

# **Virtuoso Placer User Guide**

**Product Version ICADVM20.1  
October 2020**

© 2020 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:

The publication may be used only in accordance with a written agreement between Cadence and its customer.

The publication may not be modified in any way.

Any authorized copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement.

The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration.

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

**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>	7
<u>Scope</u>	7
<u>Licensing Requirements</u>	8
<u>Related Documentation</u>	8
<u>Virtuoso Tools</u>	8
<u>Installation, Environment, and Infrastructure</u>	8
<u>Technology Information</u>	8
<u>Virtuoso Tools</u>	9
<u>Additional Learning Resources</u>	10
<u>Video Library</u>	10
<u>Virtuoso Videos Book</u>	10
<u>Rapid Adoption Kits</u>	10
<u>Help and Support Facilities</u>	10
<u>Customer Support</u>	11
<u>Feedback about Documentation</u>	12
<u>Typographic and Syntax Conventions</u>	13
<u>1</u>	
<u>Getting Started with Virtuoso Placer</u>	15
<u>Virtuoso Placer Flow</u>	16
<u>Virtuoso Placer Commands</u>	18
<u>2</u>	
<u>Creating Rows</u>	21
<u>Introducing the Row Infrastructure</u>	22
<u>Row Attributes</u>	22
<u>Row Templates</u>	23
<u>Row Template Manager</u>	25
<u>Creating and Editing a Row Template</u>	27
<u>Importing a Row Template from Another Cellview</u>	40

## Virtuoso Placer User Guide

---

<u>Deleting a Row Template</u> .....	41
<u>Layout Support for Row Regions</u> .....	42
<u>Editing Support for Row Regions</u> .....	47
<u>Importing Row Regions</u> .....	48

### 3

<u>Performing Design Placement</u> .....	49
<u>Setting Placement Options</u> .....	50
<u>Assisted Placement Options</u> .....	51
<u>Automatic Placement Options</u> .....	59
<u>Common Options</u> .....	62
<u>Running the Automatic Placer</u> .....	65
<u>Viewing the Placement Report</u> .....	70
<u>Performing Assisted Placement</u> .....	72
<u>Snapping Behavior</u> .....	75
<u>Support for Module Generators</u> .....	76
<u>Supported Properties</u> .....	79
<u>leSnapGridHorizontal</u> .....	79
<u>leSnapGridVertical</u> .....	80
<u>leSnapPatternSnapping</u> .....	80
<u>leSnapPatternSnappingHorizontalGrid</u> .....	81
<u>leSnapPatternSnappingVerticalGrid</u> .....	81

### 4

<u>Placement Post-Processing</u> .....	83
<u>Insert/Delete Dummy Fill</u> .....	84
<u>Prerequisites</u> .....	84
<u>Types of Fill</u> .....	85
<u>Inserting Fill</u> .....	88
<u>Adding New Fill Definitions</u> .....	90
<u>Verify Placement</u> .....	92

## A

<b>Virtuoso Placer Forms</b>	95
<u>Virtuoso Placer Forms:</u>	95
<u>Automatic Placement</u>	96
<u>Batch Checker (Placement Tab)</u>	97
<u>Placement Options</u>	99
<u>Row Template Manager</u>	103

## B

<b>Virtuoso Placer Environment Variables</b>	107
<u>Virtuoso Placer Environment Variables</u>	110
<u>abutDevices</u>	110
<u>adjustBoundary</u>	111
<u>effort</u>	112
<u>likeSchematicTolerance</u>	113
<u>passiveOutside</u>	114
<u>pinsOnBoundary</u>	115
<u>placePOverN</u>	116
<u>placerType</u>	117
<u>Assisted Placement Environment Variables</u>	118
<u>allowRotationFigGroup</u>	118
<u>apMove</u>	119
<u>displayResolveOverlapsColor</u>	120
<u>displaySnappedInfo</u>	121
<u>fixAdjustedForResolveOverlaps</u>	122
<u>highlightSnappableRowsForInst</u>	123
<u>highlightSnappableRowsForProxy</u>	124
<u>infoBalloon</u>	125
<u>insertColor</u>	126
<u>insertDefaultSpreadType</u>	127
<u>largeModgenEnvelopeCheck</u>	128
<u>largeModgenEnvelopeThreshold</u>	129
<u>maximizePinAccess</u>	130
<u>minNumPinAccessPts</u>	131

## Virtuoso Placer User Guide

---

<u>numSnappableRows</u>	132
<u>paddingForCells</u>	133
<u>pinDensityAwarePlacement</u>	134
<u>regenModgenPostProcess</u>	135
<u>snapColor</u>	136
<u>snapFigGroupRowCorrect</u>	137
<u>spreadEnabled</u>	138
<u>spreadType</u>	139
<u>swapRowMode</u>	140
<u>useDeviceOrder</u>	141
<u>useLayoutAsSeed</u>	142
<u>useLEReference</u>	143
<u>WSPAware</u>	144
<u>Dummy Fill Insertion Environment Variables</u>	145
<u>lobFillAsMosaic</u>	145
<u>lobDevFillPhysOnly</u>	146
<b><u>Placement Checker Environment Variables</u></b>	147
<u>constraints</u>	147
<u>rowOrGrid</u>	148
<u>spacing</u>	149
<u>tapCell</u>	150
<u>padding</u>	151
<u>minPinAccess</u>	152
<u>colorAware</u>	153
<u>reportViolations</u>	154
<u>createMarkers</u>	155
<u>selOnly</u>	156

---

# Preface

---

The Virtuoso® Placer is a unified placement solution available in the Virtuoso Layout EXL cockpit in advanced node releases. It can be used to perform row-based placement of standard cells and devices in analog, digital, and mixed-signal designs.

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

- The Virtuoso design environment and application infrastructure mechanisms designed to support consistent operations between all Cadence® tools.
- The applications used to design and develop integrated circuits in the Virtuoso design environment, notably, the Virtuoso Layout Suite, and Virtuoso Schematic Editor.
- The Virtuoso design environment technology file.
- Component description format (CDF), which lets you create and describe your own components for use with Layout XL.

This preface contains the following topics:

- [Scope](#)
- [Licensing Requirements](#)
- [Related Documentation](#)
- [Additional Learning Resources](#)
- [Customer Support](#)
- [Feedback about Documentation](#)
- [Typographic and Syntax Conventions](#)

## Scope

The functionality described in this guide can be used ONLY in the advanced node ICADVM18.1 release. It is not available in mature node releases.

## Licensing Requirements

You can launch the Virtuoso Placer only from the Layout EXL cockpit. Virtuoso Placer requires the following license to be available in addition to the usual licenses required for the tools you are using:

- 95511: Virtuoso Advanced Node Options for Layout

All Virtuoso Placer features require the Virtuoso\_Layout\_Suite\_EXL license combined with 24 GXL tokens.

For more information on the license checkout requirements for the EXL flow, see [Virtuoso Software Licensing and Configuration Guide](#).

## Related Documentation

### Virtuoso Tools

#### What's New

- [Virtuoso Placer What's New](#)

### Installation, Environment, and Infrastructure

- [Cadence Installation Guide](#)
- [Virtuoso Design Environment User Guide](#)
- [Virtuoso Design Environment SKILL Reference](#)
- [Cadence Application Infrastructure User Guide](#)

### Technology Information

- [Virtuoso Technology Data User Guide](#)
- [Virtuoso Technology Data ASCII Files Reference](#)
- [Virtuoso Technology Data SKILL Reference](#)
- [Virtuoso Technology Data Constraints Reference](#)



## **Virtuoso Tools**

### **IC6.1.8 Only**

- [\*Virtuoso Layout Suite L User Guide\*](#)
- [\*Virtuoso Layout Suite XL User Guide\*](#)
- [\*Virtuoso Layout Suite GXL Reference\*](#)

### **ICADVM18.1 Only**

- [\*Virtuoso Layout Viewer User Guide\*](#)
- [\*Virtuoso Layout Suite XL: Basic Editing User Guide\*](#)
- [\*Virtuoso Layout Suite XL: Connectivity Driven Editing Guide\*](#)
- [\*Virtuoso Layout Suite EXL Reference\*](#)
- [\*Virtuoso Concurrent Layout Editing User Guide\*](#)
- [\*Virtuoso Design Planner User Guide\*](#)
- [\*Virtuoso Multi-Patterning Technology User Guide\*](#)
- [\*Virtuoso Width Spacing Patterns User Guide\*](#)

### **IC6.1.8 and ICADVM18.1**

- [\*Virtuoso Abstract Generator User Guide\*](#)
- [\*Virtuoso Custom Digital Placer User Guide\*](#)
- [\*Virtuoso Floorplanner User Guide\*](#)
- [\*Virtuoso Layout Suite SKILL Reference\*](#)
- [\*Virtuoso Module Generator User Guide\*](#)
- [\*Virtuoso Parameterized Cell Reference\*](#)
- [\*Virtuoso Space-based Router User Guide\*](#)

## Additional Learning Resources

### Video Library

The [Video Library](#) on the Cadence Online Support website provides a comprehensive list of videos on various Cadence products.

To view a list of videos related to a specific product, you can use the *Filter Results* feature available in the pane on the left. For example, click the *Virtuoso Layout Suite* product link to view a list of videos available for the product.

You can also save your product preferences in the Product Selection form, which opens when you click the *Edit* icon located next to *My Products*.

### Virtuoso Videos Book

You can access certain videos directly from Cadence Help. To learn more about this feature and to access the list of available videos, see [Virtuoso Videos](#).

### Rapid Adoption Kits

Cadence provides a number of [Rapid Adoption Kits](#) that demonstrate how to use Virtuoso applications in your design flows. These kits contain design databases and instructions on how to run the design flow.

For specific information about the courses available in your region, visit [Cadence Training](#) or write to [training\\_enroll@cadence.com](mailto:training_enroll@cadence.com).

**Note:** The links in this section open in a separate web browser window when clicked in Cadence Help.

### Help and Support Facilities

Virtuoso offers several built-in features to let you access help and support directly from the software.

- The Virtuoso *Help* menu provides consistent help system access across Virtuoso tools and applications. The standard Virtuoso *Help* menu lets you access the most useful help and support resources from the Cadence support and corporate websites directly from the CIW or any Virtuoso application.

- The Virtuoso Welcome Page is a self-help launch pad offering access to a host of useful knowledge resources, including quick links to content available within the Virtuoso installation as well as to other popular online content.

The Welcome Page is displayed by default when you open Cadence Help in standalone mode from a Virtuoso installation. You can also access it at any time by selecting *Help – Virtuoso Documentation Library* from any application window, or by clicking the *Home* button on the Cadence Help toolbar (provided you have not set a custom home page).

For more information, see [Getting Help](#) in *Virtuoso Design Environment User Guide*.

## Customer Support

For assistance with Cadence products:

- Contact Cadence Customer Support

Cadence is committed to keeping your design teams productive by providing answers to technical questions and to any queries about the latest software updates and training needs. For more information, visit <https://www.cadence.com/support>.

- Log on to Cadence Online Support

Customers with a maintenance contract with Cadence can obtain the latest information about various tools at <https://support.cadence.com>.

## **Feedback about Documentation**

You can contact Cadence Customer Support to open a service request if you:

- Find erroneous information in a product manual
- Cannot find in a product manual the information you are looking for
- Face an issue while accessing documentation by using Cadence Help

You can also submit feedback by using the following methods:

- In the Cadence Help window, click the *Feedback* button and follow instructions.
- On the Cadence Online Support [Product Manuals](#) page, select the required product and submit your feedback by using the *Provide Feedback* box.

## Typographic and Syntax Conventions

The following typographic and syntax conventions are used in this manual.

<i>text</i>	Indicates names of manuals, menu commands, buttons, and fields.
text	Indicates text that you must type exactly as presented. Typically used to denote command, function, routine, or argument names that must be typed literally.
<i>z_argument</i>	Indicates text that you must replace with an appropriate argument value. The prefix (in this example, <i>z_</i> ) indicates the data type the argument can accept and must not be typed.
	Separates a choice of options.
{ }	Encloses a list of choices, separated by vertical bars, from which you <b>must</b> choose one.
[ ]	Encloses an optional argument or a list of choices separated by vertical bars, from which you <b>may</b> choose one.
[ ?argName t_arg ]	Denotes a <i>key argument</i> . The question mark and argument name must be typed as they appear in the syntax and must be followed by the required value for that argument.
...	Indicates that you can repeat the previous argument.
	Used with brackets to indicate that you can specify zero or more arguments.
	Used without brackets to indicate that you must specify at least one argument.
, ...	Indicates that multiple arguments must be separated by commas.
=>	Indicates the values returned by a Cadence® SKILL® language function.
/	Separates the values that can be returned by a Cadence SKILL language function.

If a command-line or SKILL expression is too long to fit within the paragraph margins of this document, the remainder of the expression is moved to the next line and indented. In code excerpts, a backslash ( \ ) indicates that the current line continues on to the next line.

# **Virtuoso Placer User Guide**

## Preface

---

---

# Getting Started with Virtuoso Placer

---

Virtuoso<sup>®</sup> Placer is a unified placement solution that can be used to achieve quick, high-quality, DRC-correct placement of standard cells and devices. It supports placement of different types of designs, such as digital designs with only standard cells, analog designs with only devices, and mixed designs that contain a mixture of both. It can also be used to generate connectivity and constraint-driven device placements.

Placer combines several advanced features such as snap patterns, width spacing patterns, and multi-patterning into a single automated flow. At the core of the Placer is the row template infrastructure that lets you create rows, which, in turn, define how standard cells and devices are placed. Rows, if made compatible with snap patterns, help achieve correct-by-construction placement.

Placer also brings together under its umbrella several existing and new tools that utilize the row template infrastructure. For example, you can use Virtuoso<sup>®</sup> Custom Digital Placer to place custom digital designs and Modgens to construct blocks of analog devices. The new automatic and assisted placement and post-processing tools can be used to fill the design with dummy devices to meet density DRC rules and to achieve better matching.

Placer is available in the Virtuoso Layout EXL cockpit in advanced node releases.



### *Important*

Before using the placer, ensure that the circuit components and connectivity are generated by using the Virtuoso Layout Suite XL layout editor. The placer uses the schematic design as the connectivity source for placing the components.

This chapter includes the following sections:

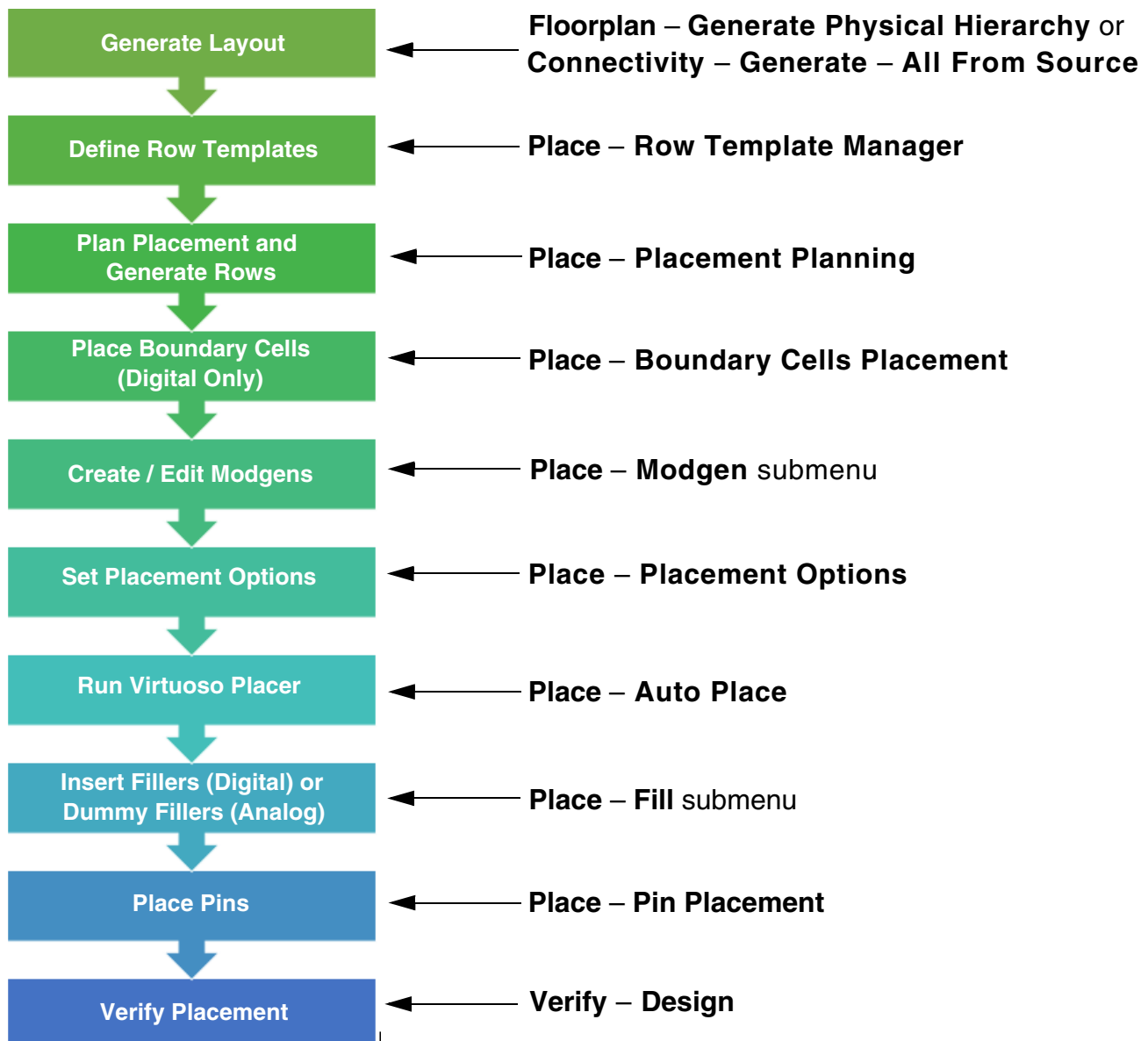
- [Virtuoso Placer Flow](#)
- [Virtuoso Placer Commands](#)

## Virtuoso Placer Flow

Virtuoso Placer is a unified, powerful placement solution. Its key benefits are:

- Accelerated layout productivity for advanced node placement.
- Unified GUIs and use-models between the various placers.
- Improved usability of the set up and placement commands.

The following diagram depicts the Virtuoso Placer flow and maps each step to the corresponding Virtuoso command:



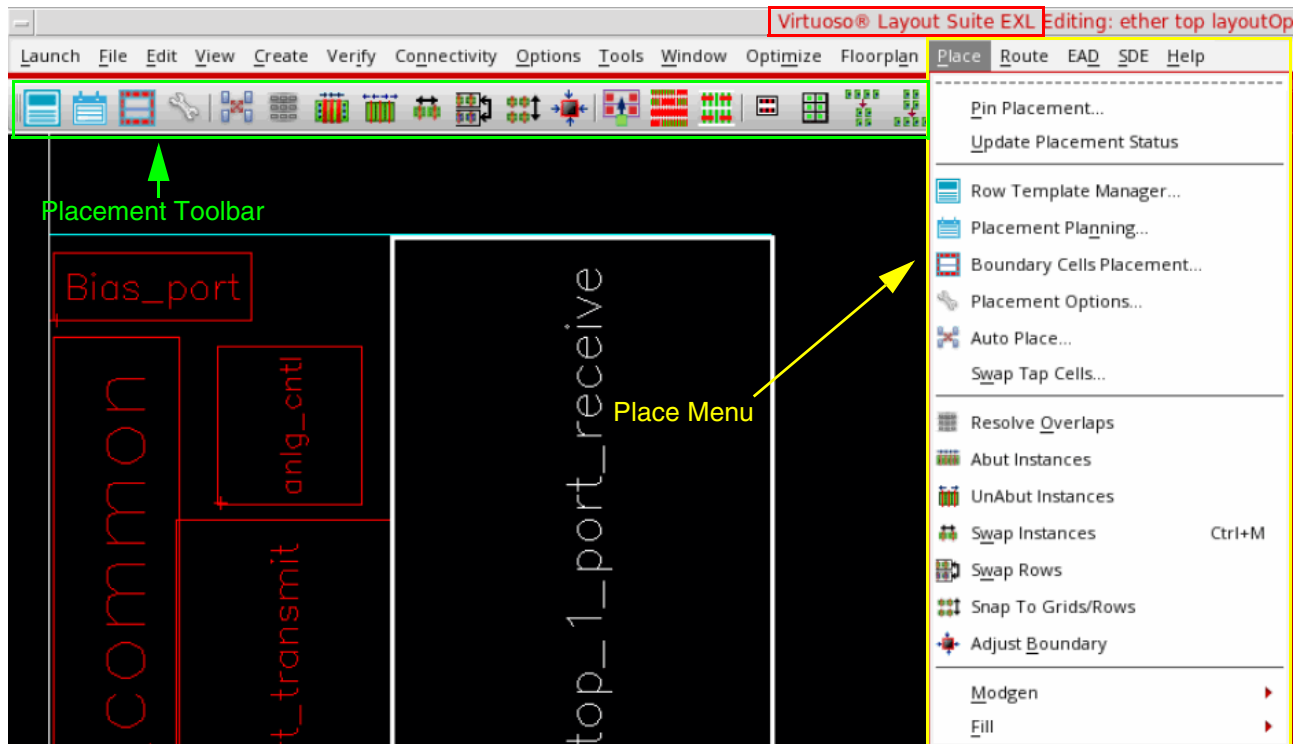


# Virtuoso Placer User Guide

## Getting Started with Virtuoso Placer

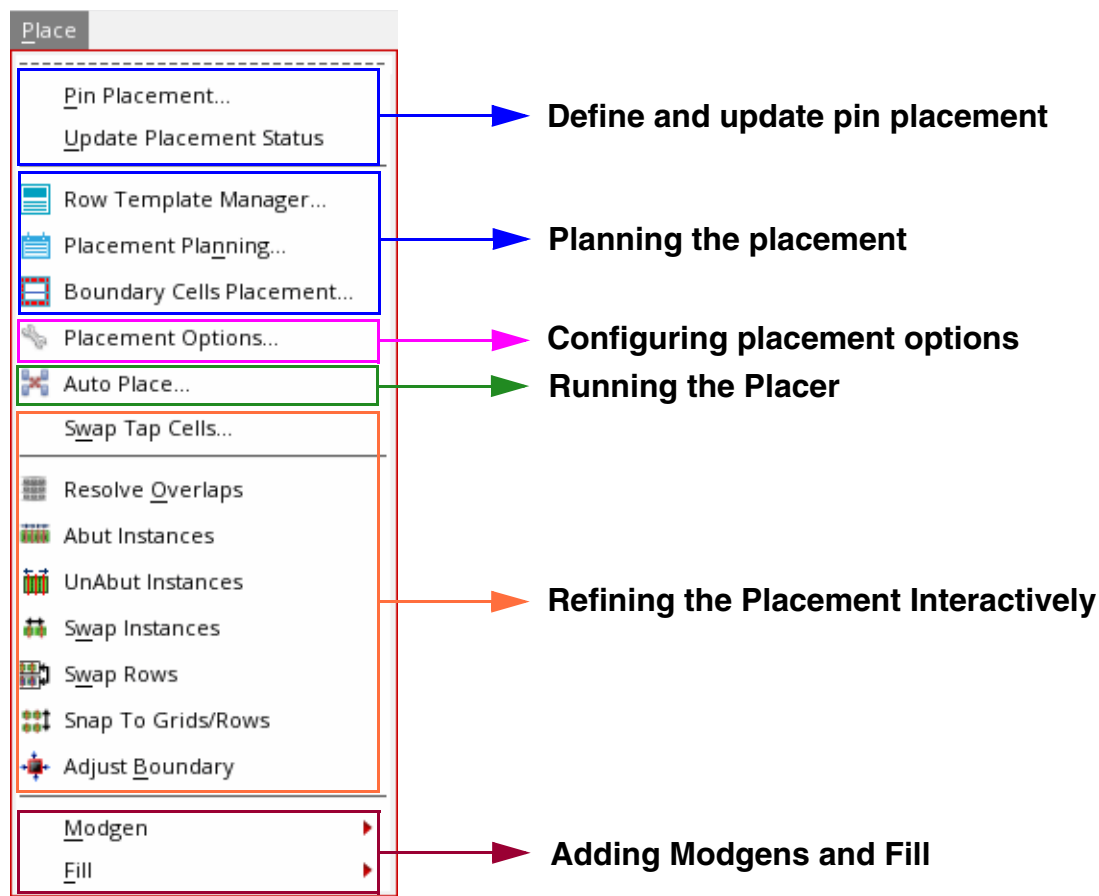
The Virtuoso Placer is available in Layout EXL. To access the Virtuoso Placer commands:

1. Open your schematic or layout design.
2. Choose *Launch - Layout EXL*



## Virtuoso Placer Commands

The following placement commands are available under the *Place* menu:











The following table provides a brief description of the commands in the *Place* menu, their corresponding icons in the Placement toolbar, and a link to related documentation.

Icon	Command Name	Description
	<i>Pin Placement</i>	<p>Opens the Pin Planner, which lets you assign and refine pin constraints and their attributes, and the Pin Optimizer, which positions pins in a manner that helps obtain the shortest possible net length at a particular level in the design. Use this command to preplace pins, independent of the main placement run.</p> <p>For more information, see <a href="#">Pin Planning</a>.</p>







## Virtuoso Placer User Guide

### Getting Started with Virtuoso Placer

	<i>Update Placement Status</i>	Lets you update the placement status of instances or pins that are loaded in Layout XL for the first time, or which have been edited outside the Layout XL or higher tiers.
	<i>Row Template Manager</i>	Lets you define row templates, which can later be used to generate rows in the layout canvas.  For more information, see <a href="#">Creating Rows</a> .
	<i>Placement Planning</i>	Lets you plan the placement region, generate rows in which the components will be placed, and select additional substrates to be inserted during placement.  For more information, see <a href="#">Placement Planning</a> .
	<i>Boundary Cells Placement</i>	Places boundary cells around core cells to protect the core cells from the undesired effects that are caused when adjacent cells are placed within close proximity.  For more information, see <a href="#">Placing Boundary Cells</a> .
	<i>Placement Options</i>	Provides options to customize assisted and automatic placement. This form can also be invoked by clicking More Options in the Automatic Placement form. You can make the required placement settings and then run the Virtuoso Placer.  For more information, see <a href="#">Placement Options</a> .
	<i>Swap Tap Cells</i>	Detects tap cells with maximum spacing violations and replaces them with tap cells that have valid maximum spacing values.  For more information, see <a href="#">Swapping Tap Cells</a> .
	<i>Auto Place</i>	Lets you place devices, standard cells, or a mix of both in rows defined in the layout canvas.  For more information, see <a href="#">Automatic Placement</a> .
	<i>Resolve Overlaps</i>	Resolves device overlaps in rows.  For more information, see <a href="#">Performing Assisted Placement</a> .
	<i>Abut Instances</i>	Abuts selected or all instances in the design.  For more information, see <a href="#">Performing Assisted Placement</a> .
	<i>UnAbut Instances</i>	Unabuts selected or all instances in the design.  For more information, see <a href="#">Performing Assisted Placement</a> .

## Virtuoso Placer User Guide

### Getting Started with Virtuoso Placer

	<i>Swap Instances</i>	Swaps the positions of selected devices. For more information, see <a href="#">Performing Assisted Placement</a> .
	<i>Swap Rows</i>	Swaps the positions of the selected rows. For more information, see <a href="#">Performing Assisted Placement</a> .
	<i>Snap to Grids/Rows</i>	Snaps the selected devices to appropriate grids or rows. For more information, see <a href="#">Performing Assisted Placement</a> .
	<i>Adjust PR Boundary</i>	Adjusts the boundary by deleting any extra rows in which devices are not placed. For more information, see <a href="#">Performing Assisted Placement</a> .
	<i>Modgen</i>	Opens a submenu that provides commands to create and edit Modgen constraints. For more information, see <a href="#">Module Generator User Guide</a> .
	<i>Filler</i>	Opens a submenu that provides options to: <ul style="list-style-type: none"> <li>■ Insert filler cells  - For digital designs</li> <li>■ Insert dummy fill  - For analog designs</li> </ul> For more information, see <a href="#">Insert/Delete Dummy Fill</a> .

---

## Creating Rows

---

Advanced node processes support multiple grid types. As a result, advanced node designs easily adapt to a row-based placement methodology. Though it is not necessary that you create rows for placement, it is recommended because row-based placement improves routability, achieves better wire length after routing, and resolves illegal placements by balancing the number of instances placed in each row. It also helps resolve overlaps between devices in each row.

The Row Template Manager feature provides the capability to define row templates. You can create row templates specific to your design type, and then use these templates to generate rows in the layout canvas. Rows are created in objects called row regions. Row regions reference a row template that determines how rows are created in that region. Rows, in turn, contain component type attributes that determine which instances are placed in that row and the manner in which they are placed.

You can save row templates and reuse them in multiple designs. Row templates can be used to place both analog and digital designs.

This chapter gives a brief introduction about rows and row templates; and describes the procedure to use the Row Template Manager to create, edit, and import row templates.

The topics covered in this chapter are:

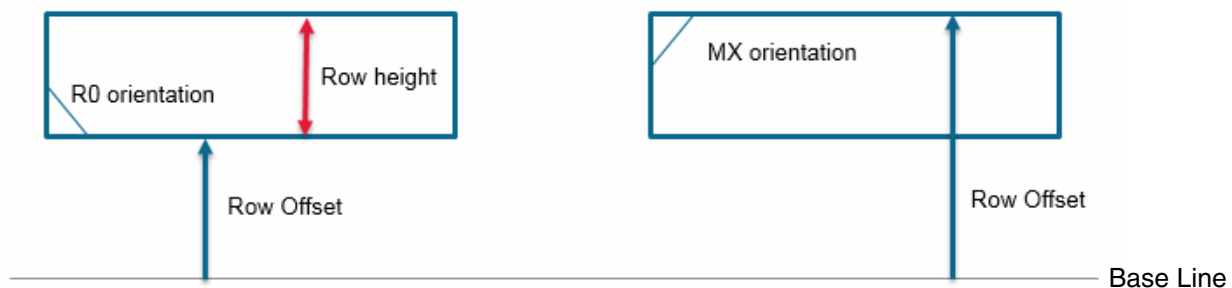
- Introducing the Row Infrastructure
  - Row Attributes
  - Row Templates
- Row Template Manager
  - Creating and Editing a Row Template
  - Importing a Row Template from Another Cellview
- Layout Support for Row Regions

## Introducing the Row Infrastructure

A row represents the location for placing standard cells, macros, or devices. In a layout design, you define rows within the PR boundary. You can then place instances within these rows.

### Row Attributes

The following diagram depicts a row and its attributes:



#### Row Edge

Row edge is defined by the row offset and row height values.

The reference edge is specified by the row offset value. The other edge is at a distance equal to the row height away from the reference edge.

**Row Orientation** Is the orientation of the row with respect to the x and y axes.

The two possible row orientations are: no orientation (R0) and mirrored orientation along the x axis (MX).

If the row orientation is R0, the row height is measured as a positive value from the bottom edge. If the row orientation is MX, the row height is measured from top to bottom.

#### Row Offset

Refers to the distance from the base line to the reference edge.

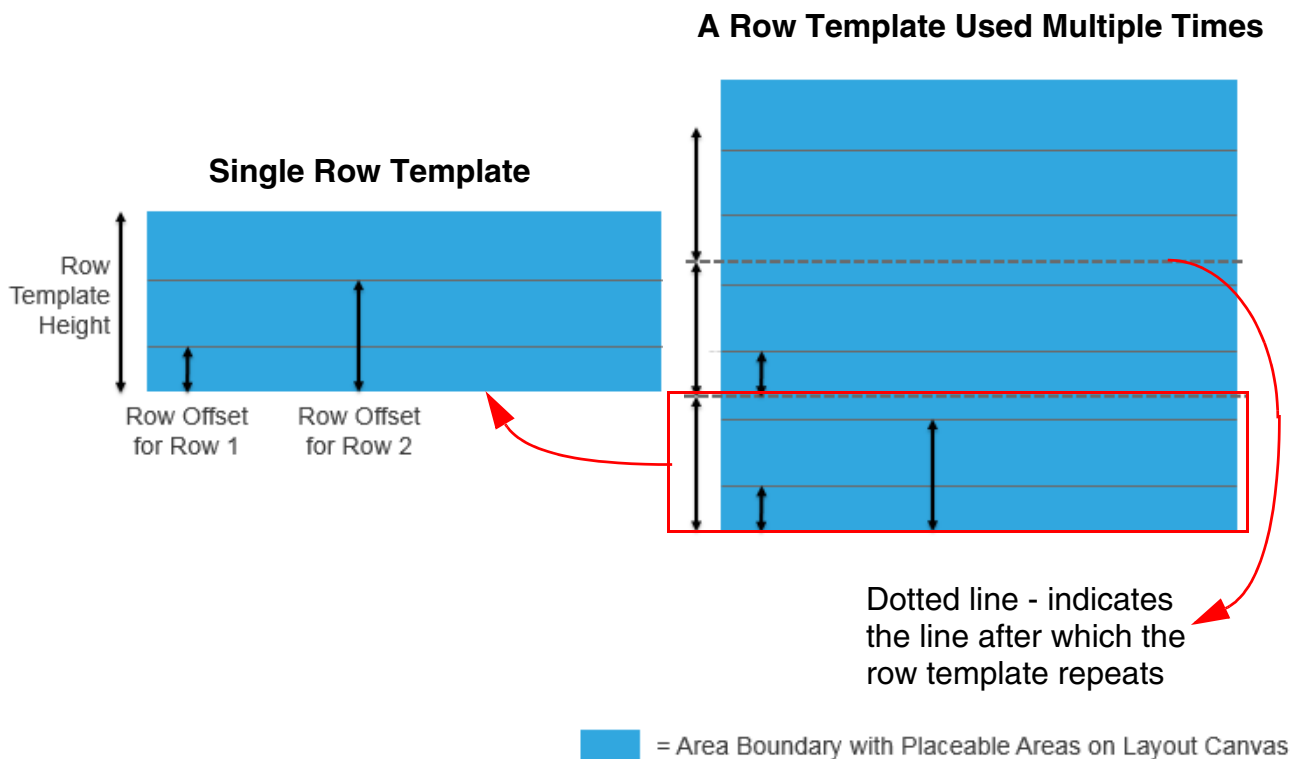
The reference edge depends on the row orientation. When the row orientation is R0, the reference edge is at the bottom and the row offset is the distance from the base line to the bottom edge. However, when the row orientation is MX (mirrored), the reference edge is at the top and the row offset defines the distance from the base line to the top edge, as indicated in the right-hand picture.

#### Row Height

Is the distance between the top and bottom edges of the row.

## Row Templates

A row template is a container that defines one or more rows. It comprises specifications for a set of rows to be generated in the layout canvas. A row template can be repeated multiple times, one on top of the other, based on the required height. The following diagram depicts how a row template can be used multiple times in a design. One instance is zoomed in to view details:



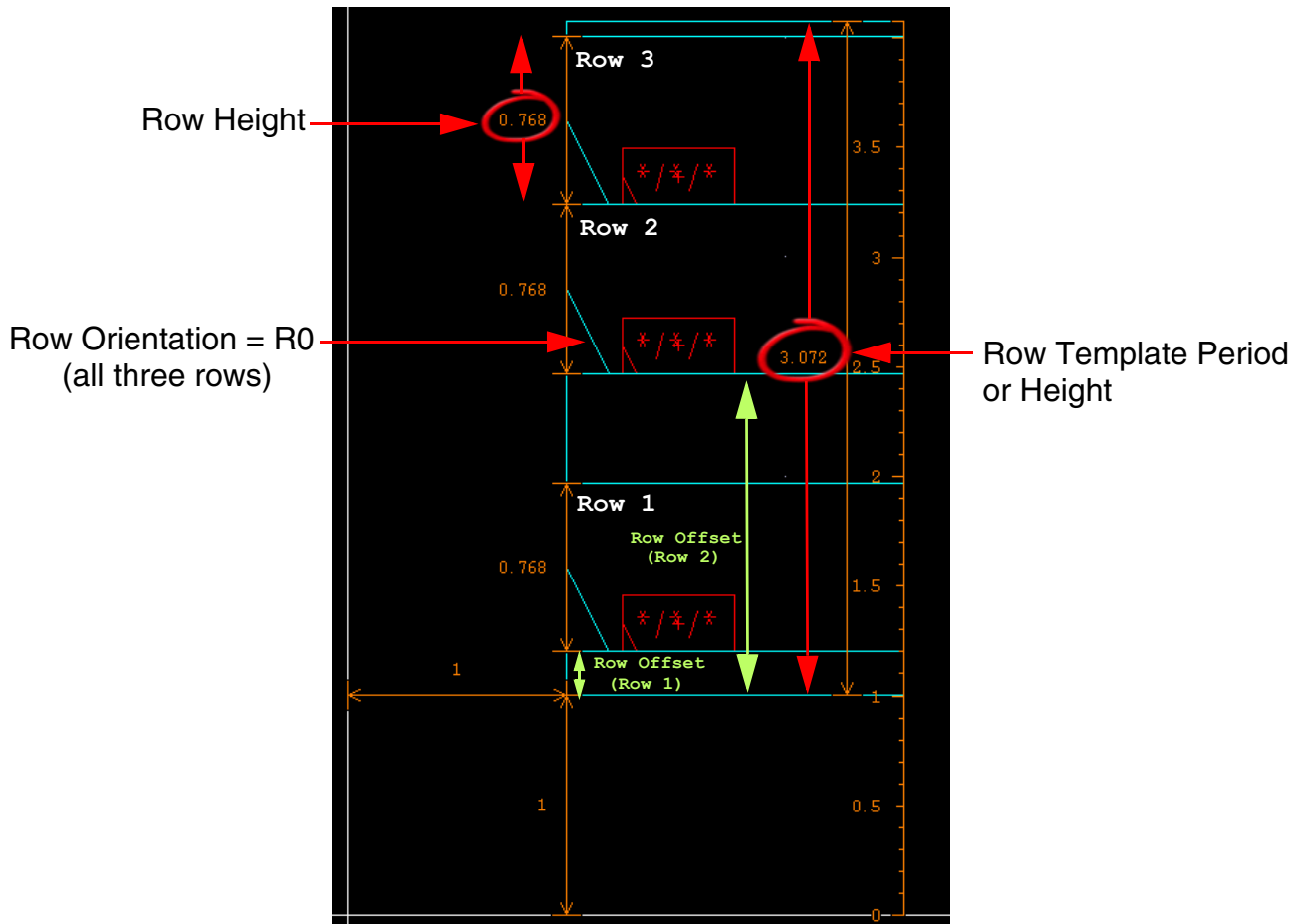
In the above diagram, you can see that:

- The row template period is the total height of the row template, inclusive of all the rows in it.
- Each row has a different row offset value.

## Virtuoso Placer User Guide

### Creating Rows

Here is a snapshot of a row template with three rows, Row 1, Row 2, and Row 3:



Before generating rows in the layout canvas using a row template, ensure that:

- The row template has at least one row.
- All rows have either a row height or a siteDef valued defined.
- All rows fit inside the template period.
- All row have an allowed orientation value set.
- All rows have either a component type or a component master filter, but not both.
- All attributes are specified for generating rails, if required.



## Row Template Manager

Before running Virtuoso Placer, you must have generated rows in the layout canvas.

Row generation involves the following steps:

1. **Defining one or more row templates:** Use the Row Template Manager to define row templates.

Row templates will typically be set up centrally for use with a particular project or process node and it's unlikely that the layout designer will have to edit them.

The Placement Planning form includes the *Row Template* field. This section covers the process of defining and generating rows using the Row Template Manager.

2. **Defining the area where rows should be generated:** Use the *Region* tab of the Placement Planning form to specify the area (*Boundary*) in which the rows must be generated.

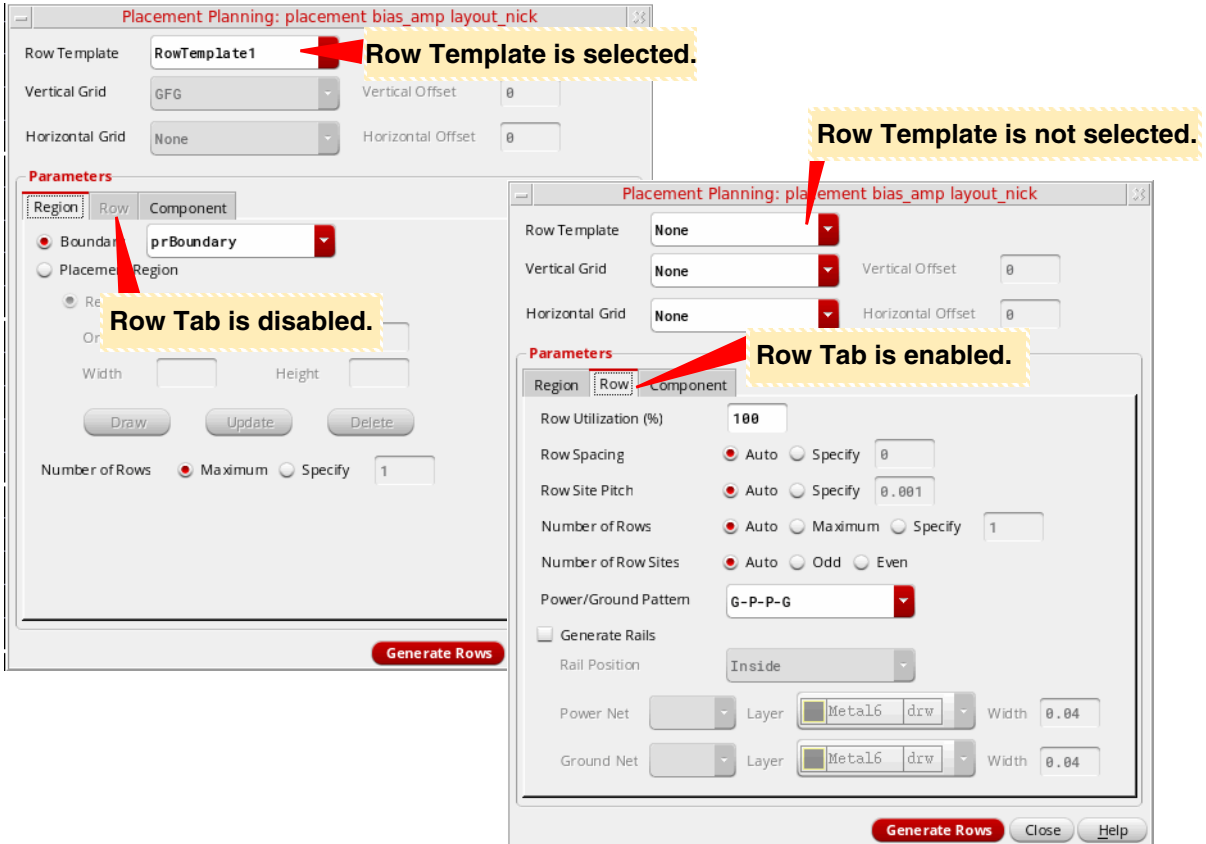
Depending on the area, which could be the PR boundary or a custom placement area, row templates are repeated to accommodate as many rows as possible within the area.

When you select a row template, the *Row* tab is automatically disabled. You cannot edit row attributes.

## Virtuoso Placer User Guide

### Creating Rows

If you do not select a row template, you can use the options on the *Row* tab to define row attributes, as shown in the following picture:



Click *Generate Rows* to generate rows in the layout canvas as per the specifications in the Placement Planning form.

For more information, see [Placement Planning](#) in *Virtuoso Custom Digital Placer User Guide*.

# Virtuoso Placer User Guide

## Creating Rows

### Creating and Editing a Row Template

Use the [Row Template Manager](#) to create and edit row templates. Click *Place – Row Template Manager* to display Row Template Manager.

**Row Template Manager**

**Edit** | **Import**

Template Name: RowTemplate1 New Delete

Template Period: 0.144 3\*0.048 ☒ Auto-compute minimum period

Region Type: Region1

**Placement Grid**

Vertical Grid: GFG Vertical Offset: 0

Horizontal Grid: GPG90 Horizontal Offset: 0

Related Snap Pattern:

**Rows**

☒ Auto-compute row offset

Row Name	Row Offset	Row Orientation	Row Height	Site Def	Background LPP
1 Row1	0	R0	0.048		
2 Row2	0.048	R0	0.048		
3 Row3	0.096	R0	0.048		

**Row Attributes: Row 2**

Component Type Definition

Types	Masters	Orientations	Align Reference	Align Offset	Inst Pitch
1 NFIN		R0	BOTTOM	0	0
2 PFIN		R0	BOTTOM	0	0

Rail Definition

Net	WSP Track	LPP	Width	Align Reference	Align Offset

Create View Close Help

The form has two tabs:

- **Edit:** Use this tab to create new row templates or to edit existing templates. Row templates are stored directly in the layout.
- **Import:** Use this tab to import existing row templates from a different cellview in your design library. The *Import* tab is displayed by default when you open the Row Template

Editor. Once a template is defined, you can reuse it any number of times. Use the options on the *Edit* tab to make further changes to an imported row template. For more information, see [Importing a Row Template from Another Cellview](#).

The tasks involved in creating a new row template are:

- [Defining Row Template Name and Placement Grid](#)
- [Defining a Row](#)
- [Defining Component Types and Rails](#)

### Defining Row Template Name and Placement Grid

With the Row Template Manager open:

1. Click the *Edit* tab.
2. Click *New* to generate a new unique row template name, which, by default, is *RowTemplateN*, where  $N = 1, 2, 3, \dots$ . You can modify this default value to specify a more descriptive name for your row template.



#### Tip

To edit an existing row template, select its name from the drop-down list.



#### Tip

At any point, click *Update* to save changes. Click *View* to preview the row template in a separate window. Whenever you make changes to a row template and click *Update*, the rows in the design and the view in the placement window are dynamically updated.

**Note:** The *Update* and *View* buttons are disabled when there are invalid entries in the table. Incorrect entries are indicated by yellow highlights. You can hover the mouse over the entry to get a detailed explanation of the error.

## Virtuoso Placer User Guide

### Creating Rows

---

3. Specify the row *Template Period*. It is the sum of the heights of all rows in the template and the spacing between these rows. In other words, it is the distance after which the template repeats.



The combo box beside the *Template Period* field specifies the number of rows that can be accommodated in the specified row template period. If you modify the number, then the *Template Period* is adjusted accordingly. For example,  $4 * 0.048$  indicates that the *Template Period* is four times the vertical grid's period (0.048). Therefore, the *Template Period* field is automatically reset to 0.192 to accommodate the vertical grid.

**Note:** To specify the grid period, select the vertical grid (*Ref Y Grid*) first.

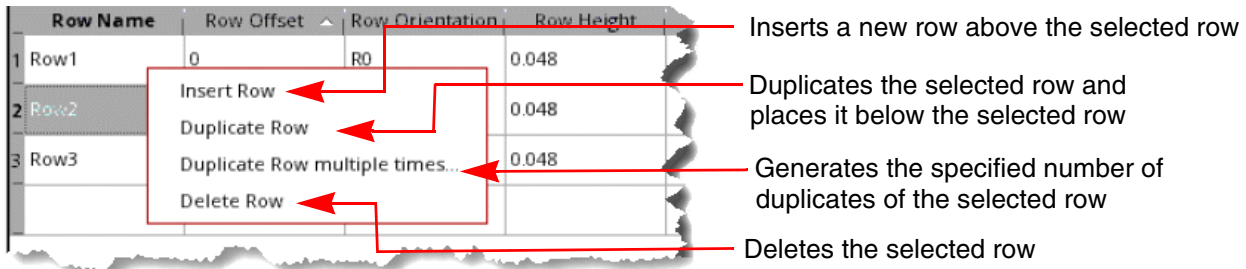
4. Specify the *Region Type*, which is a RowRegionSpec (RRS) type attribute from which settings are inherited. After generating the row region in the layout canvas, Virtuoso can automatically query the RowRegionSpec type attribute, derive the predefined actions, and run these actions on the new row region.
5. The *Placement Grid* section ensures that row placement is compatible with the Width Spacing Pattern/Snap Pattern (WSP/SP) tracks. In this section:
  - a. Optionally, select the vertical and horizontal reference grids (along the x and y axes) for snapping the devices during placement.
  - b. Optionally, specify the vertical and horizontal offset values, which indicate the reference row offset values that need to be applied to rows on each axis after they are snapped to the reference grids. The offset value is a correction factor for the SP.
  - c. Optionally, select a *Related Snap Pattern*, which is a predefined grouping of snap pattern definitions (SPDef) and width spacing snap pattern definitions (WSSPDef). If you choose a *Related Snap Pattern*, a corresponding WSP region that is equivalent to and in addition to the row region is created.

## Virtuoso Placer User Guide

### Creating Rows

#### Defining a Row

Use the *Rows* section to specify row parameters. The *Row Information* table initially has a placeholder for the first row. When you edit the row, a placeholder for the next row is automatically added. This way, you can create a list of rows to be included in the row template. You can also right-click a row to display the following shortcut menu:



Select *Auto-compute row offset* to automatically calculate and update the row offsets whenever rows are modified, for example, when a row is added, duplicated, or deleted. The offsets of all rows in the template are re-calculated and updated such that they are next to each other, without any overlaps. The row offset calculation is also done when the row height or offset values are modified.

To specify the row parameters:

1. After adding a row, edit its *Row Name*.
2. *Row Offset*, the offset from the base of the template, is initially set to 0 for the first row. You can edit the value. Row offset values must be greater than or equal to zero and less than or equal to the template period.

**Note:** You must ensure that the *Row Offset* value adheres to the *Template Period* and *Row Height* values. An example of an error condition is given [below](#).

## Virtuoso Placer User Guide

### Creating Rows

If *Auto-compute row offset* is selected, the row offset values are automatically adjusted such that rows are created next to each other, without any overlaps.

The first screenshot shows the 'Row Template Manager' dialog with 'Auto-compute row offset' unchecked. The 'Rows' table contains two rows: Row1 with an offset of 0 and Row2 with an offset of 0.048. A red arrow points to the 'Auto-compute row offset' checkbox, and a yellow callout box states: "Auto-compute row offset is not selected".

The second screenshot shows the same dialog with 'Auto-compute row offset' checked. The 'Rows' table now shows Row1 with an offset of 0 and Row2 with an offset of 0.192. A red arrow points to the 0.192 value, and a yellow callout box states: "Auto-compute row offset is selected" and "Row Offset is automatically calculated and updated".

The third screenshot shows the dialog with 'Auto-compute row offset' checked. The 'Rows' table now contains three rows: Row1 (offset 0), Row2 (offset 0.192), and a duplicate of Row2 (offset 0.24). A red arrow points to the 0.24 value, and a yellow callout box states: "A duplicate of Row2 is created" and "Row Offset is automatically calculated".

Row Name	Row Offset	Row Orientation	Row Height	Site Def	Background LPP
1 Row1	0	R0	0.192		
2 Row2	0.048	R0	0.048		

Row Name	Row Offset	Row Orientation	Row Height	Site Def	Background LPP
1 Row1	0	R0	0.192		
2 Row2	0.192	R0	0.048		

Row Name	Row Offset	Row Orientation	Row Height	Site Def	Background LPP
1 Row1	0	R0	0.192		
2 Row2	0.192	R0	0.048		
3 Row2	0.24	R0	0.048		

## Virtuoso Placer User Guide

### Creating Rows

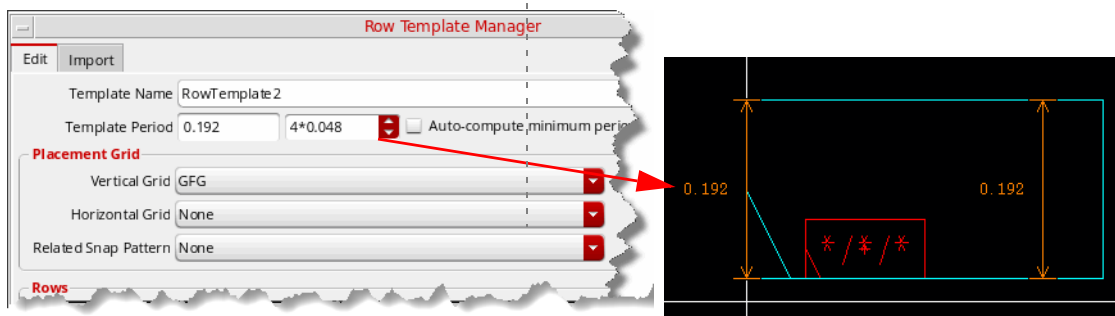
3. *Row Orientation* is the current orientation of the row. Valid orientations are **R0** (default) and **MX** (Mirrored about the x axis).



Caution

***Chains are considered incompatible if they have one or more instances with orientations that do not match the orientation of the row in which they are placed.***

4. The *Row Height* value defaults to the period of the vertical grid or to 0 if the vertical grid is not set. In the following example, the *Template Period* is set to 0.192 and the row template has a single row of height 0.192:



On adding a new row to the above definition, the *Template Period* value is highlighted, indicating that the end of the row exceeds the Template Period value. Therefore, the row is partially or fully outside of the template.

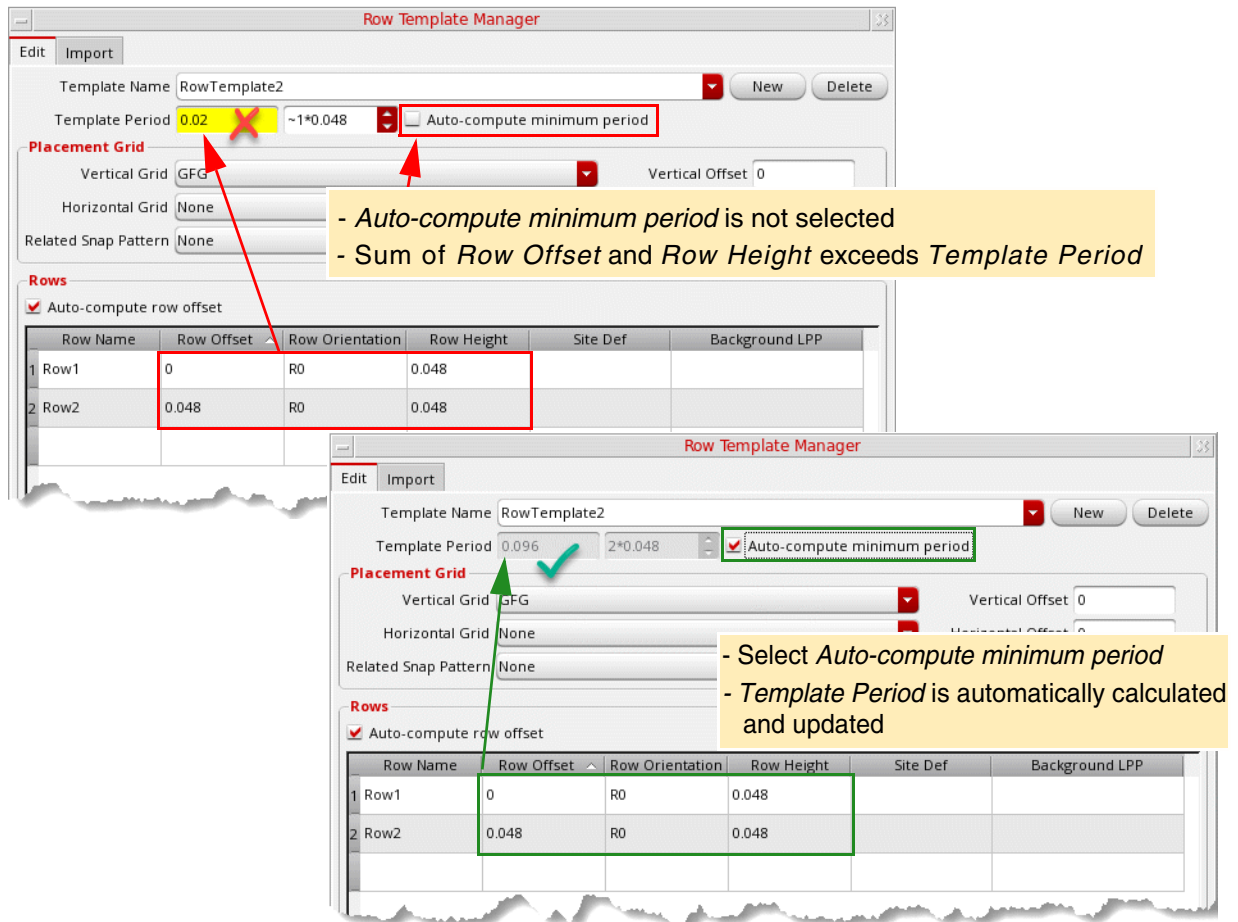
In such situations, select the *Auto compute minimum period* to automatically calculate a minimum template period value.



## Virtuoso Placer User Guide

### Creating Rows

The following figure depicts such a situation:

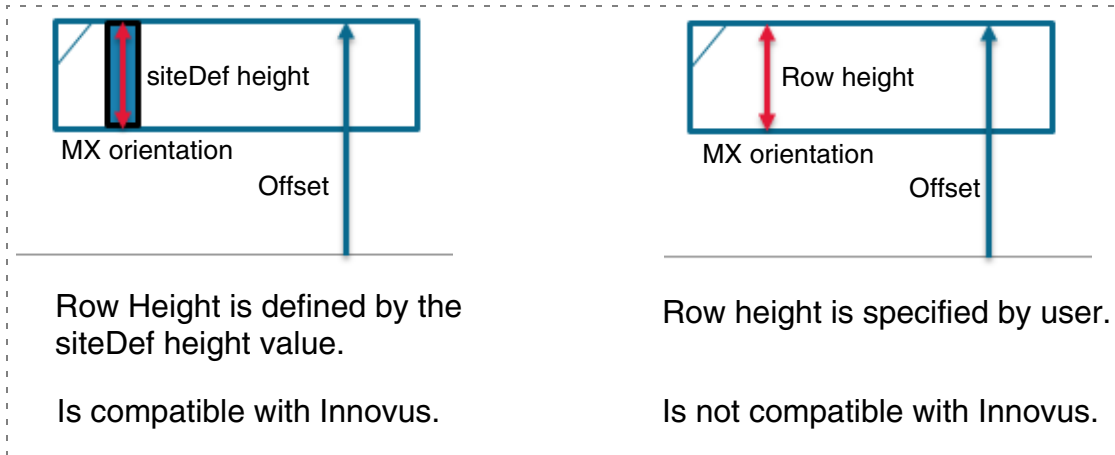


5. If you select a *Site Def*, then the specified *Row Height* is overridden by the row height value of the siteDef. The *Inst Pitch* value is also reset to the width of the siteDef, which you can further edit. Standard cells snap to the site defined in the siteDef.

## Virtuoso Placer User Guide

### Creating Rows

The following pictures depict the differences between the siteDef height and row height values:



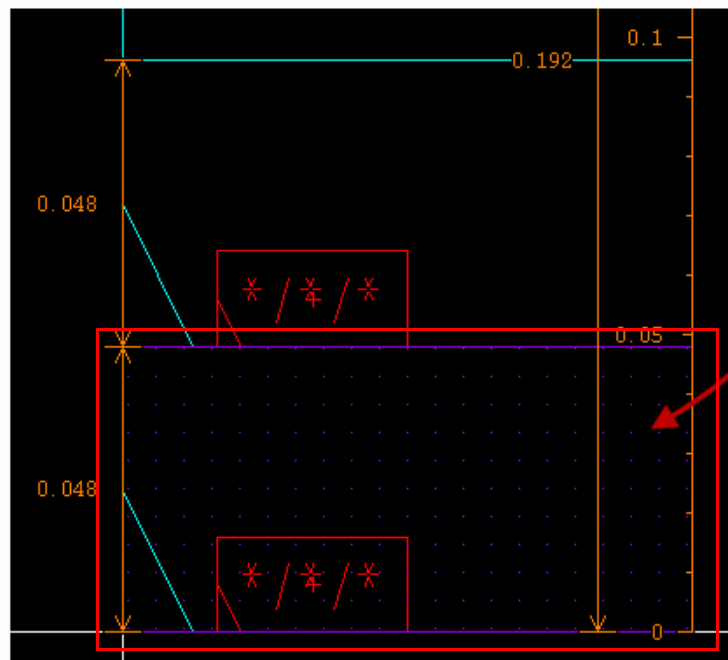
6. Select a *Background LPP* to fill the given row with a rectangle on the specified layer-purpose pair (LPP). This option is useful when you want to use well layers. You can use

## Virtuoso Placer User Guide

### Creating Rows

either an Nwell row with a Pwell behind it, or a Pwell row with an Nwell behind it. In the following example, an Nwell layer is selected as the *Background LPP*:

	Row Name	Row Offset ▲	Row Orientation	Row Height	Site Def	Background LPP
1	Row1	0	R0	0.048		NWell drawing
2	Row2	0.048	R0	0.048		

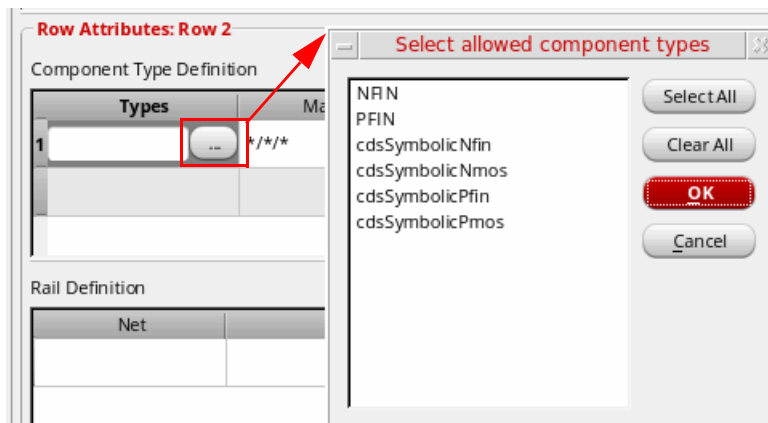


A rectangle shape is created in the row, which can be used for placing wells.

### Defining Component Types and Rails

Select a row to display its component attributes in the *Row Attributes* section. Use the options in this section to define the component types and rails.

1. In the *Types* field, specify the component types that can be placed in the row. These components are defined in the Configure Physical Hierarchy form. Click the Browse button to choose from a list of available component types.



2. If you have not specified component types, specify the device *Master*. Only the devices belonging to the specified master can be placed in a row. You can specify either a single master cellview or only the `lib`, `cell`, or `view` value. For example, *Master* can be set to `*/pfinFet/*` for a p-type row or `*/nfinFet/*` for a n-type row. Default is `*/*/*`.

#### Important

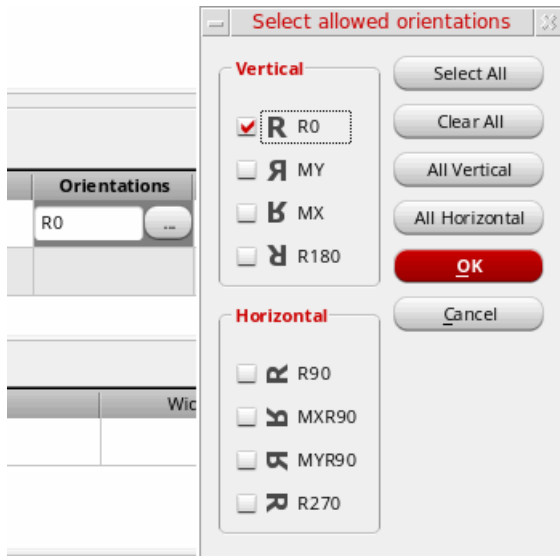
A row can use either a component type or a component master, but not both. Adding a master will delete the component type information, and vice versa.

3. Specify the component *Orientation*. This orientation is applied relative to the *Row Orientation* specified above. Therefore, if the row orientation is `R0`, the component orientation is followed as specified. However, if the row orientation is set to `MX` (mirrored) and the component orientation is also set to `MX`, then the devices are flipped twice, and

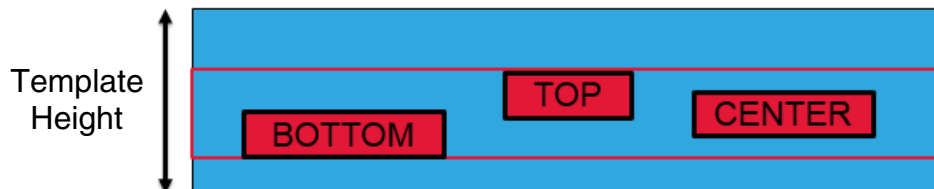
## Virtuoso Placer User Guide

### Creating Rows

so the resulting orientation will be R0. Click the Browse button to select from the list of supported orientations.



4. Choose an *Align Reference*, which specifies the row edge to which components need to be aligned. Valid values are BOTTOM, TOP, and CENTER.



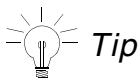
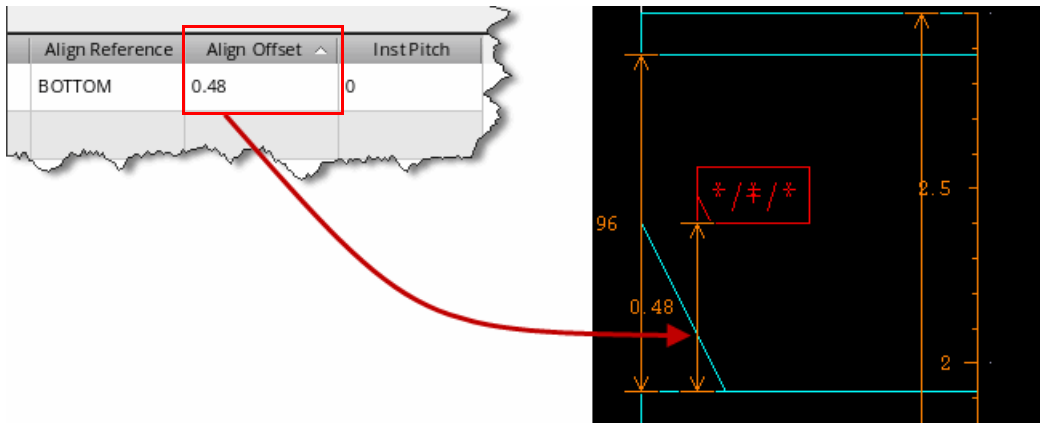
**Note:** Consider a row with *Row Orientation* set to MX and *Align Reference* set to BOTTOM edge. The devices will seem to be aligned to the top edge due to the mirrored orientation of the row. However, they are aligned to the BOTTOM edge, which happens to be the upper one as displayed in the canvas.

5. Specify an *Align Offset* value, which specifies the distance from the *Align Reference* edge that the components can be placed. In the following example, *Align Reference* is

## Virtuoso Placer User Guide

### Creating Rows

**BOTTOM** and *Align Offset* is 0.48. So, the component is placed at the specified distance from the **BOTTOM** edge.



#### Tip

Depending on your need, you can sort rows based on their *Align Offset*, *Align Reference*, or *Inst Pitch* values.

- InstPitch* specifies the minimum distance between instances of the same component in a row. If *InstPitch* is set to 0, then you can place two instances of the same component at any distance from each other. If you set *InstPitch* to any number greater than 0, then the distance between instances must be a multiple of *InstPitch*. *InstPitch* can also be used to override the default snapping definitions. A reference grid is created based on the specified value, and instances are snapped to this new grid. For example, if *InstPitch* is set to 1, then devices can snap by only one unit in the X-direction.
- Use the options in the *Rail Definition* section to generate rails in the selected row. Specify the *Net* that defines the rail connectivity. Select the *LPP* on which the rail is drawn and specify the rail *Width*.
- WSP Track* lists all WSP tracks available within the template period range. Choose a WSP track based on which the rail layer, width, and offset values must be set.
- In the *Align Reference* field, specify the reference edge for aligning the rail in relation to the row. Valid values are **TOP**, **BOTTOM**, and **CENTER**.
- Specify the *Align Offset*, which is the distance of the rail from the reference edge (*Align Reference*) specified above. *Align Offset* is automatically set to **BOTTOM** if a WSP track is selected.

## Virtuoso Placer User Guide

### Creating Rows



#### Tip

Depending on your need, you can sort rows based on their *Rail Width*, *Align Reference*, or *Align Offset* values. Here is a quick look at a design with rail definition:

The screenshot displays the 'Row Template Manager' interface. On the left is a design canvas showing a grid with dimensions 0.048 and 0.2, and a label 'avdd'. The right panel contains configuration options:

- Template Name:** RowTemplate4
- Template Period:** 0.048
- Placement Grid:** Vertical Grid: GFG, Horizontal Grid: None, Related Snap Pattern: None.
- Rows:** Auto-compute row offset is checked. The table below shows row configurations:

Row Name	Row Offset	Row Orientation	Row Height	Site Def	Background LPP
1 Row1	0	R0	0.048		
2 Row2	0.096	MX	0.048		

Below the rows table is the 'Row Attributes: Row 2' section, which includes a 'Component Type Definition' table:

Types	Masters	Orientations	Align Reference	Align Offset	Inst Pitch
1	*/*/*	R0	BOTTOM	0.48	0

At the bottom is the 'Rail Definition' table:

Net	WSP Track	LPP	Width	Align Reference	Align Offset
1 avdd	Match a Track...	Metal2 drawing	0.2	BOTTOM	0

Red arrows point from the 'MX' orientation in the Rows table to the 'Align Reference -> Bottom' text, and from the '0.2' width in the Rail Definition table to the 'Resultant Rail Alignment -> Top' text.

After creating row templates, use the Placement Planning form to generate rows in the layout canvas. For digital designs, you can use the Placement Planning form to create rows without selecting a row template, and then use the options available on the *Row* tab to specify how the rows should be created. Alternatively, you can select the required row template from the *Row Template* drop-down list and choose on the *Region* tab the required placement options. This second method helps achieve greater precision required in advanced nodes designs.

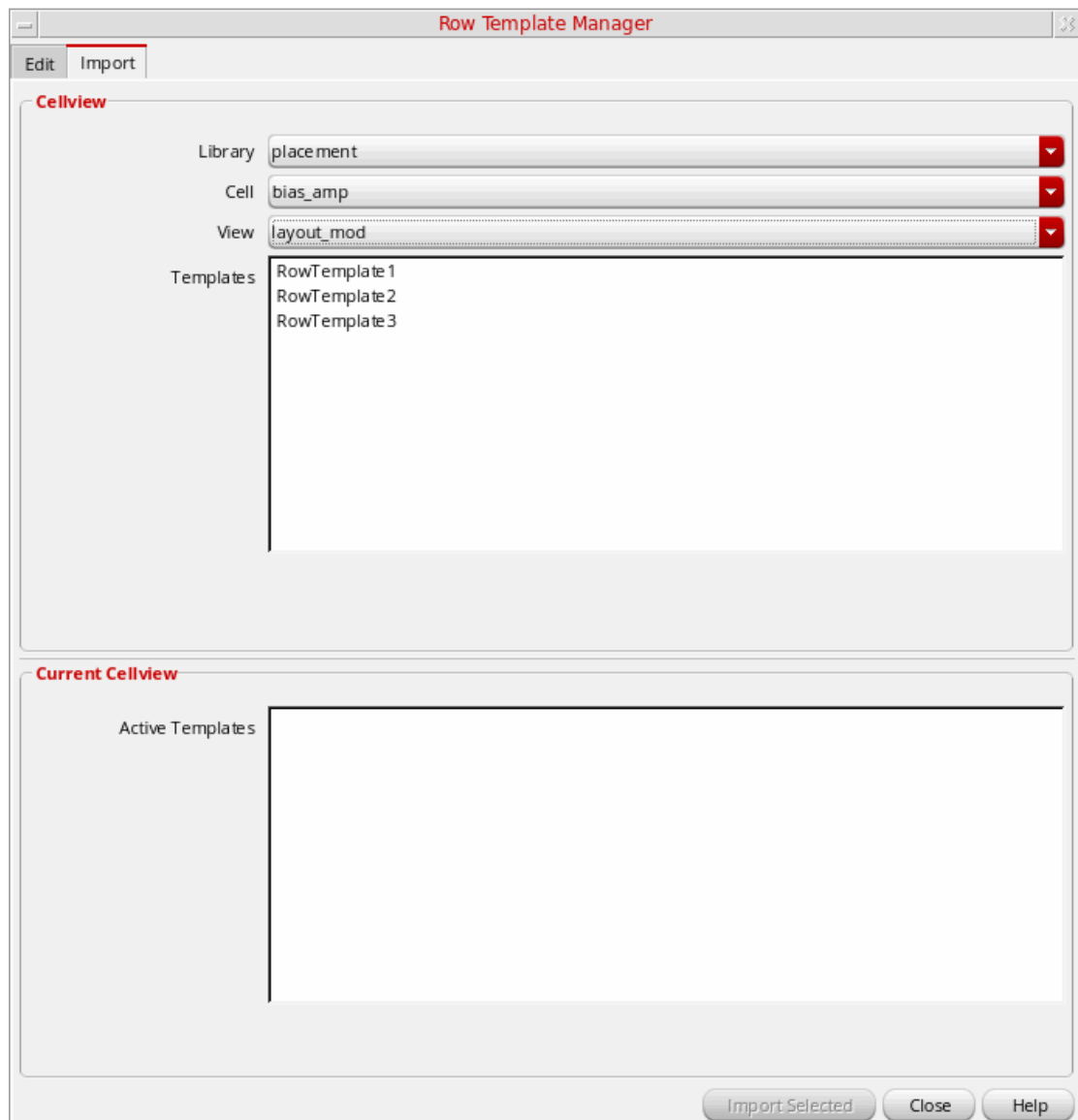
## Importing a Row Template from Another Cellview

Once defined, a row template is a reusable object, which makes it possible for you to import a row template defined for one design to another similar design. You can also define a set of row templates that can be used in a variety of designs and include them in the design library.

Use the *Import* tab of the Row Template Manager to import row templates from a different cellview in your design library.

To import existing row templates:

1. Click *Place – Row Template Manager* to display the Row Template Manager.





2. Select the *Import* tab.
3. Choose the *Library*, *Cell*, and *View* which contains the row template you want to import.
4. *Templates* lists all row templates that are available in the specified cellview. *Active Templates* in the *Current CellView* section lists all active row templates in the current cellview. Choose the required row templates.
5. Click *Import Selected*.

After importing templates, use the Placement Planning form to generate rows in the layout canvas.

## Deleting a Row Template

To delete a row template:

1. With the *Edit* tab open, choose the *Template Name* from the drop-down list.
2. Click *Delete*. The selected row template is deleted.

### *Important*

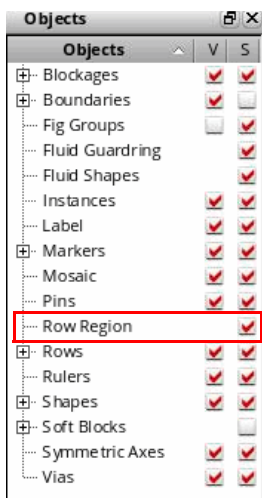
You cannot delete a template that is being used in the current design.

## Layout Support for Row Regions

This section describes the areas which have been enhanced in the layout editor to support row regions.

### Palette

Row regions can be selected from the *Objects* panel of the Palette. Only the selectability (*S*) option is supported for row regions, as shown in the figure below.



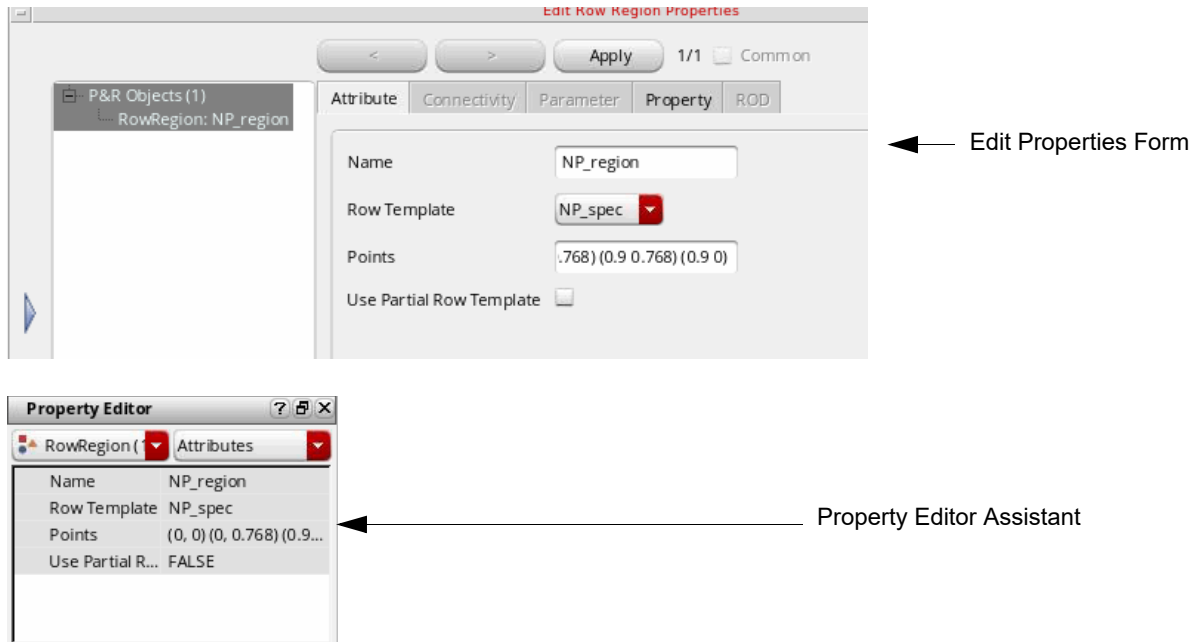
## Virtuoso Placer User Guide

### Creating Rows

---

#### Property Editor

Similar to other shapes and objects, you can use the Edit Properties form or the Property Editor assistant to edit the properties of the row regions.



#### Row Region Attributes

**Name** displays the name of the row region.

**Row Template** lets you select the row template used to create the row region.

**Points** sets the coordinates each point of the row region.

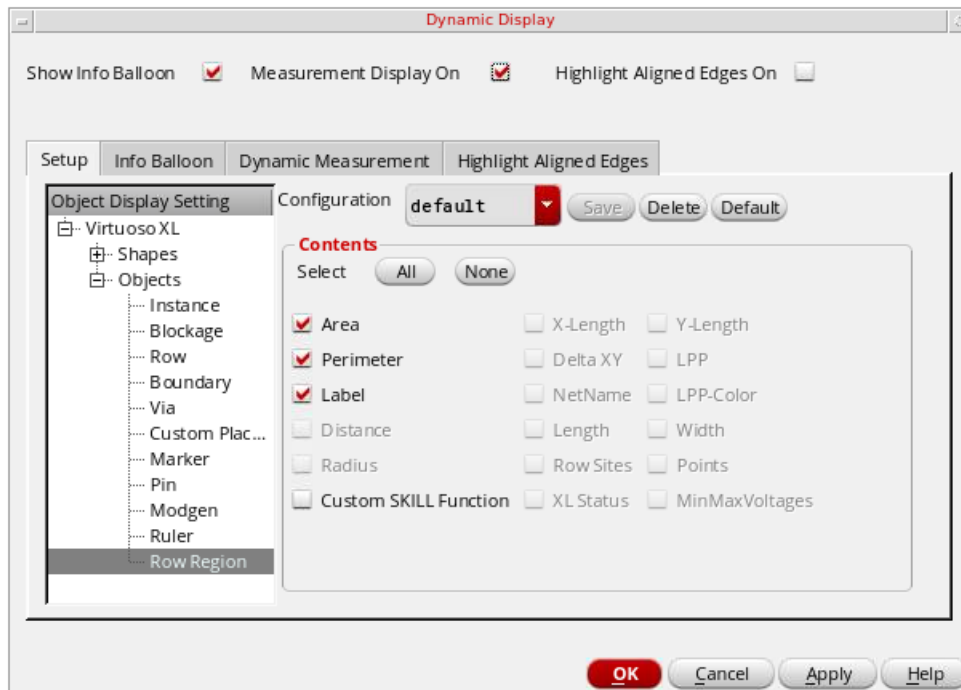
**Use Partial Row Template**, if selected, the row region spec repeats partially till there is space to fill a whole row. If not selected, only a full row region spec is created. If the space remaining on the top is less than the height of a row region spec, it is left empty.

# Virtuoso Placer User Guide

## Creating Rows

### Dynamic Display

Dynamic measurement and info balloon also support row regions. You can select the parameters to include in the *Info Balloon* and *Dynamic Measurement* display in the Dynamic Display form, as shown in the figure below.

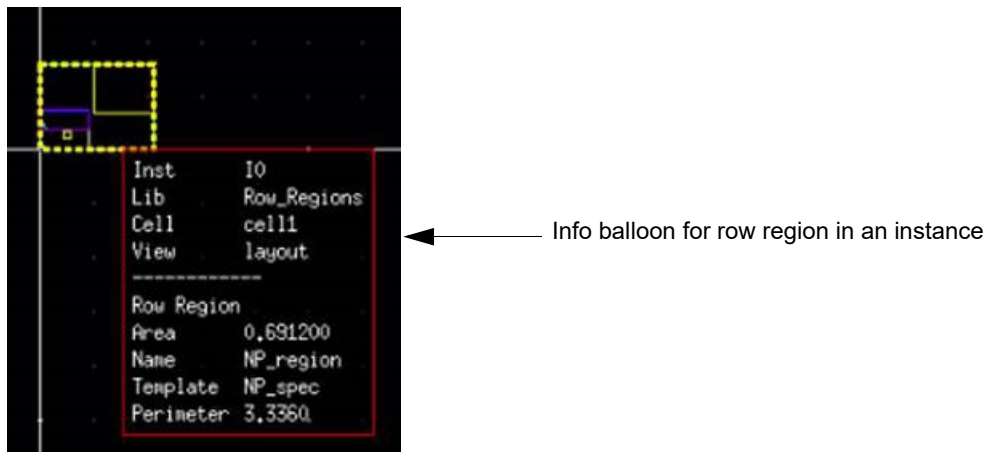
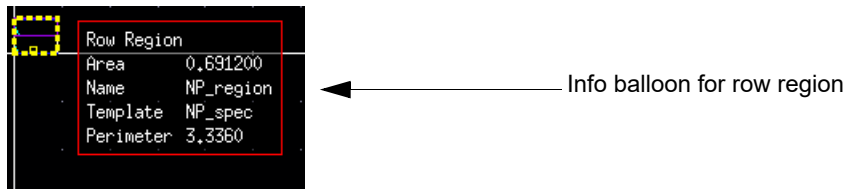


## Virtuoso Placer User Guide

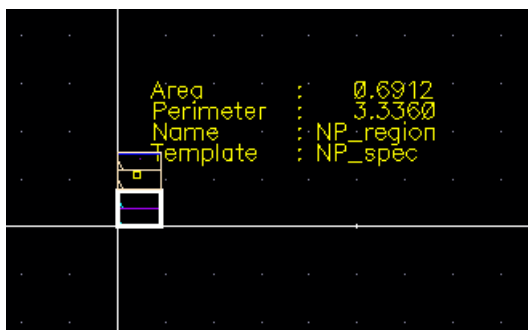
### Creating Rows

---

The figures below display the info balloon for the row regions.



The figure below displays the dynamic measurement for the row region object.



# Virtuoso Placer User Guide

## Creating Rows

### Show Selection Info Toolbar

The preselect and select information for row regions is displayed on the Show Selection Info toolbar, as shown in the figures below.

f

PreSel: Row Region Name(NP\_region)

elect: Row Region Name(NP\_region)

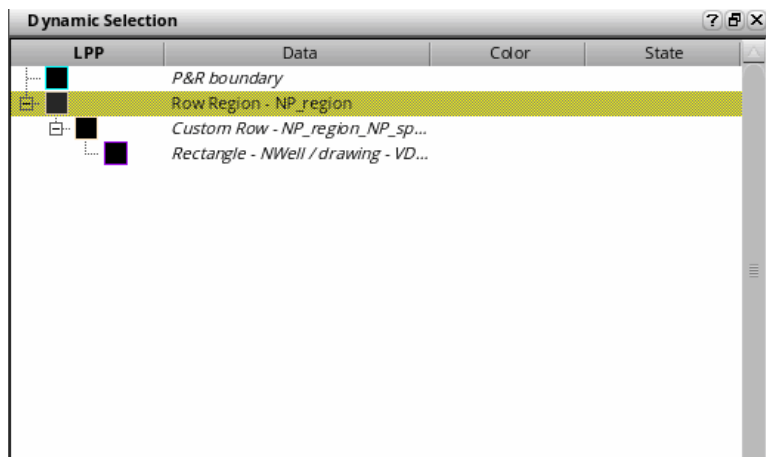
### Summary Window

You can also view the row region statistics in the Summary window, as shown in the figure below.

Row Region Statistics	
*****	
Row Region Template	Count
NP_spec	1
*****	

### Dynamic Selection Assistant

The *Dynamic Selection* assistant displays the row regions in a hierarchical manner. The properties of the row region and its subobjects are listed in the assistant, as shown in the figure below.



### Editing Support for Row Regions

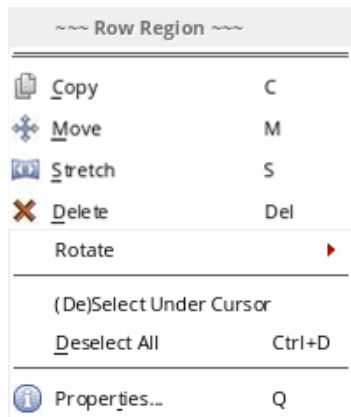
Row region objects can be edited like other shapes in the layout editor.

Some exceptions for a few commands are:

- For the `Delete` command, partial deletion (for example a vertex) is not supported.
- For the `Rotate` and `Chop` command, only orthogonal editing is supported.

**Note:** Group editing commands, such as `Add to Group` and `Ungroup` are not supported for row regions.

A context-sensitive menu is also available for row region objects. You can use the context-sensitive menu to access the commonly used editing commands.



### Snap Pattern Snapping Support

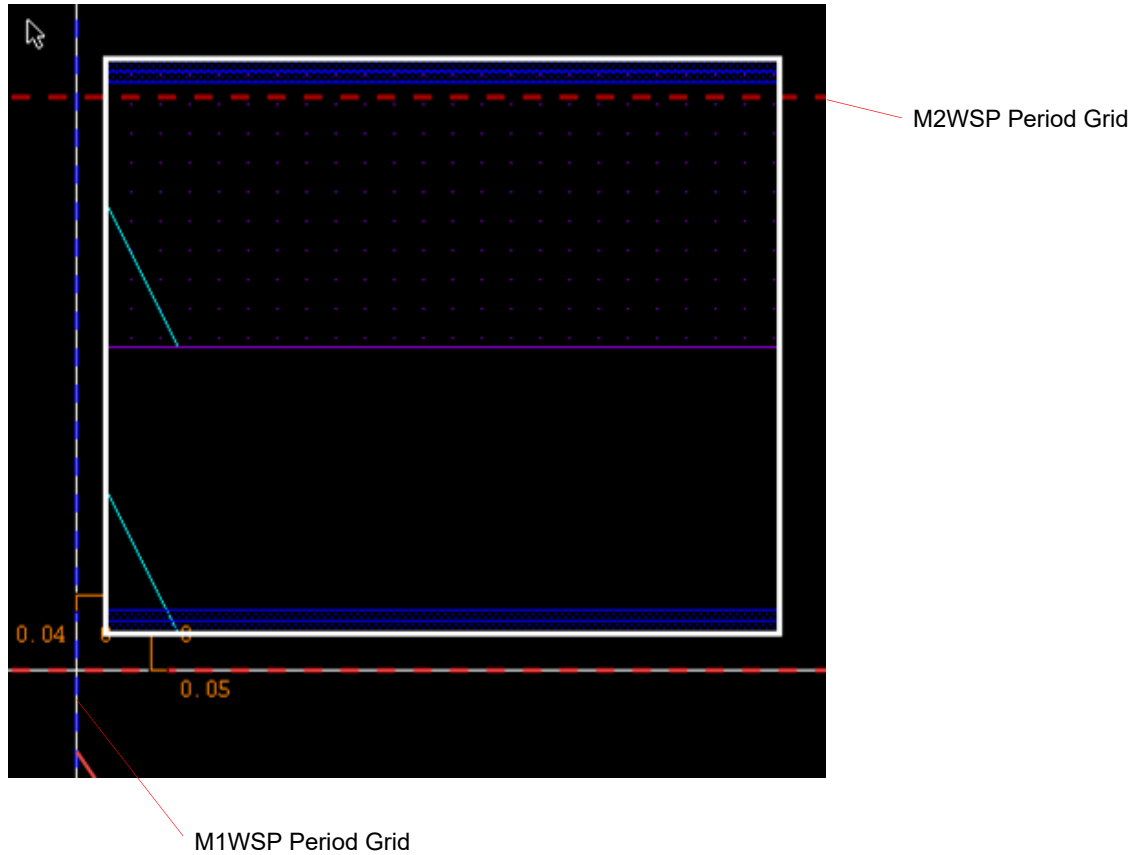
Snap pattern snapping to the reference grids specified by the row region template is also supported. You can specify the reference grid for the horizontal or vertical direction. In the

## Virtuoso Placer User Guide

### Creating Rows

---

example below, the horizontal grid is set to M1WSP and the vertical grid is set to M2WSP, with offsets (0.04 and 0.05).



## Importing Row Regions

The *Replace Rows* section of the Load Physical View form includes the *Row Regions* check box. When selected, the existing row regions in the target cellview are deleted and replaced with the row regions defined in the source cellview.

For more information, see [Load Physical View](#).



---

## Performing Design Placement

---

The Virtuoso Placer supports two placement types - automatic and assisted.

- **Automatic placement** is performed by the Virtuoso Placer.
- **Assisted placement** is the process of placing devices semi-automatically. Depending on the complexity of your design and your placement needs, you can choose to perform complete assisted placement of your design or use the assisted placement options to refine the placement after running the Virtuoso Placer.

Assisted placement options are available under the *Place* menu and on the *Assisted* tab of the Placement Options form.

The topics covered in this chapter are:

- Setting Placement Options
- Running the Automatic Placer
- Performing Assisted Placement

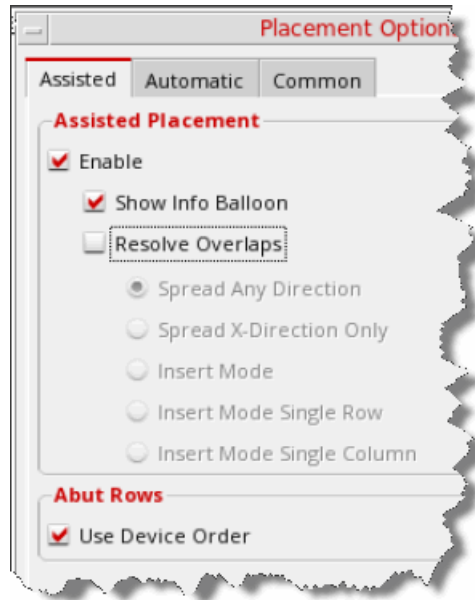
The chapter also includes the following sections:

- Viewing the Placement Report
- Support for Module Generators
- Supported Properties

## Setting Placement Options

You can use the options in the Placement Options form to customize the assisted and automatic placement features.

To display the Placement Options form, select *Place – Placement Options*.



The Placement Options form has the following three tabs:

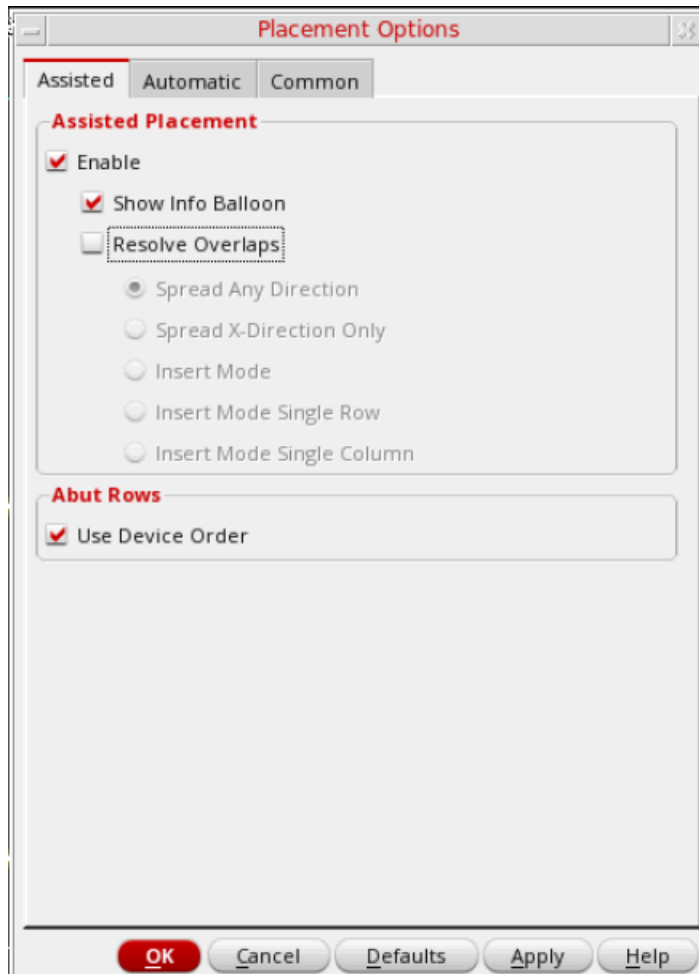
- **Assisted**, which provides options for assisted move commands and specifies assisted placement settings. This tab is displayed by default. For more information, see [Assisted Placement Options](#).
- **Automatic**, which specifies automatic placement settings. For more information, see [Automatic Placement Options](#).
- **Common**, which includes options that are relevant for both assisted and automatic placement. For more information, see [Common Options](#).

Both, assisted and automatic placement methods support snapping of individual objects and object groups to grids. For more information, see [Snapping Behavior](#).

Virtuoso Placer honors the Net Priority routing constraint set on nets. The Net Priority constraint defines the order of priority when routing a net. You can specify the priority of nets in the -128 to 128 range, where -128 is the lowest priority. The higher the priority, the closer the placer places the instances attached to the associated net.

## Assisted Placement Options

When you open the Placement Options form, the *Assisted* tab is displayed by default.



Use the options on this tab to perform the following tasks:

- ☐ Assisted Placement
- ☐ Abut Rows Use Device Order

### Assisted Placement

The *Assisted Placement* section provides the following advanced assisted features:

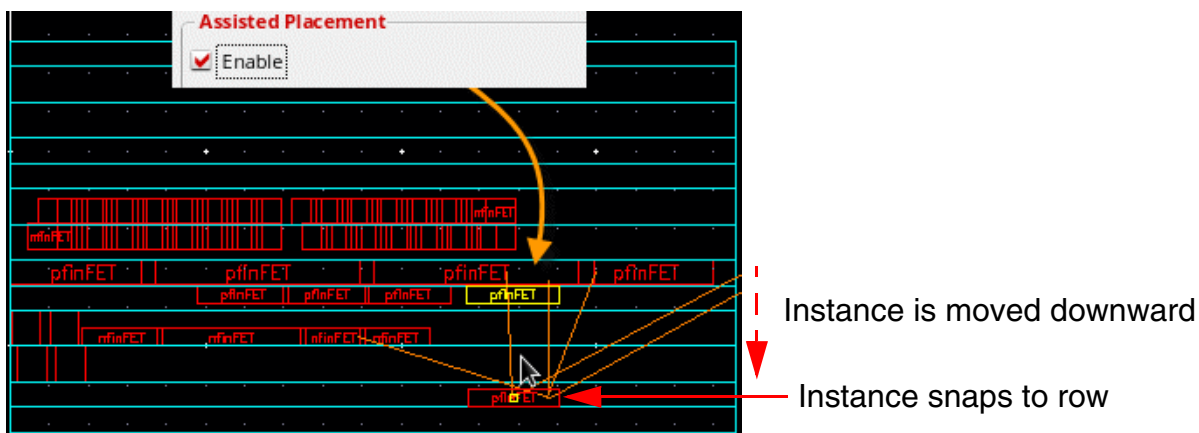
- Snaps single instances, chains, figGroups, mosaics, and Modgens to rows and snap patterns in real time. When multiple instances are moved simultaneously, each instance is snapped individually to the row over which it is moved, while honoring the existing row attributes. For more information about moving multiple instances, see [Moving Multiple Instances](#).
- Places single devices that are (incrementally) generated in the layout view using *Connectivity - Generate - Selected From Source*.
- Lets you remove overlaps post-edit and choose the direction in which devices should be moved to resolve overlaps.
- Enables constraint-aware editing.
- Supports transparent mode and edit in place.
- Lets you choose the order in which devices are abutted.



*Tip*

**Assisted Placement** is also referred to as **Assisted Move**.

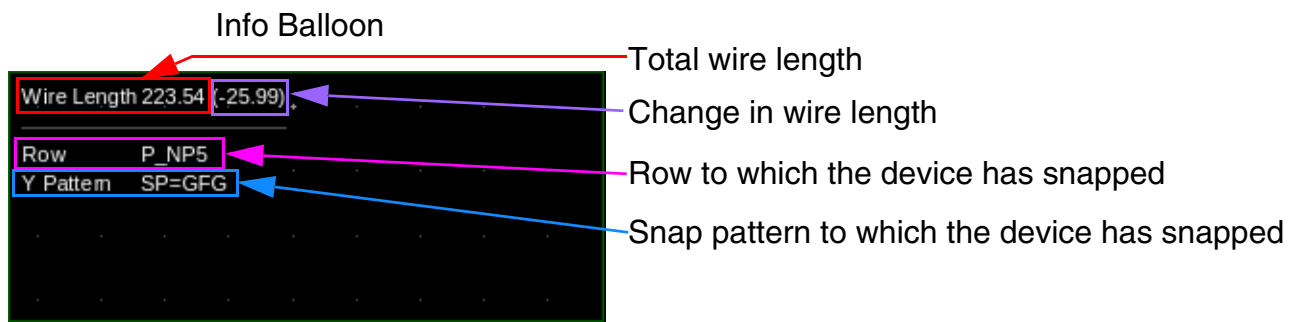
When *Enable* is selected in the *Assisted Placement* section, instances are snapped to the nearest rows instead of grids, as shown below:



#### **Important**

In Virtuoso Layout EXL, the assisted placement snapping behavior overrides the snapping behavior set in the *Snapping* section of the Layout Editor Options form. Therefore, *Instance to Row* snapping and *Snap Pattern Snapping* settings are not applicable when *Enable* is selected. For information about the snapping options in the Layout Editor Options form, see [Snapping](#).

When instances are moved with *Enable* selected, you can see an info balloon that displays the following information. Use either the *Show Info Balloon* option, or the [infoBalloon](#) environment variable to control the display of info balloon. The option is selected by default.



In the Constraint-Aware Editing (CAE) mode, instances that lie outside the cluster boundaries are highlighted. An arrow indicates the respective cluster boundary, and the cluster violation count is displayed in the info balloon. For more information, see [Constraint-Aware Editing](#) in *Virtuoso Layout Suite XL User Guide*.

The [displaySnappedInfo](#) environment variable controls the display of snapping information in the info balloon.

Snapping behavior is based on the row definition in the row template and also applies to Modgens, mosaics, and figGroups. For more information about the snapping behavior of Modgens, see [Support for Module Generators](#).

# Virtuoso Placer User Guide

## Performing Design Placement

### Moving Multiple Instances

You can select and move multiple instances simultaneously. Each instance is snapped to rows in real time, while honoring the row attributes. Consider a design with two row regions, each with different type of rows:

- **Top Region:** Rows have been generated using the Mixed row template. This template comprises two row types with similar definitions. They can contain standard cells, PFIN devices, and NFIN devices. The NFIN devices are aligned to the bottom, while the PFIN devices are aligned to the top. The PFIN devices have a mirrored orientation.
- **Bottom Region:** Rows have been generated using the deviceRow row template, which has a single row type. The row can contain PFIN and NFIN devices. All devices are bottom aligned. All device orientations are supported.

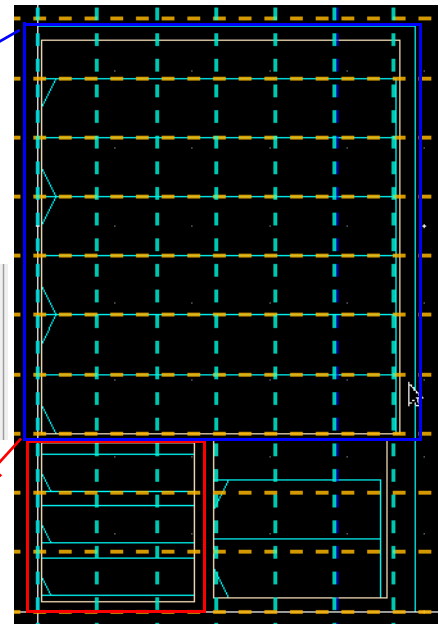
Row Template: Mixed

Component Type Definition Row 1						
Types	Masters	Orientations	Align Reference	Align Offset	Inst Pitch	
1 stdCells,STDCELL		R0	BOTTOM	0	0	
2 NFIN		R0	BOTTOM	0	0	
3 PFIN		MX	TOP	0	0	

Component Type Definition Row 2						
Types	Masters	Orientations	Align Reference	Align Offset	Inst Pitch	
1 stdCells,STDCELL		R0	BOTTOM	0	0	
2 NFIN		R0	BOTTOM	0	0	
3 PFIN		MX	TOP	0	0	

Row Template: deviceRow

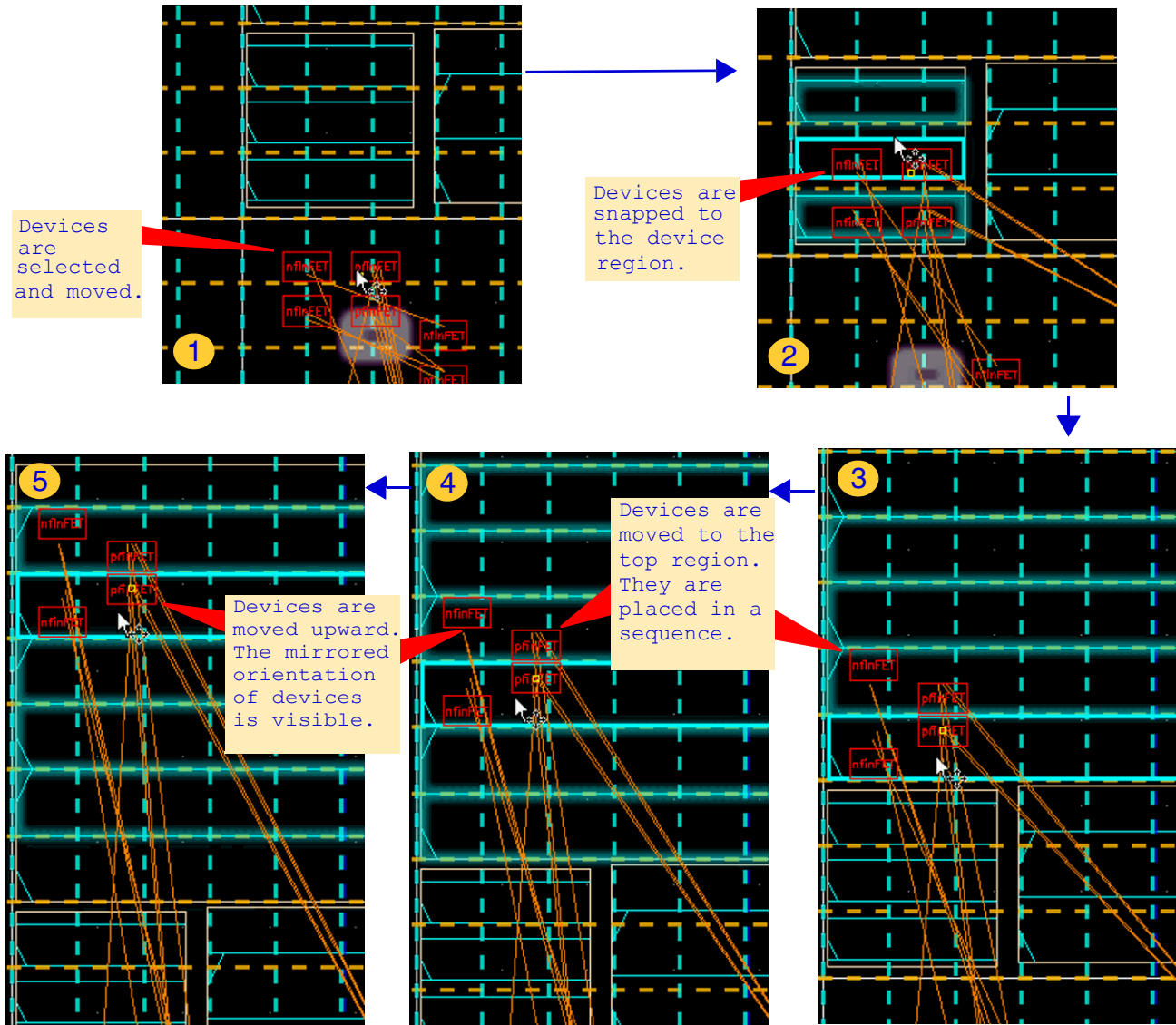
Component Type Definition						
Types	Masters	Orientations	Align Reference	Align Offset	Inst Pitch	
1 NFIN,PFIN		R0,R180,MY,MX	BOTTOM	0	0	



## Virtuoso Placer User Guide

### Performing Design Placement

The following graphics depict the snapping behavior of individual devices when two PFIN and NFIN devices are moved:



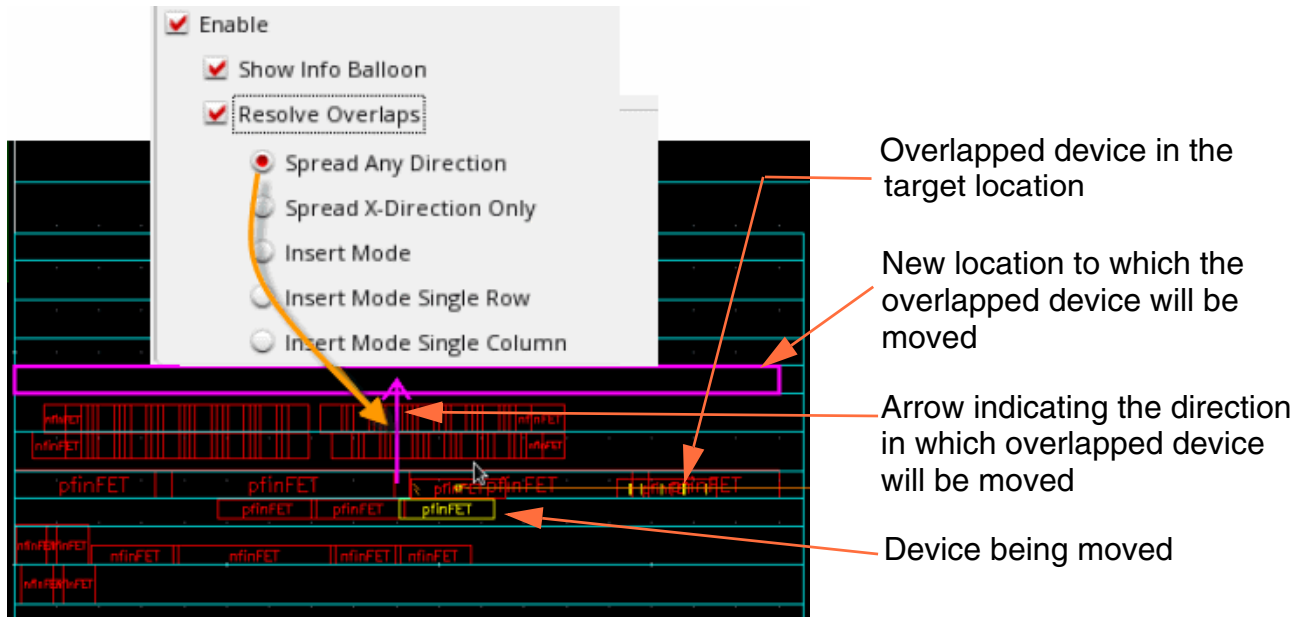
### ***Resolve Overlaps***

Select *Resolve Overlaps* to automatically resolve any overlaps resulting from moving devices in a design. An arrow indicates how the overlap will be resolved if an instance is moved in a particular way. The device being moved is placed at the target location, and the overlapped device (at the target location) is moved to a new location.

## Virtuoso Placer User Guide

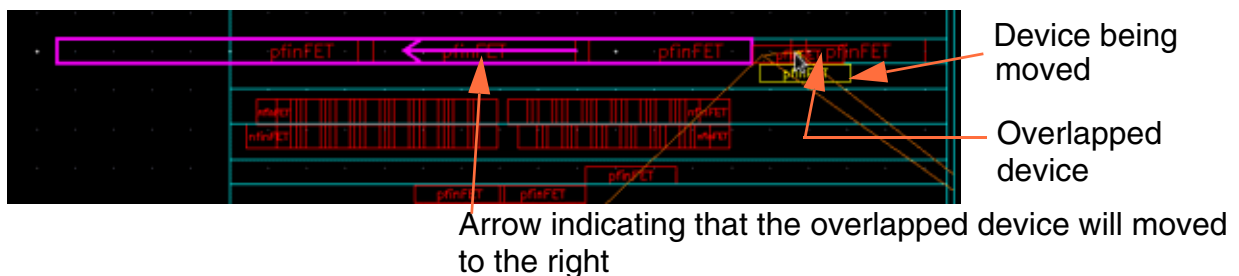
### Performing Design Placement

**Note:** *Resolve Overlaps* can be enabled only when *Enable* is selected in the *Assisted Placement* section.



*Resolve Overlaps* lists the following modes that specify the direction in which the overlapping device should be moved:

- **Spread Any Direction** (default): In this mode, the overlapped device is moved by the minimum distance in any direction. The example shown above depicts such a move.
- **Spread X-Direction Only:** In this mode, the overlapped device is moved by the minimum distance to the right or left within the same row as indicated below:

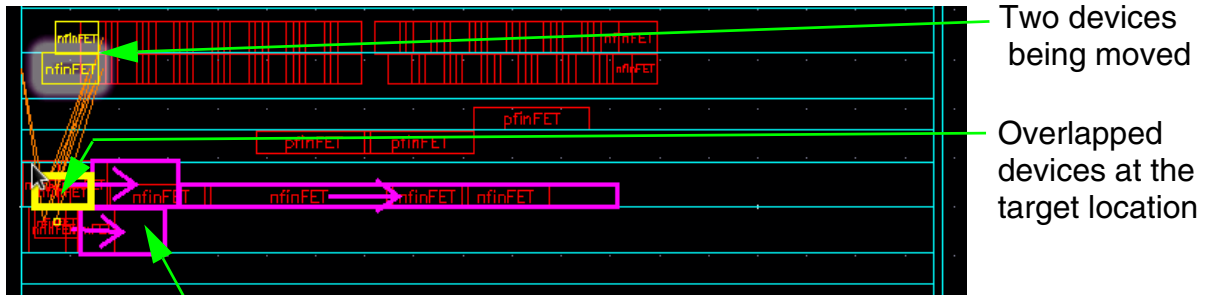




## Virtuoso Placer User Guide

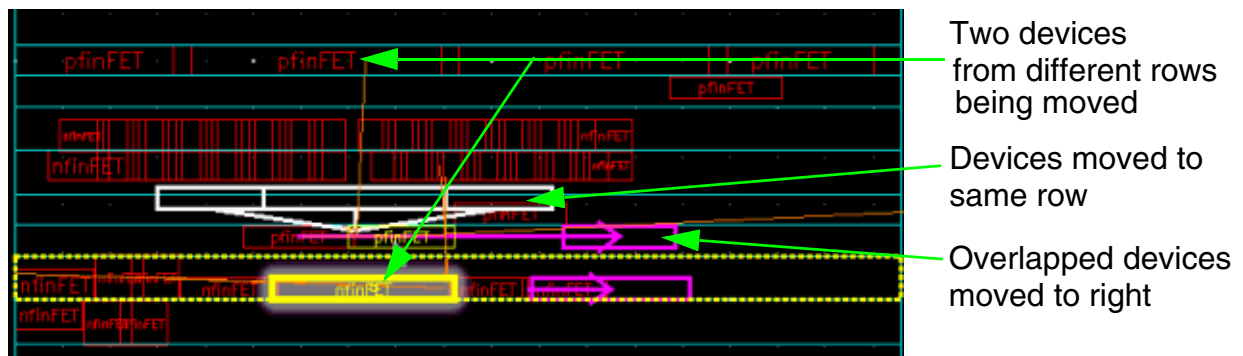
### Performing Design Placement

- **Insert Mode:** In this mode, the devices being moved are placed sequentially by row. If more than one device is being moved, the relative positions of the devices are maintained while placing them at their new locations. The overlapped devices are moved only to the right, as shown below:

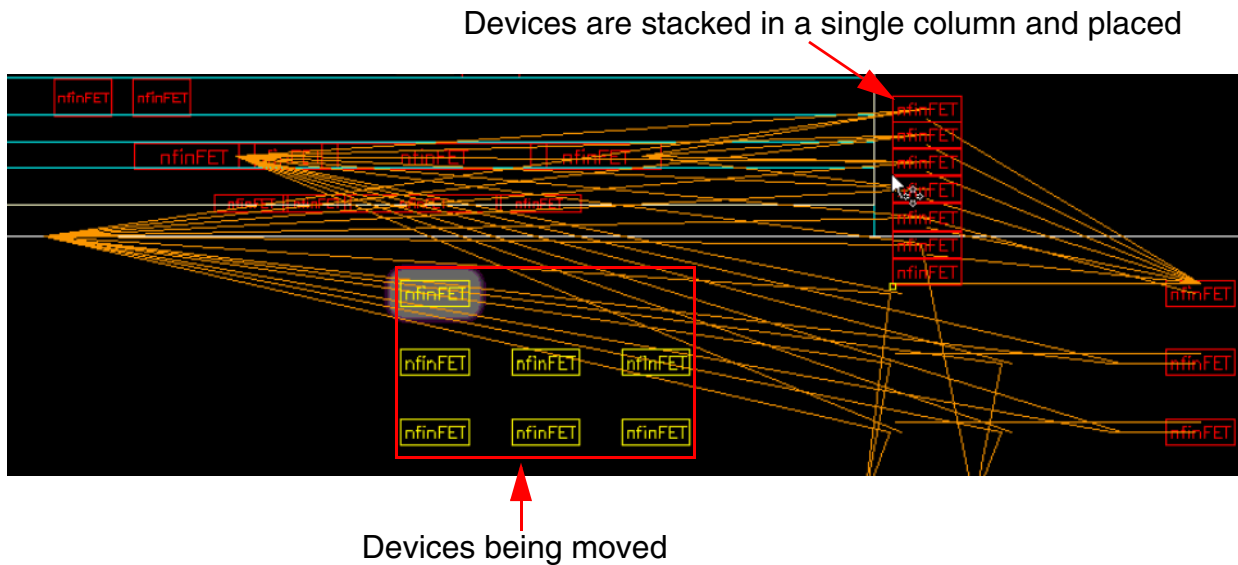


Arrows indicating that the overlapped devices will be moved to the right. Their relative positions are maintained.

- **Insert Mode Single Row:** In this mode, the devices that are moved are all placed sequentially in the same row. The overlapped devices are moved to the right as shown below:

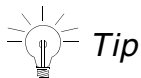


- **Insert Mode Single Column:** In this mode, the devices that are moved are all placed sequentially in the same column. The overlapped devices are moved to the right as shown below:



### Abut Rows Use Device Order

Select *Abut Rows Use Device Order* to abut instances in the same order in which they are placed in each row.



*Tip*

Disable this option when there are adjacent instances that do not have any common connectivity.

If selected, then an extra dummy poly is inserted between the two instances to allow abutment. To avoid this, clear the check box. The devices are then reorganized to achieve a more compact abutment chain.

**Note:** Abutment of rows is done using the Layout XL chaining engine. Therefore, the abutment results are subject to the settings of the Layout XL chaining environment variables. For more information about these environment variables, see [Layout XL Environment Variables](#).

## Automatic Placement Options

Virtuoso Placer performs automatic placement by choosing appropriate locations for all instances. The options must be set on the *Automatic* tab before Running the Automatic Placer. These options can also be accessed by clicking the *More Options* button in the Automatic Placement form. The *Automatic* tab includes options that are generic as well as those that are specific to the *Custom Digital* placement type.

The screenshot shows the 'Placement Options' dialog box with the 'Automatic' tab selected. The dialog is divided into several sections: 'Rules', 'Devices', 'Standard Cells', 'Pins', and 'Passive Devices'. The 'Rules' section has a 'Constraint Group' dropdown set to 'AbstractDefaultSetup(cdn20FF)' and a 'Minimum Boundary Offset' set to '0'. The 'Devices' section has checkboxes for 'Abut Devices' (unchecked), 'Preserve Abutment Chains' (checked), 'Group M Factor Devices' (checked), 'Group CMOS Pairs' (unchecked), and 'Group P over N' (unchecked). The 'Standard Cells' section has checkboxes for 'Run Spacer Within Rows' (unchecked) and 'Insert Tap Cells' (unchecked), followed by 'Component Type' (empty), 'Minimum Spacing' (empty), and 'Maximum Spacing' (empty). The 'Pins' section has checkboxes for 'Routing Direction Aware Pin Placement' (checked) and 'Pins On PR Boundary' (checked). The 'Passive Devices' section has a checkbox for 'Passive to Outside' (unchecked). At the bottom are buttons for 'OK', 'Cancel', 'Defaults', 'Apply', and 'Help'.

Use the options on this tab to perform the following tasks:

- Apply Design Rules
- Specify Device Settings
- Define Standard Cell Settings
- Apply Pin Placement Settings
- Define Passive Device Settings

### Apply Design Rules

The options in the *Rules* section are generic, and apply to all design types. Choose a *Constraint Group* from the drop-down list, which specifies the spacing rules to be applied while running the automatic placer. *Minimum Boundary Offset* specifies the minimum spacing required between the selected reference layer and the PR boundary.

### Specify Device Settings

The options in the *Devices* section are applicable only to analog designs. For more information, see [Running the Automatic Placer](#). The *Devices* section includes:

- **Abutment Options** - You can choose to *Abut Devices* so as to create abutment chains in each row. *Preserve Abutment* preserves any existing abutment chains in the design.

**Note:** *Preserve Abutment* cannot be used when *Abut Devices* is selected.

- **Grouping Options** - Choose *Group M Factor Devices* to create a cluster containing all complementary MOS devices in the design that have a multiplication factor in the schematic.

Similarly, you can *Group CMOS Pairs* that share gate connections. All such CMOS device pairs in the design that have been generated with chaining enabled are grouped together into a cluster.

Select *Group P Over N* to place all P-type devices at the top and N-type devices at the bottom of the selected region.

### Define Standard Cell Settings

The options in the *Standard Cells* section are applicable only to digital designs, and therefore apply only to the *Custom Digital* placement type in the Automatic Placement form. For more information, see [Running the Automatic Placer](#). The *Standard Cells* section includes:

- **Spacing Options:** Select *Run Spacer Within Rows* to distribute standard cells evenly in the rows by adding or removing space between standard cells within each row.
- **Tap Cells Options:** Use the options in this section to *Insert Tap Cells* in rows based on the specified *Minimum Spacing* and *Maximum Spacing* values. Specify the *Component Type* that contains the substrate contacts (tap cells) to be used. The specified component type must have the component class STDSUBCONT.

### Apply Pin Placement Settings

The options in the *Pins* section are generic, and so are applicable to all design types.

The *Routing Direction Aware Pin Placement* option is applicable to designs with width spacing patterns (WSPs). If selected (default), the placer considers the WSP direction while placing unconstrained pins. For vertical WSPs, pins are placed along the left and right edges; for horizontal WSPs, pins are placed along the top and bottom edges.

You can choose to place all *Pins on Boundary* during placement.

### Define Passive Device Settings

The *Passive to Outside* option is generic, and applicable to all design types. Selecting this option pushes large capacitors and resistors to the edge of the PR boundary, thereby separating them from the transistors in the design.



#### Tip

**What are passive devices?** Passive devices are devices that cannot be used to amplify signals. These devices do not require an external source of energy to function. Rather, they store energy in the form of voltage or current. Examples: resistors and capacitors.

## Common Options

The *Common* tab provides options common to both *Assisted* and *Automatic* placement.

The screenshot shows the 'Placement Options' dialog box with the 'Common' tab selected. The dialog is divided into four sections: General, Modgen, Device, and Standard Cell. The 'General' section has a checkbox for 'Width Spacing Pattern (WSP) Snapping' and a text field for 'Cell Boundary LPP(s)'. The 'Modgen' section has a checkbox for 'Update MODGENs to Rows Post-Placement'. The 'Device' section has text fields for 'N-Diffusion Spacing' and 'P-Diffusion Spacing'. The 'Standard Cell' section has checkboxes for 'Context Aware Placement', 'Pin Density Aware Placement', and 'Maximize Pin Access', a 'Context File' text field with a 'Browse...' button, a 'Minimum Pin Access Count' text field with the value '2', and a 'Cell Padding Value(s)' text field. At the bottom are buttons for 'OK', 'Cancel', 'Defaults', 'Apply', and 'Help'.

**Placement Options**

Assisted Automatic **Common**

**General**

☐ Width Spacing Pattern (WSP) Snapping

Cell Boundary LPP(s):

**Modgen**

☐ Update MODGENs to Rows Post-Placement

**Device**

N-Diffusion Spacing

P-Diffusion Spacing

**Standard Cell**

☐ Context Aware Placement

Context File

☐ Pin Density Aware Placement

☐ Maximize Pin Access

Minimum Pin Access Count

Cell Padding Value(s):

The *Common* tab is divided into: *General*, *Modgen*, *Device*, and *Standard Cell* sections.

In the *General* section:

- Select *Width Spacing Pattern (WSP) Snapping* to enable width-aware and constraint (CST)-aware snapping of instances and pins. Both interactive and automatic placers recognize the width spacing snap pattern grid that is defined using the `widthSpacingSnapPatternDef` rule in the technology file. The `widthSpacingSnapPatternDef` definitions are available in the technology database as well as in the design. For more information, see the [Width Spacing Pattern Layer Rules](#) section in *Virtuoso Technology Data ASCII Files Reference*.

#### **Important**

If both WSPs and Snap Patterns (SPs) are active, WSP snapping takes precedence.

WSP snapping is possible only when the specified constraints, such as the `minwidth`, `allowedWidthRanges`, and `sig type`, are met. If a constraint is not met, for example, if the pin width is less than the width specified with the `minWidth` constraint, the next track is checked.

- Specify a list of *Cell Boundary LPPs* used to derive the cell boundary for standard cells and devices when running Place Like Schematic, Analog Placer, and Digital Placer. Only the shapes on the listed LPPs are considered when deriving the boundary.

Example: (Metal1 drawing).

In the *Modgen* section:

- Select *Update MODGENS to Rows Post-Placement* to regenerate Modgens after they are moved to a row region. The orientations of the Modgen members are updated to match the orientations of the rows to which they are moved. For example, a Modgen member instance with orientation `R0` is moved to a row with orientation `MX`. The Modgen is regenerated and the orientation of the instance is changed to `MX`.

**Note:** When deselected, Modgens are snapped only to rows with matching orientations.

In the *Device* section:

- Specify the *N-Diffusion Spacing* and *P-Diffusion Spacing* values, which set the minimum spacing required between adjacent NMOS and PMOS chains, respectively.

In the *Standard Cell* section:

- You can choose to enable *Context Aware Placement*. Standard cells are placed such that each cell is aware of the context of the neighboring cell and supports placement in accordance with the `EDGETYPE` and `CELLEDGESPACE` constraints, which are either defined in the technology and macro LEF, or provided using an XML file. By default, this option is switched off. You can also specify a *Context File*, which is an

## Virtuoso Placer User Guide

### Performing Design Placement

---

XML file containing the edge spacing constraint information for performing context-aware placement. During context-aware placement, the placer honors the edge types and spacing constraints of tap cells.

- Select *Pin Density Aware Placement* to consider the pin densities of standard cells while optimizing placement and improving routability. The assisted placement options are also pin density-aware. The pin density consideration can result in larger spacing between the placed standard cells.

**Note:** Pin density awareness is a heuristic to improve routability, and therefore is not checked during placement verification.

- Select *Maximize Pin Access* to increase the number of WSP tracks that overlap the pins on standard cells. Selecting this option improves routability by providing more access points to the router. Specify the *Minimum Pin Access Count*, which refers to the minimum number of access points that should be available.
- In the *Cell Padding Value* field, specify the space to the left and right of the master. Use the following syntax to enter values in this field:

```
"(cellName right_spacing left_spacing)"
```

Example:

```
"(nand4 2.5 2.5) (inv 0 1)"
```

Here, spacing of 2.5 is applied to both sides of cell `nand4`. For cell `inv`, spacing 0 is applied to the left side and spacing 1 is applied to the right side.



## Running the Automatic Placer

The automatic placer places devices in rows defined in the layout canvas. For best results, run the automatic placer after performing Placement Planning. You can use the automatic placer for placing devices, standard cells, or a mix of both. During placement, these devices are snapped to appropriate rows and grids. For more information about the snapping, see [Snapping Behavior](#).


**Note:** The default snap depth for snap pattern mapping is set based on the display step level setting. You can use the environment variables [snapPatternSnappingDepth](#) and [snapPatternSnappingDepthMethod](#) to specify the snap depth and the method to be used to define the snap depth for snap pattern snapping, respectively.

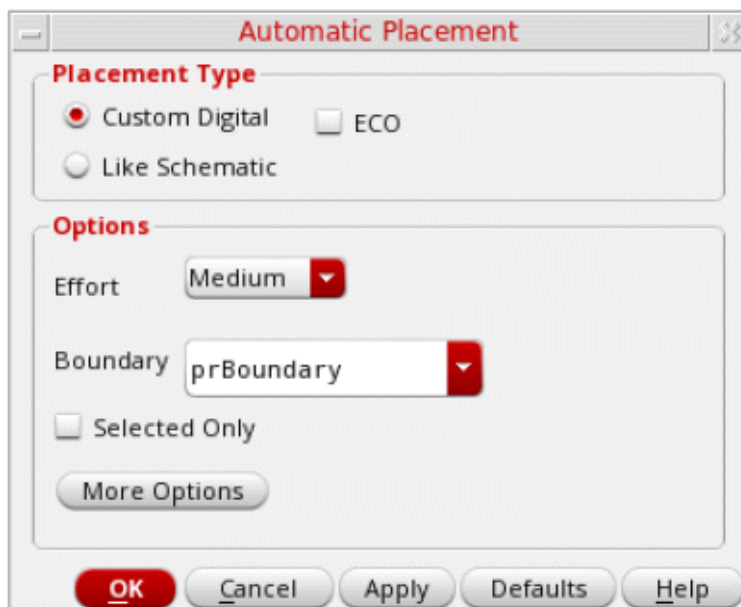
You can use assisted placement to further refine the placement to achieve the desired result. For more information about these options, see [Setting Placement Options](#).

The automatic placer supports placement of virtual figGroups. For more information, see [Placement of Virtual FigGroups](#).

To run the automatic placer:

1. Select *Place – Auto Place*. The [Automatic Placement](#) form is displayed.

**Note:** Alternatively you can choose the Auto Place  button from either the Placement toolbar or the Design Planning toolbar. For more information, see [Design Planner Toolbar](#).



2. Depending on your design type, choose one of the following placement types:

## Virtuoso Placer User Guide

### Performing Design Placement

---

- ❑ **Custom Digital:** Useful for placement in designs that have a higher proportion of standard cells; for example, designs with around 80% standard cells and 20% devices. A precise placement of the devices can also be achieved when devices have matched gates or are connected to the same net.

Selecting *Custom Digital* performs a complete initial placement of all the devices in the design. Select *Eco* to incrementally place any unplaced devices in the design.

**Note:** Master filter definitions are honored while placing standard cells. For more information about master filters, see [Defining Component Types and Rails](#).

- ❑ **Like Schematic:** Optimizes for wire length, area, and spacing patterns by placing instances as per their relative positions in the schematic design. Use the [likeSchematicTolerance](#) environment variable to specify the tolerance level. Larger tolerance levels allows the placer to use more approximate schematic correspondence to improve the placement. Valid values are 1 through 1000.

Choose a placer depending on which device type is the majority in your design. If there are more standard cells, use custom digital.

#### 3. Select one of the three *Effort* levels from the *Options* section - *Low*, *Medium*, and *High*.

The runtime of the placer and quality of results depend on the effort chosen. The time for each effort level increases in roughly linear increments. Selecting *Low* results in a comparatively less precise, but quick placement. Selecting *High* results in the best placement results, but takes more time than the *Low* and *Medium* effort levels.

Try the *Low* effort level first, because many designs perform well at this level. *Low* effort level is best suited for regular, structured designs with no wells through the center of the cell and non-conformal group boundaries.

However, if your design is less structured or includes a large number of constraints, try running the design at the *Medium* effort level. You might also consider this effort level if you have a few conformal group boundaries and wells under placer control.

If a design fails to meet your expectations at *Medium* effort, then increase the effort level to *High*. *High* effort utilizes the same algorithms as *Medium*, but exerts itself more fully on each step of the process. The default is *Medium*.

#### 4. If a restricted placement is desired, select the *Boundary* within which the placer must be run. Instead of the default PR boundary, choose a cluster boundary or a row region boundary, if available. If you select a non-PR boundary, *Adjust PR Boundary* is inactive.

**Note:** When a row region is selected as the Boundary, the placer places only those devices that are permitted in the selected row region.

#### 5. Choose *Adjust PR Boundary* to run the placer unconstrained by the existing PR boundary.

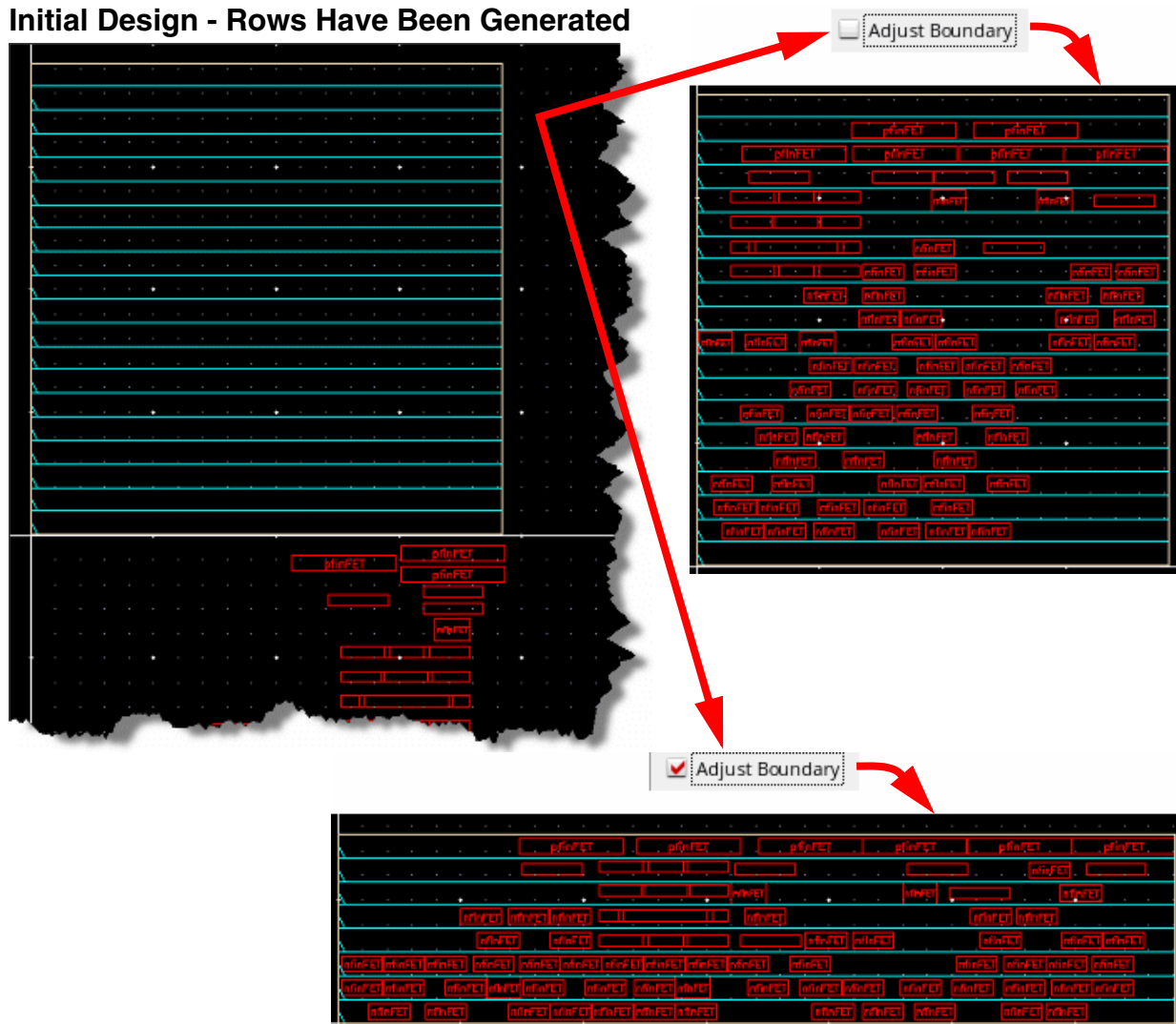
## Virtuoso Placer User Guide

### Performing Design Placement

The devices and standard cells are placed so as to achieve better QoR and wire length. After placement, the PR boundary is adjusted accordingly.

The following images depict the difference between placement with and without *Adjust PR Boundary* selected:

#### Initial Design - Rows Have Been Generated



The design is compacted on each subsequent run, and therefore the total area, wire length, and space between devices keeps decreasing with each run.

**Note:** For designs that do not have a PR boundary, if *Adjust PR Boundary* is selected, then a new PR boundary is automatically generated during placement.

6. Choose *Selected Only* to restrict placement only to the selected devices. This option is useful in situations where you have added new devices or edited existing devices after running the placer.

## Virtuoso Placer User Guide

### Performing Design Placement

7. Click *More Options* to display the *Placement Options* form. Use the options on the *Automatic* and *Common* tabs to further customize placement settings. The form includes options that are generic as well as those that are specific to the *Custom Digital* placement type. For more information about these options, see [Placement Options](#).

Placement Options	
<b>Rules</b> Constraint Group: <code>stractDefaultSetup(cdn20FF)</code> Minimum Boundary Offset: <code>0</code> <code>None</code>	Generic Options
<b>Devices</b> <input type="checkbox"/> Abut Devices <input checked="" type="checkbox"/> Preserve Abutment Chains <input checked="" type="checkbox"/> Group M Factor Devices <input type="checkbox"/> Group CMOS Pairs <input type="checkbox"/> Group P over N	Analog Mode
<b>Standard Cells</b> <input type="checkbox"/> Run Spacer Within Rows <input type="checkbox"/> Insert Tap Cells Component Type: <input type="text"/> <code>Choose...</code> Minimum Spacing: <input type="text"/> Maximum Spacing: <input type="text"/>	Custom Digital Mode
<b>Pins</b> <input checked="" type="checkbox"/> Routing Direction Aware Pin Placement <input checked="" type="checkbox"/> Pins On PR Boundary <b>Passive Devices</b> <input type="checkbox"/> Passive to Outside	Generic Options

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

8. Click *OK* or *Apply* to run the placer.

### Placement of Virtual FigGroups

Virtuoso Placer supports placement of virtual hierarchies in the *Custom Digital* and *Like Schematic* modes.

By default, virtual figGroups are placed as is. The contents of the virtual hierarchy are placed within the rectangular or rectilinear area boundary of the virtual hierarchy. The instances inside the figGroups are not modified.

## Virtuoso Placer User Guide

### Performing Design Placement

---

If a virtual figGroup is frozen, the instances inside it are not modified. The entire figGroup is placed as a unit. The placer does not break any synchronous clones that exist in a design.

When *Adjust PR Boundary* is selected in the Automatic Placer form, the boundaries of virtual hierarchies are padded based on the value of the areaBoundaryEnclosure environment variable.

After placement, whenever a stretch or chop operation is done on the area boundary of a virtual hierarchy, the *Like Schematic* placer is automatically called to ensure all the contents of the virtual hierarchy are placed within the new `areaBoundary`, which can be either rectangular or rectilinear.

# Virtuoso Placer User Guide

## Performing Design Placement

### Viewing the Placement Report

The Placement Statistics report is generated dynamically in the CIW as you run the automatic placer. The report provides the following information:

**Placement Parameters**  
Before running the placer

```
INFO (AP-10002): Started placement bias_amp layout_mod
placerType      = Refine
effort          = Medium adjustBoundary      = t
preserveChains  = t abutDevices              = nil
passiveOutside  = nil pinsOnBoundary         = t
selectedOnly    = nil boundary               = PR boundary
groupCMOSPairs  = nil groupMFactors          = t
placeP0verN     = nil useLayoutAsSeed        = nil
```

**Placement Characteristics**  
After running the placer

	Placeable	Locked or Fixed	Temp Locked	Unplaceable
Pins	10	0	0	0
Devices	16	0	0	0
Device Chains	3	0	0	0
Modgens	0	0	0	1
Mosaics	2	0	0	0

Constraints	Boundary	Rows	Symmetry	Alignment	Distance	Orientation	NetPriority
	0	20	0	0	0	0	0

Connectivity	Nets	InstPins
	24	294

**Placement Results**  
After running the placer

```
INFO (AP-10003): Completed runtime 28.37 seconds
```

	Area (micron^2)	Width (micron)	Height (micron)	Aspect Ratio	RowUtil (percent)	WireLength (micron)
Initial:	844.80	40.68	20.77	1.96	0	642.98
Solution:	466.62	35.42	13.18	2.69	7	480.07

	Violations	Spacing	Overlap	Constraint
Initial:		0	0	0
Solution:		0	0	0

**Note:** The above report is displayed only when you select the *Like Schematic* placement option. If you select the *Custom Digital* option, the Placement Status report is displayed. For information about the Placement Status report, see [VLS-GXL Placement Status](#).

The Placement Statistics report displays the following information:

- **Before running the automatic placer:** A summary of the placement parameters in the current design.
- **After running the automatic placer:**

## Virtuoso Placer User Guide

### Performing Design Placement

---

- ❑ **Placement Parameters:** A list of the number of placeable, locked or fixed, temporarily locked, and unplaceable components.

**Note:** Unplaceable components are row-based objects in the design, but none of the rows in the design match these objects. Typically this happens when there is a problem with the data setup, for example when the component definition of an object does not match any row.

- ❑ **Placement Results:** Summary of the initial placement and the final placement achieved, followed by a list of errors and warnings (if any). The number of spacing and overlap violations is also listed.

## Performing Assisted Placement

After running Automatic Placer, you might want to perform a few manual edits to some of the placed instances to improve their placement.

Start with assisted move to group devices with similar masters and similar finger lengths. No device should be adjacent to another dissimilar device. Additionally, ensure that the devices are ordered top to bottom in a manner that maintains the path of the current from the power rail to the ground rail.



### Tip

Before you move devices, in the Display Options form, set the *Show Name Of* option to *instance* or *both*.

After you have grouped the devices by type, run any of the several assisted commands to improve placement. Assisted placement commands include abutting devices, swapping devices or groups of devices, resolving overlaps, and snapping devices to grids or rows, and adjusting the boundary. Ensure that you press `Ctrl+D` to deselect all devices, and then carefully select only the ones you need before running an assisted placement command.

The *Place* menu provides the following assisted placement options:

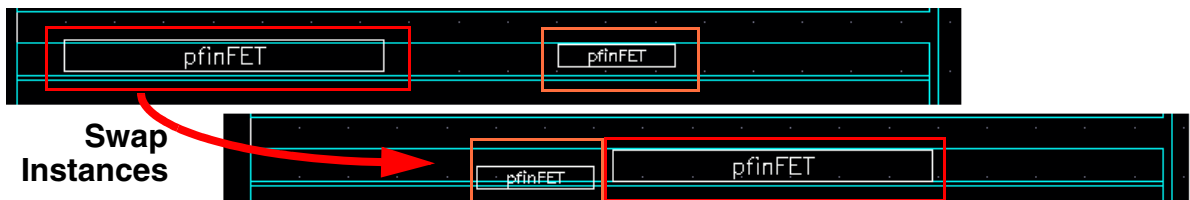
- **Resolve Overlaps:** Resolves overlaps between devices in rows, which removes any shorts that are created when you move the devices or snap them to rows.
- **Abut Instances:** Abutting devices helps avoid loss of area consumed by the extra poly fingers on either side of each device. You can abut selected or all instances in a design. If there are no instances selected before running the command, all instances in the design are abutted. The Placement Options form includes the *Abut Rows Use Device Order* option, which specifies whether abutment should follow the order in which instances were placed in each row.
- **UnAbut Instances:** Unabuts selected or all instances in the design. If no instances are selected, then all instances in the design are unabutted
- **Swap Instances:** Swaps the position of the selected instances, chains, and figGroups. To swap instances:
  - a. Choose *Place—Swap Instances*.
  - b. Click the first instance to be swapped.
  - c. Click the second instance.



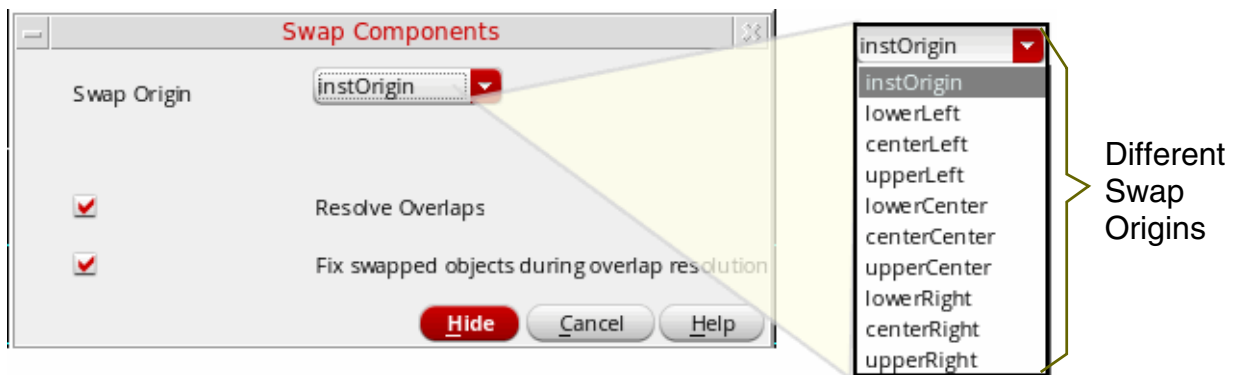
## Virtuoso Placer User Guide

### Performing Design Placement

The locations of the selected instances are switched, but their orientations are retained as in their initial position.



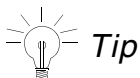
You can also select the components first and then choose *Place—Swap Instances*. After selecting the command, press **F3** to display the Swap Components form.



In this form:

- ❑ Use the *Swap Origin* option to specify the position to which the instance needs to be moved. This option is useful when there is a difference in the sizes of the instances to be swapped.
- ❑ You can choose to *Resolve Overlaps* while placing the devices at their new locations.
- ❑ Choose *Fix swapped objects during overlap resolution* to ensure that the position of the swapped device is not changed while resolving overlaps.

Related Environment variables: [highlightPotentialSwap](#), [highlightPotentialSwapColor](#)



#### Tip

For information about the generic swap command (that is available in the non-advanced node releases), see [Swapping Components](#) in *Virtuoso Layout Suite XL User Guide*.

## Virtuoso Placer User Guide

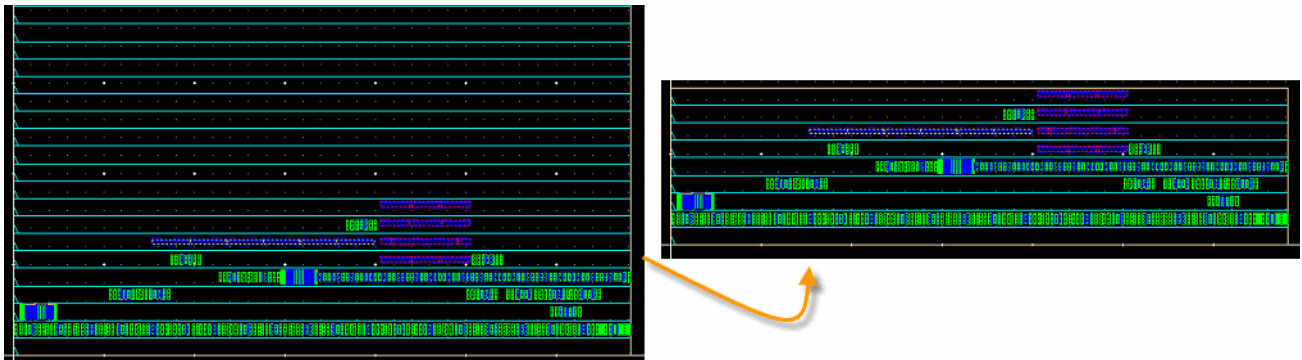
### Performing Design Placement

---

- **Swap Rows:** Swaps components of the rows that contain the selected instances, chains, or figGroups. Contents of the two rows have to be compatible to be swapped to the new row.

Boundary cells are ignored during row swapping. Locked instances, highlighted in red in the design canvas, are also ignored. All other cells in the selected row are swapped.

- **Snap to Grids/Rows:** Snaps instances to their nearest compatible rows in a Spacing Pattern (SP) or Width Spacing Patterns (WSP)-correct manner. This option is intended as a batch alternative to the assisted move commands. For more information, see [Assisted Placement](#).
- **Adjust PR Boundary:** Adjusts the PR boundary to enclose all placed instances. Unused rows are deleted, unused space is chopped, and the PR boundary is adjusted accordingly without disturbing the placement.



### Support for Transparent Group Mode

The assisted placement options support the transparent group mode. In this mode, you can select and edit individual members of user-defined groups at the top-level, while retaining them as members of their respective groups. You can edit objects in nested groups also at the top-level, without performing multiple Edit In Place operations. With *Resolve Overlaps* selected, the user-defined group instances that are at the same level as the selected member instance interact as individual members. Other groups, which are at the same or different levels, behave as groups.

For more information about the transparent mode, see [Using Transparent Group Mode](#) in the Virtuoso Layout Suite documentation.

## Snapping Behavior

Assisted and automatic placement methods support snapping of individual objects and object groups to grids based on the following priority:

1. Snap to a row region. For more information, see [Creating Rows](#).
2. Snap to a WSP grid. For more information, see [Using Width Spacing Patterns](#) in *Virtuoso Width Spacing Patterns User Guide*.
3. Snap based on the [leSnapGridHorizontal](#) and [leSnapGridVertical](#) properties.
4. Snap based on the [leSnapPatternSnapping](#) property.
5. Snap based on setting of the [snapPatternSnappingMode](#) environment variable.

### ***Controlling the Rotation of Object Groups During Placement***

The snapping behavior is impacted by the [allowRotationFigGroup](#) environment variable setting. This environment variable controls the rotation of certain complex object group types. By default, the environment variable is set to `nil`, which means that their orientation is considered fixed, and therefore they cannot be rotated during placement. For example, if the rows in a design are interposed with orientations `R0` and `MX`, a `figGroup` with orientation `R0` can be snapped only to every other row that has a matching orientation.



#### ***Tip***

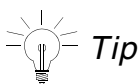
During assisted move, you can manually force a rotation of object groups using the middle mouse button or by pressing `F3`.

When [allowRotationFigGroup](#) is set to `t`, the orientations of object groups are unlocked. During placement, these objects are rotated such that their orientations match their row orientations.

### Support for Module Generators

When you run Virtuoso Placer, Modgens are automatically snapped to compatible rows, if they exist. If the Modgen instances do not align exactly to the rows, then one of the Modgen instances is snapped to the nearest row. Use the `snapFigGroupRowCorrect` to control the snapping behavior of Modgens.

Modgens that are created after the rows exist are automatically correct by construction, as shown below:



#### Tip

Modgen dummies are also snapped to nearest rows.

### Support for Modgen On-Canvas Commands

You can use the various Modgen on-canvas commands that are available under the *Place – Modgen* menu to perform simple editing operations directly in the layout editor without invoking the Modgen Editor. For example, you can use the *Split Modgen Rows* and *UnSplit Modgen Rows* commands to ensure that Modgens use as few rows as possible, while still fitting comfortably inside the PR boundary.

## Virtuoso Placer User Guide

### Performing Design Placement

For more information about these Modgen on-canvas commands, see [Using the Modgen On-Canvas Commands](#) in *Virtuoso Module Generator User Guide*.

### Support for Modgen Pattern Editors

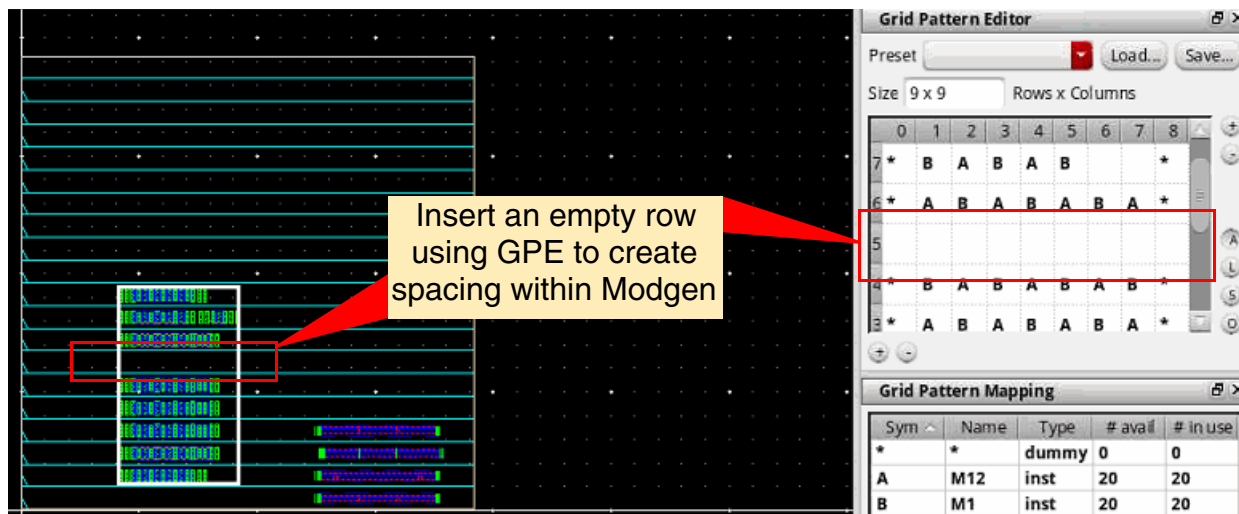
The Grid Pattern Editor (GPE) and Grid Pattern Mapping (GPM) assistants are available on-canvas. The Modgen grids directly map to the rows in the layout canvas. Therefore, whenever you make changes to the grid pattern, the Modgen instances always snap to the nearest rows.

For more information about the pattern assistants, see [Using the Modgen Editor Assistants](#) in the *Virtuoso Module Generator User Guide*.

### Inserting Space within Modgens

The alignment and spacing settings defined for the rows override the Modgen alignment and spacings settings made using the Set Member Alignment and Spacing form. Therefore, in the Virtuoso Placer flow, it is recommended that you do not use this form for making vertical alignment and spacing settings.

If you want to create an empty space within a Modgen, use the GPE to create an empty row, as shown below:



For more information about alignment and spacing settings in Modgens, see [Specifying Device Alignment and Spacing](#) in the *Virtuoso Module Generator User Guide*.

### Using Guard Rings

In the Virtuoso Placer flow, it is recommended to use either identical guard rings or MPP guard rings. Fluid guard rings can be created, but they are not compatible with the row infrastructure.

For more information about creating guard rings, see [Specifying Guard Rings](#) in the *Virtuoso Module Generator User Guide*.

## Supported Properties

### leSnapGridHorizontal

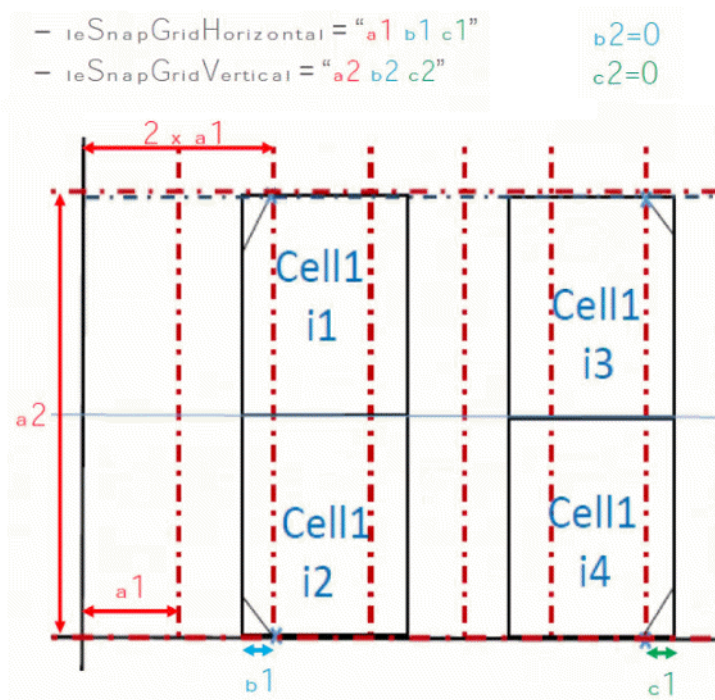
`leSnapGridHorizontal` string *f\_pitch* *f\_offset* *f\_offsetFlippedOrientation*

Defines a horizontal grid to snap devices, standard cells, block-level cells, and passives with orientations R0, MY, or R180. You can define this property either on the cell master or as an optional override on the instance.

- ***f\_pitch***: Specifies the grid pitch.
- ***f\_offset***: Specifies the grid offset.
- ***f\_offsetFlippedOrientation* (optional)**: Specifies the offset for instances with flipped orientations. If not specified, the value is considered to be equal to *f\_offset*.

**Note:** Snapping is skipped for instances that have been rotated (with orientations: R90, R270, MYR90, and MXR90).

The lower-left corner of the instance PR boundary is considered the reference for instance snapping. In the absence of a PR boundary, the instance origin is the reference.



#### Example

```
dbCreateProp(master_cellView "leSnapGridHorizontal" "string" "3 5 1")
;; Override the master's property
dbCreateProp(instance "leSnapGridHorizontal" "string" "4 5 1")
```

### leSnapGridVertical

leSnapGridVertical string *f\_pitch f\_offset f\_offsetFlippedOrientation*

Defines a vertical grid to snap devices, standard cells, block-level cells, and passives that have their orientations set to R0, MX, or R180. You can define this property either on the cell master or as an optional override on the instance.

- ***f\_pitch***: Specifies the grid pitch.
- ***f\_offset***: Specifies the grid offset.
- ***f\_offsetFlippedOrientation* (optional)**: Specifies the offset for instances with flipped orientations. If not specified, the value is considered to be equal to *f\_offset*.

**Note:** Snapping is skipped for instances that have been rotated (with orientations: R90, R270, MYR90, and MXR90).

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

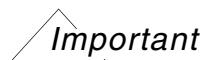
#### Example

```
dbCreateProp(master_cellView "leSnapGridVertical" "string" "3 5 1")
;; Override the master's property
dbCreateProp(instance "leSnapGridVertical" "string" "4 5 1")
```

### leSnapPatternSnapping

leSnapPatternSnapping string { off | detail | abstract { *f\_offset* | *f\_vOffset* *f\_Offset* } }

Lets you enable the Manual Mode Placement override and specify the placement mode or offset on a per instance or master basis. You can set the `leSnapPatternSnapping` property on the instance, master, or both. When set on both, the instance definition overrides the master definition.



`leSnapPatternSnapping` must be paired with either [leSnapPatternSnappingHorizontalGrid](#) or [leSnapPatternSnappingVerticalGrid](#).



Valid values are:

- **off**: No automatic snapping.
- **detail**: Force Detail Mode Placement.
- **abstract**: Force Abstract Mode Placement.
- **f\_offset**: Force Abstract Mode Placement with manual offset for either one or both directions along which snapping is done.
- **f\_vOffset f\_Offset**: Force Abstract Mode Placement with separate manual offsets. The first number indicates the vertical offset (for example, fin grid) and the second number refers to the horizontal offset (for example, poly grid).

#### **Example**

```
leSnapPatternSnapping "detail"
```

### **leSnapPatternSnappingHorizontalGrid**

```
leSnapPatternSnappingHorizontalGrid cyclic poly_grid_name
```

Specifies the horizontal reference grid to snap instances. The specified grid overrides all active snap patterns (SPs) or width spacing patterns (WSPs). If the reference grid is an SP, detailed snapping is done. If it is a WSP, period snapping is done.

#### **Example**

```
leSnapPatternSnappingHorizontalGrid "analog_poly_nom"
```

### **leSnapPatternSnappingVerticalGrid**

```
leSnapPatternSnappingVerticalGrid cyclic poly_grid_name
```

Specifies the vertical reference grid to snap instances. The specified grid overrides all active SPs or WSPs. If the reference grid is an SP, detailed snapping is done. If it is a WSP, period snapping is done.

#### **Example**

```
leSnapPatternSnappingVerticalGrid "analog_poly_nom"
```

## **Virtuoso Placer User Guide**

### **Performing Design Placement**

---

---

## Placement Post-Processing

---

This chapter covers tasks that are performed after achieving the required placement results. To achieve design correctness and conformance to advanced node layout constraints, you can insert dummy fill in the empty spaces in rows. Depending on your design type, you can insert the following types of dummy fill:

- **Filler cells:** Inserted between standard cells in digital designs.
- **Dummy fill:** Inserted between devices in analog designs.

After completing all placement tasks, use the *Batch Checker* to verify whether the placement satisfies placement constraints.

Related documentation:

- [Insert/Delete Filler Cells](#)
- [Insert/Delete Dummy Fill](#)
- [Verify Placement](#)

## Insert/Delete Dummy Fill

Insert/Delete Dummy Fill is an analog fill utility available only at advanced nodes. Dummy fill cells are inserted in the empty spaces in each row without impacting the row size.

This chapter includes the following sections:

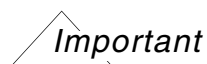
- [Prerequisites](#)
- [Types of Fill](#)
- [Inserting Fill](#)
- [Adding New Fill Definitions](#)

### Prerequisites

Before inserting dummy fill, ensure that:

- All active devices are placed in rows. Align the devices with matching concerns down the center of the rows for better symmetry.
- The rows do not include devices with different finger lengths.
- An empty row is maintained between devices with different lengths.

Depending on the placement that you have created, you might need to stretch the row region to create the required number of rows for the fill.



The second and third prerequisites are advanced node constraints that might differ depending on your particular PDK.

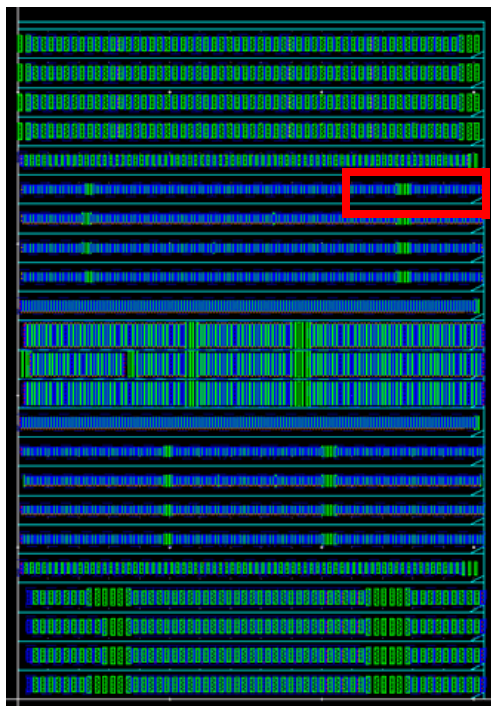
**Note:** Transition and adjacent fill can be added even when poly trim layers do not exist in the technology file.

## Types of Fill

Dummy fill can be classified into the following three types:

- **Adjacent fill:** Is a type of device fill inserted to the left side of devices, as shown in the snapshot below. These instances are created with all their parameters equal to those of the active devices. Adjacent fill are automatically snapped to the nearest compatible rows in a Spacing Patterns (SP) or Width Spacing Patterns (WSP)-correct manner.

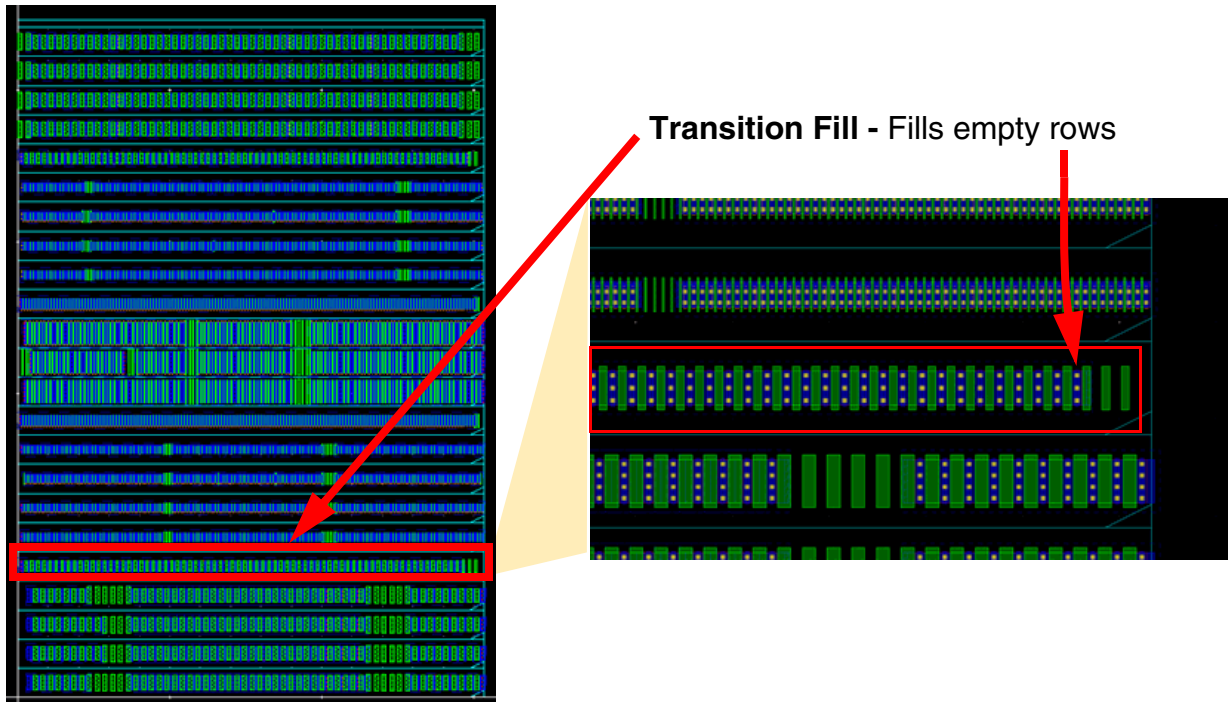
Inserting adjacent fill helps achieve better active layer density.



**Adjacent Fill** - Fills empty spaces within rows

- **Transition fill:** Is a type of device fill inserted in empty rows to mitigate the effects that a type of device might have on other devices. The transition fill utility lets you insert devices between rows of devices. Transition fill cells are automatically snapped to the nearest compatible rows in an SP or WSP-correct manner. In certain advanced node processes, active devices are surrounded by an entire region of same-sized dummy fill to achieve proper matching. In cases where this is not practical, a transition region can be used to mitigate potential mismatch. In the snapshots below, the parameters of transition fill are

the same as the neighboring devices, except that the finger length of the transition fill equals the median of the surrounding rows.



- **Poly fill:** Extends polysilicon fingers vertically to the middle of the nearest cut-poly rail. This helps achieve better poly density, as specified by the foundry and mitigates density gradient effects (DGE).

The required poly density threshold is usually much higher than the amount of poly on the instances alone. Adding poly fill satisfies this requirement.

Poly fill inherently require cut poly rails to mitigate the shorts caused due to extending the polysilicon fingers. The following two types of cut-poly rails are required:

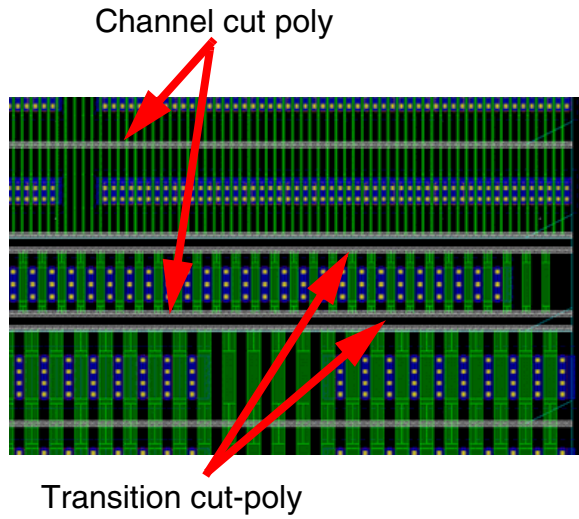
- ❑ **Channel Cut-Poly rails:** Separate polysilicon fingers between rows to avoid gates from being shorted.
- ❑ **Transition Cut-Poly rails:** Are additional cut-poly rails that surround transition devices.

## Virtuoso Placer User Guide

### Placement Post-Processing

---

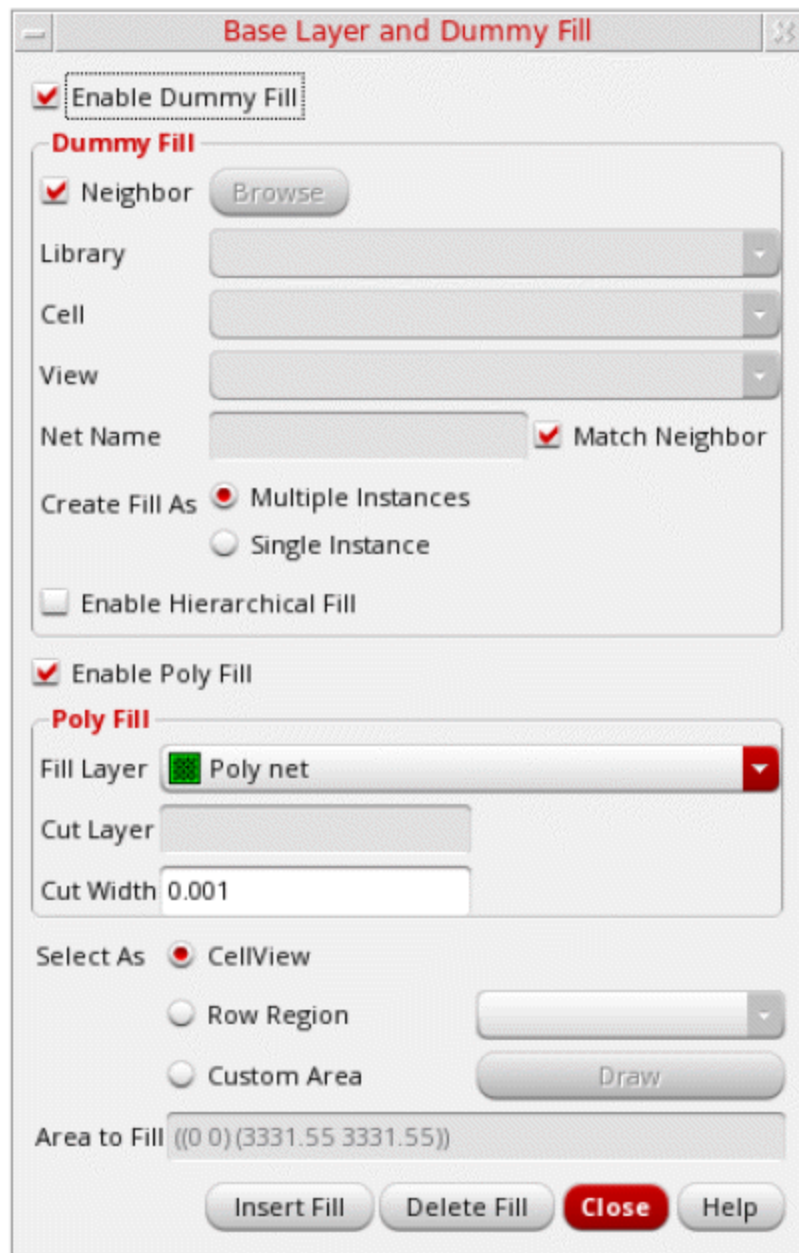
In the snapshots below, notice that a channel cut-poly is created between each row of devices. Also, extra transition cut-poly rails surround the transition rows.



## Inserting Fill

Use the options in the Insert/Delete Dummy Fill form to insert dummy fill in your design. To insert dummy fill, open the Insert/Delete Dummy Fill form by selecting *Place—Fill—Base Layer Dummy Fill*.

The Base Layer and Dummy Fill form appears.



The image shows a dialog box titled "Base Layer and Dummy Fill". It contains several sections for configuring dummy and poly fill options.

- Enable Dummy Fill:** A checked checkbox.
- Dummy Fill Section:**
  - Neighbor:** A checked checkbox with a "Browse" button next to it.
  - Library:** A dropdown menu.
  - Cell:** A dropdown menu.
  - View:** A dropdown menu.
  - Net Name:** A text input field.
  - Match Neighbor:** A checked checkbox.
  - Create Fill As:** Two radio buttons: "Multiple Instances" (selected) and "Single Instance".
  - Enable Hierarchical Fill:** An unchecked checkbox.
- Enable Poly Fill:** A checked checkbox.
- Poly Fill Section:**
  - Fill Layer:** A dropdown menu showing "Poly net" with a green square icon.
  - Cut Layer:** A text input field.
  - Cut Width:** A text input field containing "0.001".
- Select As:** Three radio buttons: "CellView" (selected), "Row Region", and "Custom Area".
- Area to Fill:** A text input field containing "((0 0) (3331.55 3331.55))".
- Buttons:** "Insert Fill", "Delete Fill", "Close" (highlighted in red), and "Help".



## **Virtuoso Placer User Guide**

### **Placement Post-Processing**

---

The options in the Base Layer and Dummy Fill form are the same as on the Fill tab of the Auto Device P&R assistant. For more information, see [Adding Base Layer Fill](#).

Before running the command, select the instances for which dummy fill must be inserted. If no instances are selected, all instances that are currently placed within the PR boundary are selected for dummy fill insertion.

## Adding New Fill Definitions

New fill definitions can be added to the drop-down list by defining and registering a SKILL procedure. A fill definition can specify the master to be used for the fill devices and set parameters on those devices.

**Note:** The fill definition `diffusionLPP` parameter can be either a single layer-purpose pair or a list of layer-purpose pairs. The fill definition `fingerLPP` parameter is always a list of layer-purpose pairs.

Here is an example of a SKILL procedure:

```
(
  name "ptap" <- Always a string
  diffusionLPP ("Active" "drawing") or (("Active" "drawing") ("Active" "dummy")) <- Either a
  single lpp pair or a list of lpp pairs
  fingerLPP (("Poly" "drawing")) <- Always a list of lpp pairs
  libName "cdn2FF" <- Strings indicating LCV
  cellName "ptap"
  viewName "layout"
  type "instance" <- For any non-identical fill, this is always "instance"
  params (("nFins" "5") ("1" "20n") ("Param" "paramFunc")) <- A list of lists containing string
  names of parameters and strings representing the value, or a string with the name of a function to
  call
  createAsMosaic "t" <- Either "t" or "nil" indicating whether to create fill as mosaic or by
  modifying finger count
  mosaicFunc "calcDxNumCols()" <- Function to run to calculate mosaic dx value. Function returns
  DX in DBU
)
```

Use the following SKILL functions to register the SKILL procedures:

- **lobRegisterAdjacentFillDefsProc**: Registers the specified user function symbol as the adjacent fill definition procedure.
- **lobRegisterTransitionFillDefsProc**: Registers the specified user function symbol as the transition fill definition procedure.
- **lobRegUserProc**: Registers a user-defined function for a specific purpose, as needed by layout objects. This function is currently used to register procedures to retrieve adjacent fill definitions.

Use the following SKILL functions to retrieve names of registered procedures:

## Virtuoso Placer User Guide

### Placement Post-Processing

---

- **lobGetRegisteredAdjacentFillDefsProc**: Returns the name of the function that is registered as the adjacent fill definition procedure.
- **lobGetRegisteredTransitionFillDefsProc**: Returns the name of the function that is registered as the transition fill definition procedure.
- **lobGetRegUserProc**: Returns a list of strings that contain the name of the specified registered user procedure.

Use the following SKILL functions to unregister procedures:

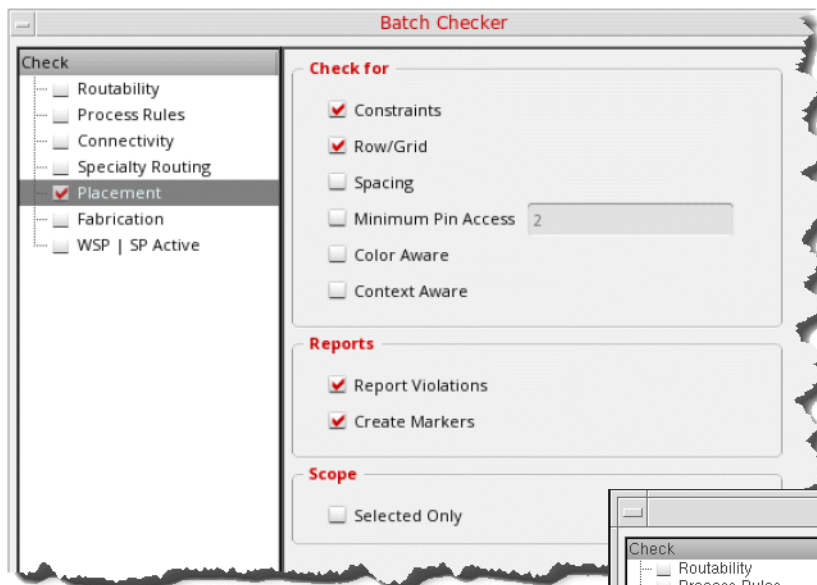
- **lobUnRegisterAdjacentFillDefsProc**: Unregisters the specified user function symbol, which is currently registered as the adjacent fill definition procedure.
- **lobUnRegisterTransitionFillDefsProc**: Unregisters the specified user function symbol, which is currently registered as the transition fill definition procedure.
- **lobUnRegUserProc**: Accepts user functions based on keyword and unregisters them.

## Verify Placement

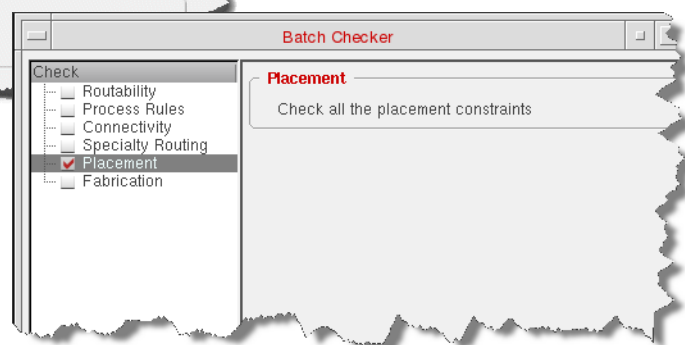
After performing all the placement tasks, you can verify whether the placement is DRC-correct. Use the *Batch Checker* form to verify constraints and width spacing pattern (WSP)-specific settings in your design.

1. Choose *Verify – Design* to open the Batch Checker form.
2. Choose *Placement* in the *Check* panel. The *Placement* options are displayed in the right panel of the form. The options on the *Placement* tab available in Virtuoso Layout EXL cockpit in advanced node releases is different from the options available in all other releases. The following snapshots show the differences:

### Advanced Node Layout EXL Only



### Other Releases



These placement checks let you verify the constraints and width spacing pattern (WSP)-specific settings in your design. To run these checks:

3. In the *Check For* group box, choose the following:
  - ☐ *Constraints*

## Virtuoso Placer User Guide

### Placement Post-Processing

---

Checks for conformance with the following constraints, which Virtuoso Placer honors:

- ☐ Alignment
- ☐ Orientation
- ☐ Symmetry
- ☐ Distance

**Note:** You can use Cadence Physical Verification System (PVS) Constraint Validation (CV) to run a comprehensive check of all placement and routing constraints. For more information, see [Running PVS-CV](#) in *Cadence Physical Verification User Guide*.

☐ *Row/Grid*

Checks whether instances are snapped to the correct rows or grids.

☐ *Spacing*

Checks instances for conformance with spacing rules that are set, for example the layer spacing rules defined in the technology file and row spacing rules defined in the Row Template Manager.

☐ *Minimum Pin Access*

Checks WSP tracks for conformance with the *Minimum Pin Access Count* specified on the *Misc* tab of the Placement Options form. For more information, see [Miscellaneous Options](#).

☐ *Color Aware*

Checks shapes for conformance with color locks assigned to them, if any.

☐ *Context Aware*

Verifies whether context-aware placement has been done for standard cells and whether the edge type and spacing constraints of tap cells have been honored during placement.

4. In the *Reports* section, choose one or both the following options:

☐ *Report Violations*

Displays detailed error and warning messages in the CIW reporting all violations that were detected while running batch checker.

☐ *Create Markers*

## Virtuoso Placer User Guide

### Placement Post-Processing

---

Displays markers in the design to indicate the locations where violations are reported.

5. In the *Scope* section, choose *Selected Only* to run batch checker only on selected objects in the design. Clear the check box to run batch checker on the entire design.

---

## Virtuoso Placer Forms

---

This section lists the controls in the Virtuoso® Placer forms.

**Note:** Many of the options described in this section have a corresponding environment variable. For a full listing of these environment variables, see [Virtuoso Placer Environment Variables](#).

### Virtuoso Placer Forms:

- [Automatic Placement](#)
- [Batch Checker \(Placement Tab\)](#)
- [Placement Options](#)
- [Row Template Manager](#)

## Automatic Placement

Use the **Automatic Placement** form to select the type of placement required, set the corresponding options, and run the placer.

**Placement Type** provides placement options for digital and analog designs.

**Custom Digital** places standard cells in the design. You can select the *ECO* mode to make incremental updates, for example to place only the unplaced devices in the design.

**Like Schematic** places devices as closely as possible to their relative locations in the schematic view.  
Environment variable: placerType

**Options** lets you set high-level options depending on the placement type. Click *More Options* to open the Placement Options form where you can refine the settings further.

**Effort** provides a choice of three placement effort levels: *Low*, *Medium*, and *High*. The runtime of the placer depends on the effort chosen. For example setting the effort level to Low results in lower effort, which means shorter runtime and potentially less good results. The time for each effort level increases in roughly linear increments.  
Environment variable: effort

**Boundary** defines the region within which the placer must run. Default value is prBoundary. Non-PR boundaries such as cluster boundaries and row regions are also supported. Environment variable: placeCluster

**Adjust PR Boundary** automatically adjusts the PR boundary to enclose all the placed objects. This option is not available when a non-PR boundary is selected. Environment variable: adjustBoundary

**Selected Only** places only the objects currently selected in the design. When unchecked, all the objects in the design are placed.

### ***Related Topics:***

[Automatic Placement](#)

## ***Related Topics***

[Running the Automatic Placer](#)

[Setting Placement Options](#)



## **Batch Checker (Placement Tab)**

Use the **Batch Checker** form to run various design checks. The following section describes the options on the *Placement* tab of the Batch Checker form. For information about other tabs, see [Batch Checker Form](#) in the *Virtuoso Layout Suite XL: Connectivity Driven Editing* user guide.

### **Placement Tab**

#### **Check For**

**Constraints** checks all elements in the design for conformance with the Alignment, Orientation, Symmetry, and Distance constraints, if set.

Environment variable: [constraints](#)

**Row/Grid** checks whether instances are snapped to the correct rows or grids.

Environment variable: [rowOrGrid](#)

**Spacing** checks instances for conformance with applicable spacing rules.

Environment variable: [spacing](#)

**Minimum Pin Access** checks WSP tracks for conformance with the *Minimum Pin Access Count* specified on the *Misc* tab of the Placement Options form.

Environment variable: [minPinAccess](#)

**Color Aware** checks shapes for conformance with color locks assigned to them, if any.

Environment variable: [colorAware](#)

**Context Aware** verifies whether context-aware placement has been done for standard cells and whether the edge type and spacing constraints of tap cells have been honored during placement.

## Reports

**Report Violations** displays detailed error and warning messages in the CIW reporting all violations that were detected while running batch checker.

Environment variable: [reportViolations](#)

**Create Markers** displays markers in the design to indicate the locations where violations are reported.

Environment variable: [createMarkers](#)

## Scope

**Selected Only** runs batch checker on the selected instances only.

Environment variable: [selOnly](#)

## ***Related Topics***

[Verify Placement](#)

[Virtuoso Placer Environment Variables](#)

## Placement Options

Use the **Placement Options** form to control how various assisted and automatic placement tasks are to be performed.

**Assisted** provides the following options to assist you while making manual edits to objects in the layout canvas.

### ***Related Topics:***

[Setting Placement Options](#)

**Assisted Placement** provides advanced assisted features while moving objects in the layout. For information about these features, see [Assisted Placement](#). Environment variable: [apMove](#)

**Enable** switches on the Resolve Overlap and its associated options.

**Show Info Balloon** displays an info balloon when objects are moved in the layout, which provides information about the wire length and snapping behavior. Environment variable: [infoBalloon](#)

**Resolve Overlaps** automatically resolves overlaps between instances after they are edited in the layout canvas by moving the instances to the left or right. Environment variables: [displayResolveOverlapsColor](#), [fixAdjustedForResolveOverlaps](#), [snapColor](#)

**Spread Any Direction** moves the overlapped device by a minimum distance in any direction.

**Spread X-Direction Only** moves the overlapped device by a minimum distance along the right or left directions only. Environment variables: [spreadType](#)

**Insert Mode** moves the overlapped device to the right only.

**Insert Mode Single Row** moves the overlapped device to the right. The instances that are moved are placed on the same row. Environment variables: [snapColor](#), [insertColor](#), [insertDefaultSpreadType](#), [paddingForCells](#)

**Insert Mode Single Column** stacks the selected devices vertically before moving them. Environment variable: [spreadType](#)

**Abut Rows Use Device Order** controls whether abutment is based on connectivity or on the order of devices in the cellview (default). Environment variable: [useDeviceOrder](#)

**Rules** section specifies the following placement rules:

**Constraint Group** specifies the technology file constraint group to be used for applying spacing rules.

**Minimum Boundary Offset** specifies the minimum offset of instances from the PR boundary to a reference layer. Any layer inside the standard cell can be assigned as the reference layer. Specify the offset value and choose the required layer from the drop-down list.

***Related Topics:***

[Setting Placement Options](#)

**Automatic** specifies settings for the automatic placer.

**Devices** section includes the following options applicable only to analog designs.

**Abut Devices** automatically chains devices during placement. Environment variable: [abutDevices](#)

**Preserve Abutment Chains** preserves existing chains during placement. This option cannot be used when *Abut Devices* is selected.

**Group M Factor Devices** and **Group CMOS Pairs** group mfactor devices and CMOS pairs defined to share connection.

**Group P over N** places all P-devices to the top and N-devices to the bottom of the selected region. Environment variable: [placePOverN](#)

**Standard Cells** section includes the following options applicable only to digital designs.

**Run Spacer Within Rows** distributes standard cells evenly in the rows by adding or removing space between them within each row. Selecting this option ensures that there is sufficient space to route between the standard cells in the rows.

**Insert Tap Cells** inserts tap cells in rows. The following options are available:

**Component Type** specifies the names of the component types that contain the substrate contacts (tap cells). The specified component type must have the component class STDSUBCONT. Click *Choose* to select from a list of available component types.

**Minimum Spacing** and **Maximum Spacing** specify the minimum and maximum contact spacing values.

**Pins** section defines the following pin placement settings:

**Routing Direction Aware Pins** specifies whether the placer must consider the WSP direction while placing unconstrained pins. This option is enabled by default.  
Environment variable: [routingAwarePinPlc](#)

**Pins on Boundary** places all pins on the PR boundary.

Environment variable: [pinsOnBoundary](#)

***Related Topics:***

[Setting Placement Options](#)

**Passive Devices** section specifies the following settings for passive devices:

**Passive to Outside** pushes passive devices such as large capacitors and resistors to the edge of the PR boundary. for more information, see [Define Passive Device Settings](#). Environment variable: [passiveOutside](#)

**Common** specifies the following settings that are applicable to both, *Assisted* and *Automatic* modes:

**General** section settings are applicable to both devices and standard cells.

**Width Spacing Pattern (WSP) Snapping** enables snapping to WSP tracks.  
Environment variable: [WSPAware](#)

**Cell Boundary LPP(s)** specifies the layer-purpose pairs (LPPs) that the placer uses to derive the cell boundary for standard cells and devices when running Place Like Schematic, Analog Placer, and Digital Placer. Example: ([Metal1](#) drawing).

**Modgen** section settings are applicable to all Modgens that are either located in or are moved to the row regions.

**Update MODGENS to Rows Post-Placement** specifies whether a Modgens regenerates after it is moved to a row region. The orientations of the Modgen members are updated to match the orientations of the rows to which they are moved. Environment variable: [regenModgenPostProcess](#)

**Device** section settings are applicable only to devices.

**N-Diffusion Spacing** and **P-Diffusion Spacing** specify the diffusion spacing for the PMOS and NMOS transistors within the row.

**Standard Cell** section settings are applicable only to standard cells.

**Context Aware Placement** specifies that the placement of standard cells should be such that each cell is aware of the context of the neighboring cell and supports placement in accordance with the EDGETYPE and CELLEDGESPACINGTABLE LEF5.8 constraints, which are either defined in the technology and macro LEF, or provided using an XML file. By default, the Context Aware Placement option is OFF.

**Context File** specifies the XML file containing the edge spacing constraint information for performing context-aware placement.

**Pin Density Aware Placement** considers the pin density of standard cells to optimize the placement and to improve routability.

**Maximize Pin Access** maximizes the number of WSP tracks that overlap the pins.

**Minimum Pin Access Count** specifies the minimum number of pin access points to achieve during the process of pin access maximization.

**Cell Padding Value(s)** specifies the space to be left on either side of the master.  
Environment variable: [paddingForCells](#)

### ***Related Topics***

[Setting Placement Options](#)

## Row Template Manager

Use the **Row Template Manager** to create new row templates with different configurations, edit existing row templates, import row templates from other cellviews, and delete row templates.

**Note:** The Row Template Manager cannot be used to generate rows in the layout canvas. Use the **Placement Planning** form to generate rows based on these row templates.

### ***Related Topics:***

[Creating Rows](#)

**Edit** tab provides the following options to create new or edit existing row templates.

**Template Name** indicates the name of the current row template. To create a new template, click *New* and edit the template name if required; to edit an existing template, select an entry from the list and update the definition as needed; or to delete an entry, select a template from the list and click *Delete*.

**Template Period** specifies the height of the row template, which is the distance after which the template repeats. Alternatively, specify the number of times the row template must be generated in the combo box beside the *Template Height* field, and the *Template Period* is calculated automatically.

**Auto Compute Minimum Period** automatically calculates the *Template Period* based on the specified *Row Height* values.

**Region Type** specifies a RowRegionSpec (RRS) type attribute from which settings can be inherited.

**Placement Grid** provides the following options:

**Vertical Grid** and **Horizontal Grid** specify the reference grids (along the y axis and x axis respectively) for snapping the devices during placement.

**Vertical Offset** and **Horizontal Offset** specify the reference row offset values (along the y axis and x axis respectively) that must be applied to rows after they are snapped to the reference grids. The offset value is a correction factor for the SP.

**Related Snap Pattern** creates a WSP region in addition to the row region.

**Rows** table defines the rows in the template. These values can be edited directly in the table.

**Auto-compute row offset** automatically calculates and updates the row offsets whenever rows are modified.

**Row Name** specifies a user-defined value that uniquely identifies a row.

**Row Offset** specifies the offset of the row from the bottom of the template. The default row offset value is 0.

**Row Orientation** specifies the default orientation of the row. Row orientation **MX** indicates a flipped row. The default row orientation is **R0**.

**Row Height** specifies the height of the row. A template can have rows of different heights. The row height must be a positive integer value. The combo box beside the *Row Height* field shows the row height as a multiple of the *Vertical Grid*' period. This combo box is available only when you have specified the *Vertical Grid*. The default row height is equal to the period of the vertical grid or 0 (if the vertical grid is not set).

***Related Topics:***

[Creating Rows](#)

**Site Def** lists all the site definitions from the technology file. Select the required site definition.

**Note:** *Row Height* and *Site Def* are mutually exclusive. If a *Site Def* is specified, the row attributes in the siteDef are honored. All row values specified in this form, such as *Row Height*, are overridden by the values in the siteDef.

**Background LPP** fills the given row with a rectangle on the specified layer-purpose pair (LPP).

**Note:** If the *Use Layer Palette LPPs Only* option is selected in Layout Editor Options form, then the *Background LPP* option is synchronized with the associated window palette.

**Row Attributes** defines the component types and rails for individual rows in the row template. Select a row in the *Rows* table to display the following values in the *Row Attributes* section.



**Component Type Definition** lists the component type definitions for the selected row. You can edit the following values in the table.

**Types** specifies the component types as defined in the Configure Physical Hierarchy (CPH) form.

**Masters** specifies the master cellview of the components. Masters and component types are mutually exclusive. Therefore, specify the masters only when the component types are not specified. You can specify either a single master cellview or only the `lib`, `cell`, or `view` value. For example, *Master* can be set to `*/pfinFet/*` for a p-type row or `*/nfinFet/*` for a n-type row. Default is `*/*/*`.

**Orientations** defines the valid orientations for devices and component types in each row (relative to the *Row Orientation*). You can choose multiple orientations for a row. Click (...) to open the Select Allowed Orientations form, where you can choose the required orientations.

**Align Reference** specifies the alignment for devices in relation to the row.

**Align Offset** specifies the offset from the *Align Reference* specified above.

**InstPitch** creates a grid such that instances can be placed only at a distance that is a multiple of the *InstPitch* apart. When specified, the value overrides any values specified for the *Site Def* and *Placement Grid* options.

**Rail Definition** contains a list of rail definitions for the row. The following rail attributes must be specified.

***Related Topics:***

[Creating Rows](#)

**Net** specifies the rail connectivity.

**WSP Track** specifies a WSP track within the template period range based on which the rail layer, width, and offset values are automatically set. *Align Offset* of the rail is set to BOTTOM if a WSP track is selected.

**LPP** specifies the metal layer-purpose pair on which the rail is drawn in the canvas.

**Width** specifies the rail width.

**Align Reference** specifies the reference point for aligning the rail in relation to the row.

**Align Offset** specifies the offset of the rail from the *Align Reference* set above.

**Import** tab lets you import a row template from the same or a different cellview.

**Library**, **Cell**, and **View** specify the cellview containing the row template to be imported.

**Templates** lists all row templates stored in the selected cellview. Select the templates to be imported.

**Active Templates** in the *Current Cellview* section lists the row templates that are available in the current cellview.

**Import Selected** imports all templates that are selected in the Templates list box.

### ***Related Topics***

[Creating Rows](#)

---

## Virtuoso Placer Environment Variables

---

This appendix provides information on the names, descriptions, and graphical user interface equivalents for the Virtuoso® Placer.

Only the environment variables documented in this chapter are supported for public use. All other placer 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 the placer are set in Layout XL. Information on these environment variables is not duplicated in this section.

For more information, see [Environment Variables](#) in the Virtuoso Layout Suite documentation.

### **Virtuoso Placer Environment Variables**

- abutDevices
- adjustBoundary
- effort
- likeSchematicTolerance
- passiveOutside
- pinsOnBoundary
- placePOverN
- placerType

### **Assisted Placement Environment Variables**

- allowRotationFigGroup
- apMove
- displayResolveOverlapsColor
- displaySnappedInfo
- fixAdjustedForResolveOverlaps
- highlightSnappableRowsForInst
- highlightSnappableRowsForProxy
- infoBalloon
- insertColor
- insertDefaultSpreadType
- largeModgenEnvelopeCheck
- largeModgenEnvelopeThreshold
- maximizePinAccess
- minNumPinAccessPts
- numSnappableRows
- paddingForCells

## Virtuoso Placer User Guide

### Virtuoso Placer Environment Variables

---

- pinDensityAwarePlacement
- regenModgenPostProcess
- snapColor
- snapFigGroupRowCorrect
- spreadEnabled
- spreadType
- swapRowMode
- useDeviceOrder
- useLayoutAsSeed
- useLEReference

#### Dummy Fill Insertion Environment Variables

- lobFillAsMosaic
- lobDevFillPhysOnly

#### Placement Checker Environment Variables

- constraints
- rowOrGrid
- spacing
- tapCell
- padding
- minPinAccess
- colorAware
- reportViolations
- createMarkers
- selOnly

## Virtuoso Placer Environment Variables

### abutDevices

```
layoutXL.AP abutDevices boolean { t | nil }
```

#### Description

Automatically abuts devices during placement.

**Note:** `abutDevices` cannot be used when *Preserve Abutment* on the *Automatic* tab of the Placement Options form is selected.

The default is `nil`.

#### GUI Equivalent

Command	<i>Place – Placement Options</i>
Field	<i>Automatic – Devices – Abut Devices (<u>Placement Options</u>)</i>

#### Examples

```
envGetVal("layoutXL.AP" "abutDevices")
envSetVal("layoutXL.AP" "abutDevices" 'boolean t)
envSetVal("layoutXL.AP" "abutDevices" 'boolean nil)
```

#### ***Related Topics***

[Virtuoso Placer Environment Variables](#)

## **adjustBoundary**

```
layoutXL.AP adjustBoundary boolean { t | nil }
```

### **Description**

Automatically adjusts the PR boundary where necessary during placement so that it encloses all the components in the design.

The default is `nil`.

### **GUI Equivalent**

Command	<i>Place – Auto Placer</i>
Field	<i>Adjust PR Boundary (<u>Automatic Placement</u>)</i>

### **Examples**

```
envGetVal("layoutXL.AP" "adjustBoundary")  
envSetVal("layoutXL.AP" "adjustBoundary" 'boolean t)  
envSetVal("layoutXL.AP" "adjustBoundary" 'boolean nil)
```

### ***Related Topics***

[Virtuoso Placer Environment Variables](#)

## effort

```
layoutXL.AP effort cyclic { "Low" | "Medium" | "High" }
```

### Description

Specifies the effort level for running the placer. The runtime of the placer and quality of results depend on the effort chosen. Valid values are:

- **Low:** Results in a comparatively less precise, but quick placement.
- **Medium:** Results in slightly better placement results, but takes more time than the `Low` effort level.
- **High:** Results in the best placement results, but takes more time than the `Low` and `Medium` effort levels.

The default is `Medium`.

### GUI Equivalent

Command	<i>Place – Auto Place</i>
Field	<i>Options – Effort (<u>Automatic Placement</u>)</i>

### Examples

```
envGetVal("layoutXL.AP" "effort")  
envSetVal("layoutXL.AP" "effort" 'cyclic "Low")  
envSetVal("layoutXL.AP" "effort" 'cyclic "High")
```

### Related Topics

[Virtuoso Placer Environment Variables](#)



## **likeSchematicTolerance**

```
layoutXL.AP likeSchematicTolerance 'int "Tolerance_value"
```

### **Description**

Specifies the tolerance level to be applied when the placerType environment variable is set to PLS. Larger tolerance levels allows the placer to use more approximate schematic correspondence to improve the placement. Valid values are 1 through 1000.

The default value is 0.

### **Argument**

*Tolerance\_value*

Non-zero positive integer specifying the tolerance level to be applied.

### **GUI Equivalent**

None

### **Examples**

```
envGetVal("layoutXL.AP" "likeSchematicTolerance")  
envSetVal("layoutXL.AP" "likeSchematicTolerance" 'int 2)
```

### ***Related Topics***

[Virtuoso Placer Environment Variables](#)

## **passiveOutside**

```
layoutXL.AP passiveOutside boolean { t | nil }
```

### **Description**

Pushes large capacitors and resistors to the edge of the boundary.

The default is `nil`.

### **GUI Equivalent**

Command	<i>Place – Placement Options</i>
Field	<i>Automatic – Passive Devices – Passive to Outside (<u>Placement Options</u>)</i>

### **Examples**

```
envGetVal("layoutXL.AP" "passiveOutside")  
envSetVal("layoutXL.AP" "passiveOutside" 'boolean t)  
envSetVal("layoutXL.AP" "passiveOutside" 'boolean nil)
```

### ***Related Topics***

Virtuoso Placer Environment Variables

## **pinsOnBoundary**

```
layoutXL.AP pinsOnBoundary boolean { t | nil }
```

### **Description**

Places all pins on the PR boundary.

The default is `t`.

### **GUI Equivalent**

Command      *Place – Placement Options*

Field          *Automatic – Pins – Pins On Boundary ([Placement Options](#))*

### **Examples**

```
envGetVal("layoutXL.AP" "pinsOnBoundary")  
envSetVal("layoutXL.AP" "pinsOnBoundary" 'boolean t)  
envSetVal("layoutXL.AP" "pinsOnBoundary" 'boolean nil)
```

### ***Related Topics***

[Virtuoso Placer Environment Variables](#)

## placePOverN

```
layoutXL.AP placePOverN boolean { t | nil }
```

### Description

Places all P-devices to the top and N-devices to the bottom of the selected region.

The default is `nil`.

### GUI Equivalent

Command      *Place – Placement Options*

Field          *Automatic – Devices – Group P over N (Placement Options)*

### Examples

```
envGetVal("layoutXL.AP" "placePOverN")
envSetVal("layoutXL.AP" "placePOverN" 'boolean t)
envSetVal("layoutXL.AP" "placePOverN6" 'boolean nil)
```

### ***Related Topics***

Virtuoso Placer Environment Variables

## placerType

```
layoutXL.AP placerType 'cyclic { "Digital" | "Digital ECO" | "Analog Auto" | "PLS"  
    | "Place" }
```

### Description

Specifies the placement type.

- **Digital:** Places all the standard cells in the design.
- **Digital ECO:** Places only the unplaced standard cells in the design.
- **Analog Auto:** Uses the Virtuoso device-level automatic placer.
- **PLS:** Places devices as closely as possible to their relative locations in the schematic view.
- **Place:** Places all standard cells and devices.

The default is `Digital`.

### GUI Equivalent

Command	<i>Place – Auto Place</i>
Field	<i>Placement Type (<u>Automatic Placement</u>)</i>

### Examples

```
envGetVal("layoutXL.AP" "placerType")  
envSetVal("layoutXL.AP" "placerType" 'cyclic "Digital")  
envSetVal("layoutXL.AP" "placerType" 'cyclic "PLS")
```

### ***Related Topics***

[Virtuoso Placer Environment Variables](#)

## Assisted Placement Environment Variables

### allowRotationFigGroup

```
layoutXL.APAssist allowRotationFigGroup 'boolean { t | nil }
```

#### Description

Controls rotation of user-created objects during assisted and automatic row-based placement.

The default is `nil`.

#### GUI Equivalent

None

#### Examples

```
envGetVal("layoutXL.APAssist" "allowRotationFigGroup")
envSetVal("layoutXL.APAssist" "allowRotationFigGroup" 'boolean t)
envSetVal("layoutXL.APAssist" "allowRotationFigGroup" 'boolean nil)
```

#### ***Related Topics***

[Assisted Placement Environment Variables](#)

## apMove

```
layoutXL.APAssist apMove 'boolean { t | nil }
```

### Description

Enables the advanced assisted features while moving or copying objects such as devices, standard cells, figGroups, chains, mosaics, Modgens, and pins in the layout. It also snaps instances, figGroups, and blocks to rows and snap patterns.

**Note:** The snapColor can be set to `t` only if `apMove` is set to `t`.

The default is `t`.

### GUI Equivalent

Command	<i>Place – Placement Options</i>
Field	<i>Assisted – Assisted Placement – Enable (<u>Placement Options</u>)</i>

### Examples

```
envGetVal("layoutXL.APAssist" "apMove")  
envSetVal("layoutXL.APAssist" "apMove" 'boolean t)  
envSetVal("layoutXL.APAssist" "apMove" 'boolean nil)
```

### Related Topics

[Assisted Placement Environment Variables](#)

## **displayResolveOverlapsColor**

```
layoutXL.APAssist displayResolveOverlapsColor string "arrow_color"
```

### **Description**

Specifies the color of arrows and boxes that are displayed to indicate the direction of movement of instances to resolve overlaps.

The default is `nil`.

### **GUI Equivalent**

None

### **Examples**

```
envGetVal("layoutXL.APAssist" "displayResolveOverlapsColor")  
envSetVal("layoutXL.APAssist" "displayResolveOverlapsColor" 'string "magenta")
```

### ***Related Topics***

[Assisted Placement Environment Variables](#)

[Placement Options](#)



## **displaySnappedInfo**

```
layoutXL.APAssist displaySnappedInfo boolean { t | nil }
```

### **Description**

Shows snapping information, which is the row name and the x and y snap pattern grid names, in the info balloon that is displayed while moving objects in assisted placement mode.

The default is `t`.

### **GUI Equivalent**

None

### **Examples**

```
envGetVal("layoutXL.APAssist" "displaySnappedInfo")
envSetVal("layoutXL.APAssist" "displaySnappedInfo" 'boolean t)
envSetVal("layoutXL.APAssist" "displaySnappedInfo" 'boolean nil)
```

### ***Related Topics***

[Assisted Placement Environment Variables](#)

[Placement Options](#)

## **fixAdjustedForResolveOverlaps**

```
layoutXL.APAssist fixAdjustedForResolveOverlaps boolean { t | nil }
```

### **Description**

Fixes the positions of objects that were adjusted so that these objects are not moved while resolving overlaps. When set to `nil` (default), the adjusted objects can also be moved.

The default is `t`.

### **GUI Equivalent**

Command      *Place – Placement Options*

Field          *Assisted – Resolve Overlaps (Placement Options)*

### **Examples**

```
envGetVal("layoutXL.APAssist" "fixAdjustedForResolveOverlaps")
envSetVal("layoutXL.APAssist" "fixAdjustedForResolveOverlaps" 'boolean t)
envSetVal("layoutXL.APAssist" "fixAdjustedForResolveOverlaps" 'boolean nil)
```

### ***Related Topics***

[Assisted Placement Environment Variables](#)

## **highlightSnappableRowsForInst**

```
layoutXL.APAssist highlightSnappableRowsForInst boolean { t | nil }
```

### **Description**

Highlights snappable rows when an instance is being moved from one row to another.

Default is t.

### **GUI Equivalent**

None

### **Examples**

```
envGetVal("layoutXL.APAssist" "highlightSnappableRowsForInst")
envSetVal("layoutXL.APAssist" "highlightSnappableRowsForInst" 'boolean t)
envSetVal("layoutXL.APAssist" "highlightSnappableRowsForInst" 'boolean nil)
```

### ***Related Topics***

[Assisted Placement Environment Variables](#)

## **highlightSnappableRowsForProxy**

```
layoutXL.APAssist highlightSnappableRowsForProxy boolean { t | nil }
```

### **Description**

Highlights snappable rows when a Modgen, mosaic, or figGroup is being moved from one row to another.

Default is `nil`.

### **GUI Equivalent**

None

### **Examples**

```
envGetVal("layoutXL.APAssist" "highlightSnappableRowsForProxy")  
envSetVal("layoutXL.APAssist" "highlightSnappableRowsForProxy" 'boolean t)  
envSetVal("layoutXL.APAssist" "highlightSnappableRowsForProxy" 'boolean nil)
```

### ***Related Topics***

[Assisted Placement Environment Variables](#)

## infoBalloon

```
layoutXL.APAssist infoBalloon boolean { t | nil }
```

### Description

Displays an info balloon in the top left corner while moving objects in assisted placement mode. The info balloon shows the total wire length, the change to wire length, the row to which the device has snapped, and the snap pattern information. It also indicates whether a figGroup, chain, mosaic, or Modgen is not snapped to a legal set of rows for the entire group.

**Note:** The [displaySnappedInfo](#) environment variable controls display of snapping information in the info balloon.

Default is `t`.

### GUI Equivalent

Command	<i>Place – Placement Options</i>
Field	<i>Assisted – Enable – Show Info Balloon (<a href="#">Placement Options</a>)</i>

### Examples

```
envGetVal("layoutXL.APAssist" "infoBalloon")
envSetVal("layoutXL.APAssist" "infoBalloon" 'boolean t)
envSetVal("layoutXL.APAssist" "infoBalloon" 'boolean nil)
```

### ***Related Topics***

[Assisted Placement Environment Variables](#)

[Placement Options](#)

## **insertColor**

```
layoutXL.APAssist insertColor string "arrow_color"
```

### **Description**

Controls the color of arrows displayed when the *Insert Mode* or the *Insert Mode Single Row* options are turned on.

Default is `white`.

### **GUI Equivalent**

None

### **Examples**

```
envGetVal("layoutXL.APAssist" "insertColor")  
envSetVal("layoutXL.APAssist" "insertColor" 'string "white")
```

### ***Related Topics***

[Assisted Placement Environment Variables](#)

[Placement Options](#)

## insertDefaultSpreadType

```
layoutXL.APAssist insertDefaultSpreadType cyclic { "Any" | "Horizontal" |  
    "Vertical" }
```

### Description

Specifies the direction in which objects should be moved in insert mode. The following arguments signify the move directions:

**Any:** Moves the overlapped device by a minimum distance in any direction.

**Horizontal:** Moves the overlapped device by a minimum distance along the right or left directions only.

**Vertical:** Moves the overlapped device by a minimum distance along the top or bottom directions only.

The default is **Horizontal**.

### GUI Equivalent

Command	<i>Place – Placement Options</i>
Field	<i>Assisted – Assisted Placement – Enable – Resolve Overlaps – Spread Any Direction, Spread X-Direction Only (Placement Options)</i>

### Examples

```
envGetVal("layoutXL.APAssist" "insertDefaultSpreadType")  
envSetVal("layoutXL.APAssist" "insertDefaultSpreadType" 'cyclic "Vertical")
```

### ***Related Topics***

[Assisted Placement Environment Variables](#)

## largeModgenEnvelopeCheck

```
layoutXL.APAssist largeModgenEnvelopeCheck boolean { t | nil }
```

### Description

Specifies that the row legality for large Modgens must be verified by checking only the first and last instance of every row, and not by checking every instance. The intermediate instances are skipped. Consider the following example:

```
XoooooX
XooooooooX
XoooX
```

For this Modgen, only the `x` instances are checked. None of the `o` instances are checked.

The default value is `t`.

### GUI Equivalent

None

### Examples

```
envGetVal("layoutXL.APAssist" "largeModgenEnvelopeCheck")
envSetVal("layoutXL.APAssist" "largeModgenEnvelopeCheck" 'boolean t)
envSetVal("layoutXL.APAssist" "largeModgenEnvelopeCheck" 'boolean nil)
```

### ***Related Topics***

[Assisted Placement Environment Variables](#)



## **largeModgenEnvelopeThreshold**

```
layoutXL.APAssist largeModgenEnvelopeThreshold int integer_number
```

### **Description**

Applies the [largeModgenEnvelopeCheck](#) environment variable only to Modgens with greater than or equal to the specified number of member instances.

The default value is 100.

### **GUI Equivalent**

None

### **Examples**

```
envGetVal("layoutXL.APAssist" "largeModgenEnvelopeThreshold")  
envSetVal("layoutXL.APAssist" "largeModgenEnvelopeThreshold" 'int 120)
```

### ***Related Topics***

[Assisted Placement Environment Variables](#)

## **maximizePinAccess**

```
layoutXL.placement maximizePinAccess boolean { t | nil }
```

### **Description**

Increases the number of WSP tracks that overlap the pins on standard cells.

The default is `nil`.

### **GUI Equivalent**

Command      *Place – Placement Options*

Field          *Common – Maximize Pin Access (Placement Options)*

### **Examples**

```
envGetVal("layoutXL.placement" "maximizePinAccess")  
envSetVal("layoutXL.placement" "maximizePinAccess" 'boolean t)  
envSetVal("layoutXL.placement" "maximizePinAccess" 'boolean nil)
```

### ***Related Topics***

[Assisted Placement Environment Variables](#)

## **minNumPinAccessPts**

`layoutXL.placement minNumPinAccessPts int integer_number`

### **Description**

Specifies the minimum number of pin access points that must be available to the router when maximizePinAccess is set to t.

The default is 2.

### **GUI Equivalent**

Command	<i>Place – Placement Options</i>
Field	<i>Common – Minimum Pin Access (<u>Placement Options</u>)</i>

### **Examples**

```
envGetVal("layoutXL.placement" "minNumPinAccessPts")  
envSetVal("layoutXL.placement" "minNumPinAccessPts" 'int 0)
```

### ***Related Topics***

Assisted Placement Environment Variables

## numSnappableRows

`layoutXL.APAssist numSnappableRows int integer_number`

### Description

Specifies the maximum number of snappable rows above and below the current row, which are highlighted when an instance is being moved.

The default is 3.

### GUI Equivalent

None

### Examples

```
envGetVal("layoutXL.APAssist" "numSnappableRows")
envSetVal("layoutXL.APAssist" "numSnappableRows" 'int 5)
```

### ***Related Topics***

[Assisted Placement Environment Variables](#)

## paddingForCells

```
layoutXL.placement paddingForCells string "padding"
```

### Description

Specifies the space to be left on either sides of the master. Use the following syntax:

```
(cellName right_spacing left_spacing)
```

Default is blank.

### GUI Equivalent

Command	<i>Place – Placement Options</i>
Field	<i>Common – Cell Padding Values</i> ( <u>Placement Options</u> )

### Examples

```
envGetVal("layoutXL.placement" "paddingForCells")  
envSetVal("layoutXL.placement" "paddingForCells" 'string "nand4 2.5 2.5")
```

### ***Related Topics***

Assisted Placement Environment Variables

## pinDensityAwarePlacement

```
layoutXL.placement pinDensityAwarePlacement boolean { t | nil }
```

### Description

Considers the pin densities of standard cells while optimizing placement and improving routability. The assisted placement options are also pin density-aware. The pin density consideration can result in larger spacing between the placed standard cells.

**Note:** Pin density awareness is a heuristic to improve routability, and therefore is not checked during placement verification.

The default is `nil`.

### GUI Equivalent

Command	<i>Place – Placement Options</i>
Field	<i>Common – Pin Density Aware Placement (<a href="#">Placement Options</a>)</i>

### Examples

```
envGetVal("layoutXL.placement" "pinDensityAwarePlacement")
envSetVal("layoutXL.placement" "pinDensityAwarePlacement" 'boolean t)
envSetVal("layoutXL.placement" "pinDensityAwarePlacement" 'boolean nil)
```

### Related Topics

[Assisted Placement Environment Variables](#)

## regenModgenPostProcess

```
layoutXL.AP regenModgenPostProcess boolean { t | nil }
```

### Description

Regenerates Modgens after they are moved to a row region. The orientations of the Modgen members are updated to match the orientations of the rows to which they are moved.

The default is `t`.

### GUI Equivalent

Command	<i>Place – Placement Options</i>
Field	<i>Common – Update MODGENS to Rows Post-Placement (<u>Placement Options</u>)</i>

### Examples

```
envGetVal ("layoutXL.placement" "regenModgenPostProcess")
envSetVal ("layoutXL.placement" "regenModgenPostProcess" 'boolean t)
envSetVal ("layoutXL.placement" "regenModgenPostProcess" 'boolean nil)
```

### ***Related Topics***

Assisted Placement Environment Variables

## **snapColor**

```
layoutXL.APAssist snapColor string "highlight_color"
```

### **Description**

Specifies the color of highlights to be displayed when Modgens, mosaics, or figGroups are moved. These highlights indicate the rows to which the Modgens can be snapped.

Default is `yellow`.

### **GUI Equivalent**

None

### **Examples**

```
envGetVal("layoutXL.APAssist" "snapColor")  
envSetVal("layoutXL.APAssist" "snapColor" 'string "yellow")
```

### ***Related Topics***

[Assisted Placement Environment Variables](#)

[Placement Options](#)



## **snapFigGroupRowCorrect**

```
layoutXL.APAssist snapFigGroupRowCorrect boolean { t | nil }
```

### **Description**

Specifies whether figGroups, Modgen, mosaics, and chains must snap to rows (when set to `t`), or can snap by reference when their member instances do not align exactly to the rows (when set to `nil`).

Default is `nil`.

### **GUI Equivalent**

None

### **Examples**

```
envGetVal("layoutXL.APAssist" "snapFigGroupRowCorrect")
envSetVal("layoutXL.APAssist" "snapFigGroupRowCorrect" 'boolean t)
envSetVal("layoutXL.APAssist" "snapFigGroupRowCorrect" 'boolean nil)
```

### ***Related Topics***

[Assisted Placement Environment Variables](#)

## spreadEnabled

```
layoutXL.APAssist spreadEnabled boolean { t | nil }
```

### Description

Specifies whether devices can be moved to resolve overlaps.

Default is `nil`.

### GUI Equivalent

Command      *Place – Placement Options*

Field          *Assisted – Resolve Overlaps (Placement Options)*

### Examples

```
envGetVal("layoutXL.APAssist" "spreadEnabled")
envSetVal("layoutXL.APAssist" "spreadEnabled" 'boolean t)
envSetVal("layoutXL.APAssist" "spreadEnabled" 'boolean nil)
```

### ***Related Topics***

[Assisted Placement Environment Variables](#)

## spreadType

```
layoutXL.APAssist spreadType cyclic { "All" | "LeftOrRight" | "Right" |  
    "RightOnSingleLine" | "InsertSingleColumn" }
```

### Description

Specifies the direction in which devices can be moved to resolve overlaps. Valid values are:

- **All:** Devices can be moved in any direction (**GUI Equivalent:** Spread Any Direction).
- **LeftOrRight:** Devices can be moved to the left or right (**GUI Equivalent:** Spread X-Direction Only).
- **Right:** Devices can be moved only to the right (**GUI Equivalent:** Insert Mode).
- **RightOnSingleLine:** Devices can be moved only to the right in the same row (**GUI Equivalent:** Insert Mode Single Row).
- **InsertSingleColumn:** Devices are stacked vertically before the interactive move (**GUI Equivalent:** Insert Mode Single Column).

Default is All.

### GUI Equivalent

Command      *Place – Placement Options*

Field          *Assisted – Resolve Overlaps (Placement Options)*

### Examples

```
envGetVal("layoutXL.APAssist" "spreadType")  
envSetVal("layoutXL.APAssist" "spreadType" 'cyclic "LeftOrRight")  
envSetVal("layoutXL.APAssist" "spreadType" 'cyclic "Right")  
envSetVal("layoutXL.APAssist" "spreadType" 'cyclic "InsertSingleColumn")
```

### Related Topics

Assisted Placement Environment Variables

## swapRowMode

```
layoutXL.APAssist swapRowMode boolean { t | nil }
```

### Description

Controls the swap mode. When set to `t`, all objects in the selected rows are swapped. When set to `nil`, the positions of the selected objects are swapped.

Default is `nil`.

### GUI Equivalent

Command	<i>Place – Swap Instances, Swap Rows</i>
Field	None

### Examples

```
envGetVal("layoutXL.APAssist" "swapRowMode")
envSetVal("layoutXL.APAssist" "swapRowMode" 'boolean t)
envSetVal("layoutXL.APAssist" "swapRowMode" 'boolean nil)
```

### ***Related Topics***

[Assisted Placement Environment Variables](#)

## useDeviceOrder

```
layoutXL.APAssist useDeviceOrder boolean { t | nil }
```

### Description

Abuts instances in the same order in which they are placed in each row.

Default is `t`, where instances maintain their current order when abutted.

### GUI Equivalent

Command      *Place – Placement Options*

Field          *Assisted – Abut Rows Use Device Order ([Placement Options](#))*

### Examples

```
envGetVal("layoutXL.APAssist" "useDeviceOrder")  
envSetVal("layoutXL.APAssist" "useDeviceOrder" 'boolean t)  
envSetVal("layoutXL.APAssist" "useDeviceOrder" 'boolean nil)
```

### ***Related Topics***

[Assisted Placement Environment Variables](#)

## **useLayoutAsSeed**

```
layoutXL.APAssist useLayoutAsSeed boolean { t | nil }
```

### **Description**

Runs the placer from its current layout. The placement results depend on the starting layout. Consider the same layout generated using two different methods. The placement results for these layouts might differ.

However, placement results will remain the same for a specific starting layout. For example, the placement result will remain the same for a layout that is always generated using Generating All Components from Source.

The default is `t`.

### **GUI Equivalent**

None

### **Examples**

```
envGetVal("layoutXL.APAssist" "useLayoutAsSeed")
envSetVal("layoutXL.APAssist" "useLayoutAsSeed" 'boolean t)
envSetVal("layoutXL.APAssist" "useLayoutAsSeed" 'boolean nil)
```

### ***Related Topics***

[Assisted Placement Environment Variables](#)

## **useLEReference**

```
layoutXL.APAssist useLEReference boolean { t | nil }
```

### **Description**

Specifies that the closest object must be considered as the reference for snapping. You can choose the left-most lowest instance as viewed when the group is oriented at R0.

The default is `t`.

### **GUI Equivalent**

None

### **Examples**

```
envGetVal("layoutXL.APAssist" "useLEReference")  
envSetVal("layoutXL.APAssist" "useLEReference" 'boolean t)  
envSetVal("layoutXL.APAssist" "useLEReference" 'boolean nil)
```

### ***Related Topics***

[Assisted Placement Environment Variables](#)

## WSPAware

```
layoutXL.APAssist WSPAware boolean { t | nil }
```

### Description

Enables snapping of devices to WSP tracks.

The default is `t`.

### GUI Equivalent

Command	<i>Place – Placement Options</i>
Field	<i>Common – Width Spacing Pattern (WSP) Snapping (<a href="#">Placement Options</a>)</i>

### Examples

```
envGetVal("layoutXL.APAssist" "WSPAware")  
envSetVal("layoutXL.APAssist" "WSPAware" 'boolean t)  
envSetVal("layoutXL.APAssist" "WSPAware" 'boolean nil)
```

### ***Related Topics***

[Assisted Placement Environment Variables](#)



## Dummy Fill Insertion Environment Variables

### lobFillAsMosaic

```
layoutXL lobFillAsMosaic boolean { t | nil }
```

#### Description

Specifies whether mosaic arrays must be generated to fill the adjacent space when the active device does not have a fingerCount type parameter.

The default is `nil`.

#### GUI Equivalent

None

#### Examples

```
envGetVal("layoutXL" "lobFillAsMosaic")
envSetVal("layoutXL" "lobFillAsMosaic" 'boolean t)
envSetVal("layoutXL" "lobFillAsMosaic" 'boolean nil)
```

#### ***Related Topics***

[Dummy Fill Insertion Environment Variables](#)

## **lobDevFillPhysOnly**

```
layoutXL lobDevFillPhysOnly boolean { t | nil }
```

### **Description**

Specifies whether the fill cells must be rendered as physical-only.

The default is `t`.

### **GUI Equivalent**

None

### **Examples**

```
envGetVal("layoutXL" "lobDevFillPhysOnly")
envSetVal("layoutXL" "lobDevFillPhysOnly" 'boolean t)
envSetVal("layoutXL" "lobDevFillPhysOnly" 'boolean nil)
```

### ***Related Topics***

[Dummy Fill Insertion Environment Variables](#)

## Placement Checker Environment Variables

These `layoutXL.APverify` environment variables are used to set options for the placement checks in the Batch Checker GUI.

### constraints

```
layoutXL.APVerify constraints 'boolean { t | nil }
```

### Description

(Advanced Node Layout EXL Only) If set, checks all elements in the design for conformance with the Alignment, Orientation, Symmetry, and Distance constraints.

Default is `t`.

### GUI Equivalent

Command:     *Verify – Design*

Field:        *Placement – Check for – Constraints*

### Examples

```
envGetVal("layoutXL.APVerify" "constraints")
envSetVal("layoutXL.APVerify" "constraints" 'boolean t)
envSetVal("layoutXL.APVerify" "constraints" 'boolean nil)
```

### ***Related Topics***

[Batch Checker \(Placement Tab\)](#)

[Verify Placement](#)

## rowOrGrid

```
layoutXL.APVerify rowOrGrid 'boolean { t | nil }
```

### Description

(Advanced Node Layout EXL Only) Checks whether instances are snapped to the correct rows or grids.

Default is t.

### GUI Equivalent

Command:     *Verify – Design*

Field:         *Placement – Check for – Row/Grid*

### Examples

```
envGetVal("layoutXL.APVerify" "rowOrGrid")  
envSetVal("layoutXL.APVerify" "rowOrGrid" 'boolean t)  
envSetVal("layoutXL.APVerify" "rowOrGrid" 'boolean nil)
```

### *Related Topics*

[Batch Checker \(Placement Tab\)](#)

[Verify Placement](#)

## spacing

```
layoutXL.APVerify spacing 'boolean { t | nil }
```

### Description

(Advanced Node Layout EXL Only) Checks instances for conformance with applicable spacing rules.

Default is `nil`.

### GUI Equivalent

Command:     *Verify – Design*

Field:        *Placement – Check for – Spacing*

### Examples

```
envGetVal("layoutXL.APVerify" "spacing")
envSetVal("layoutXL.APVerify" "spacing" 'boolean t)
envSetVal("layoutXL.APVerify" "spacing" 'boolean nil)
```

### ***Related Topics***

[Batch Checker \(Placement Tab\)](#)

[Verify Placement](#)

## tapCell

```
layoutXL.APVerify tapCell 'boolean { t | nil }
```

### Description

(Advanced Node Layout EXL Only) Checks whether tap cells are inserted correctly, as per their specifications on the *Automatic* tab of the Placement Options form.

Default is `nil`.

### GUI Equivalent

None

### Examples

```
envGetVal("layoutXL.APVerify" "tapCell")  
envSetVal("layoutXL.APVerify" "tapCell" t)  
envSetVal("layoutXL.APVerify" "tapCell" nil)
```

### ***Related Topics***

Batch Checker (Placement Tab)

Verify Placement

## padding

```
layoutXL.APVerify padding 'boolean { t | nil }
```

### Description

(Advanced Node Layout EXL Only) Checks whether the spacing on either side of the master are in conformance with the *Cell Padding* value specified on the *Common* tab of the Placement Options form.

Default is `nil`.

### GUI Equivalent

None

### Examples

```
envGetVal("layoutXL.APVerify" "padding")
envSetVal("layoutXL.APVerify" "padding" 'tapCell t)
envSetVal("layoutXL.APVerify" "padding" 'tapCell nil)
```

### ***Related Topics***

Batch Checker (Placement Tab)

Verify Placement

## minPinAccess

```
layoutXL.APVerify minPinAccess 'boolean { t | nil }
```

### Description

(Advanced Node Layout EXL Only) Checks WSP tracks for conformance with the *Minimum Pin Access Count* specified on the *Common* tab of the Placement Options form.

Default is `nil`.

### GUI Equivalent

Command:     *Verify – Design*

Field:         *Placement – Check for – Minimum Pin Access*

### Examples

```
envGetVal("layoutXL.APVerify" "minPinAccess")
envSetVal("layoutXL.APVerify" "minPinAccess" 'boolean t)
envSetVal("layoutXL.APVerify" "minPinAccess" 'boolean nil)
```

### Related Topics

Batch Checker (Placement Tab)

Verify Placement



## colorAware

```
layoutXL.APVerify colorAware 'boolean { t | nil }
```

### Description

(Advanced Node Layout EXL Only) Checks shapes for conformance with color locks assigned to them, if any.

Default is `nil`.

### GUI Equivalent

Command:     *Verify – Design*

Field:         *Placement – Check for – Color Aware*

### Examples

```
envGetVal("layoutXL.APVerify" "colorAware")  
envSetVal("layoutXL.APVerify" "colorAware" 'boolean t)  
envSetVal("layoutXL.APVerify" "colorAware" 'boolean nil)
```

### *Related Topics*

[Batch Checker \(Placement Tab\)](#)

[Verify Placement](#)

## reportViolations

```
layoutXL.APVerify reportViolations 'boolean { t | nil }
```

### Description

(Advanced Node Layout EXL Only) Displays detailed error and warning messages in the CIW reporting all violations that were detected while running batch checker.

Default is `nil`.

### GUI Equivalent

Command:     *Verify – Design*

Field:        *Placement – Reports – Report Violations*

### Examples

```
envGetVal("layoutXL.APVerify" "reportViolations")  
envSetVal("layoutXL.APVerify" "reportViolations" 'boolean t)  
envSetVal("layoutXL.APVerify" "reportViolations" 'boolean nil)
```

### ***Related Topics***

[Batch Checker \(Placement Tab\)](#)

[Verify Placement](#)

## **createMarkers**

```
layoutXL.APVerify createMarkers 'boolean { t | nil }
```

### **Description**

(Advanced Node Layout EXL Only) Displays markers in the design to indicate the locations where violations are reported.

Default is `nil`.

### **GUI Equivalent**

Command:     *Verify – Design*

Field:        *Placement – Reports – Create Markers*

### **Examples**

```
envGetVal("layoutXL.APVerify" "createMarkers")  
envSetVal("layoutXL.APVerify" "createMarkers" 'boolean t)  
envSetVal("layoutXL.APVerify" "createMarkers" 'boolean nil)
```

### ***Related Topics***

[Batch Checker \(Placement Tab\)](#)

[Verify Placement](#)

## **selOnly**

```
layoutXL.APVerify selOnly 'boolean { t | nil }
```

### **Description**

(Advanced Node Layout EXL Only) Runs batch checker on the selected instances only.

Default is `nil`.

### **GUI Equivalent**

Command:     *Verify – Design*

Field:         *Placement – Scope – Selected Only*

### **Examples**

```
envGetVal("layoutXL.APVerify" "selOnly")
envSetVal("layoutXL.APVerify" "selOnly" 'boolean t)
envSetVal("layoutXL.APVerify" "selOnly" 'boolean nil)
```

### ***Related Topics***

[Batch Checker \(Placement Tab\)](#)