>Nets</i>	Shows the net associated with the marker. Note: This column is hidden by default.
	<i>Type</i>	Shows the marker type. Note: This column is hidden by default.

Description Pane

The *Description* pane displays the message stored on the current marker. Note that the current marker is colored magenta in the browser and haloed in the canvas. If there are multiple markers selected, the current marker and its description changes as you use the *Go To The Next Selected Marker* and *Go To The Previous Selected Marker* buttons to browse the selected set. (This works even if you select the markers in the canvas and their parent category is not expandable in the browser pane.)

Related Topics

[Annotation Browser](#)

[Annotation Browser Context Menus](#)

[Annotation Browser Forms](#)

[Grouping Markers in the Annotation Browser](#)

[Sorting Markers in the Annotation Browser](#)

[Filtering Markers in the Annotation Browser](#)

[Layout XL Assistants](#)

Annotation Browser Context Menus

The Annotation Browser has two context menus.

Click the right mouse button in the **table header** to show or hide different columns in the table.

Click the right mouse button in the **table body** to access the following functions. (These are enabled and disabled dynamically depending on what is selected in the browser window.)

Command Name	Description
<i>Open (<mode>/<tab>)</i>	Opens the cellview containing the selected marker with the behavior specified in the command name. For example, <i>Open (edit/new tab)</i> opens the cellview in edit mode in a new tab; <i>Open (read/current tab)</i> opens the cellview in read mode in the current tab. You specify the default behavior for this command in the Annotation Browser Options form. To open a different view of the current cell, use the <i>Open</i> command instead.
<i>Open...</i>	Brings up the Open form where you can specify ■ Which view of the current cell you want to open ■ Whether that view is opened in edit or read mode ■ Whether it is opened in a new tab, the current tab, or in a new window.
<i>Expand All Below</i>	Expands the hierarchy to display everything underneath the currently selected node. Note: You can expand a single level of hierarchy for a selected entry using the Right arrow on your keyboard.
<i>Collapse All Below</i>	Collapses the hierarchy to hide everything underneath the currently selected node. Note: You can collapse a single level of hierarchy for a selected entry using the Left arrow on your keyboard.
<i>Select All Below</i>	Expands the hierarchy and selects all the marker entries underneath the currently selected node.

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

<i>Check/Uncheck</i>	Sets the <i>Checked</i> state of the selected markers. When a marker is set to <i>Checked</i>, you can use the <i>Hide Checked Markers</i> button to hide it in the Annotation Browser. Note: This command has no effect on the marker display in the canvas, only in the Annotation Browser.
<i>Set/Unset Highlight State</i>	Sets the highlight state of the selected markers in the design window. Using this command on a node sets the visibility for all the markers under that node. Note: A grayed-out version of the eye icon displayed against a node indicates that only some of the markers under that node are highlighted.
<i>Suppress/Unsuppress Expansion</i>	Suppresses (or unsuppresses) the expansion of the node under the cursor. You suppress expansion globally by setting a limit on the number of violations using the <i>Suppress expansion</i> option in the Annotation Browser Options form. You then use these commands to dynamically unsuppress (and suppress) the expansion for a selected node in the browser. Note: These commands appear only when expansion has been suppressed as a result of the limit set using the <i>Suppress expansion</i> option.
<i>Highlight Color</i>	Sets the color used to draw the corresponding marker in the canvas. Using this command for a node sets the color for all the markers under that node. Choose <i>cycle</i> to let Layout XL select the color automatically by cycling through a predefined list. Note: This command does not change the color assigned to the marker in the database, only the color used to draw the marker in the canvas.
<i>Delete</i>	Deletes the selected markers from the Annotation Browser and the design window
<i>Select Nets</i>	Selects in the layout canvas and Navigator assistant the nets associated with the currently highlighted markers.

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

<i>Group By</i>	Opens the Group By form where you specify how the list of annotations is grouped and sorted in the browser. For more information, see Grouping Markers in the Annotation Browser.
<i>Incomplete Net Filter</i>	Opens the Incomplete Net Filter form where you filter the incomplete net markers shown in the browser based on the set of currently selected nets in the layout canvas or Navigator assistant. For more information, see Setting an Incomplete Net Filter.
<i>Options</i>	Opens the Annotation Browser Options form, where you specify ■ The number of violations, which prevent a node from being expanded ■ The levels of hierarchy searched for violation markers ■ The default open behavior ■ How new markers are highlighted

You can delete and set the checked state of markers only in cellviews that are open in the edit mode. If you try to use the *Delete* or *Check/Uncheck* commands (or their [Browser Pane](#) equivalents) on a read-only cellview, the application prompts you to confirm that you want to re-open the cellview in edit mode before performing the requested operation.

Note: This prompt represents a behavior change from previous releases, where the software automatically re-opened the cellview in edit mode if this was required in order to delete a marker. To restore this behavior, set one or both of the following environment variables to `nil`.

- [deleteConfirmModeChange](#)
- [checkStateConfirmModeChange](#)

Related Topics

[Annotation Browser](#)

[Annotation Browser Graphical User Interface](#)

[Annotation Browser Forms](#)

[Grouping Markers in the Annotation Browser](#)

[Sorting Markers in the Annotation Browser](#)

[Filtering Markers in the Annotation Browser](#)

[Layout XL Assistants](#)

Annotation Browser Forms

Use the following forms to customize the way marker information is displayed in the Annotation Browser.

- [Annotation Browser Options](#)
- [Group By](#)
- [Incomplete Net Filter](#)
- [Custom Filter](#)

Related Topics

[Annotation Browser Graphical User Interface](#)

[Annotation Browser Context Menus](#)

[Grouping Markers in the Annotation Browser](#)

[Sorting Markers in the Annotation Browser](#)

[Filtering Markers in the Annotation Browser](#)

[Layout XL Assistants](#)

Annotation Browser Options

Use the **Annotation Browser Options** form to specify the look and behavior of the Annotation Browser. To open the form, click the right mouse button in the browser pane and choose *Options*.

Suppress expansion at specifies in 1000s the number of violations which prevents a node from being expanded. If the number of entries under a particular node exceeds the limit, the node is not expanded by default.

Note: If you select markers in the canvas and their parent category is not expandable in

the browser pane, you can still use the *Go To The Next Selected Marker* and *Go To The Previous Selected Marker* buttons in the Annotation Browser toolbar to cycle through the selected markers. Additionally, you can override the global limit for an individual node by positioning your cursor over the node, clicking the right mouse button, and choosing *Unsuppress Expansion* from the context menu.

Hierarchy depth specifies the number of levels of hierarchy searched for markers. The default is 3.

Default Open specifies the default open behavior when you descend into a cellview.

Mode specifies whether the cellview is opened for *edit* or *read*.

Open in specifies whether the cellview is opened in a *new tab*, the *current tab*, or a *new window*.

Marker Highlighting

Remove highlighting when browser is closed removes the highlighting in the design canvas when you close the Annotation Browser assistant. The default is off, which means the markers remain highlighted in the canvas when you close the browser.

Automatic highlighting controls the highlight state of newly-created markers in the design canvas. For example, when you delete part of a completed route in the canvas, an open marker is created automatically. This option controls whether the dragline representing that marker is automatically displayed or not.

Highlight new marker categories specifies that draglines are displayed automatically in the canvas to represent newly-created marker categories.

Highlight new markers specifies that draglines are displayed automatically in the canvas to represent newly-created individual markers.

Inherit from parent category specifies that a newly-created marker inherits the highlight state from its parent category. Note that if the category is only partially highlighted, the dragline representing the new marker is not automatically highlighted.

Related Topics

[Annotation Browser](#)

[Annotation Browser Forms](#)

[Layout XL Assistants](#)

Group By

Use the **Group By** form to specify how the list of annotations is grouped and sorted in the browser.

To open the form, click the right mouse button in the browser pane and choose *Group By*. Choose the first category to group by from the **Group items by** pull down, which lists all the columns available in the current tab. Click *Ascending* to sort the lowest number or the beginning of the alphabet first for that grouping, or *Descending* to sort the highest number or the end of the alphabet first.

If required, do the same for second, third, and fourth-level groupings using the **Then by** pull-downs in the form.

Click *Apply* to apply your changes to the markers currently displayed in the Annotation Browser. Click *OK* to apply the changes and to close the Group By form. Click *Defaults* to restore the default grouping for the tab (for example, the default grouping for the *Connectivity* tab is *Type/Net*).

For more information, see [Grouping Markers in the Annotation Browser](#).

Related Topics

[Annotation Browser](#)

[Annotation Browser Forms](#)

[Layout XL Assistants](#)

Incomplete Net Filter

Use the **Incomplete Net Filter** form to filter the incomplete net markers shown in the browser, based on the set of currently selected nets in the layout canvas or Navigator assistant.

To open the form, click the right mouse button in the browser pane and choose *Incomplete Net Filter*. Use the pull-down at the top of the form to select the filter type.

No Filter means that the list of markers shown in the browser is not filtered.

Show nets only from list shows only those markers associated with the listed nets.

Hide nets from list hides the markers associated with the listed nets.

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

When a filter has been applied, the *Incomplete Nets* category label in the Annotation Browser is displayed in blue text.

Populate the *List of Nets* using the buttons on the right of the form.

Add Selected Nets adds the names of the currently selected nets in the layout window to the list of nets in the form

Remove removes one or more selected nets from the list.

Remove All clears the list of nets completely.

Click *Apply* to apply your changes to the markers currently displayed in the Annotation Browser; or *OK* to apply the changes and to close the Incomplete Net Filter form.

Related Topics

[Annotation Browser](#)

[Annotation Browser Forms](#)

[Layout XL Assistants](#)

Custom Filter

Use the **Custom Filter** form to define more complex filtering criteria, which reduce the number of annotations displayed in the Annotation Browser. To open the form, click the header of the column for which you want to define the custom filter and choose *Custom* from the list.

Add criteria adds further filtering criteria to the list. Choose whether you want filtering to *Match all of the following* or *Match any of the following* criteria.

For each criterion,

1. Choose the operator from the first pull-down list.

Operator	Meaning
<	Less than
<=	Less than or equal to
==	Exactly equal to

Operator	Meaning
>=	Greater than or equal to
>	Greater than
!=	Not equal to

2. Choose the column content you want to filter by from the second pull-down list. The content of this list is determined automatically by the system.
3. If required, click *Delete* to remove the corresponding criterion from the list.

Click *Apply* to apply your changes to the markers currently displayed in the Annotation Browser; or *OK* to apply the changes and dismiss the Custom Filter form.

For more information, see [Defining and Applying a Custom Filter](#) under [Filtering Markers in the Annotation Browser](#).

Related Topics

[Annotation Browser](#)

[Annotation Browser Forms](#)

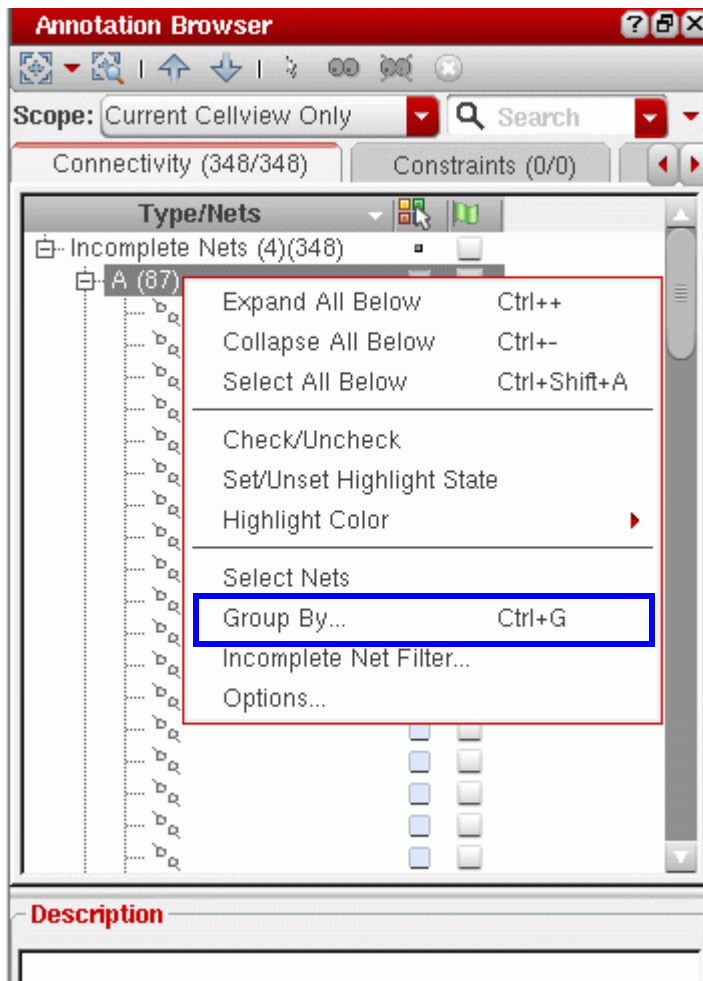
Grouping Markers in the Annotation Browser

To change the way the markers displayed in the Annotation Browser are grouped,

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

1. In the browser pane, click the right mouse button and choose *Group By*.



Virtuoso Layout Suite XL User Guide

Layout XL Assistants

The Group By form is displayed showing the current groupings. In this example, we will change these to group first by *Type*, then by *Layers*, and finally by *Nets*, with each group sorted in *Ascending* order.

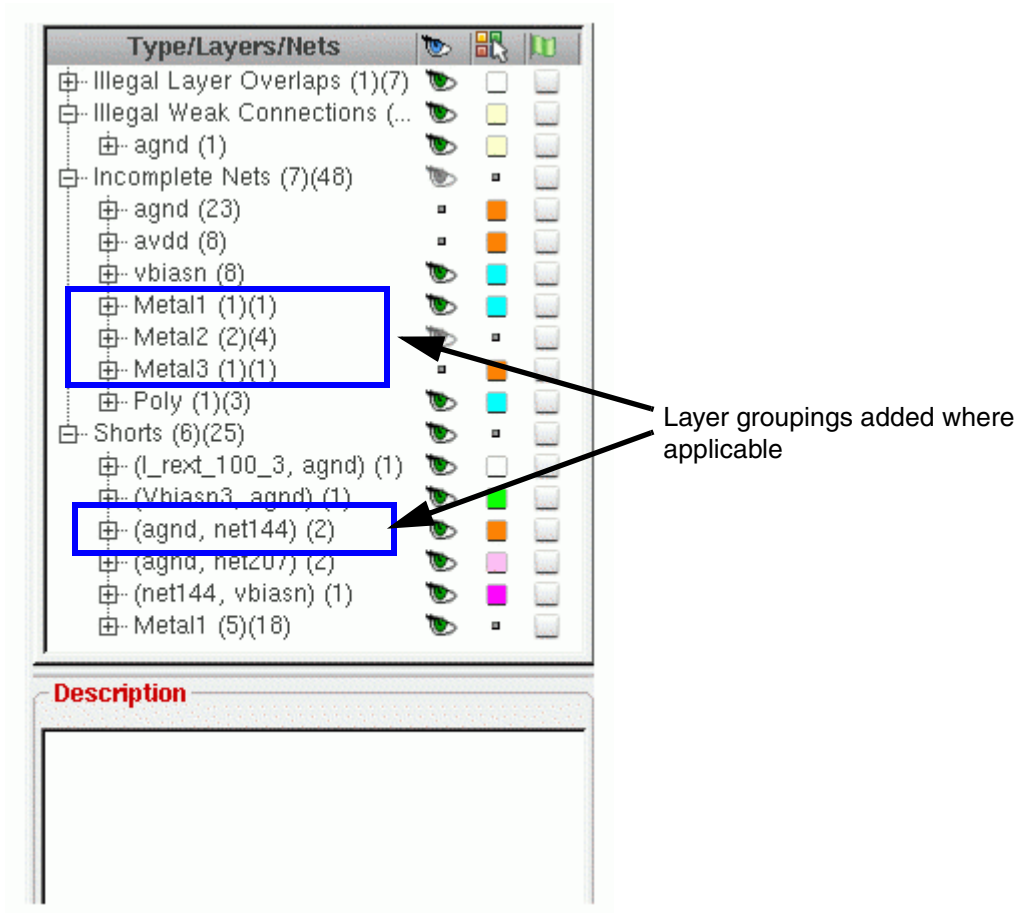
The screenshot shows the 'Group By' dialog box. It has a title bar with the text 'Group By' and standard window controls (minimize, maximize, close). The dialog contains four sections for grouping:

- Group items by:** A dropdown menu showing 'Type' and two radio buttons: 'Ascending' (selected) and 'Descending'.
- Then by:** A dropdown menu showing 'Layers' and two radio buttons: 'Ascending' (selected) and 'Descending'.
- Then by:** A dropdown menu showing 'Nets' and two radio buttons: 'Ascending' (selected) and 'Descending'.
- Then by:** A dropdown menu showing 'No Group' and two radio buttons: 'Ascending' (selected) and 'Descending'.

At the bottom of the dialog are five buttons: 'OK' (highlighted in red), 'Cancel', 'Defaults', 'Apply', and 'Help'.

2. Click *OK* to apply the changes and to close the form.

Where layer information is available, the markers in the browser pane are grouped by layers.



Restoring Default Groupings in the Annotation Browser

To restore the default grouping defined for the tab:

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

- Click *Defaults* in the Group By form.

Group By

Group items by

Type

Ascending

Descending

Then by

Layers

Ascending

Descending

Then by

Nets

Ascending

Descending

Then by

No Group

Ascending

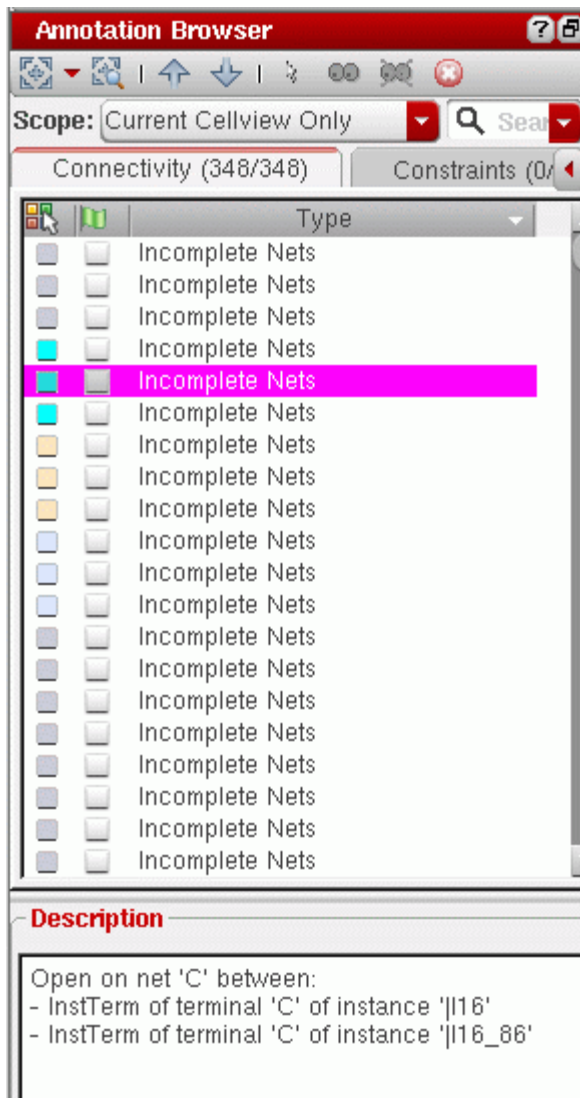
Descending

OK Cancel Defaults Apply Help

Click *Defaults* to restore the default groupings for the tab

Annotation Browser Display with No Grouping Rules

If you specify no grouping, the Annotation Browser tree view is flattened, the tree column is removed, and the previous Top grouping column is restored and used to define how the markers are sorted.



Related Topics

[Annotation Browser Graphical User Interface](#)

[Annotation Browser Context Menus](#)

Annotation Browser Forms

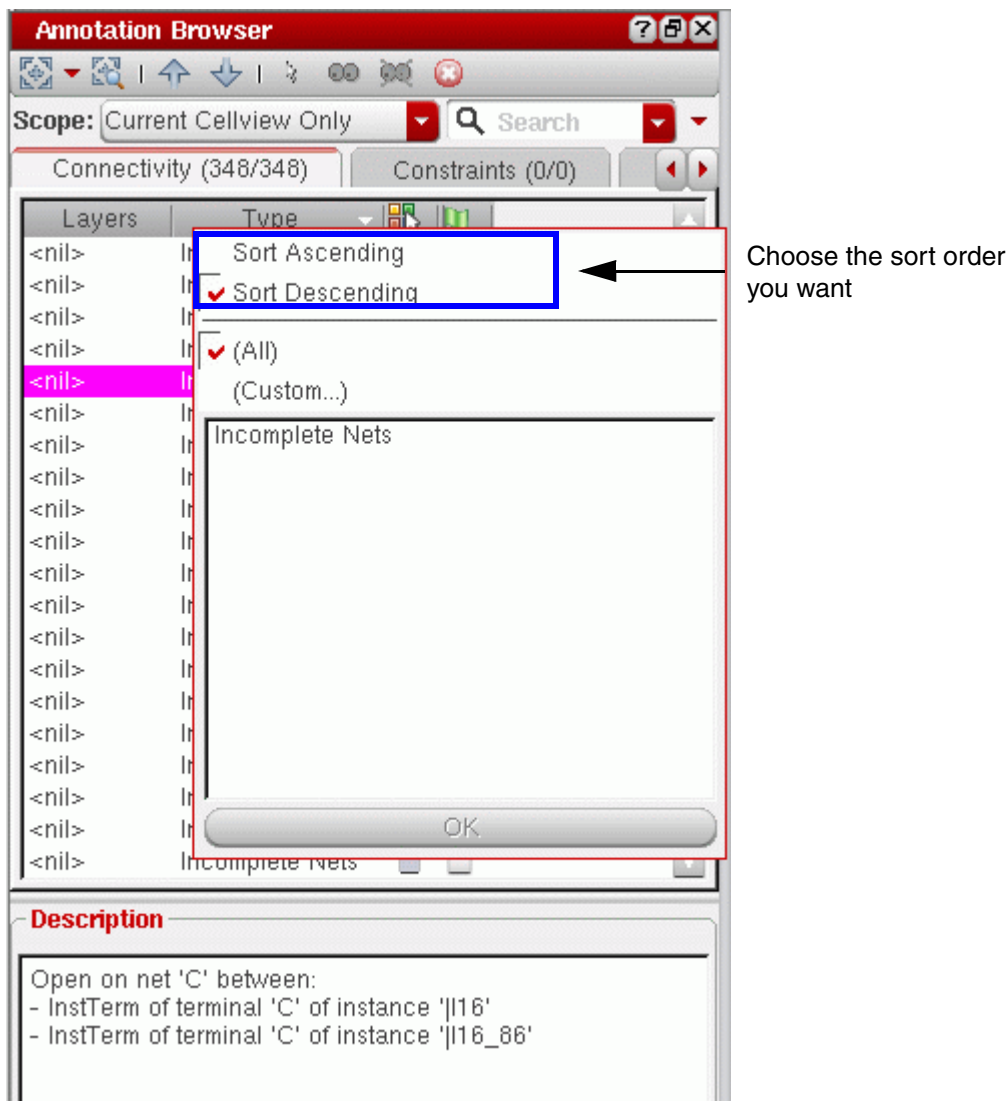
Sorting Markers in the Annotation Browser

Filtering Markers in the Annotation Browser

Sorting Markers in the Annotation Browser

To sort the markers differently in the browser pane,

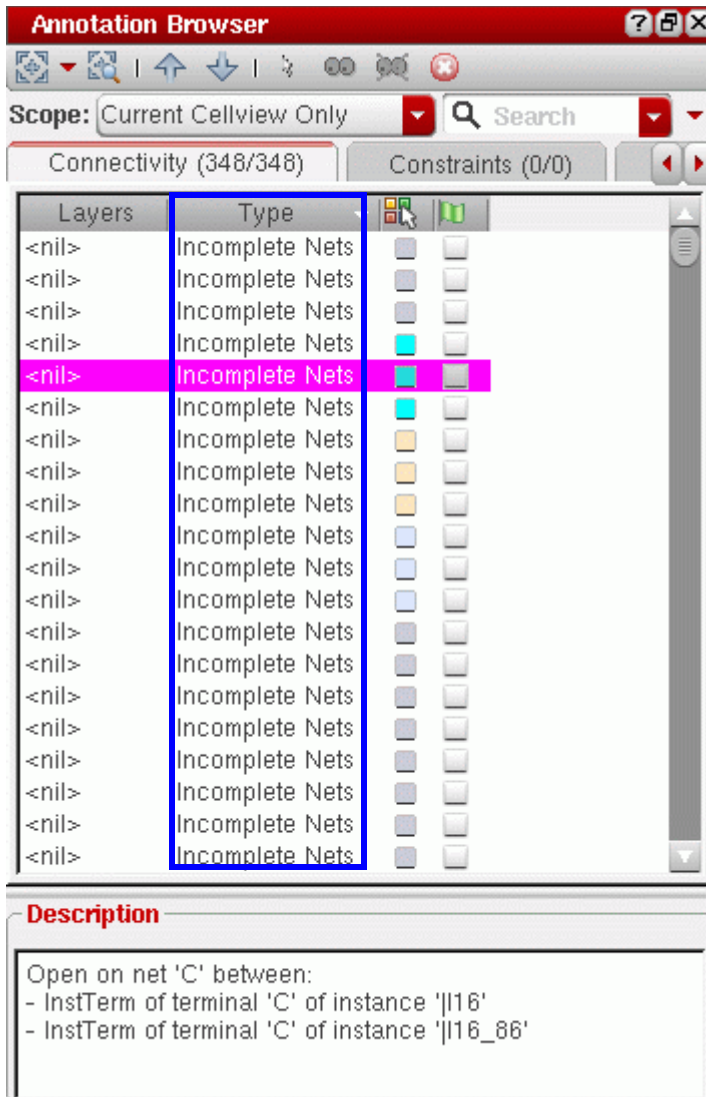
1. Click in the header of the column you want to sort by.
2. Choose either *Sort Ascending* or *Sort Descending* from the popup menu.



Virtuoso Layout Suite XL User Guide

Layout XL Assistants

The table entries are re-sorted based on your selection.



3. To sort the entries by a different column, repeat steps 1 and 2 for the column in question.

Note: You cannot sort entries based on the highlight color.

Related Topics

[Annotation Browser Graphical User Interface](#)

[Annotation Browser Context Menus](#)

[Annotation Browser Forms](#)

[Grouping Markers in the Annotation Browser](#)

[Filtering Markers in the Annotation Browser](#)

Filtering Markers in the Annotation Browser

You can filter the markers shown in the Annotation Browser display using either a predefined column filter, or by defining your own filter using the [Custom Filter](#) form. Additionally, you can use the [Incomplete Net Filter](#) filter form to filter incomplete net markers based on the set of currently selected nets in the layout canvas or Navigator assistant. For more information, see

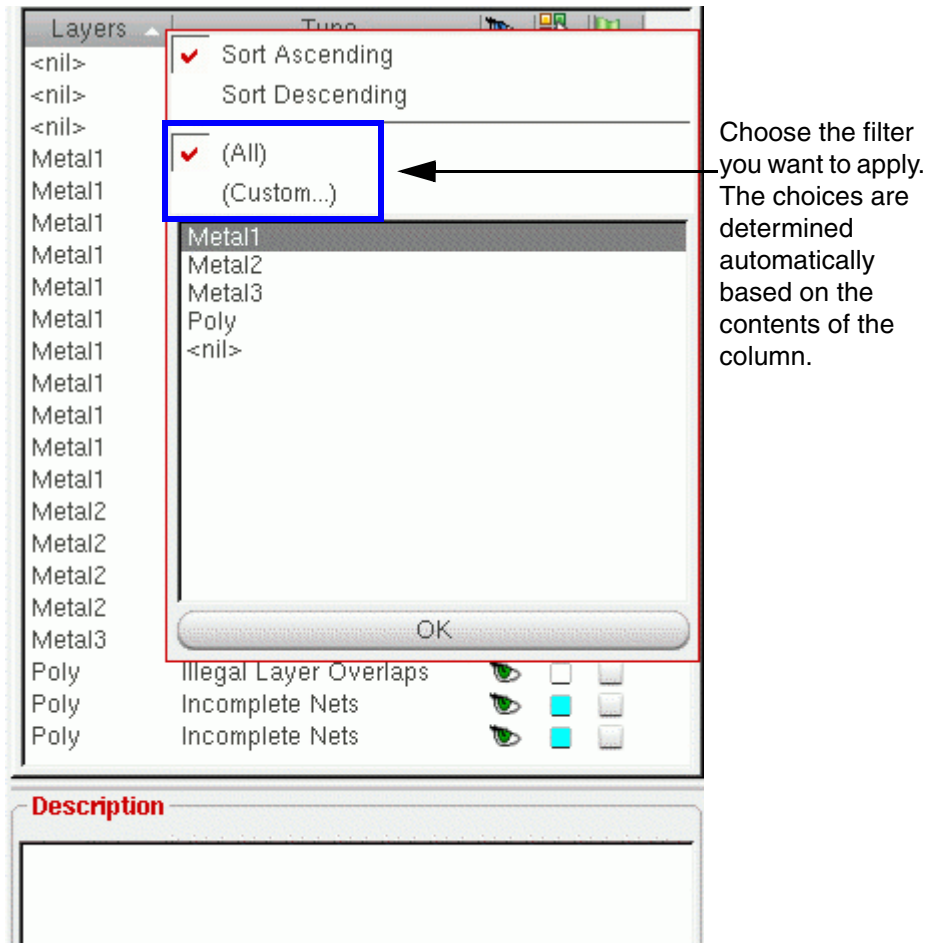
- [Setting a Predefined Column Filter](#)
- [Defining and Applying a Custom Filter](#)
- [Setting an Incomplete Net Filter](#)

Setting a Predefined Column Filter

To use a predefined filter to show, for example, only the markers on a specified layer,

1. Click the header of the column to which you want to apply the filter.

2. Choose *Metal1* from the list of simple filters and click *OK*.



The browser updates to show markers only on the Metal1 layer.

The column header is displayed in bold blue text to indicate that there is a filter set on that column.

If required, you can apply a predefined or custom filter on another column. The browser shows only those markers that match all the filter criteria for all the columns.

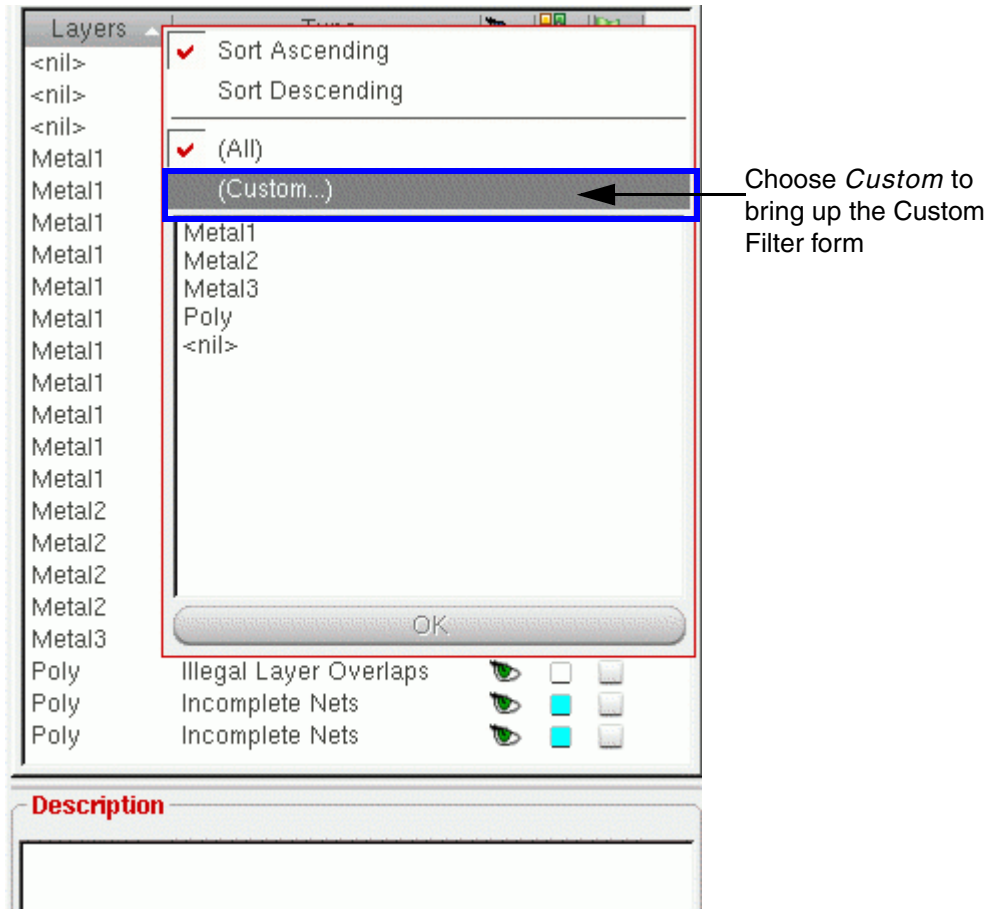
Defining and Applying a Custom Filter

To define and apply a custom filter:

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

1. Click in the header of the column for which you want to define the custom filter and choose *Custom* from the list.



The Custom Filter form is displayed.



2. Click *Add criteria* to add another criterion to the list and *Match any of the following*.
3. For each criterion, choose the operator from the first pull-down list and the column content you want to filter by from the second pull-down list.

For example, the custom filter shown below will display all the markers on layers Metal1 and Poly.

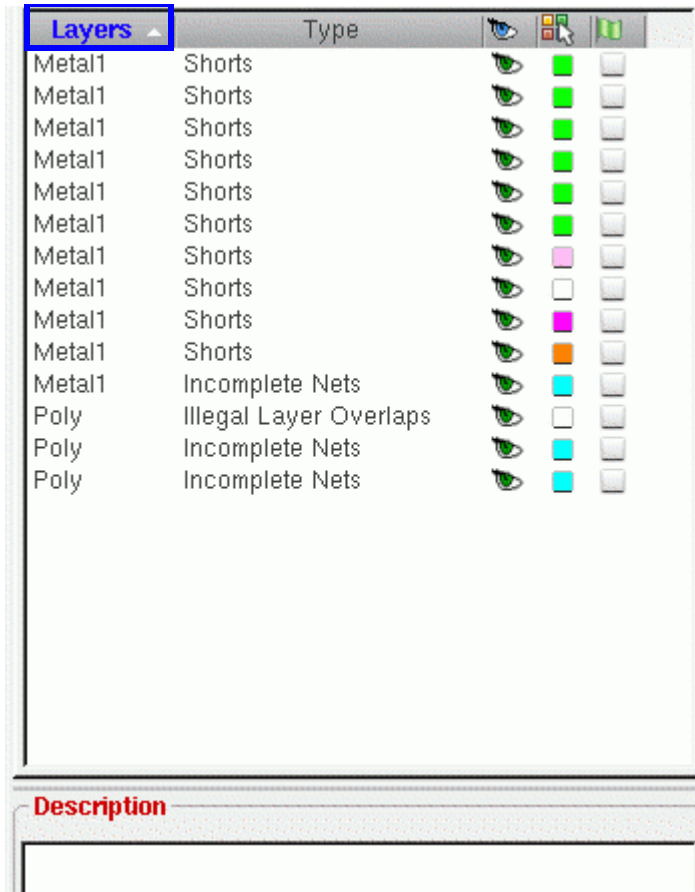


4. Click *OK* to apply the changes and dismiss the Custom Filter form.

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

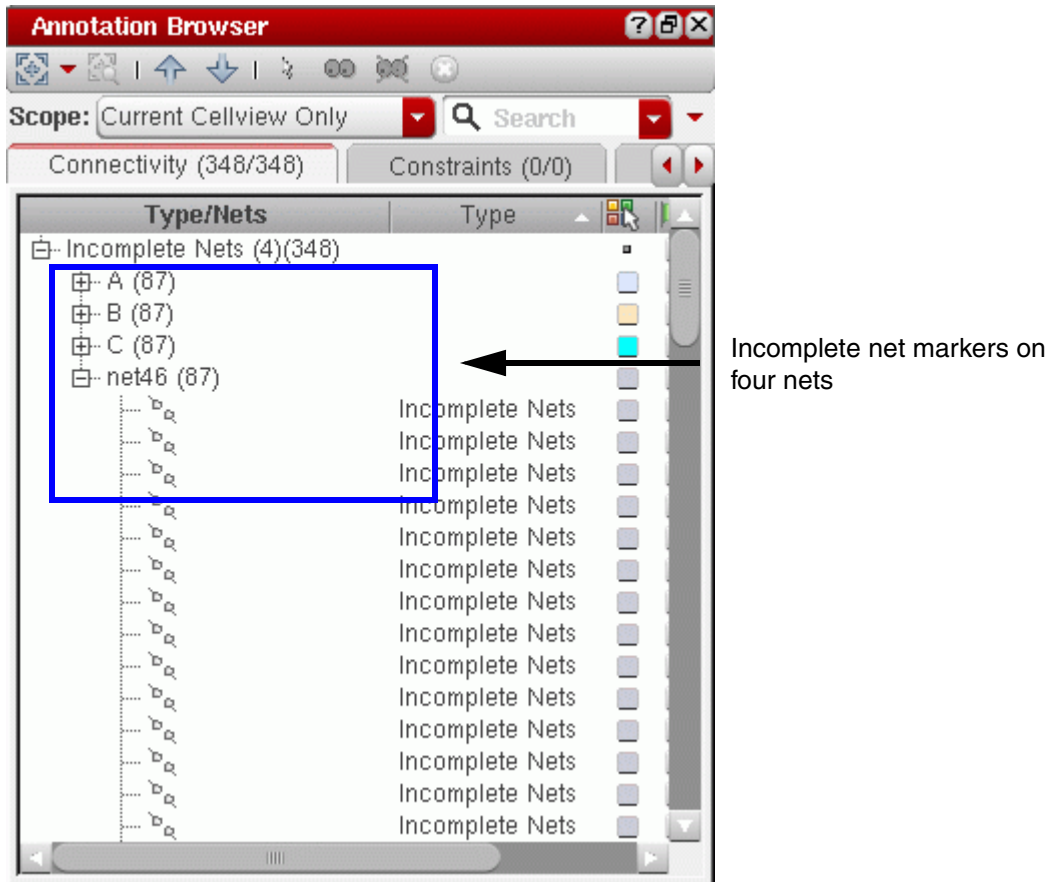
The browser updates to show only the markers on the Metal1 and Poly layers. The column header is drawn in bold blue text to indicate that there is a filter set on that column.



If required, you can apply a predefined or custom filter on another column. The browser shows only those markers that match all the filter criteria for all the columns.

Setting an Incomplete Net Filter

Consider the picture below, which shows the Annotation Browser with incomplete net markers associated with 4 nets: A, B, C, and net46.



Now, select nets A and B from the layout Navigator Assistant. To filter the incomplete net markers shown in the Annotation Browser based on the selected net in the layout window,

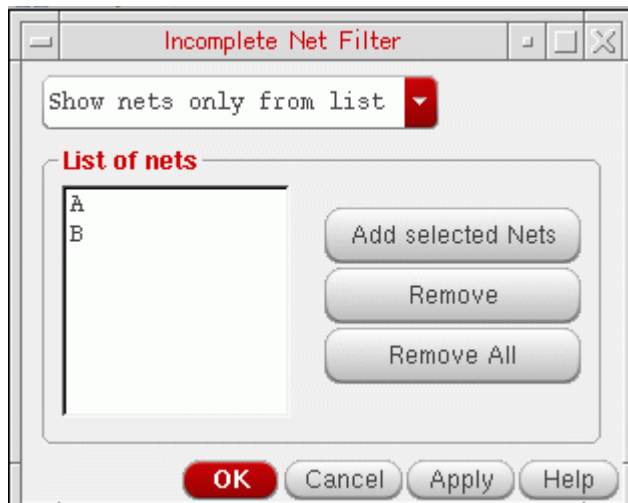
1. Click the right mouse button in the browser pane and choose *Incomplete Net Filter*.

The Incomplete Net Filter form is displayed.

2. Click *Add Selected Nets* to populate the list of nets with the names of nets currently selected in the layout window.

Nets A and B are added to the list.

3. From the pull-down at the top of the form, choose *Show nets only from list*.

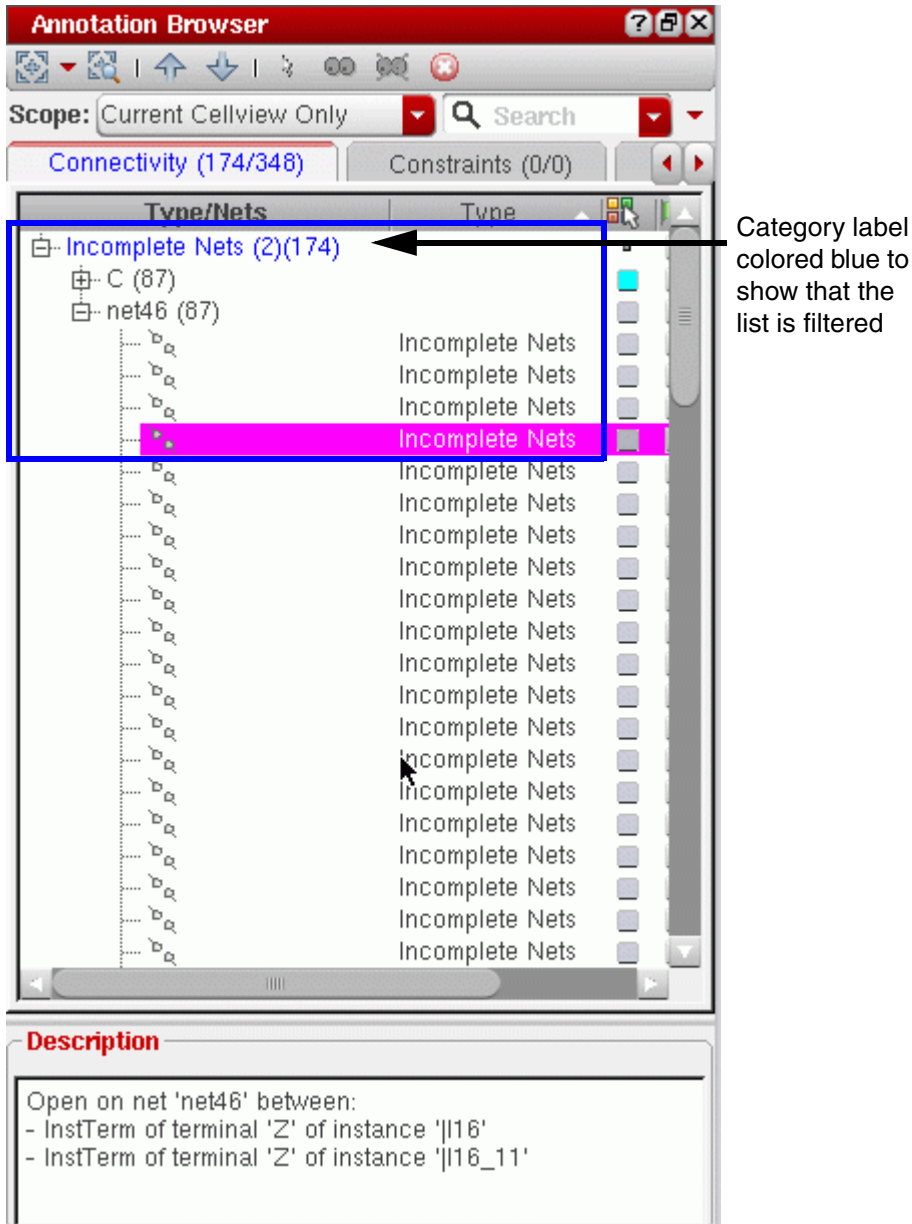


4. Click *OK*.

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

The browser updates to hide the incomplete net markers associated with the listed nets. All other incomplete net markers are shown.



The label for the *Incomplete Nets* category and the tab label are colored blue to indicate that a filter has been applied. The marker count in the tab label also indicates that some markers are hidden.

Related Topics

[Annotation Browser Graphical User Interface](#)

[Annotation Browser Context Menus](#)

[Annotation Browser Forms](#)

[Grouping Markers in the Annotation Browser](#)

[Sorting Markers in the Annotation Browser](#)

Assigning Colors to Markers

To assign a different color to a marker in the layout canvas,

1. In the [Annotation Browser](#) assistant, click the *Set Highlight Color* column next to the entry for the marker whose color you want to change.
2. Do one of the following.
 - ☐ Choose the color you want from the predefined list.
 - ☐ Choose *cycle* to automatically select the next color in the predefined list.

The marker in the canvas changes to the color you specified.

To assign colors to all the markers under a particular node in the browser,

1. In the [Annotation Browser](#) assistant, click the *Set Highlight Color* column for the parent node of the markers you want to change.
2. Do one of the following.
 - ☐ To show all the markers contained in the node in the same color, choose the color you want from the drop-down list.
 - ☐ To show each of the markers contained in the node in a different color, choose *cycle*. The markers are assigned colors automatically based on the predefined list.

The markers change to the colors you specified.

Related Topics

[Annotation Browser Graphical User Interface](#)

[Annotation Browser Context Menus](#)

[Annotation Browser Forms](#)

[Grouping Markers in the Annotation Browser](#)

[Sorting Markers in the Annotation Browser](#)

[Zooming In on Markers](#)

Zooming In on Markers

To zoom in on specific markers in the layout canvas,

- In the [Annotation Browser](#) assistant, select the markers you are interested in.
- Click the *Zoom To Selected* button in the Annotation Browser toolbar.



The canvas display zooms to the flight lines for the selected nets. The level of detail you see depends on how many nets are selected and their relative locations in the design.

To automatically zoom in on the canvas representation of the currently selected marker,

- Click the *Auto Zoom* button in the [Annotation Browser](#) assistant toolbar.



The display automatically zooms and pans to the marker representing the currently selected entry in the Annotation Browser. When the selection changes, the display updates accordingly.

For more information on *Auto Zoom*, see the toolbar section in [Annotation Browser Graphical User Interface](#).

Related Topics

[Annotation Browser Graphical User Interface](#)

[Annotation Browser Context Menus](#)

[Annotation Browser Forms](#)

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

[Grouping Markers in the Annotation Browser](#)

[Sorting Markers in the Annotation Browser](#)

[Assigning Colors to Markers](#)

[Layout XL Assistants](#)

Constraint Manager

Note: This section describes only the extended Constraint Manager functionality enabled in Layout XL. For detailed information on the Constraint Manager, see Constraint Manager Assistant in the *Virtuoso Unified Custom Constraints User Guide*.

Use the **Constraint Manager** assistant to add, modify, check or delete constraints in your design.

Examples of constraints include matching parameters, symmetry, orientation, matched orientation, IR drop, parasitic filtering, parasitic estimation, clustering, alignment, area, distance, boundaries, power structure/guard rings (layout only), custom module generators (modgens), cell plans, and analog groups.

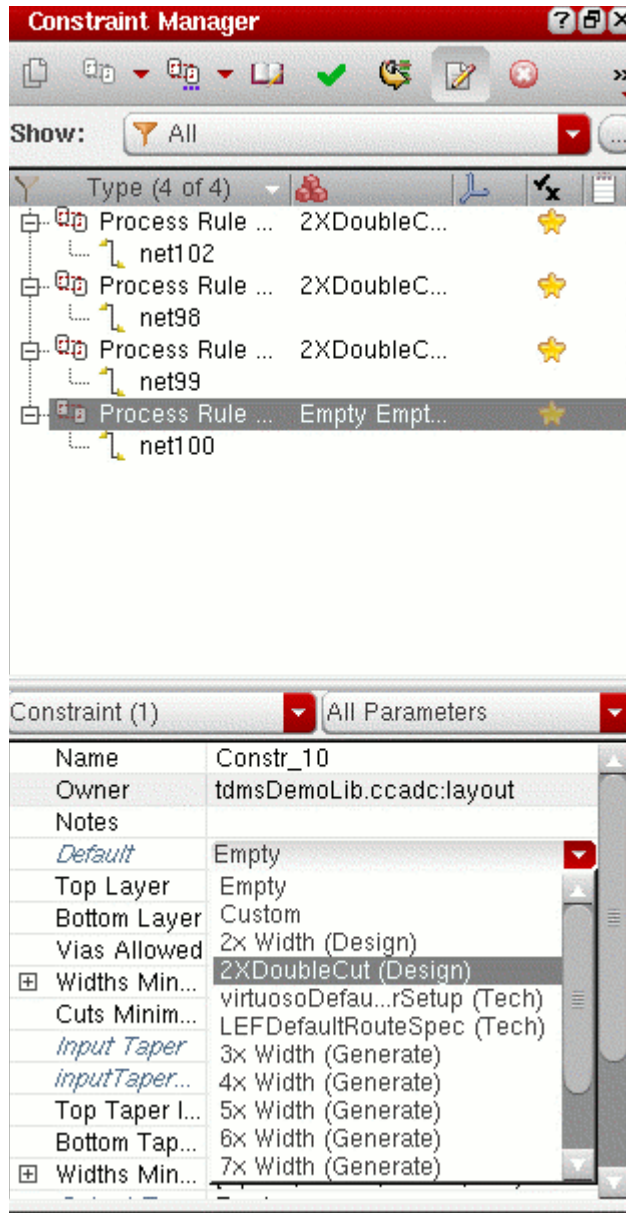
The Constraint Manager displays a full set of constraints for a design wherever you are in the design hierarchy and wherever the constraints were created in that hierarchy. It displays the constraints in a logical manner, and shows which constraints are currently met and which have been overridden during the course of the physical implementation.

The Constraint Manager user interface comprises two main component parts; the Constraint Manager table at the top, which lets you browse the constraints in your design; and the

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

Constraint Editor underneath it, which lets you change the values of one or more selected constraints.



Constraint Manager table

Constraint Editor

For more information on the Constraint Manager Toolbar, see [Constraint Manager Toolbar](#).

The Constraint Manager has a dedicated Batch Checker in the toolbar, which automatically enables the placement, specialty routing, and process constraints. In addition, the checker verifies and updates the constraint status and leaves them back as is in the 'Verify Design' form.

To run other checks, you should simply invoke *Verify Design* and click OK after enabling the required check.

For more information on verifying design constraints, see [Check Constraints and Batch Checker](#).

Related Topics

[Constraint Manager Assistant](#)

[Updating Constraints From Schematic to Layout](#)

[Comparing Constraints Between Schematic and Layout](#)

[Overriding Process Rules](#)

[Creating a Cluster Boundary Using the Constraint Manager](#)

[Creating and Editing Net Constraints](#)

[Layout XL Assistants](#)

Updating Constraints From Schematic to Layout

To update constraints in the layout to match those set in the schematic,

- In the Constraint Manager toolbar, click the *Update Layout Constraints* button.



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

For more information, see [Constraint Transfer](#) on page 218.

Related Topics

[Constraint Manager](#)

[Comparing Constraints Between Schematic and Layout](#)

[Overriding Process Rules](#)

[Creating a Cluster Boundary Using the Constraint Manager](#)

[Creating and Editing Net Constraints](#)

[Layout XL Assistants](#)

Comparing Constraints Between Schematic and Layout

The **Constraint Comparison Report** is a web page listing matched and unmatched constraints in the schematic and layout views.

The report sorts the differences into four categories and summarizes the contents at the top of the page.

Mismatched Constraints are constraints that are different in the schematic and layout views.

Matched Constraints are constraints that are the same in the schematic and layout views.

Unique Source Constraints are constraints that are present in the schematic view but not in the layout view.

Unique Target Constraints are constraints that are present in the layout view but not in the schematic view.

The scope of the report is limited to the parameters and attributes displayed in the Constraint Manager assistant. Process overrides defined in the Process Rule Editor are ignored.

Multiple layout instances generated from a single mfactored schematic instance are correctly handled as a single constraint member for comparison purposes and are not reported.

To see the constraint comparison report,

- In the Constraint Manager toolbar, click the *Constraint Comparison Report* button.



Virtuoso Layout Suite XL User Guide

Layout XL Assistants

The system launches a web browser and loads the Constraint Comparison Report for the current Layout XL session.



Related Topics

[Constraint Manager](#)

[Updating Constraints From Schematic to Layout](#)

[Overriding Process Rules](#)

[Creating a Cluster Boundary Using the Constraint Manager](#)

[Creating and Editing Net Constraints](#)

[Layout XL Assistants](#)

Overriding Process Rules

To launch the Process Rule Editor,

- In the Constraint Manager toolbar, click the *Process Rule Editor* button.



Related Topics

[Constraint Manager](#)

[Updating Constraints From Schematic to Layout](#)

[Comparing Constraints Between Schematic and Layout](#)

[Creating a Cluster Boundary Using the Constraint Manager](#)

[Creating and Editing Net Constraints](#)

[Layout XL Assistants](#)

Creating a Cluster Boundary Using the Constraint Manager

To create a cluster boundary constraint using the Constraint Manager graphical user interface,

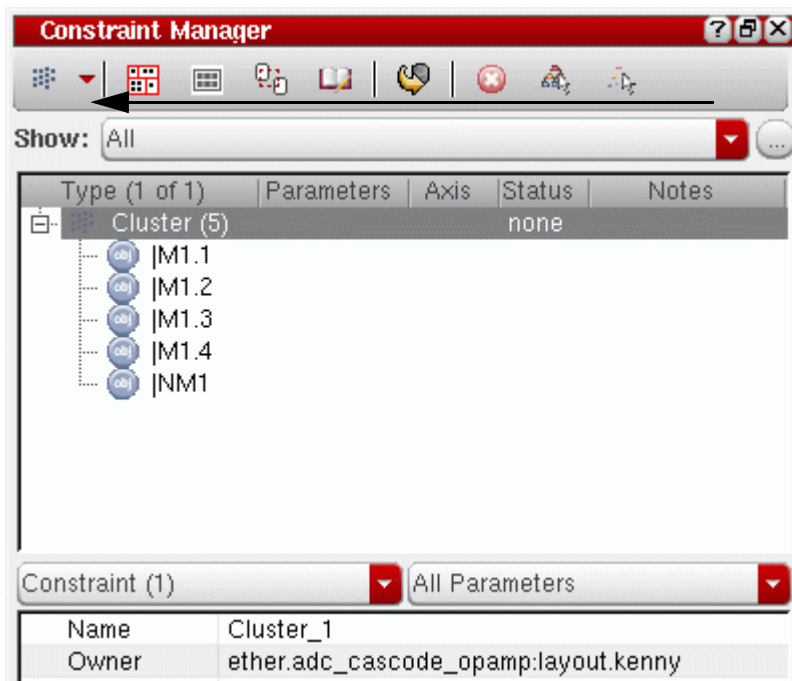
Virtuoso Layout Suite XL User Guide

Layout XL Assistants

1. In the Constraint Manager assistant, select an existing cluster around which you want to create a cluster boundary.

Note: You can create a cluster boundary only for an existing cluster.

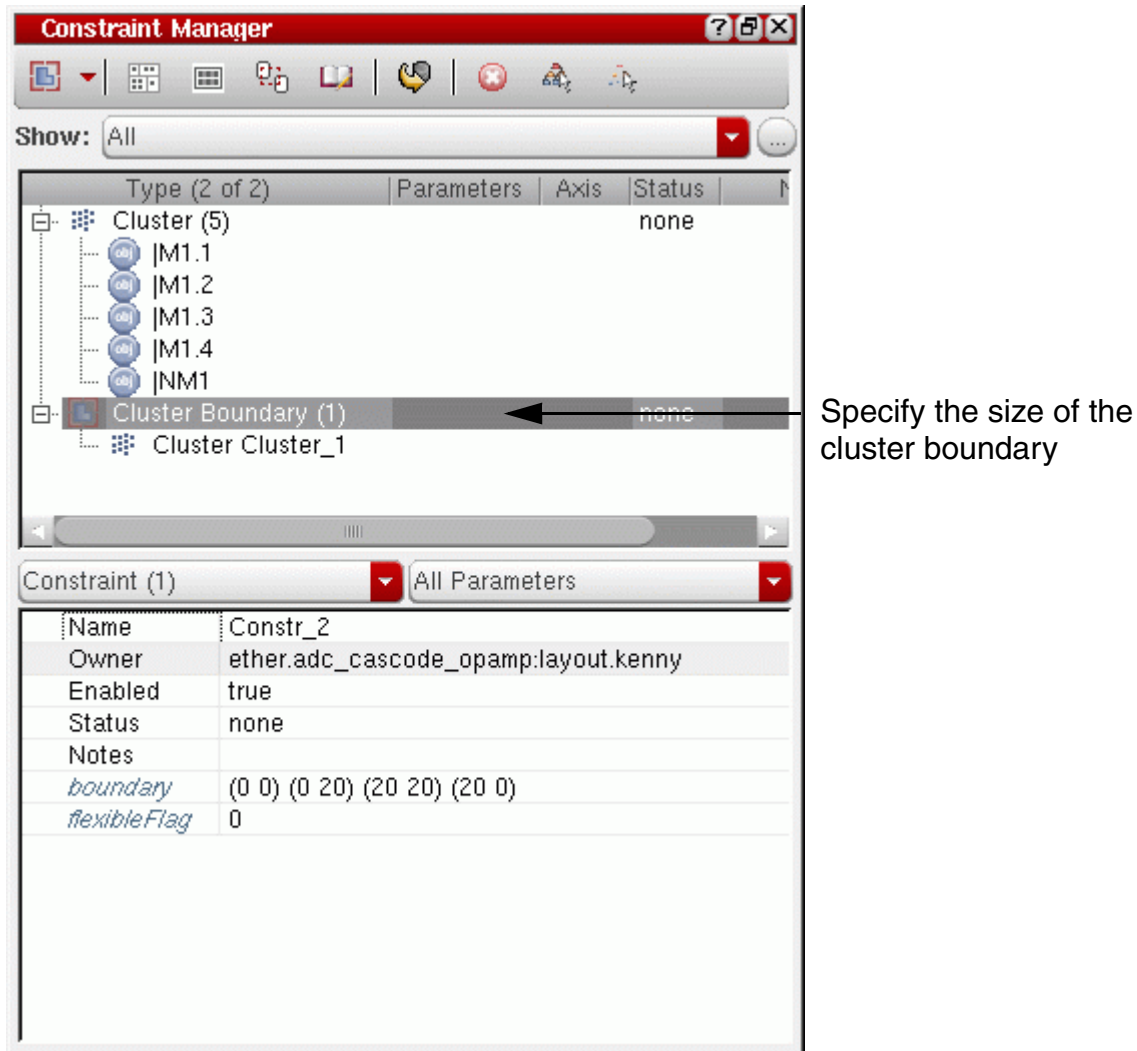
2. Choose *Cluster Boundary* from the constraint creation pull-down.



Choose *Cluster Boundary* from the pull-down

A cluster boundary constraint is created with the selected cluster constraint as a member. The cluster boundary is created with a default size. You are not prompted to draw the cluster boundary on the layout canvas.

3. Specify the size for the cluster boundary by typing the required coordinates into the appropriate fields in the Constraint Manager assistant.



Related Topics

[Constraint Manager](#)

[Updating Constraints From Schematic to Layout](#)

[Comparing Constraints Between Schematic and Layout](#)

[Overriding Process Rules](#)

[Creating and Editing Net Constraints](#)

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

Layout XL Assistants

Navigator

Note: This section describes only the extended Navigator functionality enabled in Layout XL. For detailed information on the Navigator, see [The Navigator Assistant](#) in the *Virtuoso Schematic Editor L User Guide*.

The **Navigator Assistant** is a hierarchical representation of all the devices, nets, and pins that exist in the design. The instances may include both, generic as well as parameterized cells (Pcells). Due to its hierarchical display, the Navigator Assistant simplifies the task of navigating through a hierarchical design to find the components you are looking for.

An unelaborated instance in the layout displays a [+] sign before the icon; indicating the instance holds some devices, nets, or pins within. To elaborate an instance, you must click the instance once. If the sub instance is a leaf, it does not carry the [+] sign. Else, the sub instances carry the [+] sign till you explore them and browse to the leaf level.

When using Layout XL, there is a Navigator available in the schematic window and one in the layout window. Selecting a component at the current level of hierarchy in either of the Navigators selects the corresponding component in the other Navigator and in the canvases. Similarly, selecting a component at the current level in either of the canvases selects the component in both the Navigators.

Note: Similar to cross-selecting, cross probing across the two views only works for objects selected at the current level of hierarchy. Probes created elsewhere in the hierarchy still propagate through the hierarchy but they do not cross probe to the schematic. See [Adding a Probe using the Right Mouse Button](#).

Related Topics

[The Navigator Assistant](#)

[Viewing Pin, Cell, and Signal Types in the Navigator Assistant](#)

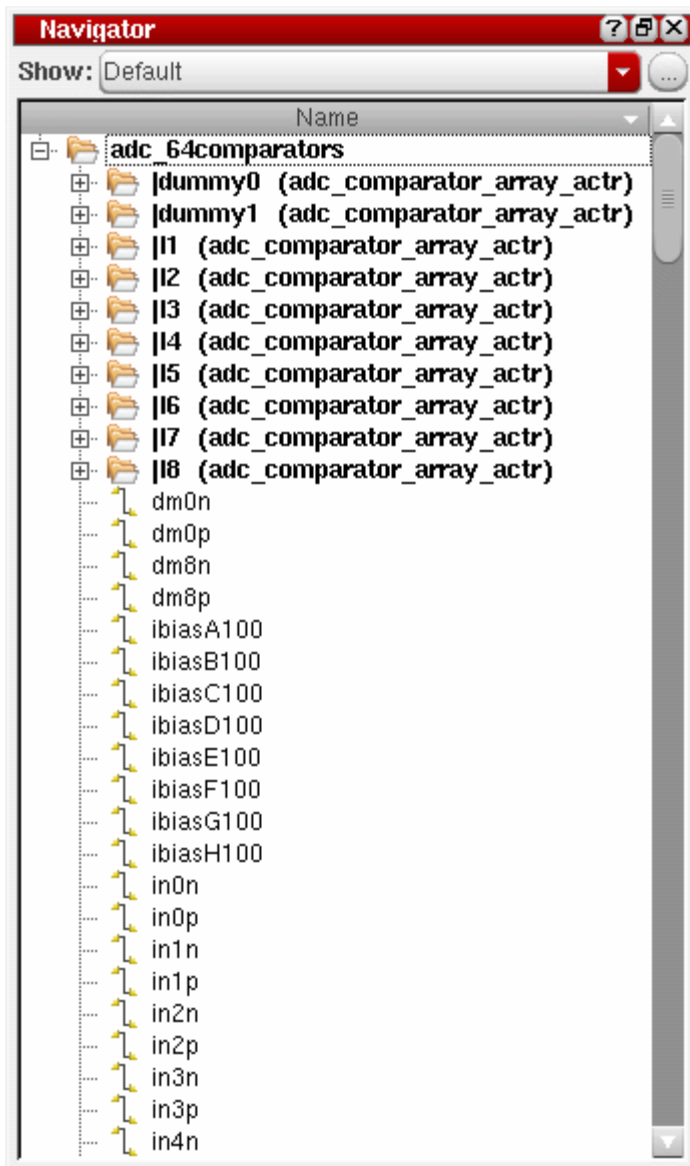
[Customizing the Navigator Filters](#)

[Viewing XL Status](#)

[Selecting and Deselecting Nets](#)

[Adding a Probe using the Right Mouse Button](#)

Viewing Pin, Cell, and Signal Types in the Navigator Assistant



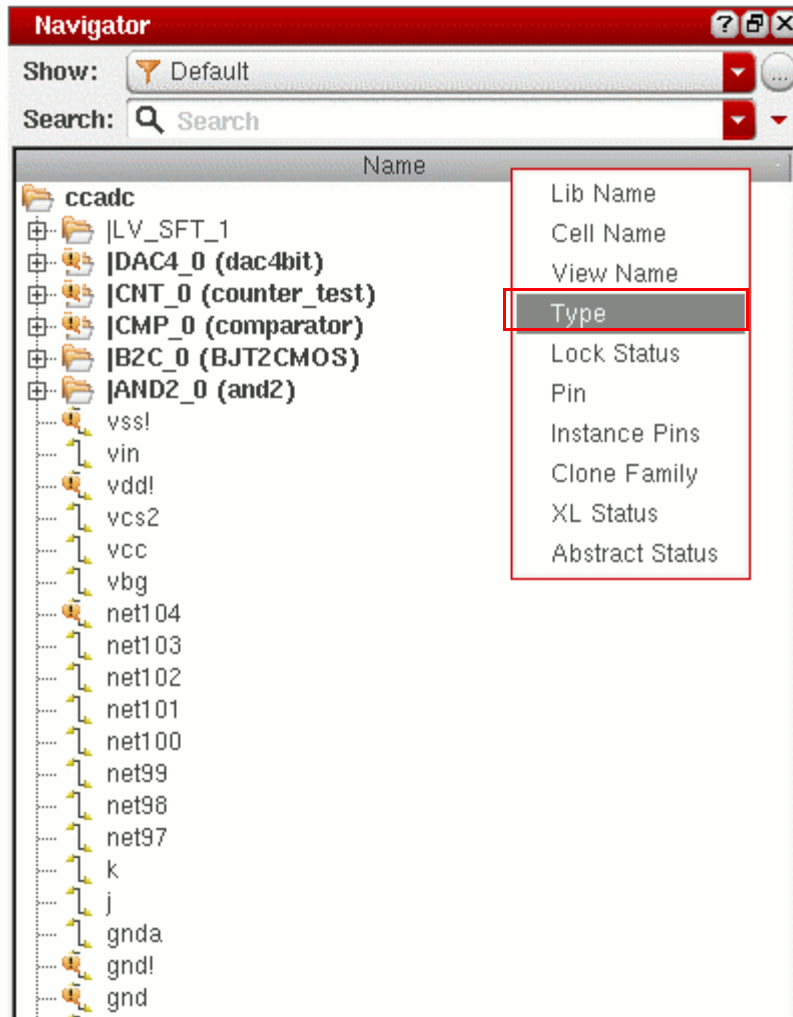
You can customize the Navigator Assistant to display the pin type, the cell type of an instance master, or the signal type of a net in the design. To do this:

1. With your cursor in the *Name* column in the Navigator Assistant, click the right mouse button.

Virtuoso Layout Suite XL User Guide

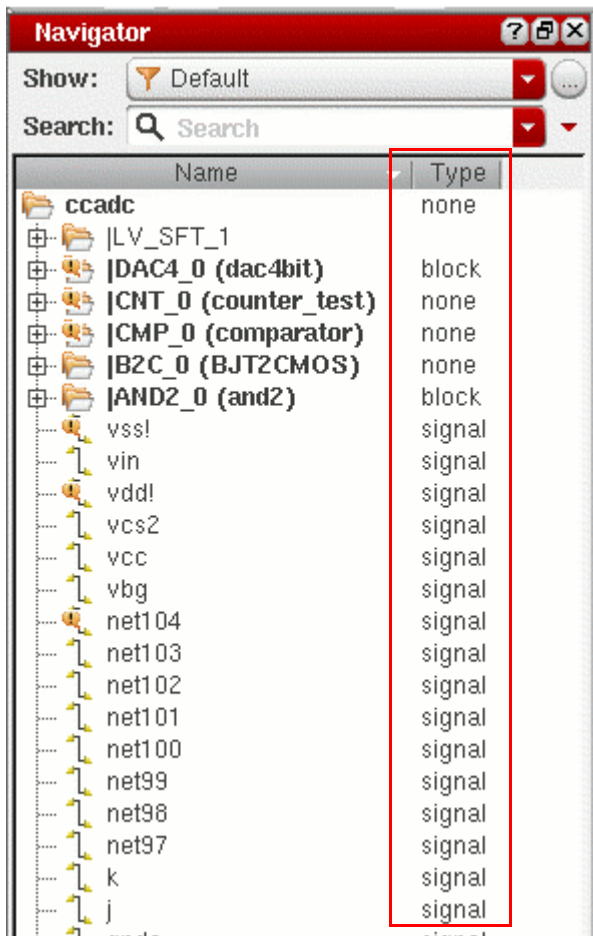
Layout XL Assistants

2. From the context menu, choose *Type*.



The *Type* column is added to the Navigator Assistant. This shows the pin type (for example, *input* or *output*); instance master cell type (for example, *softMacro* or

blockBlackBox) for each instance, and the signal type (for example, *power* or *signal*) for each net in the design.



Note: A signal type, *various*, indicates a bus or bundled net in the schematic for which at least one of the scalar components has a different signal type than the others.

Customizing the Navigator Filters

In addition to using the Navigator for displaying pin, cell, and signal types; you can use the in-built Navigator Assistant filters to display objects of a specific type. For example, if you want to see a list of all the fig groups in the current cellview, you can select the Fig Groups filter. Likewise, to see a list of all the ungenerated objects in the design—objects that are available in the Schematic view but are not yet generated in the layout—you can select the Ungenerated filter. For more information, see [Viewing Ungenerated Instances](#).

The Navigator has several in-built filters, such as *Default*, that include default settings for displaying object types, groupings, and probes. However, you can modify these default filter

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

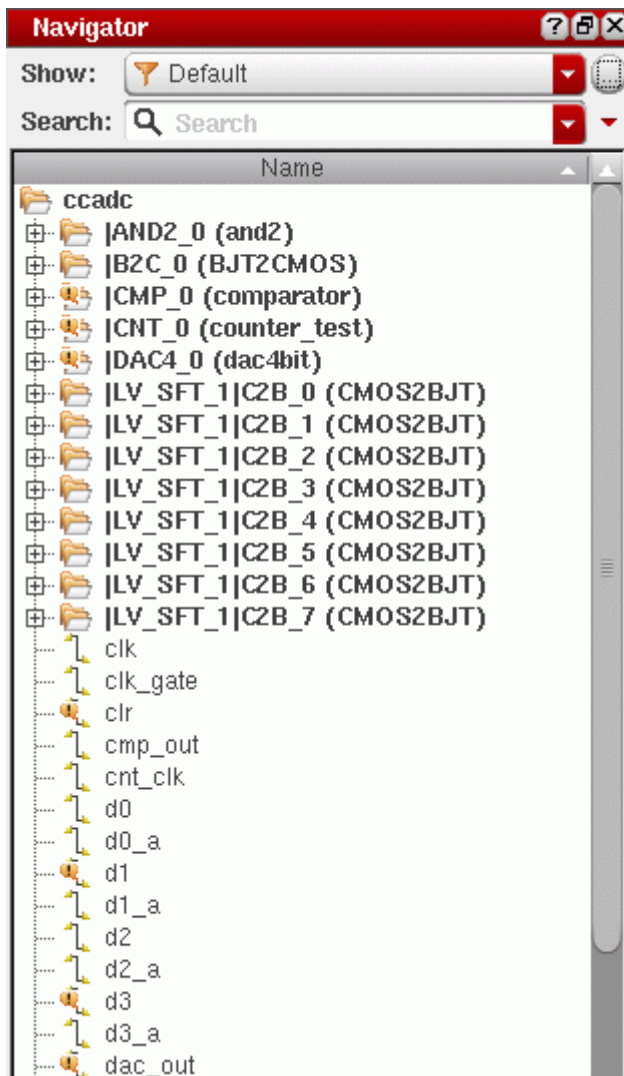
settings by selecting the filter and customizing it. For example, when looking for Instances, you can choose to select the *Instance* filter and customize it to display information based on your preferences.



Video

For a video demonstration on customizing a Navigator filter, see [An Introduction to the Navigator Assistant in VSE XL](#). Note that this video shows the assistant in the context of Schematics XL.

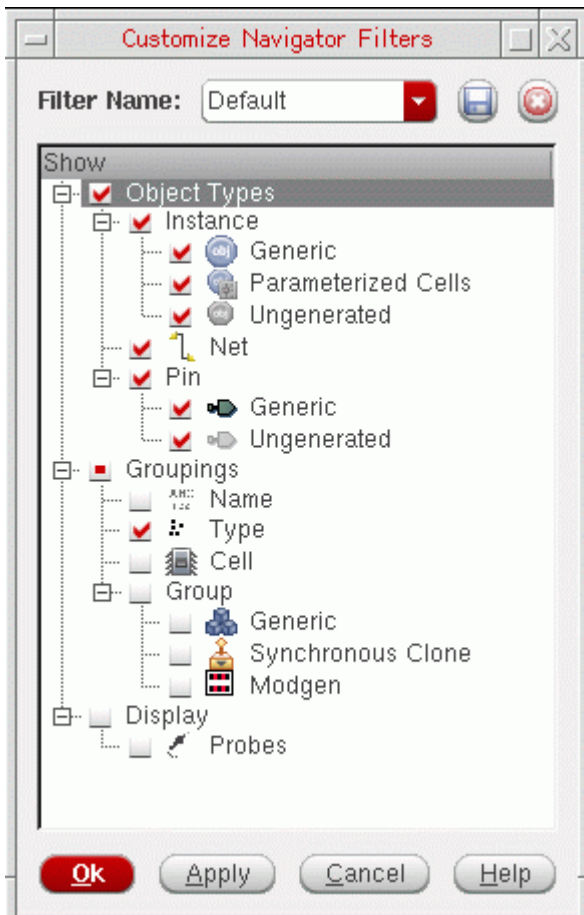
To customize a Navigator Assistant filter:



1. Select a filter to customize. In this case, the selected filter is the **Default** filter.

2. Click the Customize Navigator Filters button (the small ellipsis button to the right of the *Show* list box).

The Customize Navigator Filters form appears.



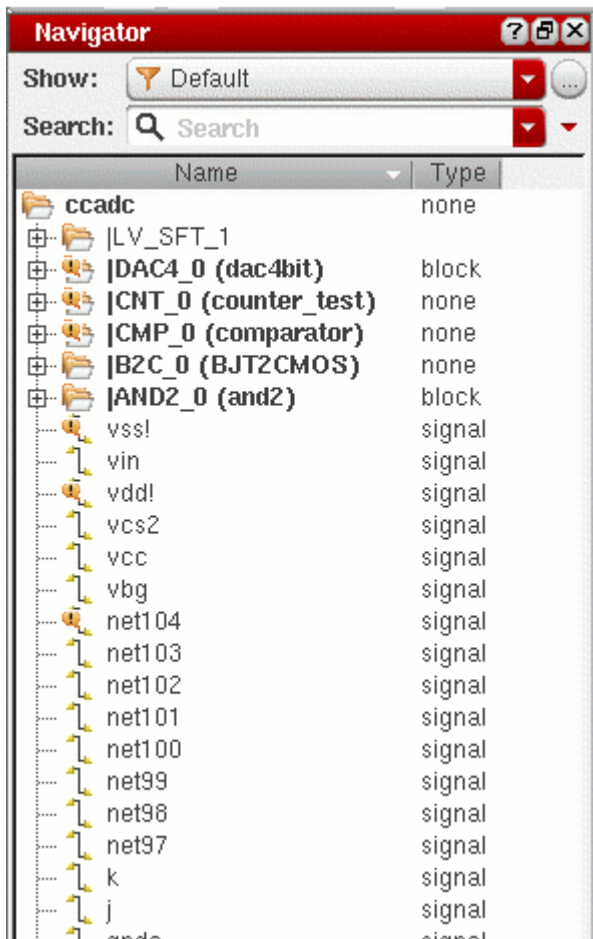
3. Select the appropriate check boxes under the Object Types section to specify the types of objects the selected filter should display.

To display all the three object types—Instance, Net, and Pin—select the **Object Types** check box. Selecting the **Object Types** check box automatically selects all the instance types—Generic, Parameterized, and Ungenerated objects. As a result, the filter lists all the generic, parameterized, and ungenerated objects in the current cellview. For this example, let us select the Object Types check box to display all the object types.

4. Select an appropriate option under **Groupings** to group the objects. The Default filter shows the objects grouped as Generic objects, Synchronous Clones, and Modgens. You can, however, customize the filter to display the objects in the groups of your choice.

For this example, let us choose to group the objects by Type.

5. Select the **Display** check box to display probes for the selected objects. The Default filter has the Display check box selected by default.
6. Click **Ok** to apply the selections to the filter.



The Default filter now displays the objects grouped by Type.

Similarly, you can customize other available search filters to display data based on your preferences. For example, you can use the Pcells filter to customize the way the SKILL Pcells are displayed in the Navigator and view or update their abstraction status using the appropriate shortcut commands.

For more information, see [Using the Context-Sensitive Menu](#).

Viewing Ungenerated Instances

Ungenerated instances are displayed as inactive objects at the end of the object list in the *Default* Navigator filter. The Navigator also provides an *Ungenerated* filter that you can use to view only the ungenerated instances in the design. You can customize the filter in the standard way by using the Customize Navigator Filters form. For more information, see [Customizing the Navigator Filters](#).

The display of ungenerated instances in the Navigator provides you a direct access to all the ungenerated instances in a design. In fact, you can quickly select a set of ungenerated instances in the Navigator and place them in the layout—the operation being similar to a *Pick From Schematic* operation.

You can also select ungenerated instances in the layout Navigator to verify the device correspondence before the physical object and binding is created.

If you select an ungenerated instance in the layout Navigator, the corresponding schematic instance is selected in the schematic Navigator and in the schematic canvas. If you select an m-factored, ungenerated instance in the schematic Navigator or in the canvas, all the corresponding layout instances are selected in the layout Navigator.

To place an ungenerated instance in the layout:

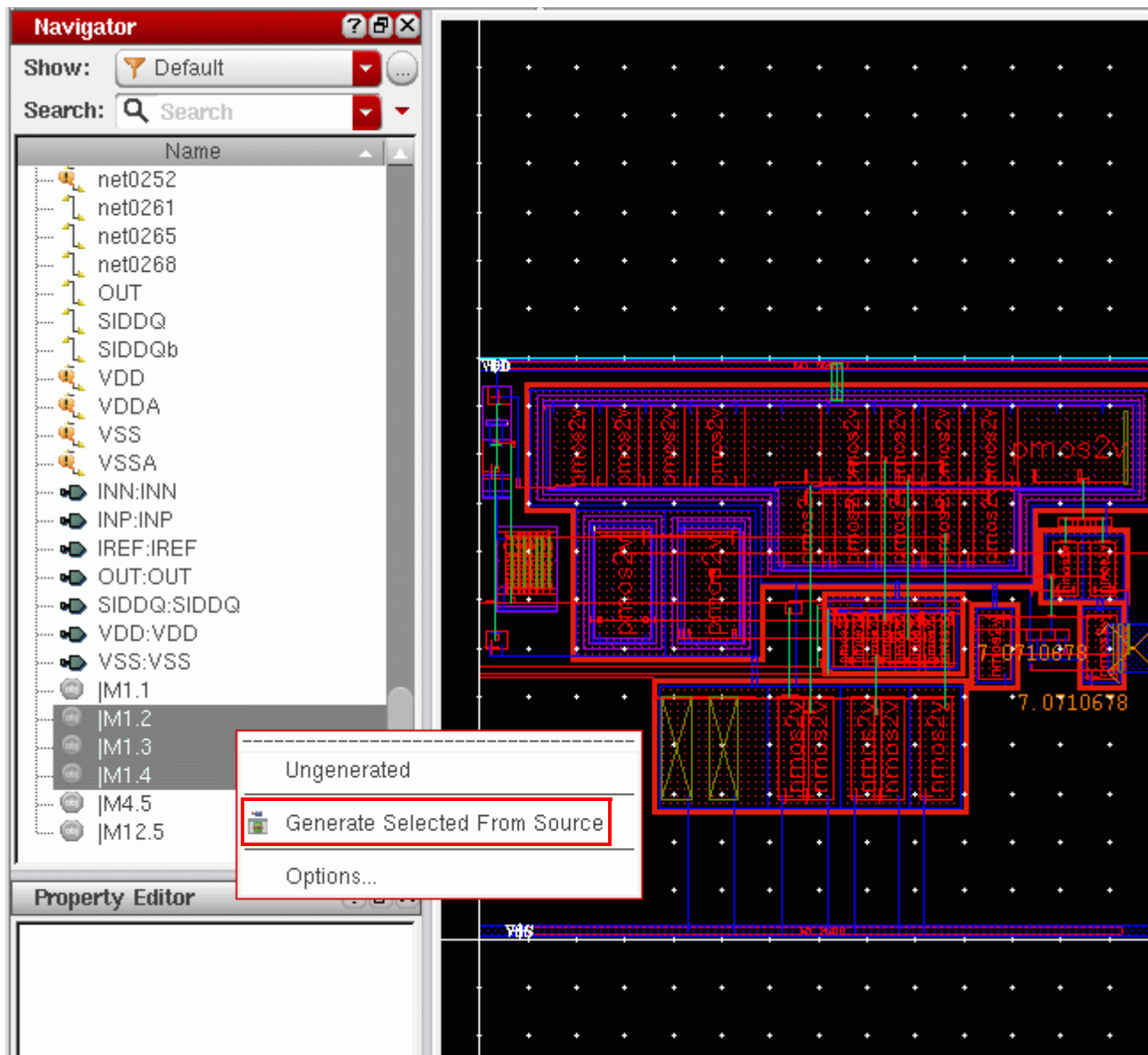
1. Right-click an ungenerated instance in the Navigator and choose *Generate Selected From Source*.

Note: You can also select multiple ungenerated instances from the Navigator and

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

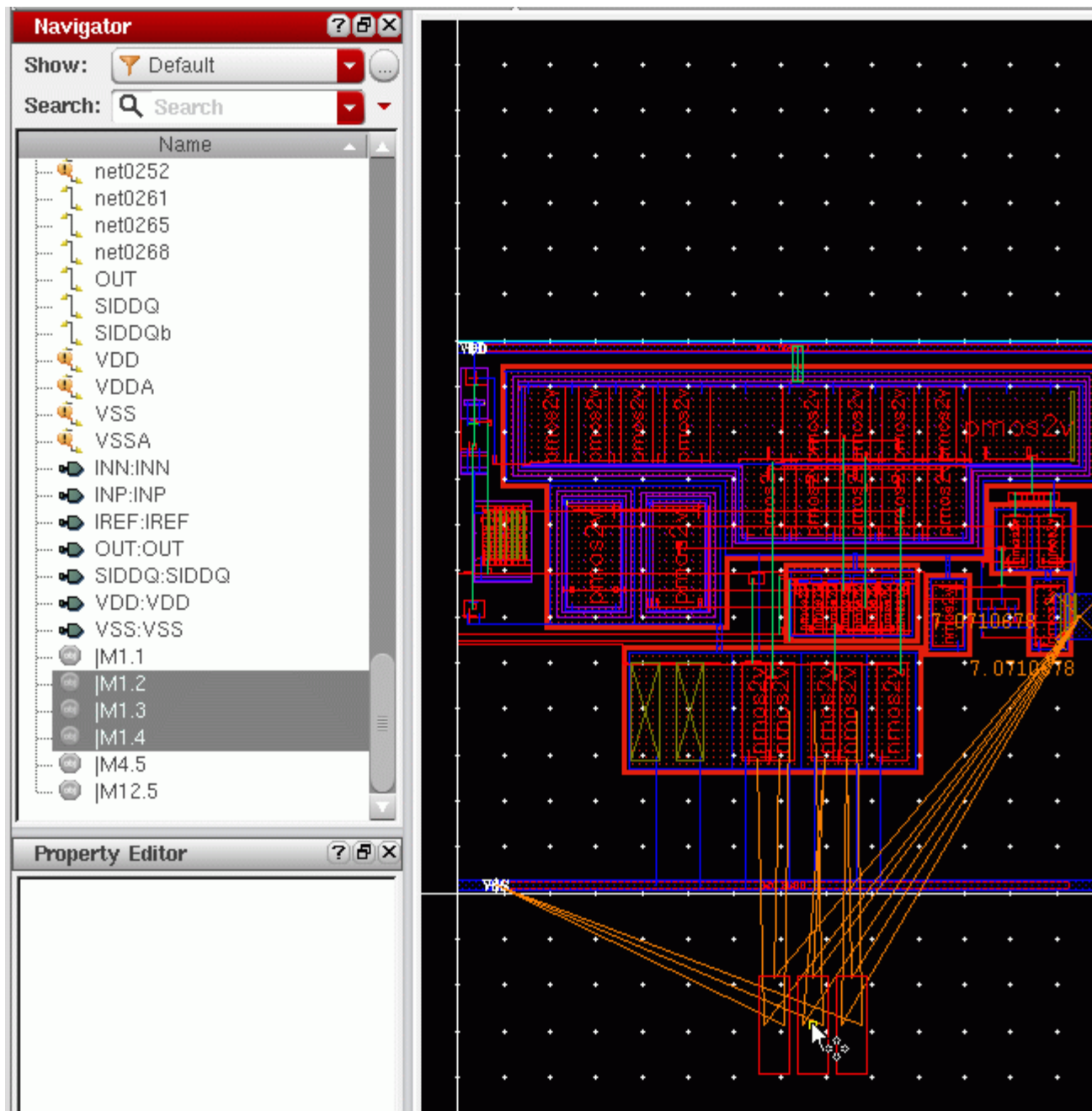
choose *Generate Selected From Source* to place all the instances on the layout canvas.



Virtuoso Layout Suite XL User Guide

Layout XL Assistants

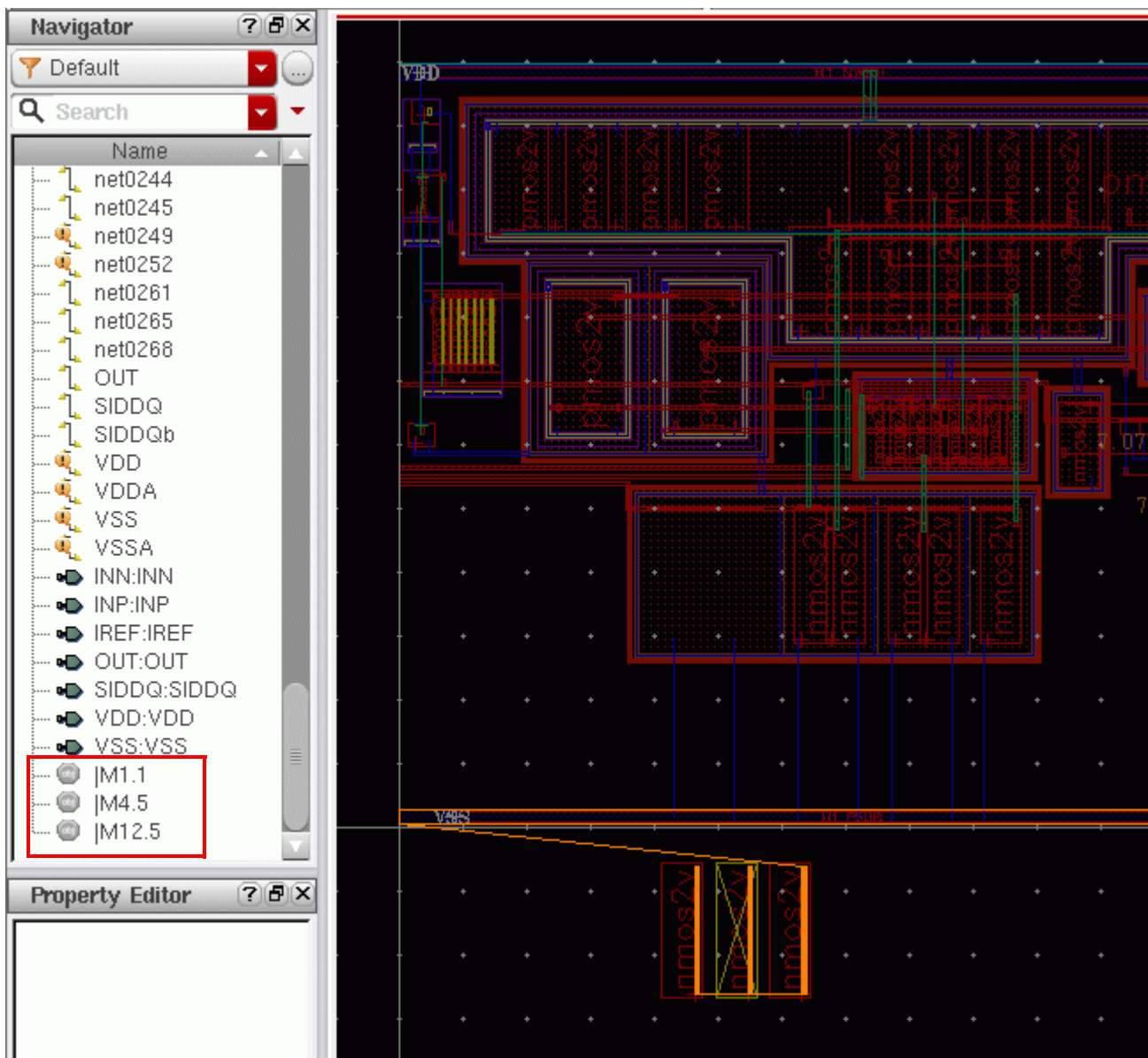
2. Select a location on the layout canvas where you want to place the selected instances.



The selected instances are placed in the layout, as displayed in the figure below, and the list of ungenerated instances in the Navigator is instantly updated to display only those instances that are still ungenerated.

Virtuoso Layout Suite XL User Guide

Layout XL Assistants



Video

For a video demonstration of the feature, see [Placing Ungenerated Instances by using the Navigator](#).

Viewing XL Status

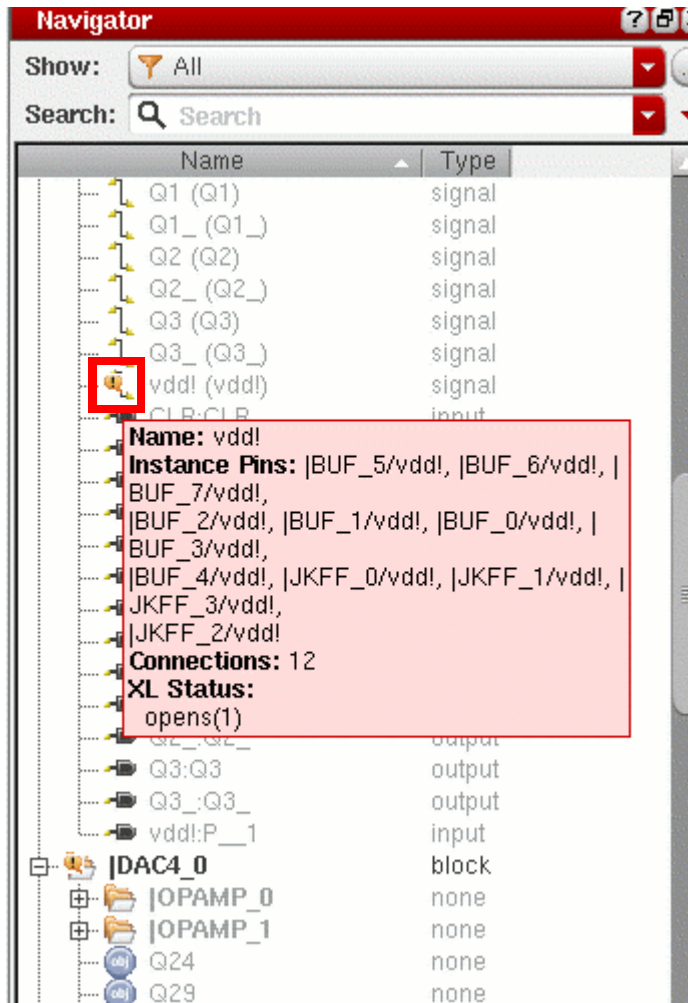
By default, the XL status information is available through an **Attention icon (!)** in the Navigator Assistant. The Attention icon (!) appears in the Navigator alongside those objects

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

that have a design issue. The icon is intended to draw your attention to the issue so that you can take corrective action.

To view the tooltip information associated with an Attention icon, you need to point to the icon. Based on the type of the issue, you are required to take appropriate corrective action. This is when the Attention icon disappears.



For example, an instance displaying an Attention icon may show the XL status in the information balloon as “placed outside prBoundary”. For the status to be converted to OK or Complete and the Attention icon to be removed, you will need to move the instance inside the prBoundary.

Note:

- ❑ For instances and pins with no associated problems, the XL status displays as “OK”.

- ❑ For nets with no opens or shorts, the XL status displays as “Complete”. Else, the status displays the number of opens and shorts.

In addition to being prompted to design issues using the Attention icon, depending on your preference, there are two methods for you to view the XL Status information for an object. These include using the:

- **Navigator column** in the assistant. See [Viewing XL Status Using the Navigator Assistant Column](#).
- **Tooltip/balloon** information on the canvas. See [Viewing XL Status using the Canvas Balloon](#).

Both these features are intended to draw your attention to the objects that have an issue and need your intervention. Although both the features provide the same information, they are available to cater to the varying work preferences of users.

Viewing XL Status Using the Navigator Assistant Column

The advantage of using the XL Status column in the Navigator Assistant is that you get to view the XL Status of all your design objects in one go. You can further use the Search feature to filter out the objects with the same status.

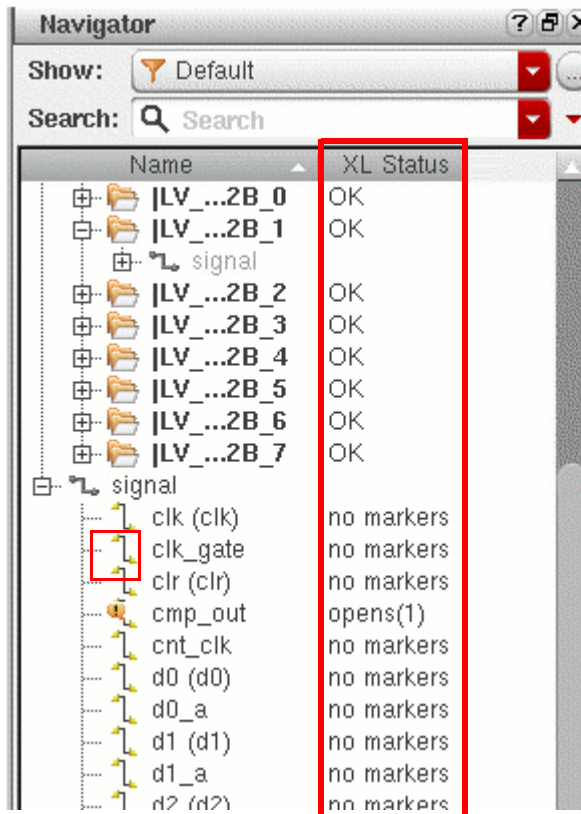
To view the XL Status in the Navigator Assistant Column:

1. With your cursor in the *Name* column in the Navigator Assistant, click the right mouse button.
2. From the pop-up menu, select *XL Status*.

Virtuoso Layout Suite XL User Guide

Layout XL Assistants

The *XL Status* column is added to the Navigator Assistant. This shows the XL status information for each instance, pin, and net.



When viewing in the Navigator Assistant, the XL Status information for the various design objects can be:

Object	XL Status
<i>Instance</i>	unbound parameter mismatch wrong master connectivity difference placed outside prBoundary
<i>Shape</i>	shorts(N) opens(M)
<i>Pin</i>	unbound

placed outside prBoundary

Viewing XL Status using the Canvas Balloon

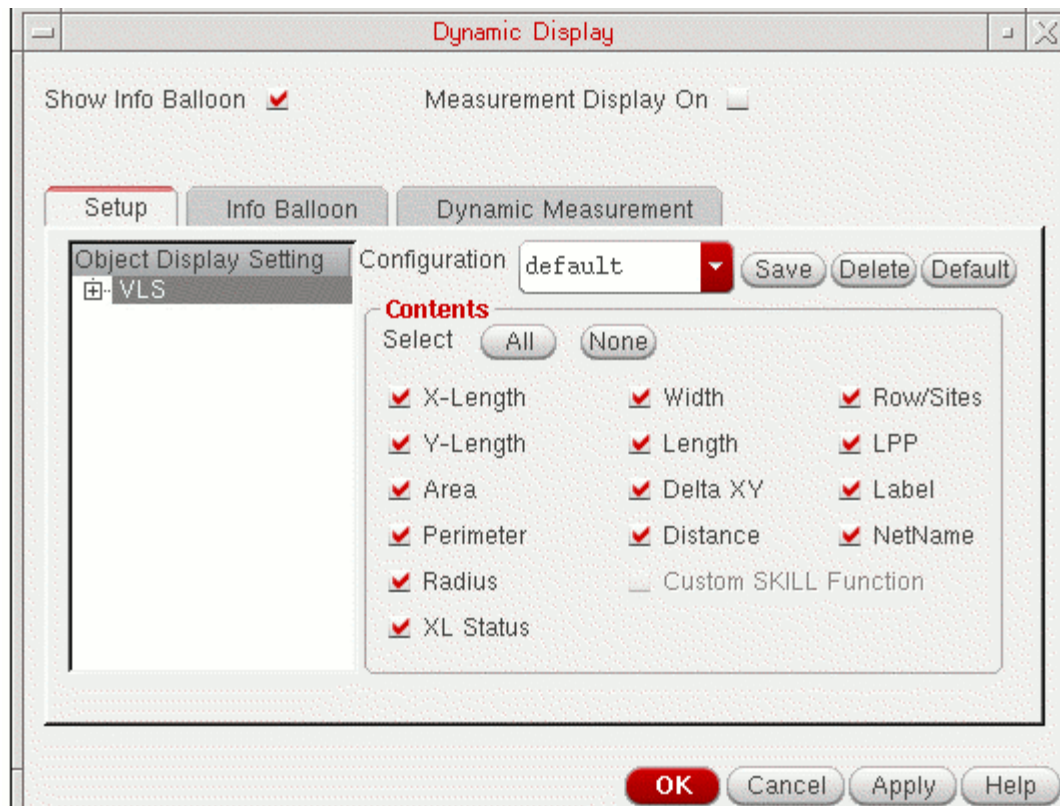
If you prefer looking up the XL Status information on the canvas, instead of using the Navigator Assistant; you can do so by pointing over the object for which you want to look up the information. The advantage of using the balloon info to view XL Status is that you can get the required information straight on the canvas without having to pull up the Navigator Assistant.

To view the XL Status information on the canvas:

1. Select the **Options – Dynamic Display** command.

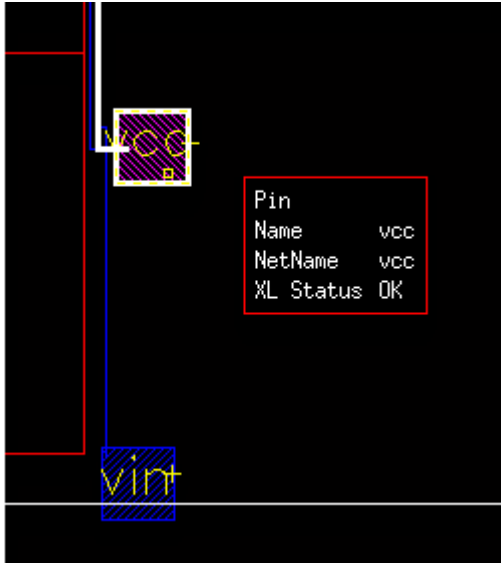
The Dynamic Display form appears.

2. Select the **Show Info Balloon** check_box.
3. In the Setup tab, select the **XL Status** check box.



4. Click **OK**.

Pointing over a design object pops up the associated information balloon displaying the XL status information.



Note:

- ❑ The XL Status information is displayed on the canvas only for instances, shapes, and pins. However, if a shape is attached to a net, the balloon also displays information about that net.

When viewing on the canvas, the XL Status information for the various design objects can be:

Object	XL Status
<i>Instance</i>	unbound
	parameter mismatch
	wrong master
	connectivity difference
	placed outside prBoundary
<i>Shape</i>	shorts(N)
	opens(N)
<i>Pin</i>	unbound
	placed outside prBoundary

Related Topics

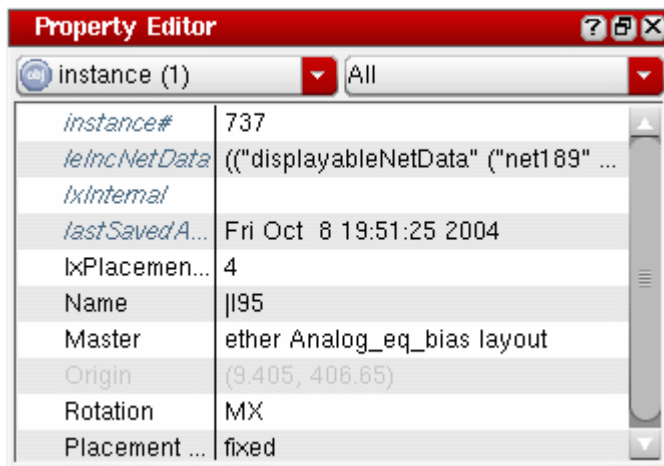
[The Navigator Assistant](#)

[Selecting and Deselecting Nets](#)

Property Editor

Note: For detailed information on the Property Editor, see [The Property Editor Assistant](#) in the *Virtuoso Schematic Editor XL User Guide*.

The **Property Editor** assistant lets you view and edit object property values, such as database attributes, CDF parameters, or user-defined database properties, on one or more components in your design. By default, each object property is displayed in a separate table row in the Property Editor.



Video

For a video demonstration of the Property Editor assistant, see [An Introduction to the Property Editor Assistant](#). Note that this video shows the assistant in the context of Schematics XL.

Related Topics

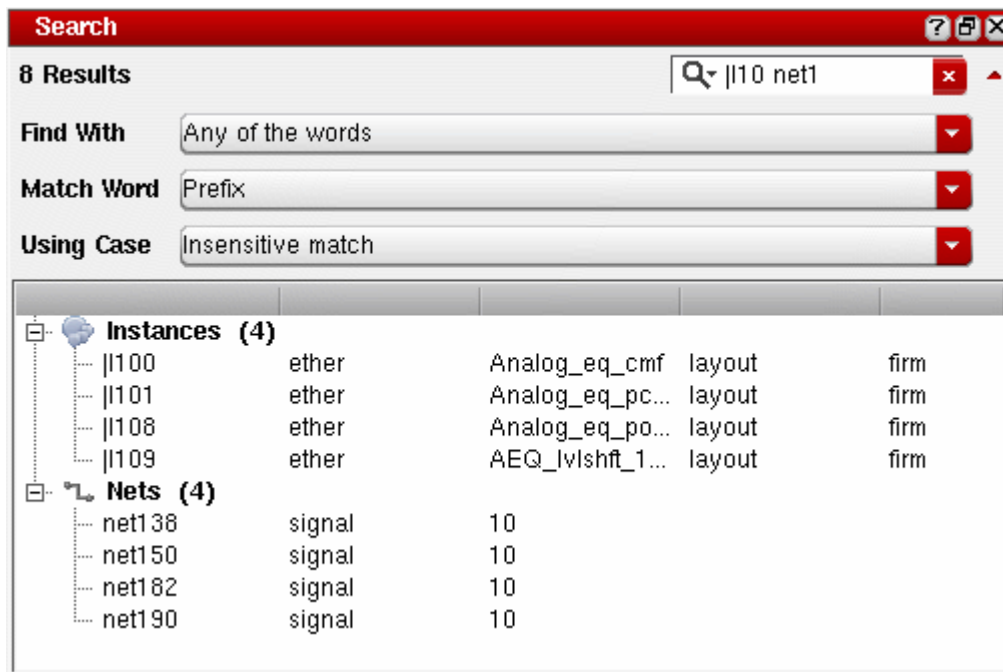
[The Property Editor Assistant](#)

[Editing Net Attributes and Properties](#)

Search

Note: For detailed information on the Search assistant, see [The Search Assistant and Toolbar](#) in the *Virtuoso Schematic Editor XL User Guide*.

The **Search** assistant and toolbar provide a wide range of design search facilities, including dynamic context search categories, fast, keyword-directed searching with immediate feedback, and an intuitive user interface that makes searching for design data simple and productive.



Search uses Virtuoso platform data structures and commands, and presents them through an interactive user interface allowing you to iteratively refine your queries based on the results fed back. This facilitates searching within search results in highly-structured data; often already represented in accessible data formats and generally containing unique identifiers for netlisting and back annotation purposes.

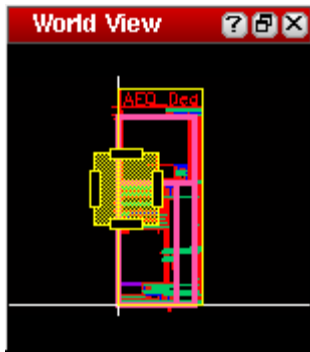
Related Topics

[The Search Assistant and Toolbar](#)

World View

Note: For detailed information on the World View assistant, see [The World View Window](#) in the *Virtuoso Layout Suite L User Guide*.

The **World View** assistant is a navigation tool especially useful in large designs. It shows you a complete picture of your entire design and marks that part of the design that is currently displayed in the drawing area.



Related Topics

[The World View Window](#)