cadence®

Virtuoso Placer User Guide

Product Version IC23.1
March 2023

Printed in the United States of America.

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

Contents

1

Introduction to Virtuoso Placer

Virtuoso® 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® 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.

The key benefits of Virtuoso Placer are:

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

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.

Related Topics

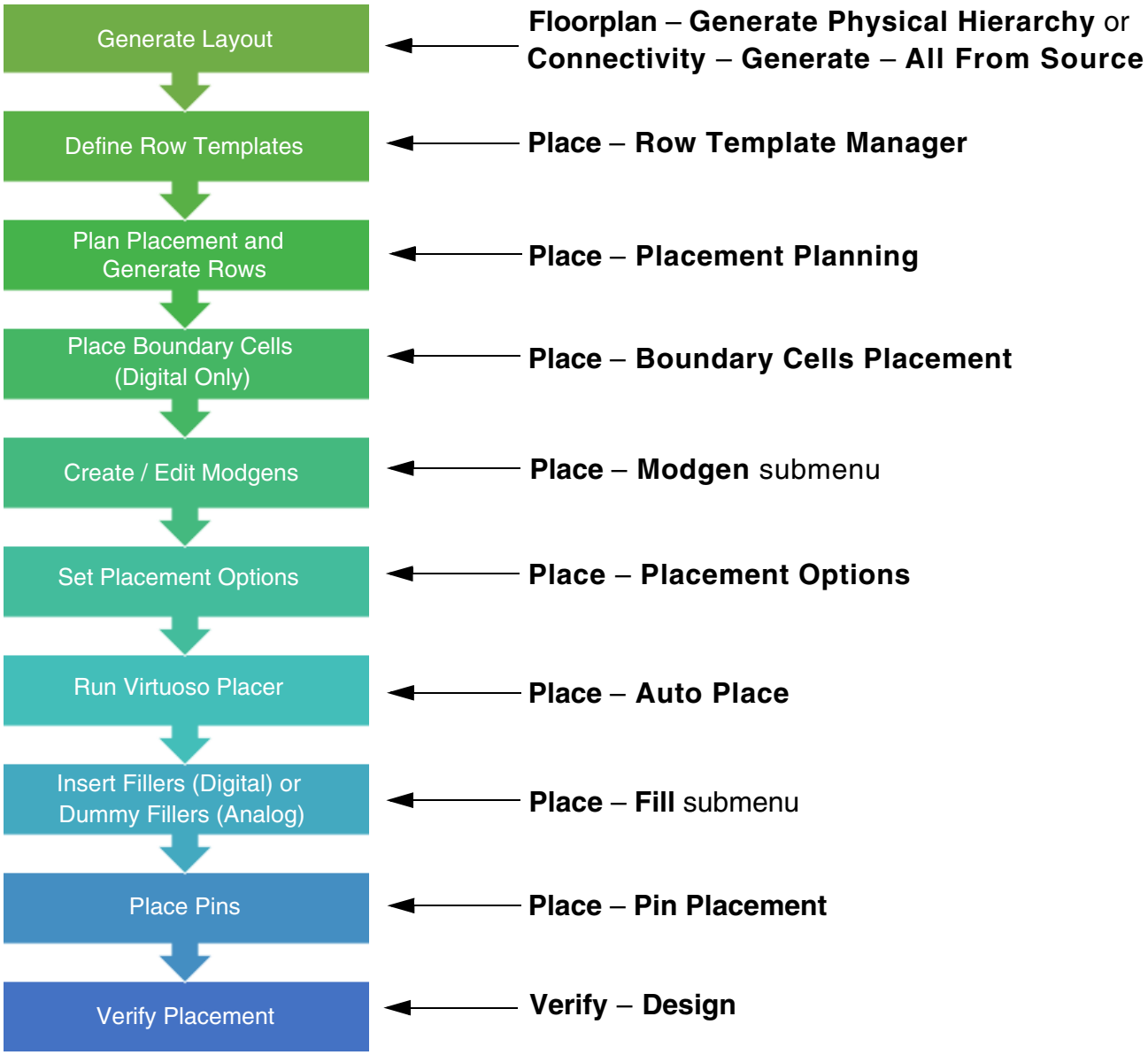
Virtuoso Placer Flow

Virtuoso Placer Menu

Generating a Layout

Virtuoso Placer Flow

Virtuoso Placer is a unified, powerful placement solution that 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. The following figure depicts the Virtuoso Placer flow and maps each step to the corresponding Virtuoso feature.



Related Topics

Introduction to Virtuoso Placer

Virtuoso Placer Menu

Virtuoso Placer Menu

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*

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

Icon	Command Name	Description
Commands to Define and Update Pin Placement		
	Pin Placement	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.
	Update Placement Status	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.
Commands to Plan the Placement		
	*Row Template Manager*	Lets you define row templates, which can later be used to generate rows in the layout canvas.
	*Placement Planning*	Lets you plan the placement region, generate rows in which the components are placed, and select additional substrates to be inserted during placement.
	*Boundary Cells Placement*	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.
Commands to Configure Placement Options		
	*Placement Options*	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.
	Swap Tap Cells	Detects tap cells with maximum spacing violations and replaces them with tap cells that have valid maximum spacing values.

Command to Run the Placer		
	*Auto Place*	Lets you place devices, standard cells, or a mix of both in rows defined in the layout canvas.
Commands to Refine the Placement Interactively		
	*Resolve Overlaps*	Resolves device overlaps in rows.
	*Abut Instances*	Abuts selected or all instances in the design.
	*UnAbut Instances*	Unabuts selected or all instances in the design.
	*Swap Instances*	Swaps the positions of selected devices.
	*Swap Rows*	Swaps the positions of the selected rows.
	*Snap to Grids/ Rows*	Snaps the selected devices to appropriate grids or rows.
	*Adjust PR Boundary*	Adjusts the boundary by deleting any extra rows in which devices are not placed.
Commands to Add Modgens and Fill		
	Modgen	Opens a submenu that provides commands to create and edit Modgen constraints.
	Filler	Opens a submenu that provides options to: ■ Insert filler cells  - For digital designs ■ Insert dummy fill  - For analog designs

Related Topics

Pin Planning

Placement Options Form

Swapping Tap Cells

Row Generation

2

Introduction to Row Infrastructure

Advanced node processes support multiple grid types. As a result, advanced node designs easily adapt to row-based placement methodology. 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.

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.

Related Topics

Row Attributes

Row Templates

Row Generation

Row Attributes

The following figure depicts a row and its attributes:

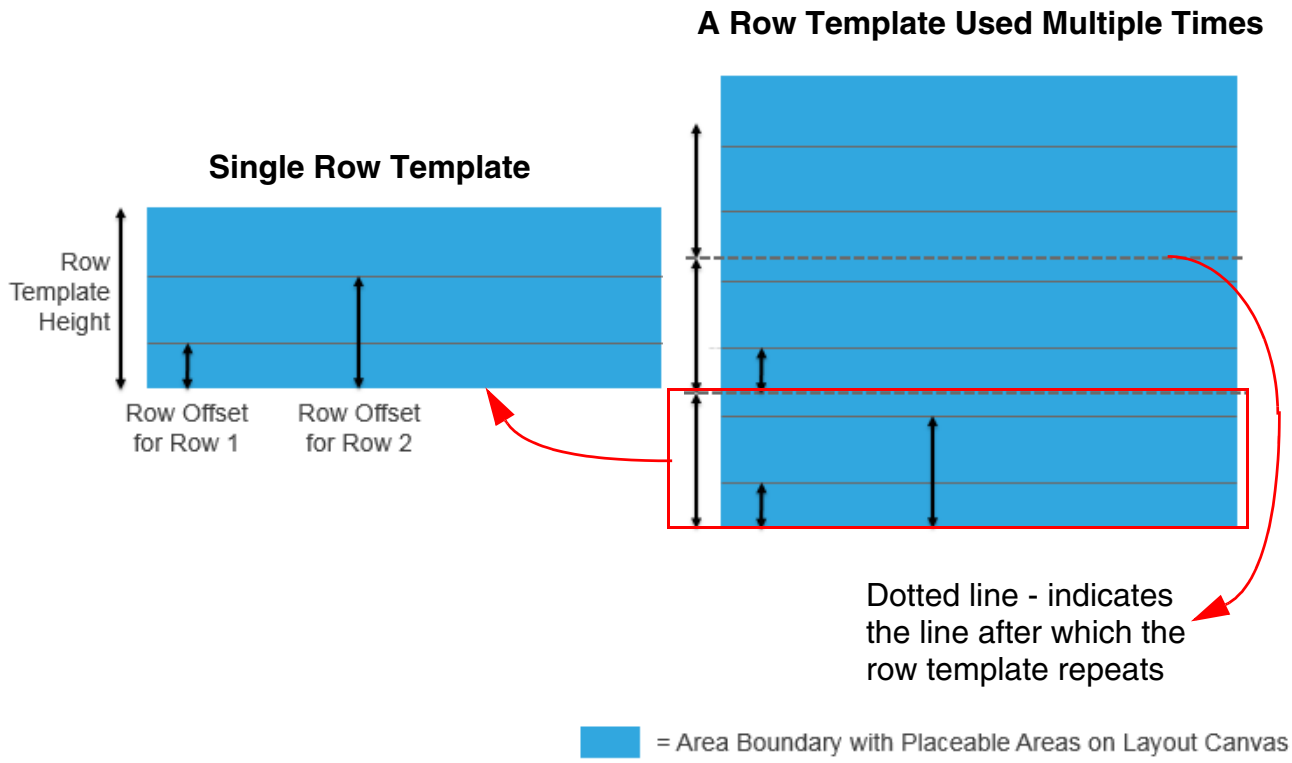
Term	Description
Row Edge	Row edge is defined by the row offset and row height values. The reference edge is specified by the row offset value. The other edge is at a distance equal to the row height away from the reference edge.
Row Orientation	Is the orientation of the row with respect to the x and 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 figure.
Row Height	Is the distance between the top and bottom edges of the row.

Related Topics

Introduction to Row Infrastructure

Row Templates

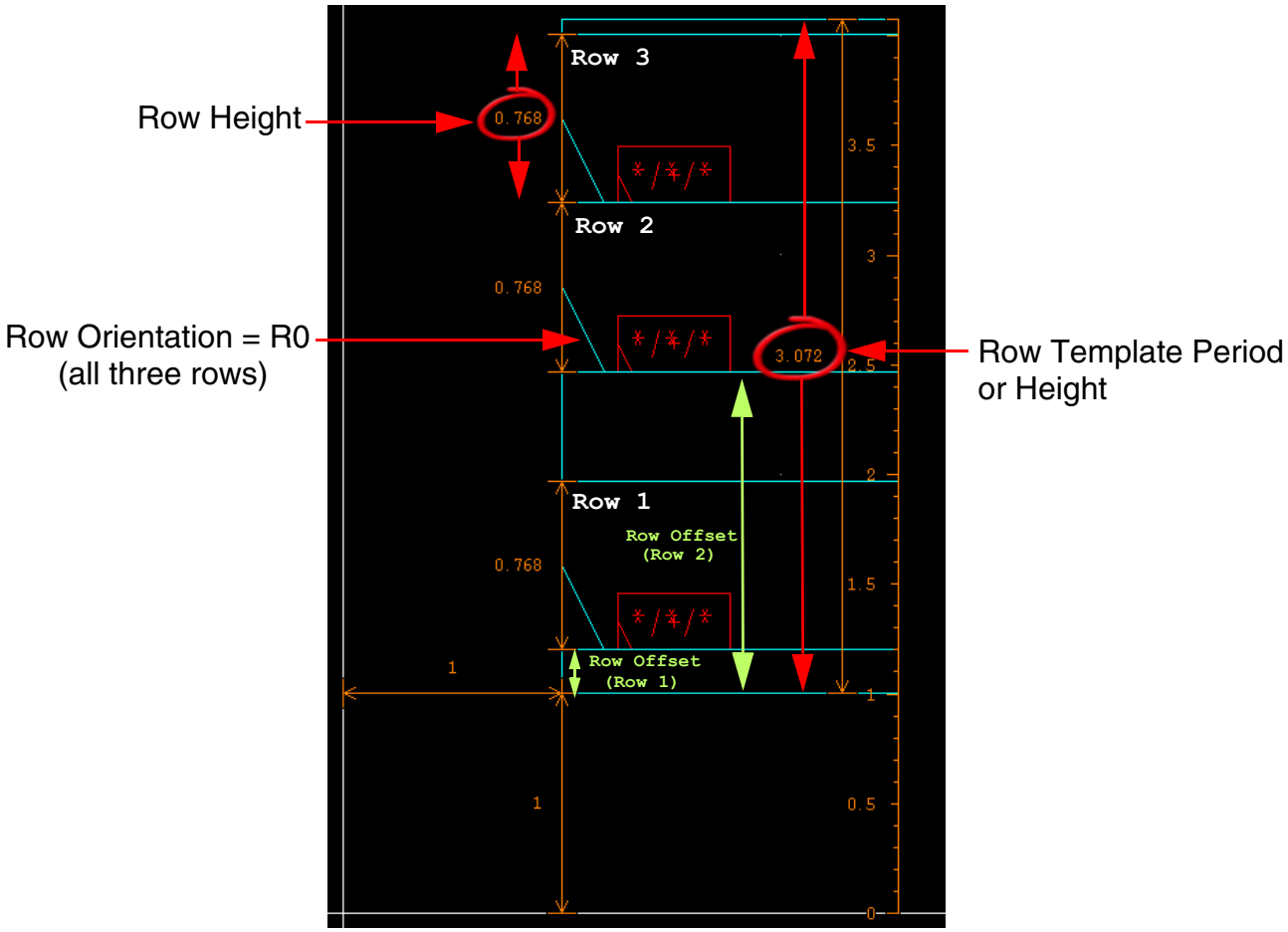
A row template is a container that defines one or more rows. It comprises specifications for a set of rows to be generated in the layout canvas. A row template can be repeated multiple times, one on top of the other, based on the required height. The following figure depicts how a row template can be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



In the above figure, 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.

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 rows 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.

Related Topics

Introduction to Row Infrastructure

Row Generation

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 are typically set up centrally for use with a particular project or process node and it is unlikely that the layout designer will have to edit them.

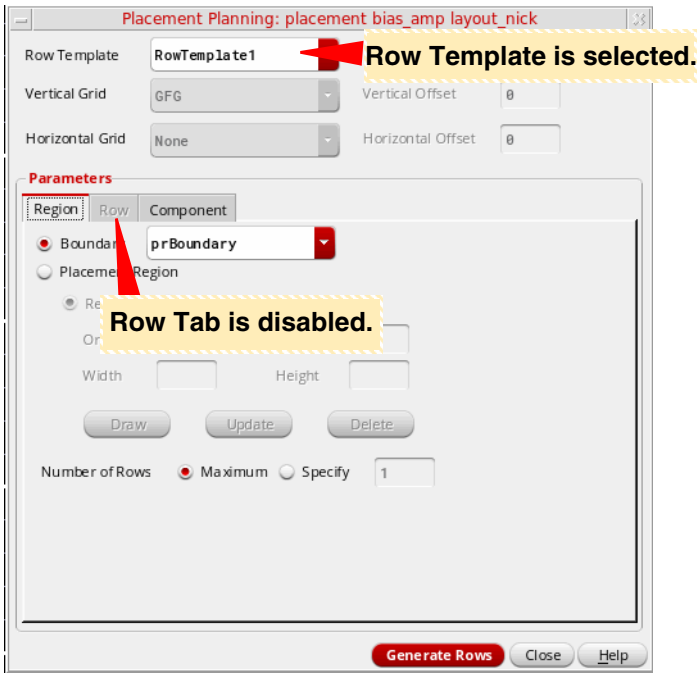
   The Placement Planning form includes the *Row Template* field. This section covers the process of defining and generating rows using the Row Template Manager.

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.

   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 figure:



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

Related Topics

Placement Planning

Row Attributes

Creating, Editing, and Deleting a Row Template

Importing a Row Template from Another Cellview

Creating, Editing, and Deleting a Row Template

Use the Row Template Manager to create, edit, and delete row templates. The tasks in creating a new row template are:

- Defining Row Template Name and Placement Grid
- Defining Row Parameters
- Defining Component Types and Rails

Important

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

Defining Row Template Name and Placement Grid

To define a row template and placement grid, with the Row Template Manager open:

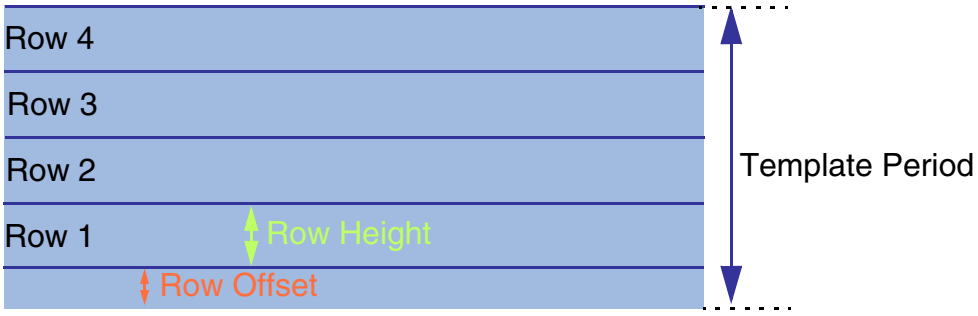
1. Click the *Edit* tab.
2. Click *New* to generate a new unique row template name, which, by default, is *RowTemplateN*, where *N* = 1, 2, 3, … You can modify this default value to specify a more descriptive name for your row template.

   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. 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.

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 RRS 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.

Defining Row Parameters

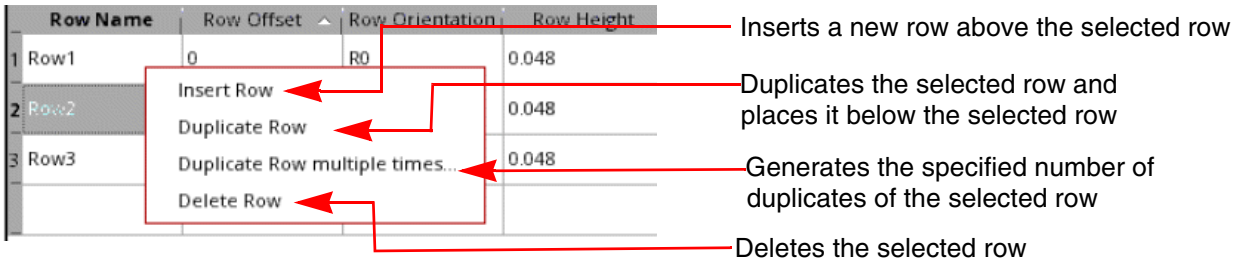
Use the *Rows* section to specify row parameters.

To define a row:

1. Add a row by editing the default placeholder for the first row in the *Row Information* table.

   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.



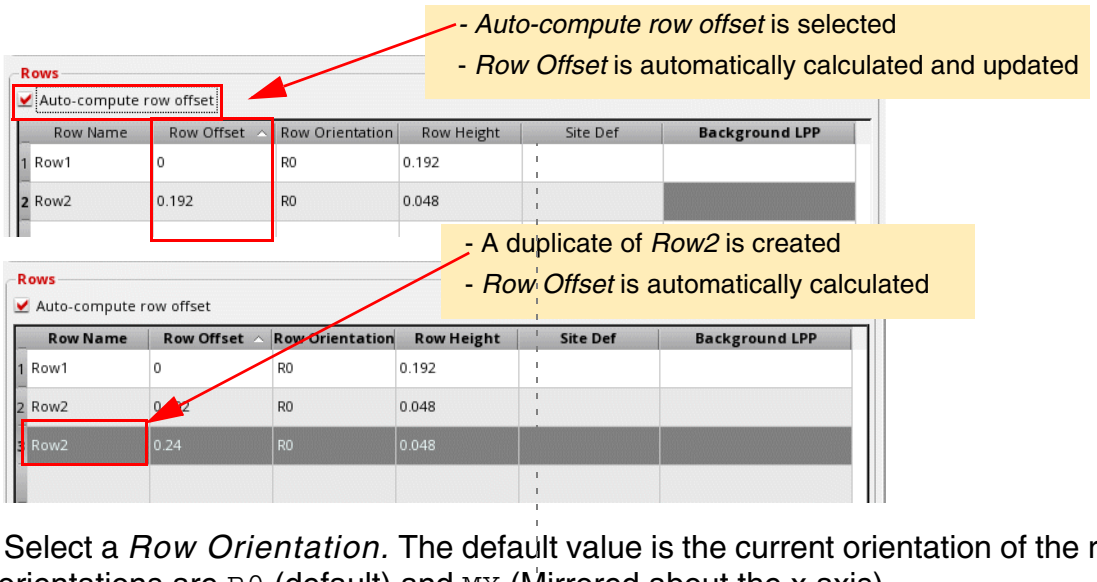
2. Edit the *Row Name*, if required.

3. Specify the row offset from the base of the template. For the first row, the default value is `0`, which you can edit. Row offsets must be greater than or equal to zero and less than or equal to the template period.

   Instead of specifying absolute values, 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.

   You must ensure that the *Row Offset* values adhere to the *Template Period* and *Row Height* values. An example of an error condition is given below.

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.



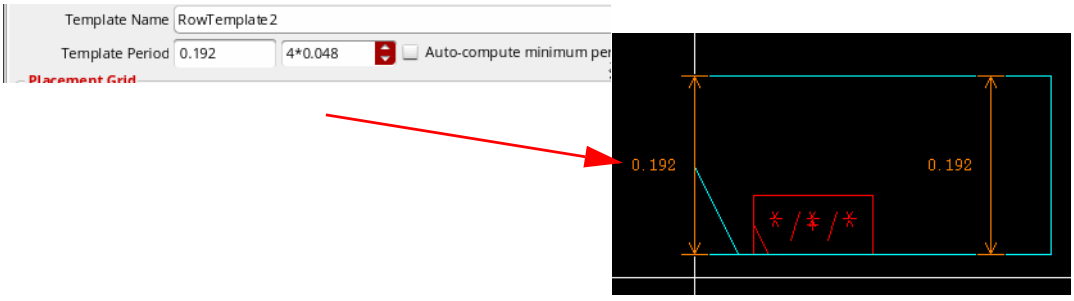
4. Select a *Row Orientation.* The default value is the current orientation of the row. Valid orientations are `R0` (default) and `MX` (Mirrored about the x axis).

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.

5. Specify the row height using one of the following methods:

   a. Specify an absolute *Row Height* value. The default value is the period of the vertical grid or `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`.

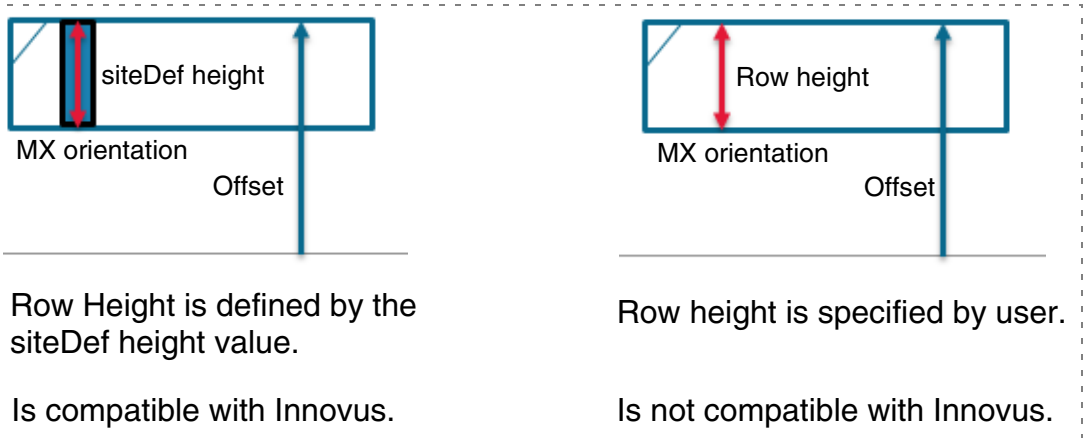


On adding a new row to the above definition, the *Template Period* value is highlighted, indicating that the end of the row exceeds the Template Period value. Therefore, the row is partially or fully outside of the template. In such situations,

select the *Auto compute minimum period* to automatically calculate a minimum template period value.

b. Select a *Site Def*. *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.

The following figure depicts the differences between the siteDef height and row height values:



Row Height is defined by the siteDef height value.

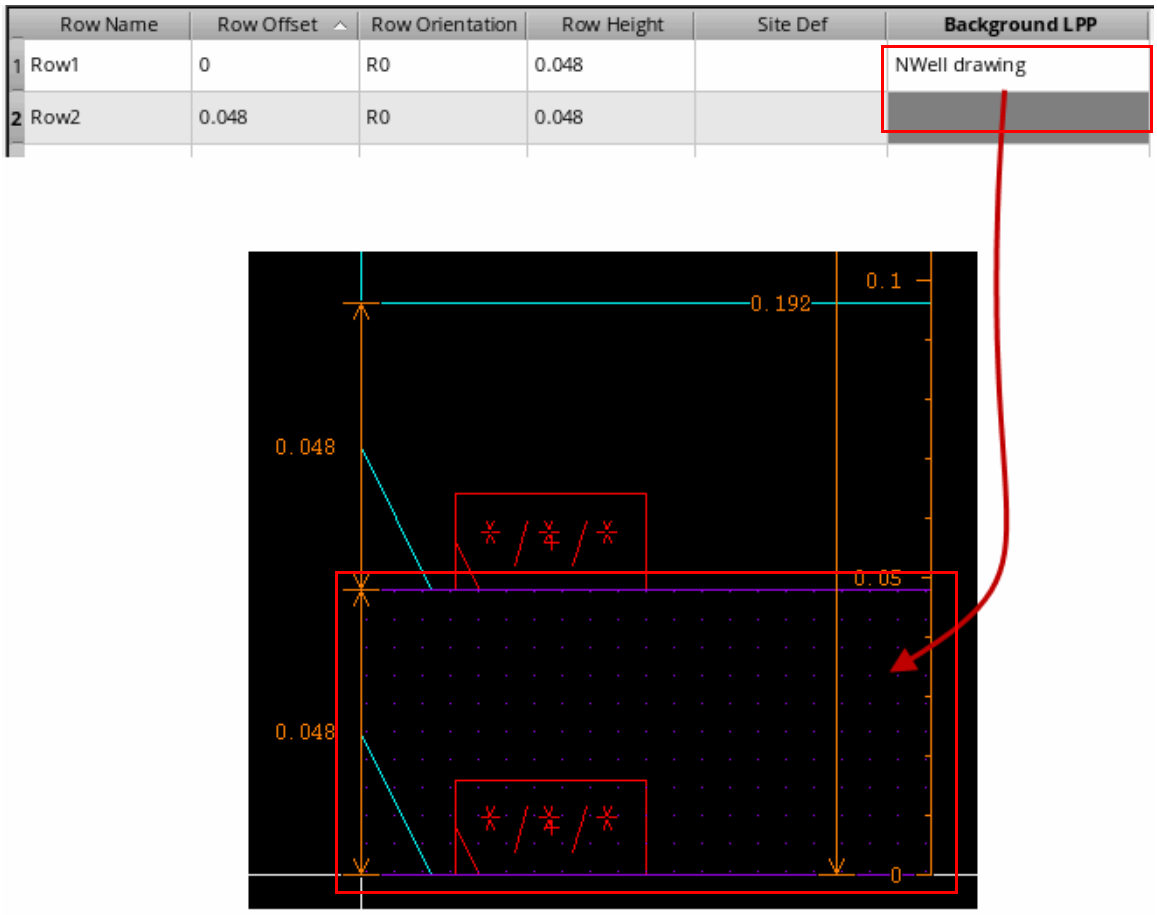
Is compatible with Innovus.

Row height is specified by user.

Is not compatible with Innovus.

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

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*:

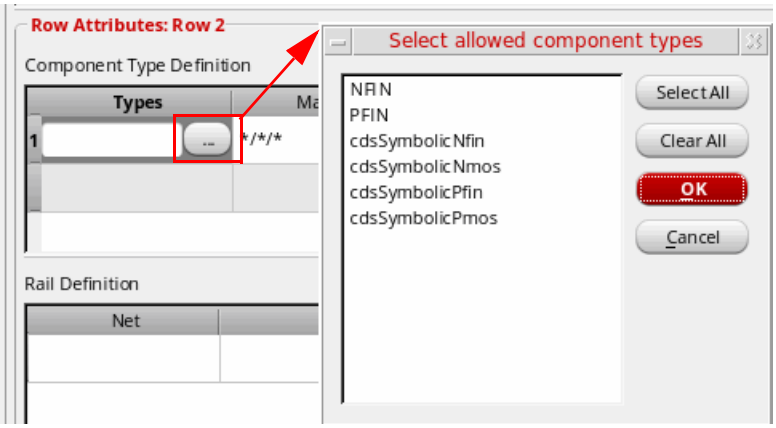


A rectangle shape is created in the row, which can be used for placing wells.

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.



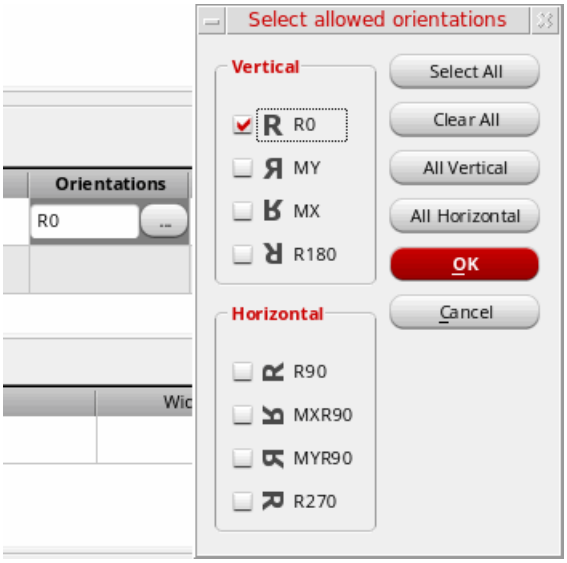
2. If you have not specified component types, specify the device *Master*. Only the devices belonging to the specified master can be placed in a row. You can specify either a single master cellview or only the `lib`, `cell`, or `view` value. For example, *Master* can be set to `*/pfinFet/*` for a p-type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

Important

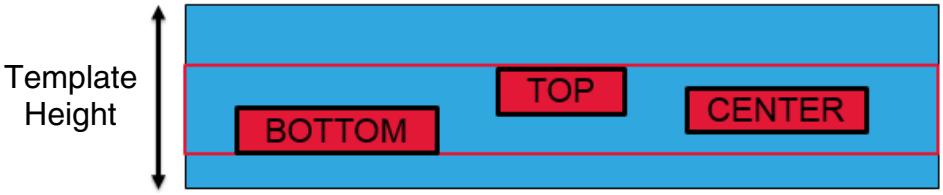
A row can use either a component type or a component master, but not both. Adding a master will delete the component type information, and vice versa.

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

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



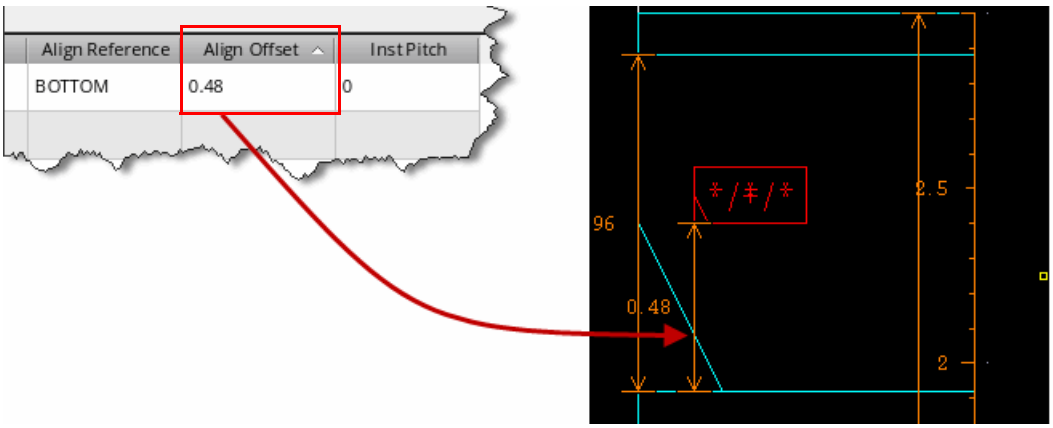
4. Choose an *Align Reference*, which specifies the row edge to which components need to be aligned. Valid values are `BOTTOM`, `TOP`, and `CENTER`.



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

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



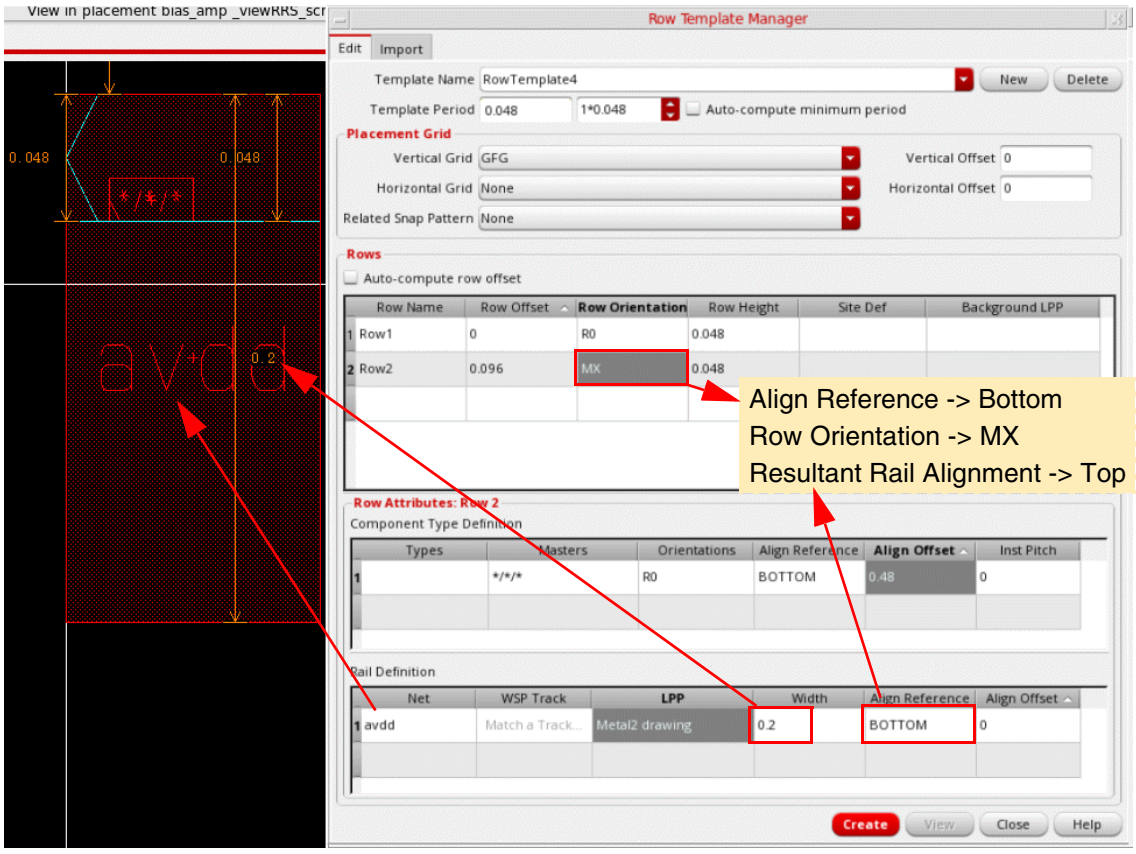
Tip

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

6. Specify an *InstPitch* value, which is the minimum distance between instances of the same component in a row.
7. 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*.
8. Select a WSP track based on which the rail layer, width, and offset values must be set. *WSP Track* lists all WSP tracks available within the template period range.
9. 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`.
10. 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.

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:



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 node designs.

Deleting a Row Template

To delete a row template, select the required row template in the *Template Name* list and click *Delete*.

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

Related Topics

Importing a Row Template from Another Cellview

Row Attributes

Placement Planning

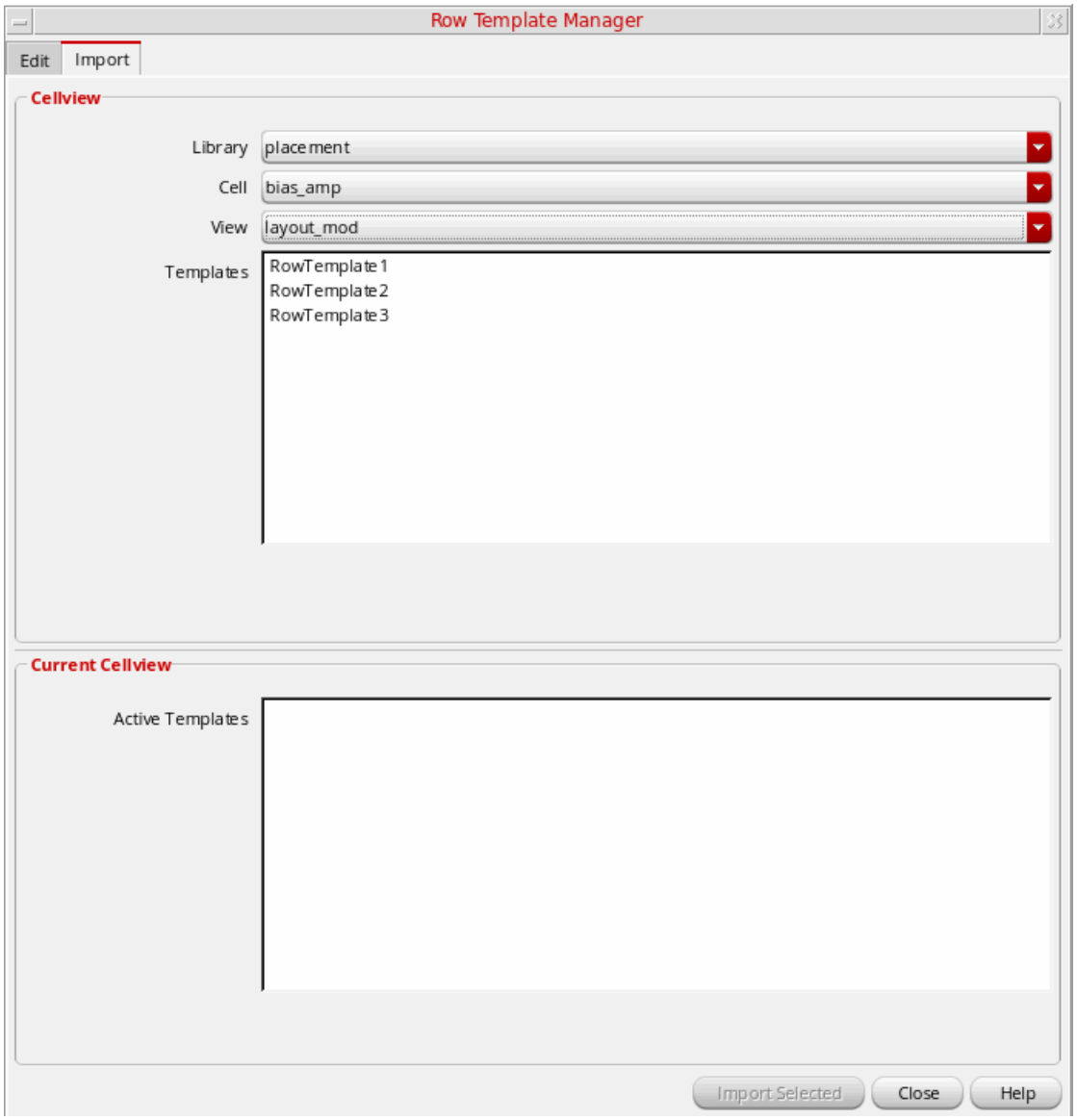
Row Generation

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.

To import existing row templates from a different cellview in your design library:

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



2. Select the *Import* tab.

3. Choose the *Library*, *Cell*, and *View* which contains the row template you want to import.
4. Select the row templates you want to import. *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.

Related Topics

Row Template Manager Form

Placement Planning

Row Attributes

Row Generation

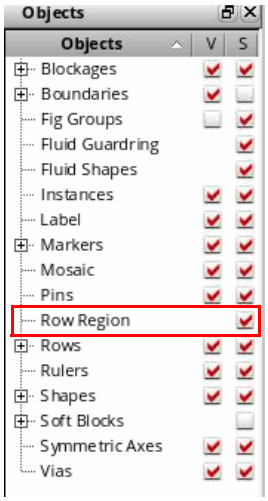
Creating, Editing, and Deleting a Row Template

Layout Support for Row Regions

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

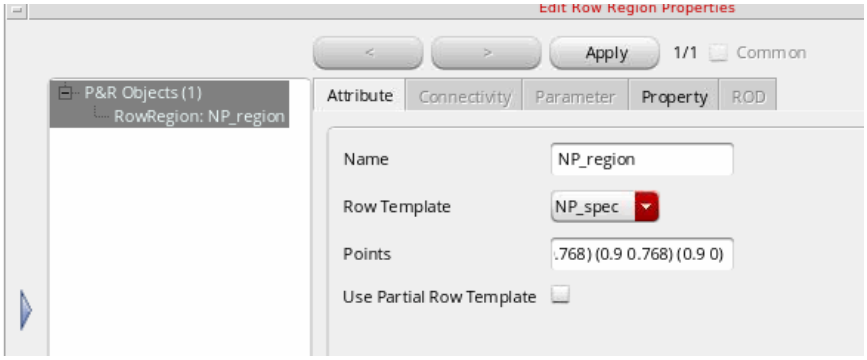
Palette

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.

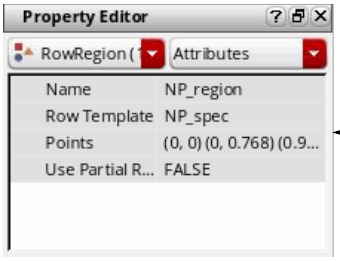


Property Editor and Edit Properties Form

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.



Edit Properties Form



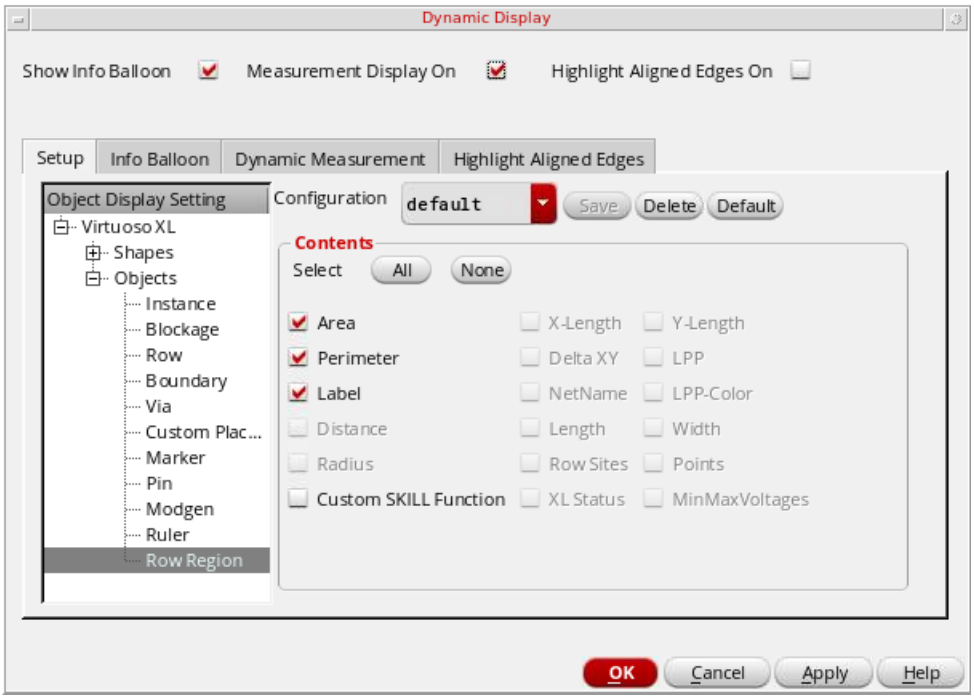
Property Editor Assistant

Row Region Attributes

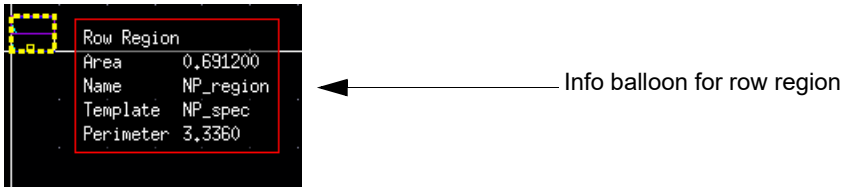
Name	Displays the name of the row region.
Row Template	Specifies the row template to be used to create the row region.
Points	Sets the coordinates each point of the row region.
Use Partial Row Template	Repeats the partially till there is space to fill a whole row. If not selected, only a full row region spec is created. If the space remaining on the top is less than the height of a row region spec, it is left empty.

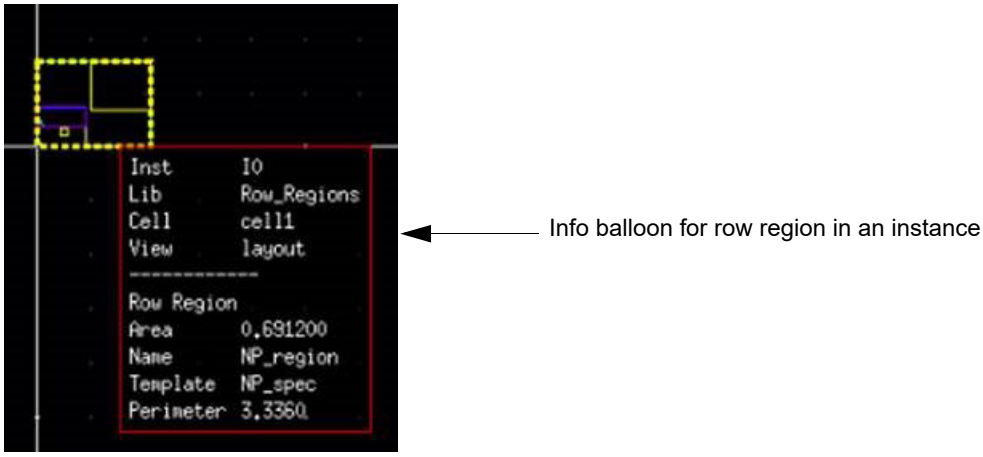
Dynamic Display

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.

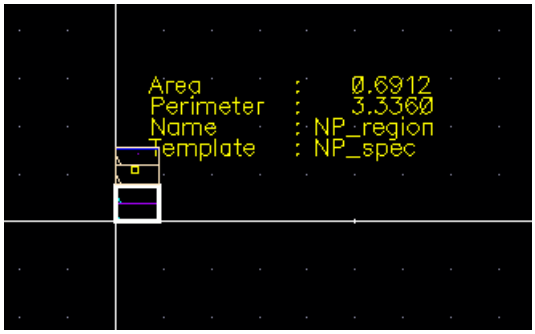


The figures below display the info balloon for the row regions.





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

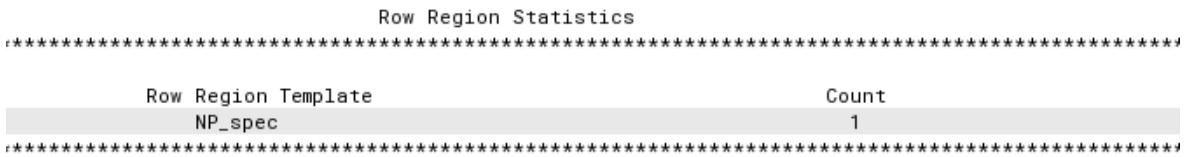


Show Selection Info Toolbar

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

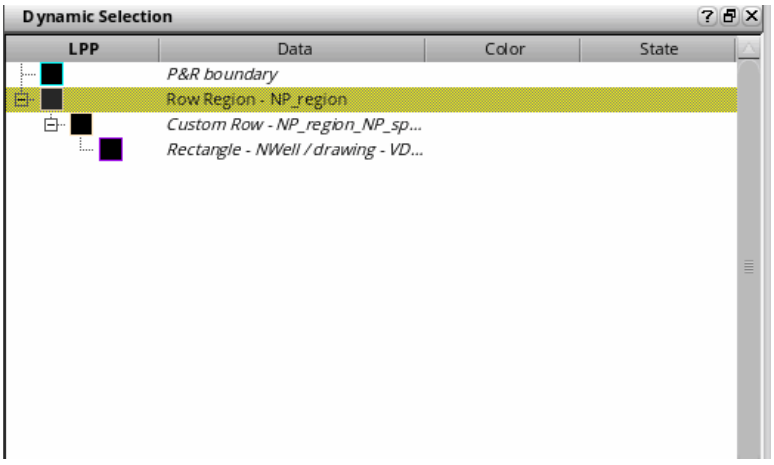
Summary Window

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



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.



Related Topics

Row Generation

Editing Support for Row Regions

Editing Support for Row Regions

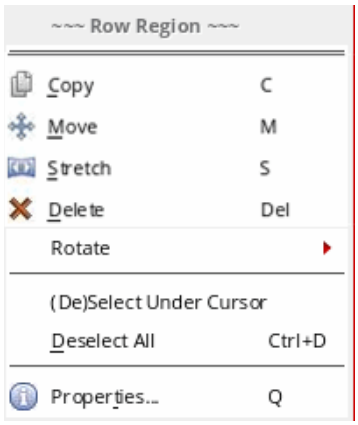
You can use various commands to edit row region objects like other shapes in the layout editor.

Some exceptions for a few commands are:

- For the *Delete* command, partial deletion (for example a vertex) is not supported.
- For the *Rotate* command, only orthogonal editing is supported.
- For the *Chop* command, grid-aware orthogonal snapping is supported for the line, rectangle, and polygon chop shapes if grids are specified in the row template manager. For the line chop shape, grid-aware orthogonal snapping is supported for no spacing and the *User Spacing* type.

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

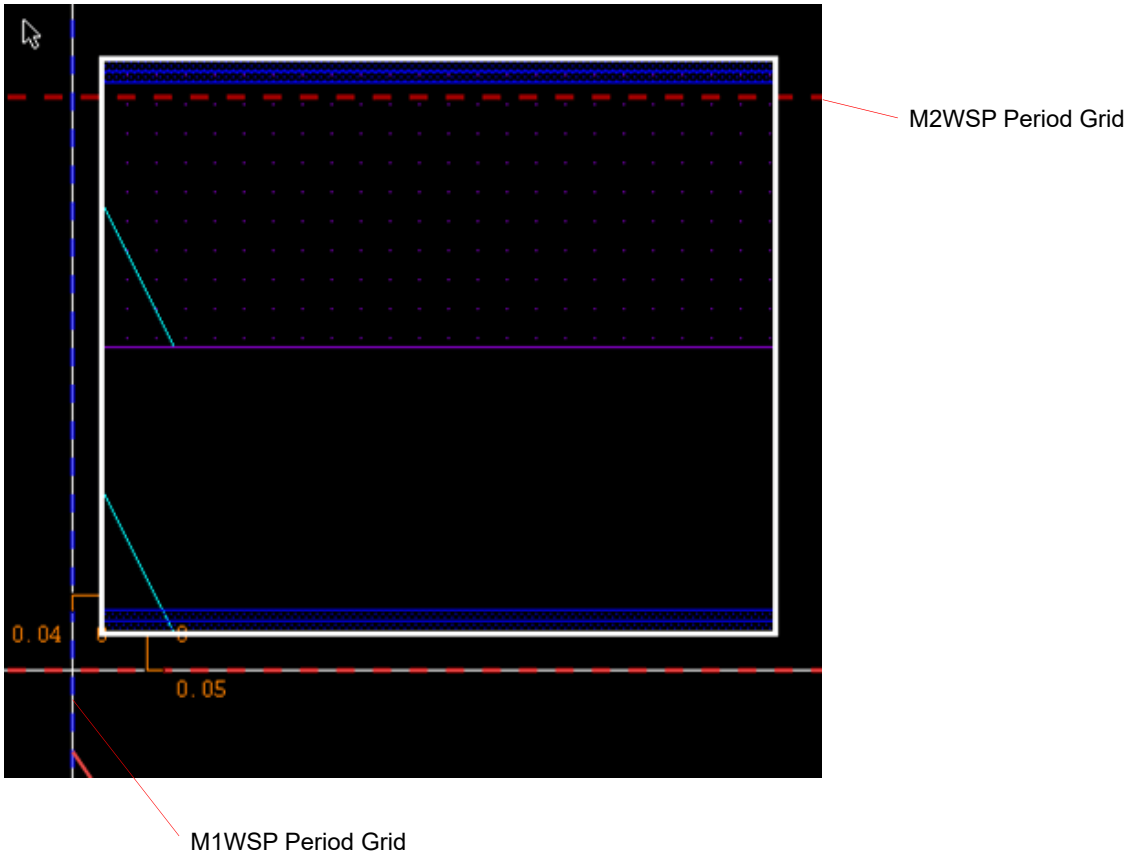
A context-sensitive menu is also available for row region objects. You can use the context-sensitive menu to access commonly used editing commands.



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

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.

Related Topics

Row Generation

Layout Support for Row Regions

Load Physical View

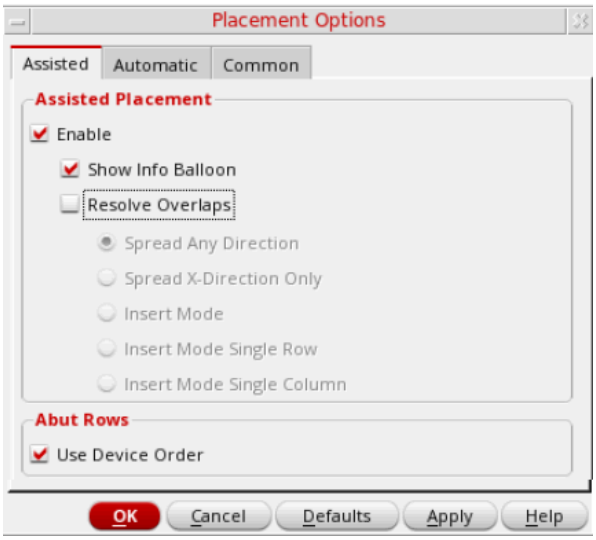
3

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.

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



The Placement Options form has the following tabs:

- *Assisted*, which provides options for assisted move commands and specifies assisted placement settings. This tab is displayed by default.
- *Automatic*, which specifies automatic placement settings.
- *Common*, which includes options that are relevant for both assisted and automatic placement.

Both, assisted and automatic placement methods support snapping of individual objects and object groups to grids.

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.

Related Topics

Setting the Assisted Placement Options in the Placement Options Form

Setting Automatic Placement Options in the Placement Options Form

Setting Common Options in the Placement Options Form

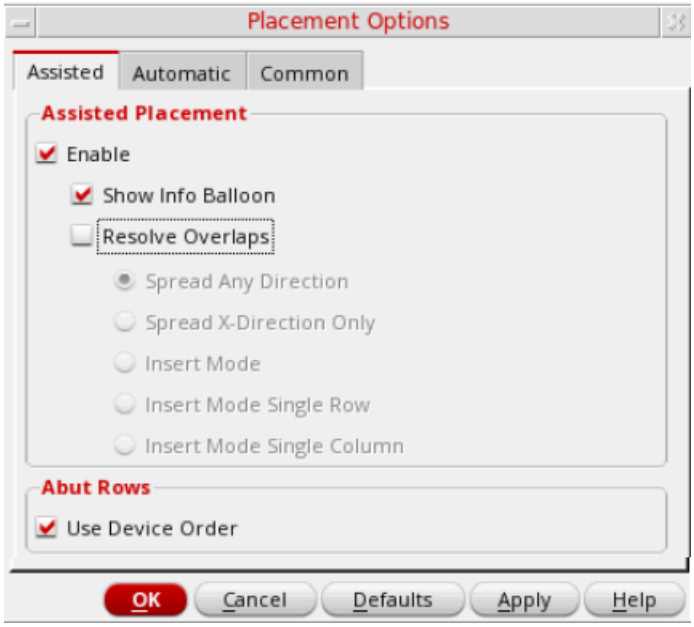
Snap Behavior

Setting the Assisted Placement Options in the Placement Options Form

To specify assisted placement options:

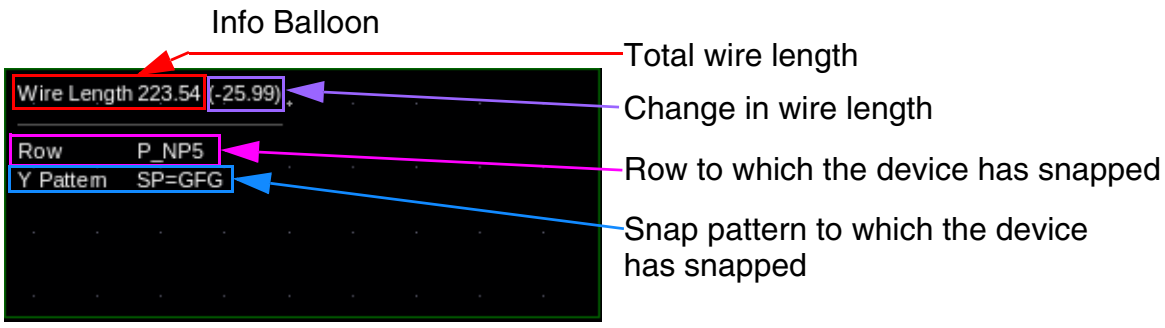
1. Select *Place – Placement Options* to display the Placement Options form.

   The *Assisted* tab is displayed by default.



2. Select *Enable* in the *Assisted Placement* section to specify that instances are to be snapped to the nearest rows instead of grids, as shown below:

3. Select *Show Info Balloon* option to display an info balloon when instances are moved with *Enable* selected. An info balloon displays the following information:



4. Select *Resolve Overlaps* to automatically resolve any overlaps resulting from moving devices in a design. An arrow indicates how the overlap is 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.

5. Select the mode for resolving overlaps.
6. Select *Use Device Order* in the *Abut Rows* section to abut instances in the same order in which they are placed in each row.

Related Topics

Design Placement

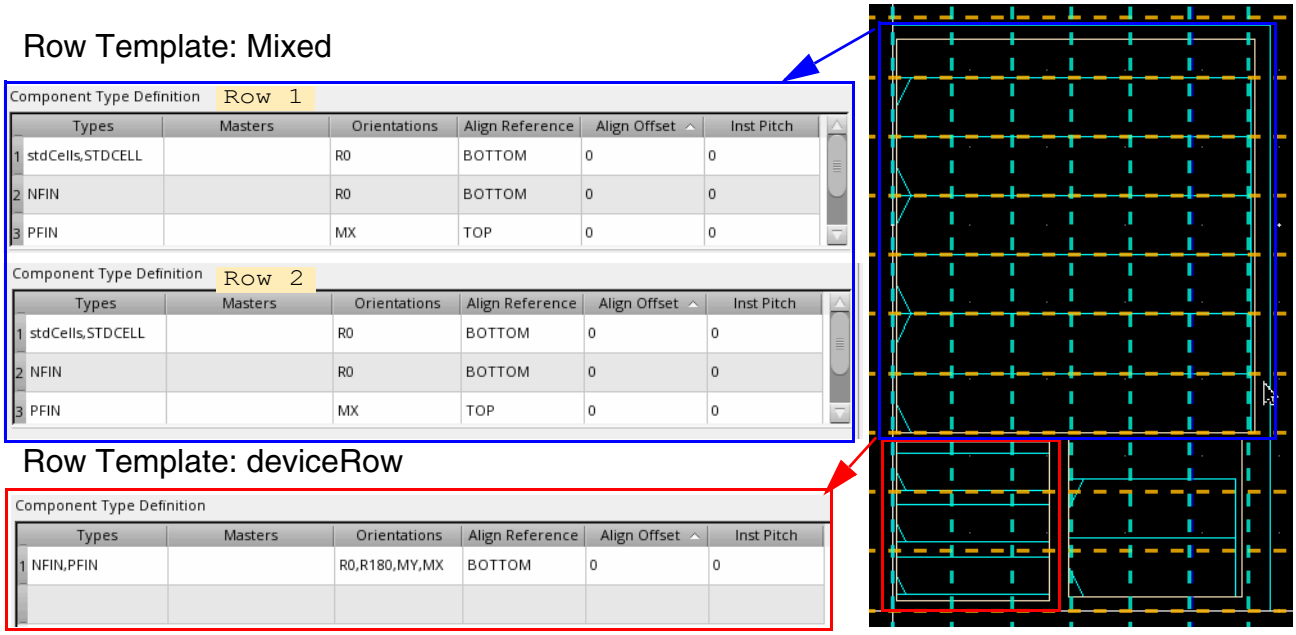
Layout XL Environment Variables

Assisted Placement of Multiple Device Instances

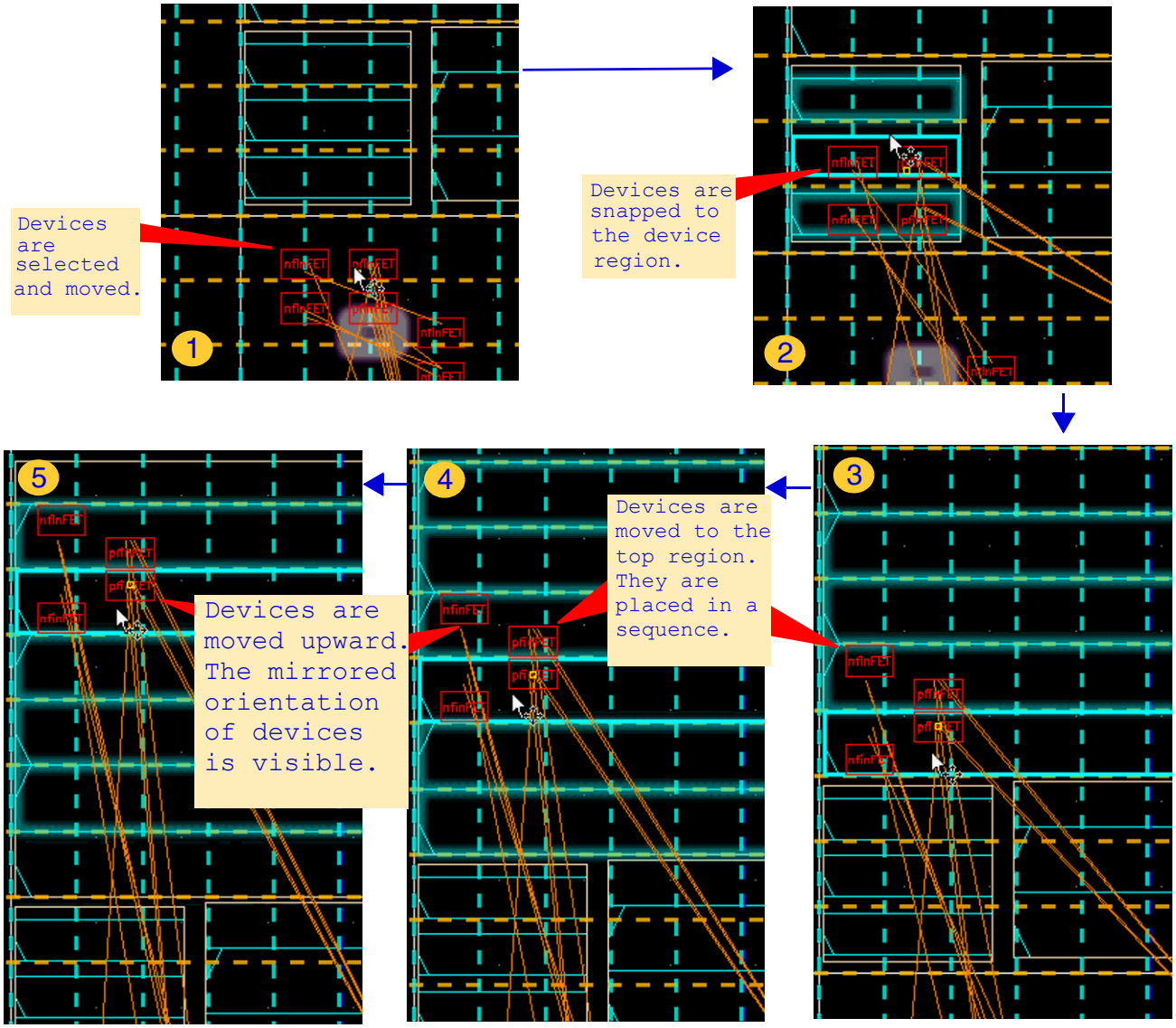
Assisted Placement of Multiple Device Instances

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.



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



Related Topics

Placement Options Form

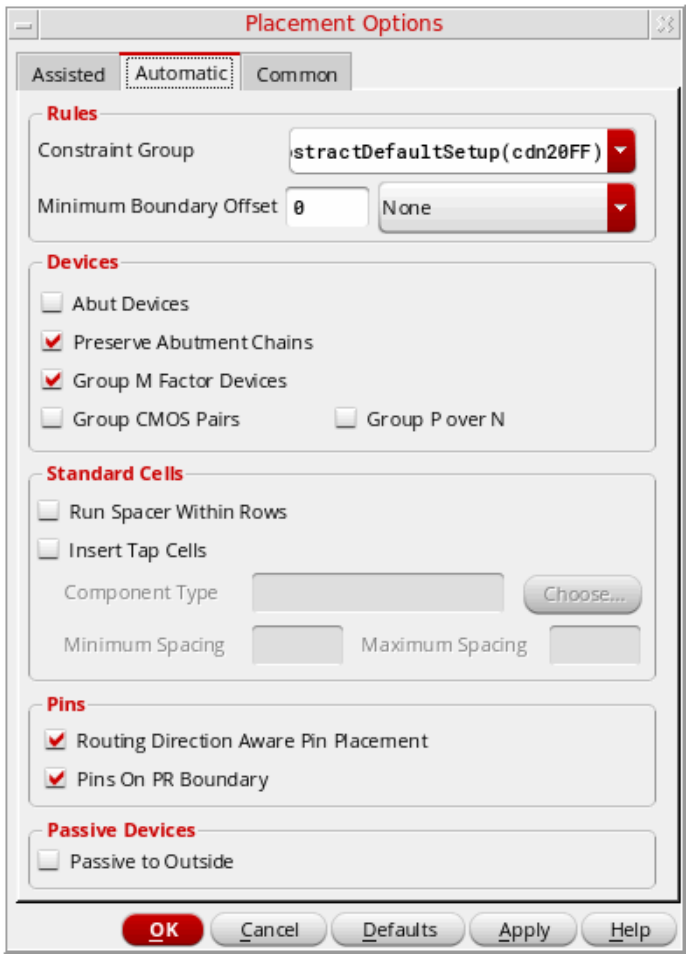
Setting the Assisted Placement Options in the Placement Options Form

Setting Common Options in the Placement Options Form

Running the Automatic Placer

Setting Automatic Placement Options in the Placement Options Form

Before running the automatic placer, you must set the placement options on the *Automatic* tab of the Placement Options form. This tab includes options that are generic as well as those that are specific to the *Custom Digital* placement type.



To specify automatic placement settings:

1. Open the Placement Options form by using one of the following methods:
   - Choose *Place – Placement Options.*
   - Click *More Options* in the Automatic Placement form.
2. Click the *Automatic* tab.

3. In the *Rules* section, select a *Constraint Group* that contains the spacing rules to be applied while running the automatic placer.
4. Select *Minimum Boundary Offset* to specify that minimum spacing is required between the selected reference layer and the PR boundary.
5. In the *Devices* section, select *Abut Devices* to create abutment chains in each row.
6. Select *Preserve Abutment* to preserve existing abutment chains in the design.
7. Select *Group M Factor Devices* to create a cluster containing all complementary MOS devices in the design that have a multiplication factor in the schematic.
8. Select *Group CMOS Pairs* to group CMOS devices that share gate connections.
9. 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.
10. In the *Standard Cells* section, select *Run Spacer Within Rows* to distribute standard cells evenly in the rows by adding or removing space between standard cells within each row.
11. Specify tap cells parameters:
    - Select *Insert Tap Cells* to insert tap cells in rows.
    - Specify the *Minimum Spacing* and *Maximum Spacing* values.
    - Specify the *Component Type* that contains the substrate contacts (tap cells) to be used.
12. In the *Pins* section, select *Routing Direction Aware Pin Placement* to let the placer consider the WSP direction while placing unconstrained pins.
13. Select *Pins on Boundary* to place all pins along the boundary during placement.
14. In the *Passive Devices* section, select *Passive to Outside* to push large capacitors and resistors to the edge of the PR boundary.

Related Topics

Placement Options Form

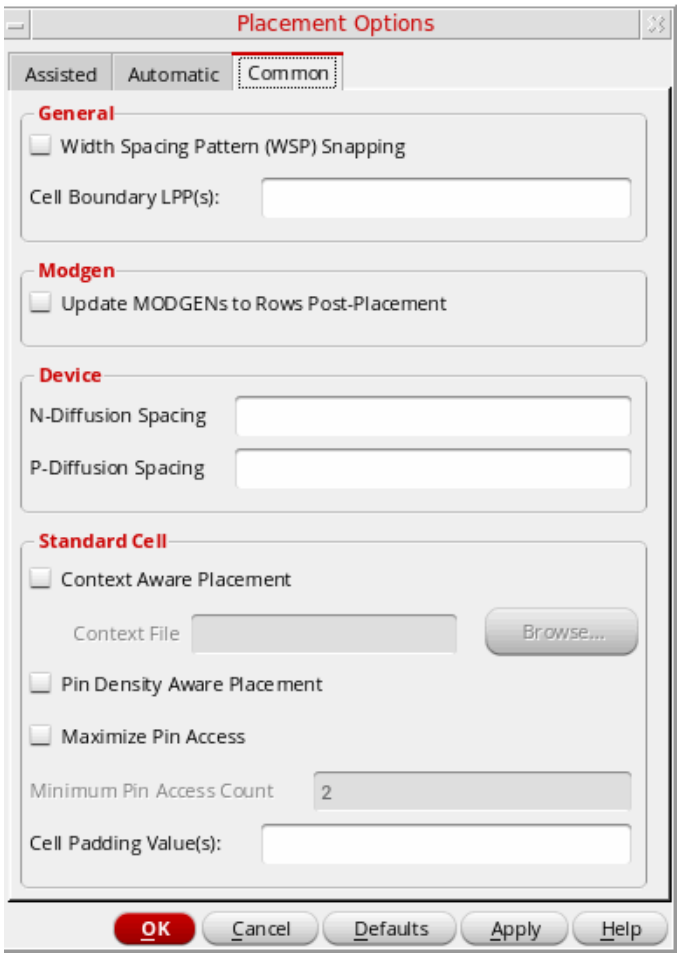
Setting the Assisted Placement Options in the Placement Options Form

Setting Common Options in the Placement Options Form

Running the Automatic Placer

Setting Common Options in the Placement Options Form

The *Common* tab of the Placement Options form provides settings that are common to both *Assisted* and *Automatic* placement.



To specify common settings:

1. Open the Placement Options form by using one of the following methods:
   - Choose *Place – Placement Options.*
   - Click *More Options* in the Automatic Placement form.

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

   a. Select *Width Spacing Pattern (WSP) Snapping* to enable width-aware and constraint (CST)-aware snapping of instances and pins.

   b. 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.

3. In the *Modgen* section, select *Update MODGENS to Rows Post-Placement* to regenerate Modgens after they are moved to a row region.

4. 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.

5. In the *Standard Cell* section:

   a. Select *Context Aware Placement* to let standard cells be placed such that each cell is aware of the context of its neighboring cell.

   b. Select *Pin Density Aware Placement* to consider the pin densities of standard cells while optimizing placement and improving routability.

   c. Select *Maximize Pin Access* to increase the number of WSP tracks that overlap the pins on standard cells.

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

   e. In the *Cell Padding Value* field, specify the space to the left and right of the master.

Related Topics

Placement Options Form

Setting the Assisted Placement Options in the Placement Options Form

Setting Automatic Placement Options in the Placement Options Form

Width Spacing Pattern Layer Rules

Running the Automatic Placer

Running the Automatic Placer

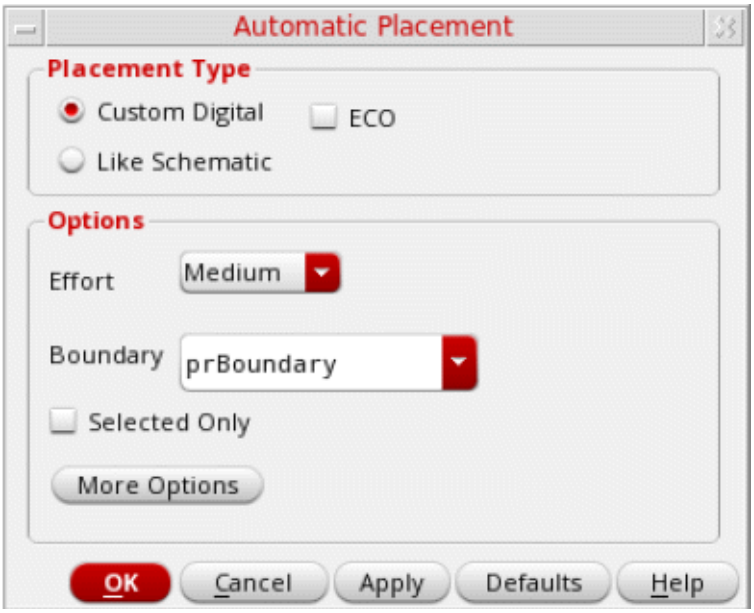
The automatic placer places devices in rows defined in the layout canvas. For best results, run the automatic placer after performing Placement Planning. You can use the automatic placer for placing devices, standard cells, or a mix of both. During placement, these devices are snapped to appropriate rows and grids. The automatic placer supports placement of virtual figGroups.

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.

To run the automatic placer:

1. Select *Place – Auto Place.* The Automatic Placement form is displayed.

   Alternatively you can choose the Auto Place button from either the Placement toolbar or the Design Planning toolbar.



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

   - ***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.

- ***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.

   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

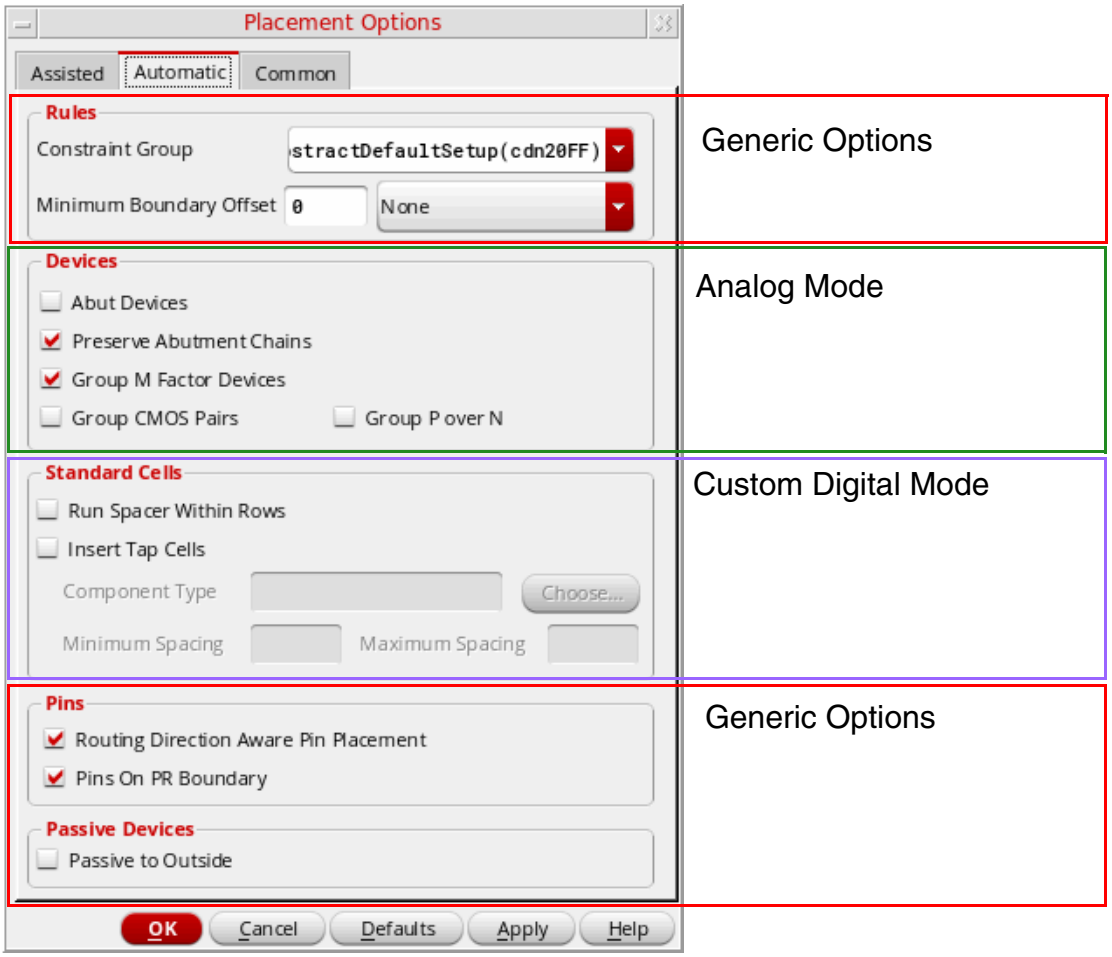
The design is compacted on each subsequent run, and therefore the total area, wire length, and space between devices keeps decreasing with each run.

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.

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.



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

Related Topics

Automatic Placement Form

Setting the Assisted Placement Options in the Placement Options Form

Setting Automatic Placement Options in the Placement Options Form

Setting Common Options in the Placement Options Form

Placement of Virtual FigGroups

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.

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.

Related Topics

Automatic Placement Form

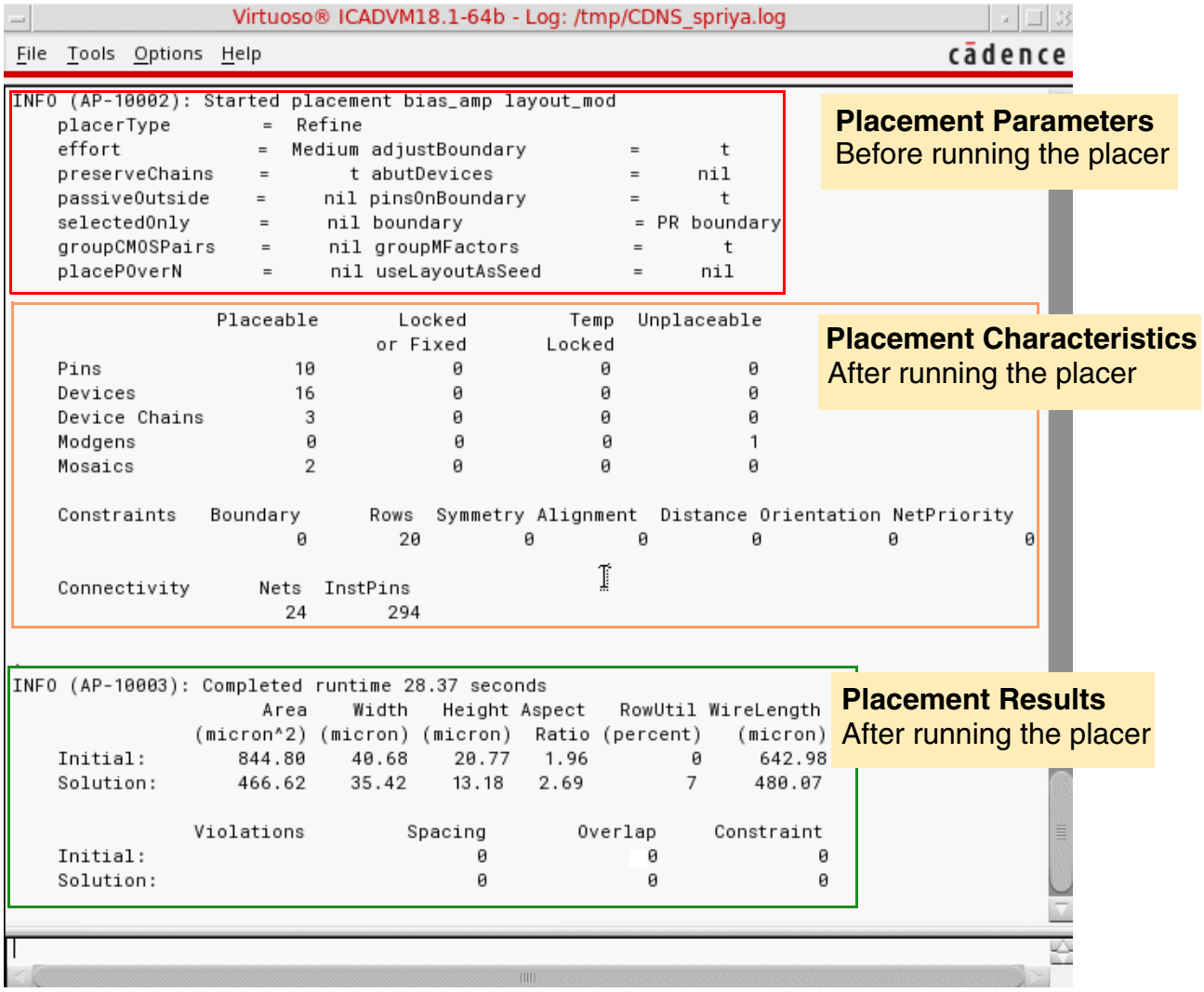
Setting the Assisted Placement Options in the Placement Options Form

Setting Automatic Placement Options in the Placement Options Form

Setting Common Options in the Placement Options Form

The Placement Statistics Report

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



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.

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:**

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

  Note: Unplacable components are row-based objects in the design, but none of the rows in the design match these objects. 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.

Related Topics

Setting the Assisted Placement Options in the Placement Options Form

Setting Automatic Placement Options in the Placement Options Form

Setting Common Options in the Placement Options Form

Running the Automatic Placer

VLS-GXL Placement Status

Assisted Placement of Devices

Assisted placement refers to the manual adjustments done to placed device instances after running Virtuoso Placer. Assisted placement is generally done to improve device placement as per your design requirements.

Assisted placement commands are available in the *Place* menu. These commands support transparent group mode, in which you can edit individual members of user-defined groups.

Note: Before moving devices using the assisted placement commands, in the Display Options form, set the *Show Name Of* option to *instance* or *both*.

Guidelines for running the assisted placement commands are:

- Start with assisted move to group devices with similar masters and similar finger lengths. No device should be adjacent to another dissimilar device.
- 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.
- After you have grouped devices by type, run the required assisted placement commands to improve placement.
- Press `Ctrl+D` to deselect all devices, and then carefully select only the ones you need before running an assisted placement command.

The following assisted placement commands are available in the *Place* menu:

Resolve Overlaps	Resolves overlaps between devices in rows to remove 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 instances in the design. If no instances are selected before running the command, all instances in the design are unabutted.

Swap Instances	Swaps the positions of the selected instances, chains, and figGroups. The locations of the selected instances are switched but their orientations are retained as in their initial position.
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.
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.
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.

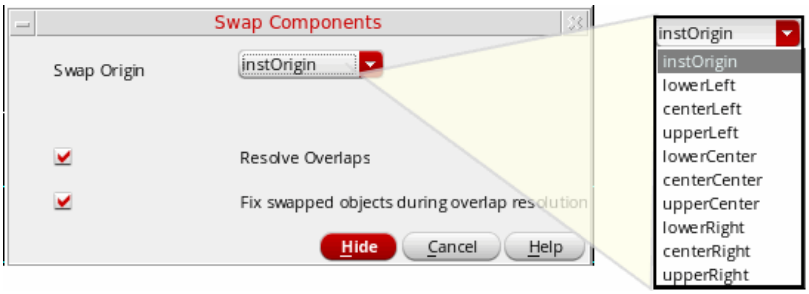
Swapping Instances

Use the Swap Instances assisted placement command to swap positions of the selected instances, chains, or figGroups.

1. Choose *Place—Swap Instances*.
2. Click the first instance to be swapped.
3. Click the second instance.

   Note: You can also select the components first and then choose *Place—Swap Instances*. After selecting the command,

4. Press `F3` to display the Swap Components form.



5. Select a position from the *Swap Origin* list 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.
6. Select *Resolve Overlaps* to automatically resolve overlaps while placing the devices at their new locations.
7. Select *Fix swapped objects during overlap resolution* to ensure that the position of the swapped device is not changed while resolving overlaps.
8. Click *Hide* to run the command on the selected instances.

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

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.

Related Topics

Setting the Assisted Placement Options in the Placement Options Form

Setting Automatic Placement Options in the Placement Options Form

Running the Automatic Placer

Assisted Placement of Devices

Swapping Components

Using Transparent Group Mode

Snap 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.
2. Snap to a WSP grid.
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.

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.

Related Topics

Setting the Assisted Placement Options in the Placement Options Form

Setting Automatic Placement Options in the Placement Options Form

Setting Common Options in the Placement Options Form

Running the Automatic Placer

Assisted Placement of Devices

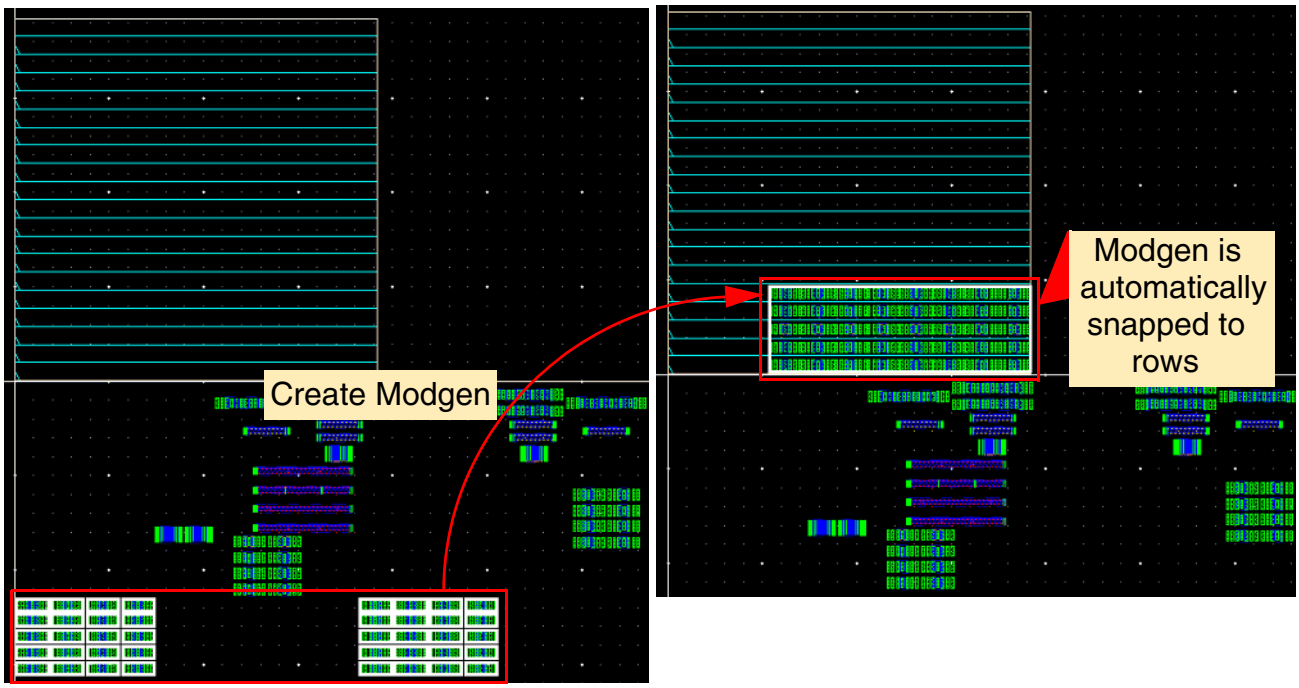
Row Generation

Using Width Spacing Patterns

Virtuoso Placer 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.

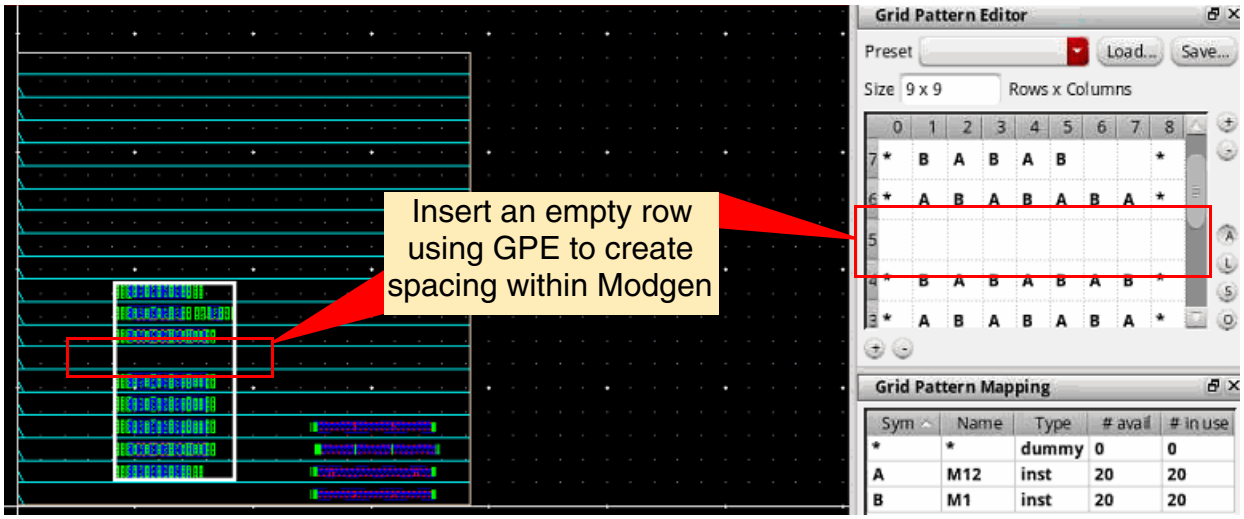
Support for Modgen Pattern Editors

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.

Insertion of Spaces 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:



Guard Rings in Virtuoso Placer Flow

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.

Related Topics

Using the Modgen On-Canvas Commands

4

Placement Post-Processing

Placement post-processing tasks 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 Topics

Inserting and Deleting Filler Cells

Base Layer and Dummy Fill Insertion

Verifying Placement

Types of Fill

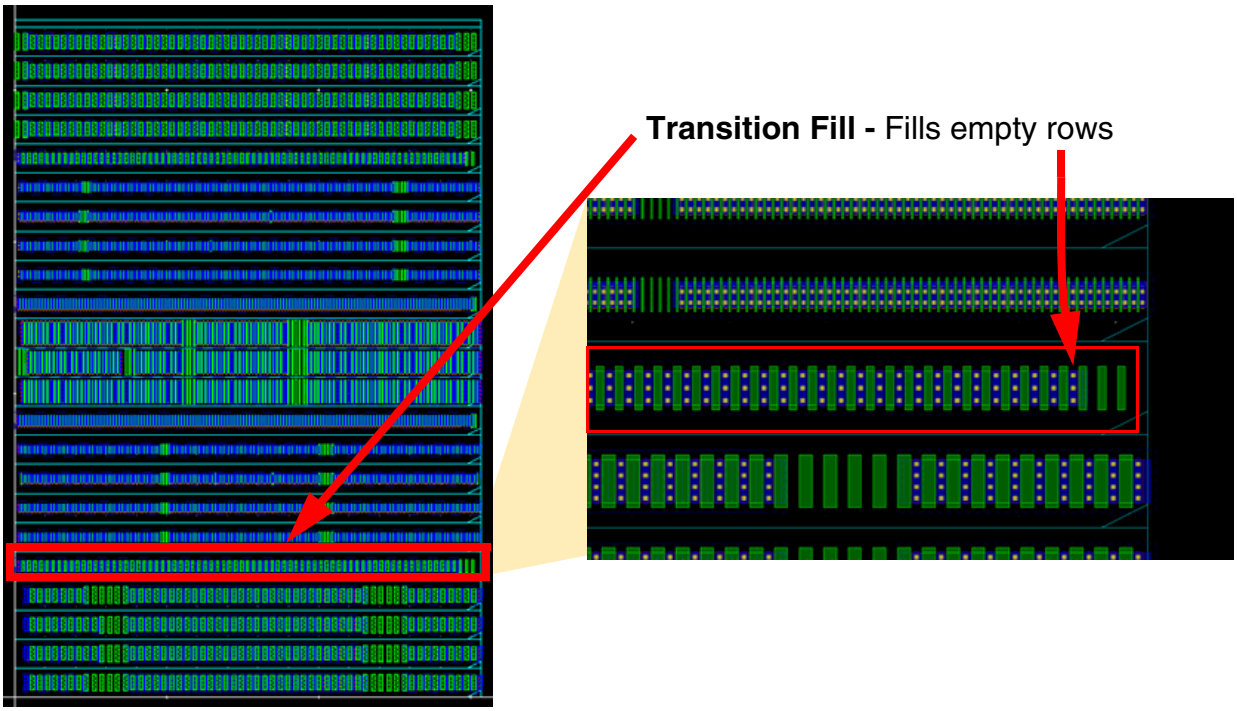
Dummy fill can be classified into the following types:

- **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 is 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.

- **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.



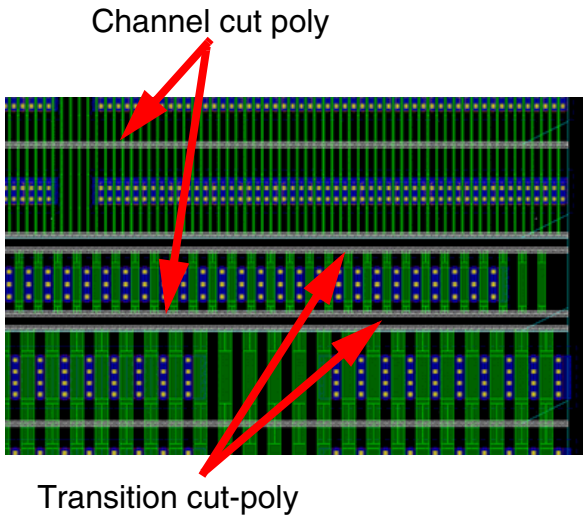
- **Poly fill:** Extends polysilicon fingers vertically to the middle of the nearest cut-poly rail. This helps achieve better poly density, as specified by the foundry and mitigates density gradient effects (DGE).

  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.

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.



Related Topics

Base Layer and Dummy Fill Insertion

Fill Definitions for the Virtuoso Placer Flow

Base Layer and Dummy Fill Insertion

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.

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.

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

Use the options in the Base Layer and Dummy Fill form to insert dummy fill in your design.

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.

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.

Related Topics

Base Layer Fill in the Automated Device Placement and Routing FlowFill

Placement Post-Processing

Types of Fill

Fill Definitions for the Virtuoso Placer Flow

Fill Definitions for the Virtuoso Placer Flow

Fill definitions can be created 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") ("l" "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:

- **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.

Related Topics

Base Layer and Dummy Fill Insertion

Types of Fill

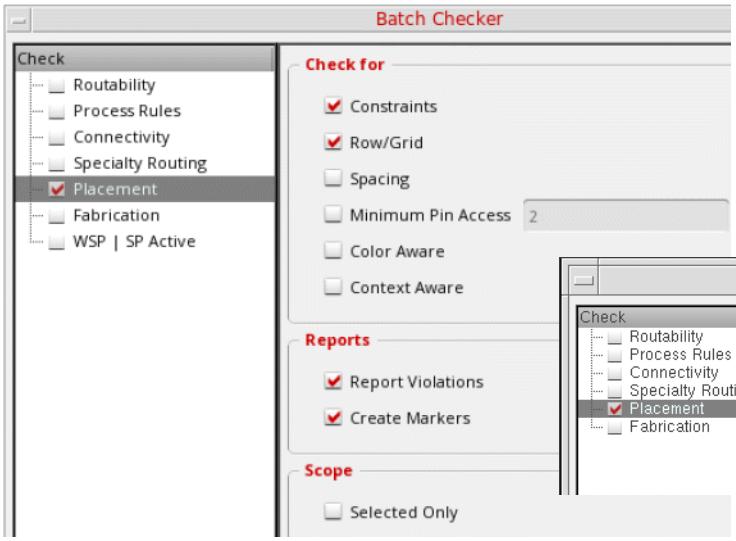
Base Layer and Dummy Fill Insertion

Verifying Placement

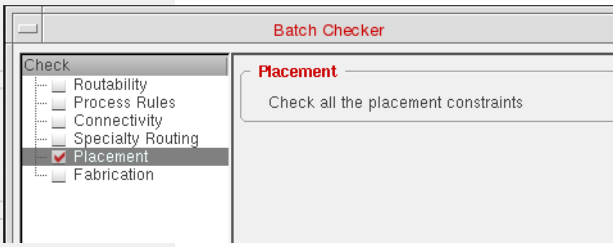
After running Virtuoso Placer, you can verify whether the placement is DRC-correct. Use the Batch Checker form to verify constraints and width spacing pattern (WSP)-specific settings in your design.

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:

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 required checks: *Constraints*, *Row/Grid*, *Spacing*, *Minimum Pin Access*, *Color Aware*, and *Context Aware*.

   You can use Cadence Physical Verification System (PVS) Constraint Validation (CV) to run a comprehensive check of all placement and routing constraints.

   Use the ignoreOverlapCheckProp environment variable to ignore overlaps between top-level instances or figGroups when verifying the placement.

4. In the *Reports* section, select how errors are to be highlighted: *Report Violations* or *Create Markers.*

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.

Related Topics

Batch Checker Form

Placement Options Form

A

Virtuoso Placer Forms

This topic lists the forms that are used to in the Virtuoso Placer flow.

- Automatic Placement Form
- Placement Options Form
- Row Template Manager Form
- Swap Components Form

Automatic Placement Form

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

Field	Description
Placement Type	This group box specifies the placement options for digital and analog designs. Environment variable: placerType
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.
Options	This group box sets 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

Running the Automatic Placer

Setting the Assisted Placement Options in the Placement Options Form

Setting Automatic Placement Options in the Placement Options Form

Setting Common Options in the Placement Options Form

Placement Options Form

The Placement Options form lets you control how various assisted and automatic placement tasks are to be performed. The form contains the following tabs.

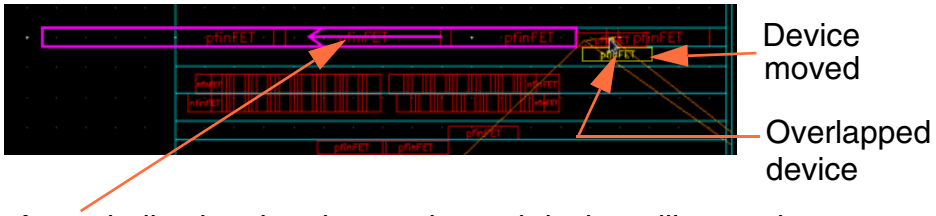
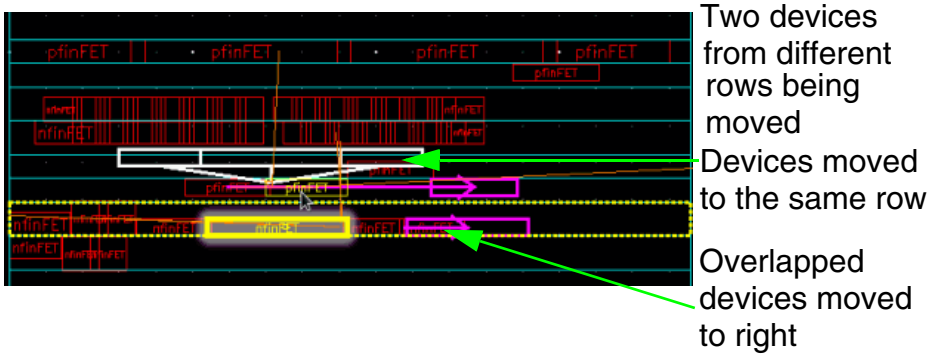
Tab	Description
Assisted	Specifies settings for the assisted placer.
Automatic	Specifies settings for the automatic placer.
Common	Specifies settings that are applicable to both, *Assisted* and *Automatic* modes.

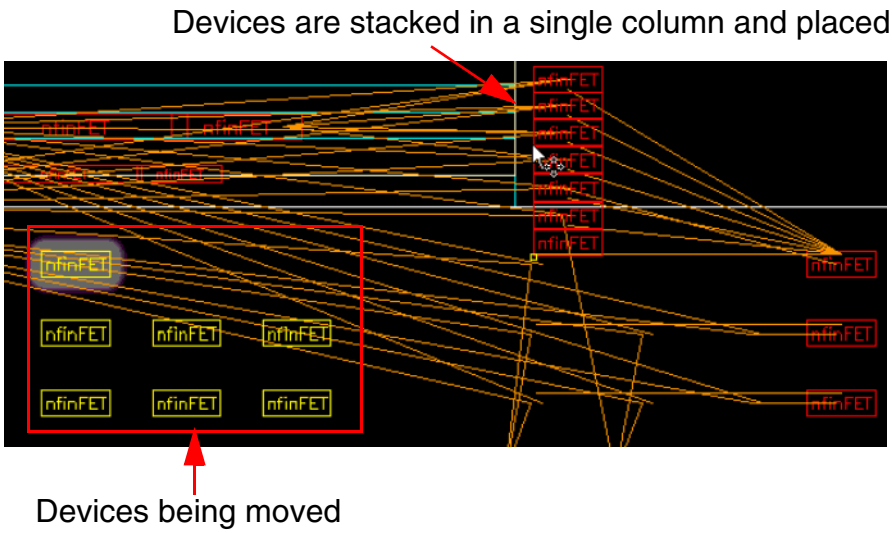
Assisted

The following table describes the fields available on the *Assisted* tab of the Placement Options form.

Field	Description
Assisted Placement	This group box provides advanced assisted features while moving objects in the layout. **Note: Assisted Placement** is also referred to as **Assisted Move**. Environment variable: apMove

Field	Description
Enable	Enables snapping of instances to the nearest rows instead of grids. The option also switches on *Resolve Overlaps* options. The tool considers tap cells and boundary cells as fixed cells and so does not snap them to rows. Adding tap cells and boundary cells helps avoid latch-up effects and DRC violations. Boundary cells are cells with component class `BOUNDARYCELL` and tap cells are cells with component class `STDSUBCONT`. For more information about boundary cells and tap cells, see Placement Planning. *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. Environment variable: apMove
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 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. 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.

Field	Description
Resolve Overlaps	*A*utomatically 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. Environment variables: spreadType, insertDefaultSpreadType
Spread X-Direction Only	Moves the overlapped device by a minimum distance along the right or left directions within the same row as indicated below:  Environment variables: spreadType, insertDefaultSpreadType
Insert Mode	Moves the overlapped device to the right only. 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. below: 

Field	Description
Insert Mode Single Row	Moves the overlapped device to the right. The instances that are moved are placed sequentially in the same row. Environment variables: insertDefaultSpreadType
Insert Mode Single Column	Stacks the selected devices vertically before moving them. The overlapped devices are moved to the right as shown below:  Environment variable: spreadType
Abut Rows	This group box specifies device abutment settings.
Use Device Order	Controls whether abutment is based on connectivity or on the order of devices in the cellview (default). Environment variable: useDeviceOrder 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.

Automatic

The following table describes the fields available on the *Automatic* tab of the Placement Options form.

Field	Description
Rules	This group box specifies the automatic placement rules. The options in the *Rules* section are generic, and apply to all design types.
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.
Devices	This section includes the following options applicable only to analog designs. These options are 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. All such CMOS device pairs in the design that have been generated with chaining enabled are grouped together into a cluster.
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	This group box includes options that are applicable only to digital designs, and therefore apply only to the *Custom Digital* placement type in the Automatic Placement form.

Field	Description
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	This group box defines the following pin placement settings. These settings are generic, and so are applicable to all design types.
Routing Direction Aware Pins	Specifies whether the placer must consider the WSP direction while placing unconstrained pins. This option is enabled by default and is applicable only to designs with width spacing patterns (WSPs). For vertical WSPs, pins are placed along the left and right edges; for horizontal WSPs, pins are placed along the top and bottom edges. Environment variable: routingAwarePinPlc
Pins on Boundary	Places all pins on the PR boundary. Environment variable: pinsOnBoundary
Passive Devices	This group box specifies settings for passive devices:

Field	Description
Passive to Outside	Pushes passive devices such as large capacitors and resistors to the edge of the PR boundary, thereby separating them from the transistors in the design. This option is generic, and applicable to all design types.  *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. Environment variable: passiveOutside

Common

The following table describes the fields available on the *Common* tab of the Placement Options form.

Field	Description
General	This group box specifies settings that are applicable to both devices and standard cells.

Field	Description
Width Spacing Pattern (WSP) Snapping	Enables 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. *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. 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.
Modgen	This group box specifies settings for 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. 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`. When deselected, Modgens are snapped only to rows with matching orientations. Environment variable: regenModgenPostProcess
Device	This group box specifies device-specific settings.
N-Diffusion Spacing	Specify the diffusion spacing for the NMOS transistors within the row.

Field	Description
P-Diffusion Spacing	Specify the diffusion spacing for the PMOS transistors within the row.
Standard Cell	This group box specifies settings that 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, this option is switched off. You can also specify a *Context File*, which is an 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.
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. The assisted placement options are also pin density-aware. The pin density consideration can result in larger spacing between the placed standard cells. Pin density awareness is a heuristic to improve routability, and therefore is not checked during placement verification. Environment variable: pinDensityAwarePlacement
Maximize Pin Access	Maximizes the number of WSP tracks that overlap the pins. 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. Environment variable: maximizePinAccess
Minimum Pin Access Count	Specifies the minimum number of pin access points to achieve during the process of pin access maximization. Environment variable: minNumPinAccessPts

Field	Description
Cell Padding Value(s)	Specifies the space to be left on either side 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. Environment variable: paddingForCells

Related Topics

Setting the Assisted Placement Options in the Placement Options Form

Setting Automatic Placement Options in the Placement Options Form

Setting Common Options in the Placement Options Form

Row Template Manager Form

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. The form contains the following tabs:

Tab	Description
Edit	Provides options to create, edit, and delete row templates. Row templates are stored directly in the layout.
Import	Lets you import a row template from the same or a different cellview. 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.

Edit

The following table describes the fields available on the *Edit* tab of the Row Template Manager.

Field	Description
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.

Field	Description
Placement Grid	This group box specifies the row grid values.
Vertical Grid	Specifies the reference grids along the y axis for snapping the devices during placement. Environment variable: verticalGrid
Horizontal Grid	Specifies the reference grids along the x axis for snapping the devices during placement. Environment variable: horizontalGrid
Vertical Offset	Specifies the reference row offset values along the y axis to be applied to rows after they are snapped to the reference grids. The offset value is a correction factor for the SP. Environment variable: verticalOffset
Horizontal Offset	Specifies the reference row offset values along the x axis to be applied to rows after they are snapped to the reference grids. The offset value is a correction factor for the SP. Environment variable: horizontalOffset
Related Snap Pattern	Creates a WSP region in addition to the row region. Environment variable: relatedSnapPattern
Rows	This group box provides a table that 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`.

Field	Description
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).
Site Def	Lists all the site definitions from the technology file. Select the required site definition. *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). 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	This group box 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 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 `*/*/*`.

Field	Description
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. If *InstPitch* is set to `0`, 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.
Rail Definition	This group box contains a list of rail definitions for each row.
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

The following table describes the fields available on the *Import* tab of the Row Template Manager.

Field	Description
Library, Cell, and View	Specifies the cellview containing the row template to be imported. Environment variables: importLibName, importCellName, importViewName
Templates	Lists all row templates stored in the selected cellview. You can select the templates to be imported.
Current Cellview	This group box provides information about row templates in the current cellview.
Active Templates	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

Introduction to Row Infrastructure

Creating, Editing, and Deleting a Row Template

Swap Components Form

The Swap Components form lets you control how the positions of the selected instances, chains, and figGroups are swapped.

Field	Description
Swap Origin	Specifies 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.
Resolve Overlaps	Resolves instance overlaps.
Fix swapped objects during overlap resolution	Ensures that the position of the swapped device is not changed while resolving overlaps.

Related Topics

highlightPotentialSwap

highlightPotentialSwapColor

Assisted Placement of Devices

B

Virtuoso Placer Environment Variables

The list below provides information on the names, descriptions, and graphical user interface equivalents for the environment variables used in the Virtuoso Placer flows.

Virtuoso Placer Environment Variables		
abutDevices	adjustBoundary	effort
likeSchematicTolerance	passiveOutside	pinsOnBoundary
placePOverN	placerType	
Row Template Manager Environment Variables		
horizontalGrid	horizontalOffset	importCellName
importLibName	importViewName	relatedSnapPattern
verticalGrid	verticalOffset	
Assisted Placement Environment Variables		
allowRotationFigGroup	apMove	displayResolveOverlapsColor
displaySnappedInfo	fixAdjustedForResolveOverlaps	highlightSnappableRowsForInst
highlightSnappableRowsForProxy	infoBalloon	insertColor
insertDefaultSpreadType	largeModgenEnvelopeCheck	largeModgenEnvelopeThreshold
maximizePinAccess	minNumPinAccessPts	numSnappableRows
paddingForCells	pinDensityAwarePlacement	regenModgenPostProcess
snapColor	snapFigGroupRowCorrect	spreadEnabled
spreadType	swapRowMode	useDeviceOrder
useLEReference		

Dummy Fill Insertion Environment Variables		
lobFillAsMosaic	lobDevFillPhysOnly	rowTempInfoAwareFillerInsertion
Placement Checker Environment Variables		
constraints	rowOrGrid	spacing
tapCell	padding	minPinAccess
colorAware	reportViolations	createMarkers
selOnly		

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	*Place – Placement Options*
Field	*Automatic – Devices – Abut Devices*

Examples

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

Related Topics

Placement Options Form

Setting Automatic Placement Options in the Placement Options Form

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	*Place – Auto Placer*
Field	*Adjust PR Boundary*

Examples

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

Related Topics

Automatic Placement Form

Running the Automatic Placer

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	*Place – Auto Place*
Field	*Options – Effort*

Examples

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

Related Topics

Automatic Placement Form

Running the Automatic Placer

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 allow the placer to use more approximate schematic correspondence to improve the placement. Valid values are `1` through `1000`.

The default value is `0`.

GUI Equivalent

None

Examples

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

Related Topics

Automatic Placement Form

Running the Automatic Placer

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	*Place – Placement Options*
Field	*Automatic – Passive Devices – Passive to Outside*

Examples

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

Related Topics

Placement Options Form

Setting Automatic Placement Options in the Placement Options Form

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*

Examples

```
envGetVal("layoutXL.AP" "pinsOnBoundary")
envSetVal("layoutXL.AP" "pinsOnBoundary" 'boolean nil)
```

Related Topics

Placement Options Form

Setting Automatic Placement Options in the Placement Options Form

Virtuoso Placer Environment Variables

placePOverN

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

Description

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

The default is `nil`.

GUI Equivalent

Command	*Place – Placement Options*
Field	*Automatic – Devices – Group P over N*

Examples

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

Related Topics

Placement Options Form

Setting Automatic Placement Options in the Placement Options Form

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	*Place – Auto Place*
Field	*Placement Type*

Examples

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

Related Topics

Automatic Placement Form

Running the Automatic Placer

Virtuoso Placer Environment Variables

horizontalGrid

```
layoutXL.APAssist horizontalGrid string "grid_name"
```

Description

Specifies the reference grids along the x axis for snapping the devices during placement.

The default is `"none"`.

GUI Equivalent

Command	*Place – Row Template Manager – Edit*
Field	*Placement Grid – Horizontal Grid*

Examples

```
envGetVal("layoutXL.APAssist" "horizontalGrid")
envSetVal("layoutXL.APAssist" "horizontalGrid" 'string "GPG90")
```

Related Topics

Creating, Editing, and Deleting a Row Template

Row Template Manager Form

Virtuoso Placer Environment Variables

horizontalOffset

```
layoutXL.APAssist horizontalOffset float horizontal_Offset
```

Description

Specifies the reference grids along the x axis for snapping the devices during placement.

The default is `0`.

GUI Equivalent

Command	*Place – Row Template Manager – Edit*
Field	*Placement Grid – Horizontal Offset*

Examples

```
envGetVal("layoutXL.APAssist" "horizontalOffset")
envSetVal("layoutXL.APAssist" "horizontalOffset" float 3)
```

Related Topics

Row Template Manager Form

Creating, Editing, and Deleting a Row Template

Virtuoso Placer Environment Variables

importCellName

```
layoutXL.APAssist importCellName string "cell_name"
```

Description

Specifies the cell containing the row template to be imported.

The default is `""`.

GUI Equivalent

Command	*Place – Row Template Manager – Import*
Field	*Cell*

Examples

```
envGetVal("layoutXL.APAssist" "importCellName")
envSetVal("layoutXL.APAssist" "importCellName" 'string "myCell")
```

Related Topics

Row Template Manager Form

Creating, Editing, and Deleting a Row Template

Virtuoso Placer Environment Variables

importLibName

```
layoutXL.APAssist importLibName string "lib_name"
```

Description

Specifies the library containing the row template to be imported.

The default is `""`.

GUI Equivalent

Command	*Place – Row Template Manager – Import*
Field	*Library*

Examples

```
envGetVal("layoutXL.APAssist" "importLibName")
envSetVal("layoutXL.APAssist" "importLibName" 'string "myLib")
```

Related Topics

Row Template Manager Form

Creating, Editing, and Deleting a Row Template

Virtuoso Placer Environment Variables

importViewName

```
layoutXL.APAssist importViewName string "view_name"
```

Description

Specifies the view containing the row template to be imported.

The default is `""`.

GUI Equivalent

Command	*Place – Row Template Manager – Import*
Field	*View*

Examples

```
envGetVal("layoutXL.APAssist" "importViewName")
envSetVal("layoutXL.APAssist" "importViewName" 'string "myView")
```

Related Topics

Row Template Manager Form

Creating, Editing, and Deleting a Row Template

Virtuoso Placer Environment Variables

relatedSnapPattern

```
layoutXL.APAssist relatedSnapPattern string "pattern_name"
```

Description

Specifies the snap pattern of the WSP region created in addition to the row region.

The default is `""`, in which case no WSP region is created.

GUI Equivalent

Command	*Place – Row Template Manager – Edit*
Field	*Related Snap Pattern*

Examples

```
envGetVal("layoutXL.APAssist" "relatedSnapPattern")
envSetVal("layoutXL.APAssist" "relatedSnapPattern" 'string "pattern_1")
```

Related Topics

Row Template Manager Form

Creating, Editing, and Deleting a Row Template

Virtuoso Placer Environment Variables

verticalGrid

```
layoutXL.APAssist verticalGrid string "vGrid"
```

Description

Specifies the reference grid along the y axis for snapping the devices during placement.

The default is `"none"`, in which case devices are not snapped to a vertical grid.

GUI Equivalent

Command	*Place – Row Template Manager – Edit*
Field	*Vertical Grid*

Examples

```
envGetVal("layoutXL.APAssist" "verticalGrid")
envSetVal("layoutXL.APAssist" "verticalGrid" 'string "grid_1")
```

Related Topics

Row Template Manager Form

Creating, Editing, and Deleting a Row Template

Virtuoso Placer Environment Variables

verticalOffset

```
layoutXL.APAssist verticalOffset float vOffset
```

Description

Specifies the reference row offset values along the y axis to be applied to rows after they are snapped to the reference grids.

The default is `0`.

GUI Equivalent

Command	*Place – Row Template Manager – Edit*
Field	*Placement Grid – Vertical Offset*

Examples

```
envGetVal("layoutXL.APAssist" "verticalOffset")
envSetVal("layoutXL.APAssist" "verticalOffset" float 3)
```

Related Topics

Row Template Manager Form

Creating, Editing, and Deleting a Row Template

Virtuoso Placer 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)
```

Related Topics

Placement Options Form

Setting the Assisted Placement Options in the Placement Options Form

Setting Automatic Placement Options in the Placement Options Form

Virtuoso Placer Environment Variables

apMove

```
APR.device.place 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: snapColor can be set to `t` only if apMove is set to `t`.

The default is `t`.

GUI Equivalent

Command	*Place – Placement Options*
Field	*Assisted – Assisted Placement – Enable*

Examples

```
envGetVal("APR.device.place" "apMove")
envSetVal("APR.device.place" "apMove" 'boolean nil)
```

Related Topics

Placement Options Form

Setting the Assisted Placement Options in the Placement Options Form

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 `magenta`.

GUI Equivalent

None

Examples

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

Related Topics

Placement Options Form

Setting the Assisted Placement Options in the Placement Options Form

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 nil)
```

Related Topics

Placement Options Form

Setting the Assisted Placement Options in the Placement Options Form

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*

Examples

```
envGetVal("layoutXL.APAssist" "fixAdjustedForResolveOverlaps")
envSetVal("layoutXL.APAssist" "fixAdjustedForResolveOverlaps" 'boolean nil)
```

Related Topics

Placement Options Form

Setting the Assisted Placement Options in the Placement Options Form

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 nil)
```

Related Topics

Placement Options Form

Setting the Assisted Placement Options in the Placement Options Form

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)
```

Related Topics

Placement Options Form

Setting the Assisted Placement Options in the Placement Options Form

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	*Place – Placement Options*
Field	*Assisted – Enable – Show Info Balloon*

Examples

```
envGetVal("layoutXL.APAssist" "infoBalloon")
envSetVal("layoutXL.APAssist" "infoBalloon" 'boolean nil)
```

Related Topics

Placement Options Form

Setting the Assisted Placement Options in the Placement Options Form

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 "green")
```

Related Topics

Placement Options Form

Setting the Assisted Placement Options in the Placement Options Form

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	*Place – Placement Options*
Field	*Assisted – Assisted Placement – Enable – Resolve Overlaps – Spread Any Direction, Spread X-Direction Only*

Examples

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

Related Topics

Placement Options Form

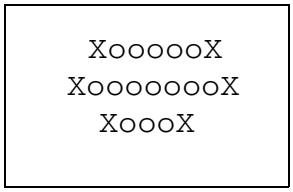
Setting the Assisted Placement Options in the Placement Options Form

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:



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 nil)
```

Related Topics

Placement Options Form

Setting the Assisted Placement Options in the Placement Options Form

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

Placement Options Form

Setting the Assisted Placement Options in the Placement Options Form

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*

Examples

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

Related Topics

Placement Options Form

Setting Common Options in the Placement Options Form

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	*Place – Placement Options*
Field	*Common – Minimum Pin Access Count*

Examples

```
envGetVal("layoutXL.placement" "minNumPinAccessPts")
envSetVal("layoutXL.placement" "minNumPinAccessPts" 'int 4)
```

Related Topics

Placement Options Form

Setting Common Options in the Placement Options Form

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

Placement Options Form

Setting Common Options in the Placement Options Form

paddingForCells

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

Description

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

(*cellName left_spacing right_spacing*)

Default is `""`, in which case no padding is added.

GUI Equivalent

Command	*Place – Placement Options*
Field	*Common – Cell Padding Values*

Examples

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

Related Topics

Placement Options Form

Setting Common Options in the Placement Options Form

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	*Place – Placement Options*
Field	*Common – Pin Density Aware Placement*

Examples

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

Related Topics

Placement Options Form

Setting Common Options in the Placement Options Form

regenModgenPostProcess

```
APR.device.place 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	*Place – Placement Options*
Field	*Common – Update MODGENS to Rows Post-Placement*

Examples

```
envGetVal("APR.device.place" "regenModgenPostProcess")
envSetVal("APR.device.place" "regenModgenPostProcess" 'boolean nil)
```

Related Topics

Placement Options Form

Setting Common Options in the Placement Options Form

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 "green")
```

Related Topics

Placement Options Form

Setting Common Options in the Placement Options Form

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)
```

Related Topics

Placement Options Form

Setting Common Options in the Placement Options Form

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*

Examples

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

Related Topics

Placement Options Form

Setting the Assisted Placement Options in the Placement Options Form

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*

Examples

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

Related Topics

Placement Options Form

Setting the Assisted Placement Options in the Placement Options Form

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	*Place – Swap Instances, Swap Rows*
Field	None

Examples

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

Related Topics

Automatic Placement Form

Running the Automatic Placer

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*

Examples

```
envGetVal("layoutXL.APAssist" "useDeviceOrder")
envSetVal("layoutXL.APAssist" "useDeviceOrder" 'boolean nil)
```

Related Topics

Placement Options Form

Setting the Assisted Placement Options in the Placement Options Form

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 nil)
```

Related Topics

Placement Options Form

Setting the Assisted Placement Options in the Placement Options Form

WSPAware

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

Description

Enables snapping of devices to WSP tracks.

The default is `t`.

GUI Equivalent

Command	*Place – Placement Options*
Field	*Common – Width Spacing Pattern (WSP) Snapping*

Examples

```
envGetVal("layoutXL.APAssist" "WSPAware")
envSetVal("layoutXL.APAssist" "WSPAware" 'boolean nil)
```

Related Topics

Placement Options Form

Setting Common Options in the Placement Options Form

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)
```

Related Topics

Placement Options Form

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 nil)
```

Related Topics

Placement Options Form

constraints

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

Description

(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 nil)
```

Related Topics

Batch Checker Form

Verifying Placement

rowOrGrid

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

Description

(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 Form

Verifying Placement

rowTempInfoAwareFillerInsertion

```
layoutXL.placement rowTempInfoAwareFillerInsertion 'boolean { t | nil }
```

Description

(Layout EXL Only) Uses the row template information for filler cell insertion. Comptypes and master filters are used to define filler cells in the row template file.

Default is `nil`.

GUI Equivalent

None

Examples

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

Related Topics

Creating, Editing, and Deleting a Row Template

Row Template Manager Form

Virtuoso Placer Environment Variables

spacing

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

Description

(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)
```

Related Topics

Batch Checker Form

Verifying Placement

tapCell

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

Description

(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)
```

Related Topics

Batch Checker Form

Verifying Placement

Placement Options Form

padding

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

Description

(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)
```

Related Topics

Batch Checker Form

Verifying Placement

minPinAccess

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

Description

(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)
```

Related Topics

Placement Options Form

Verifying Placement

Batch Checker Form

colorAware

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

Description

(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)
```

Related Topics

Placement Options Form

Verifying Placement

Batch Checker Form

reportViolations

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

Description

(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)
```

Related Topics

Placement Options Form

Verifying Placement

Batch Checker Form

createMarkers

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

Description

(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)
```

Related Topics

Placement Options Form

Verifying Placement

Batch Checker Form

selOnly

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

Description

(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)
```

Related Topics

Placement Options Form

Verifying Placement

Batch Checker Form

B

Virtuoso Placer Properties

leSnapGridHorizontal

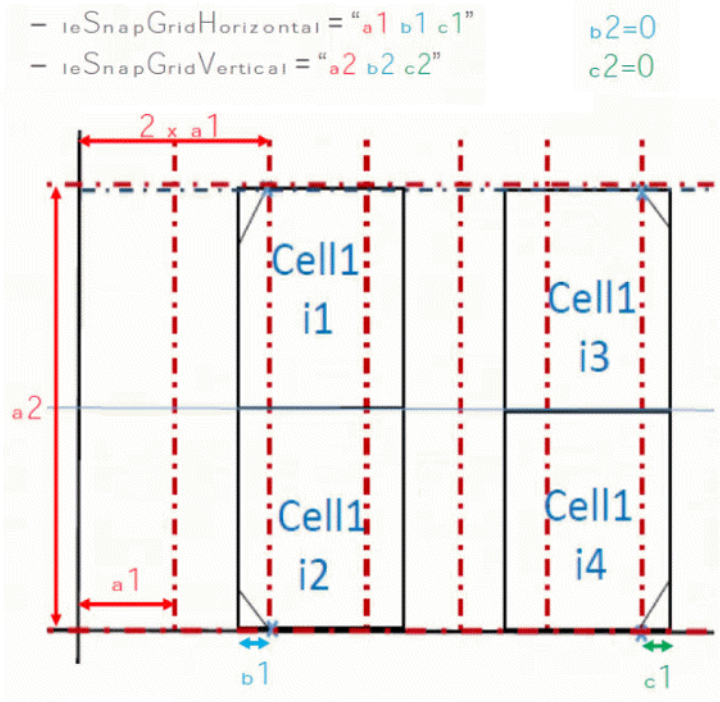
```
leSnapGridHorizontal string f_pitch f_offset f_offsetFlippedOrientation
```

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")
```

Related Topics

Setting the Assisted Placement Options in the Placement Options Form

Setting Automatic Placement Options in the Placement Options Form

Setting Common Options in the Placement Options Form

Running the Automatic Placer

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")
```

Related Topics

Setting the Assisted Placement Options in the Placement Options Form

Setting Automatic Placement Options in the Placement Options Form

Setting Common Options in the Placement Options Form

Running the Automatic Placer

leSnapPatternSnapping

```
leSnapPatternSnapping string { off | detail | abstract | f_offset | f_vOffset
     f_hOffset }
```

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.

You can also set the `leSnapPatternSnapping` property as a CDF parameter. The CDF parameter overrides the property on the instance and master.

Important

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 both directions along which snapping is done.
- *`f_vOffset f_hOffset`***:** 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).

Examples

```
leSnapPatternSnapping "detail"
leSnapPatternSnapping "abstract"
leSnapPatternSnapping "0.01 0.03"
```

Related Topics

Setting the Assisted Placement Options in the Placement Options Form

Setting Automatic Placement Options in the Placement Options Form

Setting Common Options in the Placement Options Form

Running the Automatic Placer

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"
```

Related Topics

Setting the Assisted Placement Options in the Placement Options Form

Setting Automatic Placement Options in the Placement Options Form

Setting Common Options in the Placement Options Form

Running the Automatic Placer

leSnapPatternSnappingHorizontalGridDetail

`leSnapPatternSnappingHorizontalGridDetail string` *`grid_name track_number`*

Defines the tracks of a global horizontal grid to which devices can be snapped in detail mode snapping.

- *`grid_name`*: Specifies the name of the global horizontal grid.
- *`track_number`*: Specifies a space-separated list of track numbers within a period to which devices can be snapped.

You can define this property for a master cellview or an instance. Detail mode snapping is performed only if the `snapGridPeriod` period constraint is defined for the specified global grid and the period value is greater than 1.

Example

The following example defines that devices can be snapped to the second and third tracks within each period of the global horizontal grid, `GPG`.

```
leSnapPatternSnappingHorizontalGridDetail "GPG 2 3"
```

Related Topics

snapGridPeriod

leSnapPatternSnappingVerticalGridDetail

Detail Mode Snapping to Specific Tracks of a Global Grid

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"
```

Related Topics

Setting the Assisted Placement Options in the Placement Options Form

Setting Automatic Placement Options in the Placement Options Form

Setting Common Options in the Placement Options Form

Running the Automatic Placer

leSnapPatternSnappingVerticalGridDetail

`leSnapPatternSnappingVerticalGridDetail string` *`grid_name track_number`*

Defines the tracks of a global vertical grid to which devices can be snapped in detail mode snapping.

- *`grid_name`*: Specifies the name of the global vertical grid.
- *`track_number`*: Specifies a space-separated list of track numbers within a period to which devices can be snapped.

You can define this property for a master cellview or an instance. Detail mode snapping is performed only if the `snapGridPeriod` period constraint is defined for the specified global grid and the period value is greater than 1.

Example

The following example defines that devices can be snapped to the fourth track of each period of the global vertical grid, `GFG`.

```
leSnapPatternSnappingVerticalGridDetail "GFG 4"
```

Related Topics

snapGridPeriod

leSnapPatternSnappingHorizontalGridDetail

Detail Mode Snapping to Specific Tracks of a Global Grid