Virtuoso Custom Digital Placer User Guide

Product Version IC23.1
March 2023

Contents

1 Introduction

The Virtuoso® Custom Digital Placer automatically places transistors, devices, and cells in block and cell designs.

The Custom Digital Placer is a generic placer that can be used for placement in different types of designs such as designs that contain only standard cells, designs that contain only devices, or designs that contain both standard cells and devices. You can also use the placer for generating connectivity and constraint-driven placement.

However, before you can use the placer to automatically place devices, you must generate the layout connectivity and circuit components by using Layout XL editor. The placer uses the schematic design as the connectivity source for placing the components.

This chapter covers the following topics:

Key Features

- Connectivity and constraint-driven placement to achieve overall shortest wire length.
- Placement of designs that contain:

- Only standard cells
- Only devices
- Both, standard cells and devices

- Support for the following placement types:
  - Row-based placement
  - Area-based placement
- Support for incremental and selective placement.
- Support for the following placement constraints
  - Alignment (floating axis only)
  - Cluster
  - Distance
  - Fixed
  - Symmetry (fixed and floating axis)
- Support for the following routing constraint
  - Net priority
- Automatic row generation for both standard cell and device-level MOS and FinFET designs

  Note: Automatic row generation for FinFET designs is supported only in Advanced Node for Layout Standard.
- Automated pin placement user interface that lets you:
  - Assign pins to a particular edge of the placement boundary
  - Specify the ordered and unordered pins
  - Specify the fixed pin positions
  - Place groups of pins at any given spacing
  - Place multiple pins on the same net

The following sections contain more information about using the Custom Digital placer.

- Setting Up Layout XL for Placement on page 24

- Setting Placement Constraints on page 26
- Defining Component Types and Placement Parameters on page 33
- Troubleshooting Placement on page 37

For information about the placer forms, see Custom Digital Placer Forms.

The Placement Commands

In Layout XL, the placement commands supported by the Virtuoso Custom Digital Placer are available under the *Place* menu. These commands are:

- *Custom Digital – Placement Planning:* Use this command to open the Placement Planning form for creating, modifying, and deleting rows for placing the design components.

  Note: Alternatively, you can open the Placement Planning form by using the Placement Planning icon, , on the Custom Digital Placer toolbar.

- *Custom Digital – Placer:* Use this command to run the automatic placer for performing placement.

  Note: Alternatively, you can open the automatic placer by using the Digital Placer icon, , on the Custom Digital Placer toolbar.

In addition to the placement commands listed above, the *Place* menu supports the following commands that can be used for placement:

- *Pin Placement*: Use this command to specify the pin location constraints and to preplace pins, independent of the main placement run.
- *Update Placement Status*: Use this command to 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 environment.

Other Commands Used with the Placer

- *Edit – Component Types* opens the Configure Physical Hierarchy window in the Component Types mode. Use this window to update your library with information that guides the placement of components in rows, and to define parameters for MOS device chaining and folding.

  Note: (Virtuoso Advanced Node for Layout Standard) The Configure Physical Hierarchy window can also be used to define parameters for FinFET chaining and folding.

For more information, see Component Types in the *Virtuoso Layout Suite XL User Guide*.

- *Create – P&R Objects – Row* opens the Create Row form. Use this form to create rows for placement.

  For more information about the method for creating rows manually, see Creating Rows in the Virtuoso Layout Suite documentation.

- *Create – P&R Objects – Custom Placement Area* opens the *Create Custom Placement Area* form. Use this form to create a custom area for placement.

  For more information about the method for creating a custom placement area, see Creating a Custom Placement Area in the Virtuoso Layout Suite documentation.

- *Create – P&R Objects – Clusters* opens the *Create and Edit Clusters* form. Use this form to specify the components that must be grouped together during placement.

- *Create – P&R Objects – Cluster Boundary* opens the *Create Cluster Boundary* form. Use this form to define a boundary to enclose one or more clusters. You can then specify the cluster boundary you want to place during placement planning.

  For more information on creating clusters and cluster boundaries, see Creating Clusters.

Important

To modify the existing component type assignments and to update the default component alignment and orientation before the components are placed, use the Dressing Template Editor form. You can open this form by clicking the **Dressing Template Editor** button on the *Create Row, Create Custom Placement Area, or the* Placement Planning form.

Placement Types

The Custom Digital Placer supports the following placement types:

- Row-based placement
- Area-based placement

Row-based placement

Row-based placement is a structured placement type that lets you place specific component types within rows. This placement type is most suited for placing standard cells and devices that have at least one “standard” dimension. For this reason, row-based placement is considered most suitable for designs that contain only standard cells.

In addition to providing a structured placement, row-based placement lets you constrain the components within a row to specific orientations and alignments. Given the uniform and structured placement it helps to obtain, row-based placement is considered a more economical and optimal placement type.

You can choose from the following modes for defining rows:

- User-defined (manual) mode: Use this mode to manually define the geometry of a series of rows and the components within. Using this placement mode, you can specify not only the number of rows to be placed but also define their location on the canvas and the alignment and orientation of the components to be placed within these rows.
- Component-assisted mode: Use this mode to automatically determine the row geometries based on the specified area utilization goals and other design goals. Using this placement mode, you can manipulate various design factors to explore different placement outcomes.

Area-based placement

Area-based placement is less restrictive of the two placement types supported by the Custom Digital Placer.

This placement type places the components anywhere within the available placement boundary, unless the components are locked outside the boundary. Area-based placement does not require component types or constraints to be defined. However, if any individual constraints are placed on the components, they are honored.

If individual constraints on components or groups of components do not exist, the placer only targets to reduce the overall wire length and achieve a balanced placement. Therefore, in the absence of constraints, the components tend to spread out using all the area allowed for placement.

Placement Of Different Component Types

You can use the Custom Digital Placer to place designs that have only standard cells, only devices, or a combination of both, standard cells and devices.

Standard (or custom) cells are placed in horizontal or vertical rows. Depending on their number, the placer can place all the cells within a single row or across multiple rows.

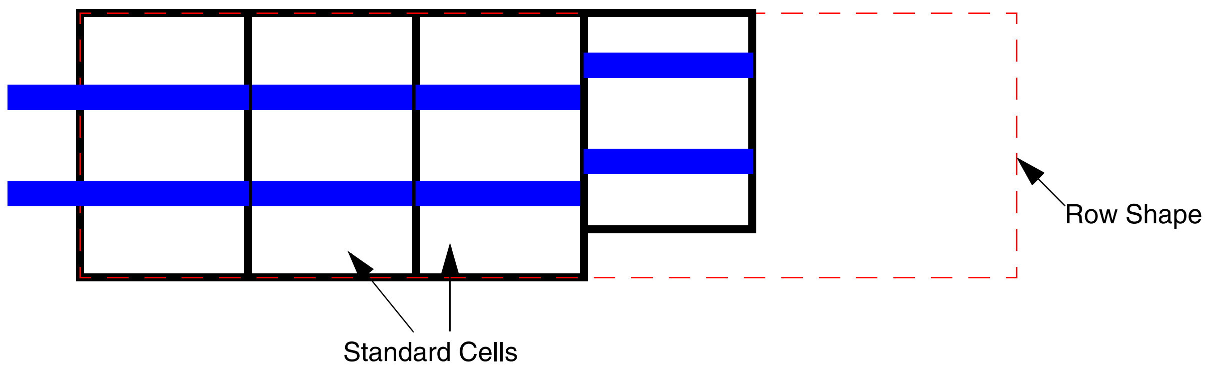


Figure 1-1 Standard cells placed in a row

For more information about placing components within rows, see Row-based placement.

Devices follow a row-based MOS, transistor-level placement.

Since MOS devices support diffusion-sharing to enable chaining, you can automatically or interactively chain devices at the time of layout generation to optimize placement. Such placement maximizes the diffusion sharing to minimize the diffusion gaps in the generated layout.

For designs with a mix of component types, such as *PMOS*, *NMOS*, *STDCELL*, *STDSUBCONT*, and *FILLER*, depending on the cell type, both row-based and MOS transistor-level placement is followed.

Placing NFIN and PFIN Components (Virtuoso Advanced Node for Layout Standard)

The Custom Digital Placer can be used for placing *NFIN* and *PFIN* component types.

Global Snap Pattern Grid Support

VCP has been enhanced to support snap patter pattern (SP) / width spacing snap pattern (WSP) grid definitions:

- The Snap Pattern (SP) grid defined using the `snapPatternDef` rule in the technology file.

  For detailed information about `snapPatternDef`, see Snap Pattern Definition Requirements in the “Technology File Requirements for FinFET-based Design” chapter of the *Virtuoso FinFET Support Overview* manual.

- The Width Spacing Snap Pattern (WSP) grid defined using the `widthSpacingSnapPatternDef` rule in the technology file.

  For definition of the `widthSpacingSnapPatternDef` rule and related technology file information, see Width Spacing Patterns.

To support snapping of devices and standard cells to the required snap pattern grid, VCP has undergone the following key enhancements:

- Row creation follows pattern-aware snapping.

  Row creation is based on reference grids and their respective offsets selected by the user at the time of Placement Planning. *Ref X Grid* and *Ref Y Grid* in the Placement Planning form specify the reference grids for snapping devices and rows. *Ref X Offset* and *Ref Y Offset* are the offsets applied to rows after they are snapped to the *Ref X Grid* and *Ref Y Grid*. If the *Ref Y Grid* is set to *NONE*, which is the default behavior, then the rows are snapped to the `siteDefs` width defined in the technology file.

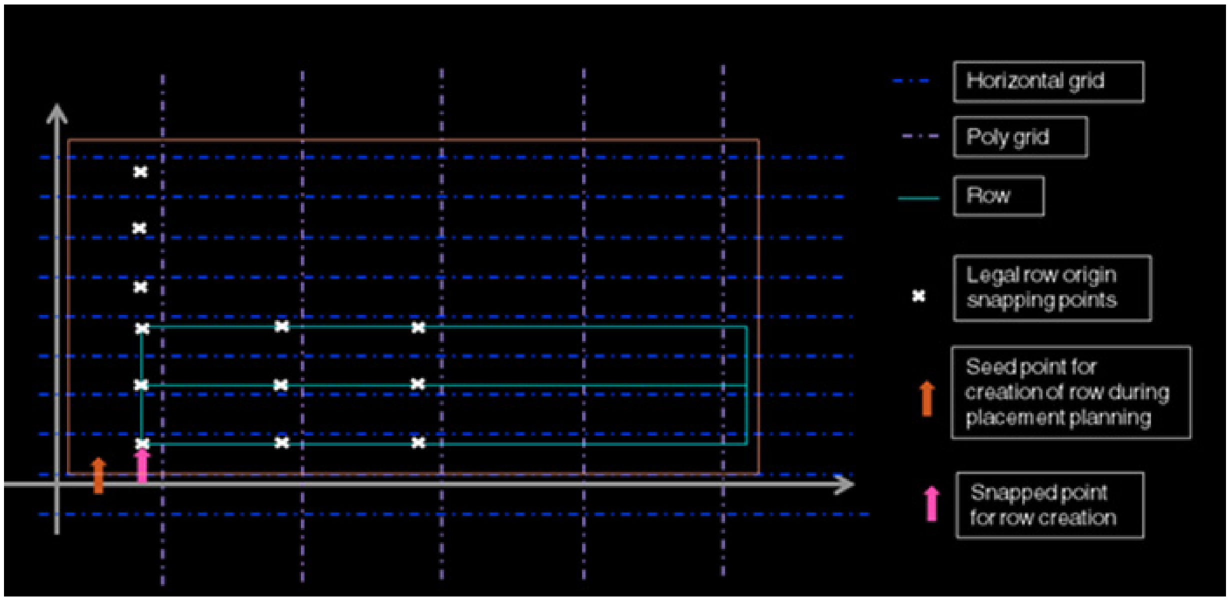
  For more information, see Planning a Placement.

- Auto Placer follows pattern-aware snapping.

  Auto Placer now honors the local/global snap pattern definitions in the technology file or in the design. During placement, devices are automatically snapped to the snap pattern grid.

  Note: Standard cells are placed on the `siteDefs` width defined in the technology file.

Here is an illustration representing how the seed point for row creation during placement planning is different from the point of row creation, which is snapped to the grid specified in the Placement Planning form.


Horizontal grid
Poly grid
Row
Legal row origin snapping points
Seed point for creation of row during placement planning
Snapped point for row creation

The Placement Flow

The placement flow for row-based designs typically involves the following steps:

1. Define component types using the Component Type mode of Configure Physical Hierarchy to assign the devices to rows and set the parameters for device chaining and folding.

   See Component Types in the *Virtuoso Layout Suite XL User Guide*.

2. (Optional) Generate layout representations for critical components by using the Layout XL *Connectivity - Generate - Selected From Source* command.

   See Generating Selected components from Source in the *Virtuoso Layout Suite XL User Guide.*

   Note: Make a preliminary placement of the critical components by using the object editing commands. Critical components should preferably be hand-placed and locked in their final positions to prevent them from being moved during automatic placement.

3. Generate a layout and a preliminary placement for the remaining design components by using the *Connectivity - Generate - All From Source* command or the *Connectivity - Update - Components and Nets* command.

   See Generating All Components from Source and Updating Components And Nets in the *Virtuoso Layout Suite XL User Guide.*

4. (Optional) Place and constrain pins by using the *Place – Pin Placement* command.

   See Pin Planning.

5. Create rows for placing the components, specify the component types to be placed in them, align the components to the respective rows, and specify the component orientation by using the Placement Planning command.

6. (Optional) Set the appropriate placement constraints.

   See Pin Placement Constraints.

7. (Optional) Set the Net Priority routing constraint for assigning higher priority to critical nets.

   See Net Priority in the *Virtuoso Custom Constraints User Guide*.

8. Run the automatic placer by using the *Place – Custom Digital – Placer* command to automatically position the unplaced design components on the layout canvas.

   See Running the Placer.

9. Examine the placement. If required, update the constraints you set and run the placer again.

Video

For a demonstration of the basic custom digital placer flow, see the Virtuoso Customer Digital Placer flow video.

Setting Up Layout XL for Placement

This section describes some Layout XL environment variables you might want to set and some design-related decisions you should make before you begin working with the placer.

- Setting the Netlisting Mode
- Defining the Placement Region
- Abutting Standard Cells
- Setting MOS Chaining and Folding Parameters
- Using Automatic Abutment During Placement

Setting the Netlisting Mode

To enable the placer to correctly handle permutable pins, ensure that the CDF parameters are evaluated correctly by setting the `CDS_Netlisting_Mode` shell environment variable to `Analog` or `Compatibility`.

Caution

Do not set the netlisting mode to Compatibility if you intend to chain or fold your devices.

To set the `CDS_Netlisting_Mode` environment variable for the current session,

➤ Type the following commands in the CIW.

```
setShellEnvVar("CDS_Netlisting_Mode=Analog")

cdsSetNetlistMode()
```

To check the mode in which the environment variable is currently set:

➤ Type the following command in the CIW.

```
cdsGetNetlistMode()
```

Note: If you do not have permutable pins in your design, and make no other use of the CDF data, you can get faster netlisting performance by setting `CDS_Netlisting_Mode` to `Digital`.

For more information about the `CDS_Netlisting_Mode`, see Customizing the Simulation Environment in the *Open Simulation System Reference*.

Important

Several placement environment variables have equivalent variables to control the Virtuoso custom routers. Placer variables always override their router equivalents during placement.

Defining the Placement Region

The prBoundary object representing the place and route boundary defines the available placement region.The prBoundary can be a single rectangle or a polygon.

Abutting Standard Cells

By default, all standard cells—cells defined with component type `STDCELL`—are designed to support abutment. This implies that all standard cells abut together without creating design violations. Therefore, when placing abutted standard cells, the placer only checks the design rules between the boundaries of the adjacent standard cells. Rules between the internal objects in a standard cell are not validated.

In other words, the placer considers the boundary of each standard cell as its abutment edge. This ensures that the placement is optimal.

Note: Although the placer does not check the design rules between the objects within a standard cell, it does run a full design rule check between standard cells—devices of component class `STDCELL`—and devices belonging to other component classes.

The placer determines the cell boundary for each standard cell by using the following precedence:

1. The list of layer-purpose pairs defined in the vcpCellBoundaryLPPs environment variable.
2. The (`prboundary drawing`) layer-purpose pair.
3. The (`prboundary boundary`) layer-purpose pair.

4. The (`instance drawing`) layer-purpose pair.

If none of the above layer-purpose pairs exists in the cell, a boundary is derived equal in size to the sum of the objects within the cell.

To ensure that the objects within adjacent standard cells can be shared or overlapped during placement without generating any abutment violations, the cell boundary must be defined by using the vcpCellBoundaryLPPs environment variable.

Setting MOS Chaining and Folding Parameters

To place the MOS devices optimally, you must set the following `.cdsenv` environment variables to chain and fold them before placement.

- `lxDeltaWidth`
- `lxWidthTolerance`
- `lxAllowPseudoParallelNets`
- `lxStackPartitionParameters`

Using Automatic Abutment During Placement

If switched on in Layout XL, automatic abutment is performed during layout generation for assisted MOS devices. To use automatic abutment:

- Select the *Abut transistors* option on the Connectivity form.
- Set the appropriate abutment properties on your design components including those on MOS device parameterized cells.

  See Setting Up Pcells for Abutment in the *Virtuoso Layout Suite XL User Guide*.

  For an example of a MOS device parameterized cell, see the `spcnmos` and `spcpmos` devices in the sample parameterized cells library. For more information, see the *Sample Parameterized Cells Installation and Reference*.

Setting Placement Constraints

To constrain the placement of devices or pins in your design, you can directly set *geometric* or *pin placement* constraints using the Constraint Manager or using an associated SKILL Function.

- Geometric Constraints are used to control the placement of objects. You can create them using the Constraint Manager assistant or by using an associated SKILL function.
    - ❑ For information about the types of constraints and how they are supported by the custom digital placer, see Geometric Constraints.
    - ❑ For information about creating constraints through the Constraint Manager assistant, see the *Virtuoso Unified Custom Constraints User Guide*.
    - ❑ For information about creating geometric constraints using a SKILL function, see the *Virtuoso Layout Suite SKILL Reference*.
- Pin Placement Constraints are used to control the placement of pins. You can create them using the Pin Placement form. The form lets you place different types of constraints on multiple pins at one time and automatically moves the pins to their assigned locations.

    Note: Alternatively, you can create pin placement constraints using the Constraint Manager or by using an associated SKILL function.

    For more information, see Pin Placement Constraints.

Geometric Constraints

You can create the following constraints in the Constraint Manager assistant for placement of objects:

- Alignment Constraints
- Cluster Constraints
- Distance Constraints
- Fixed Constraints
- Symmetry Constraints

Alignment Constraints

The placer supports alignment references on any layer or on the instance bounding box. The first object selected is considered as the alignment reference object and the remaining objects selected are aligned to that reference.

The Custom Digital Placer supports the following Alignment Constraint options:

- Matched orientation
- Align

- Reference layer or Bounding box
- Ordering

Important

When using an axis as a reference for aligning devices, the axis must be floating. You cannot align objects to a fixed reference axis.

For more information about the Alignment constraint, see Alignment under **Default Constraint Types** in the *Virtuoso Unified Custom Constraints User Guide*.

Cluster Constraints

The placer honors non-hierarchical cluster constraints that are set to specify the logical association among the instances within a cellview. Each cluster is associated with one or more rectilinear cluster boundaries within which the instances associated with that cluster are placed.

Depending on the members it accepts, a cluster can be:

- inclusive - Instances other than the instances in this cluster can be placed within its associated cluster boundaries. However, the instances that belong to this cluster cannot be placed outside of its associated boundaries.
- suggested - Instances can be placed inside or outside the associated cluster boundaries.
- exclusive - Only those instances that belong to this cluster can be placed within its associated cluster boundaries.

Note:

- The cluster boundary associated with a cluster of type “inclusive” or “suggested” is considered to be a soft fence for the cluster.
- The cluster boundary associated with a cluster of type “exclusive” is considered to be a hard fence for the cluster.

To define a cluster constraint on a selected set of components, run the Cluster command from the Placement constraints available in the Constraint Manager. To define a boundary for the cluster, select the *Create – P&R Objects – Cluster Boundary* command. To set the type of a cluster, choose a value for the Type field in the right panel on the Create and Edit Clusters form. To open the form, select the *Create – P&R Objects – Clusters* command.

For more information about the Alignment constraint, see Cluster under Default Constraint Types in the *Virtuoso Unified Custom Constraints User Guide*.

Related Topic

Create Cluster Boundary

Distance Constraints

The placer honors the distance constraint between two or more devices. However, when setting constraints for multiple devices, you must define the acceptable distance parameters between each pair of devices for the placer to interpret them.

Note: When a Distance constraint has more than two members, the constraint applies between the first and any of the following objects. That is, each of the second and following objects must be within the specified range from the first object.

You can constrain the devices in one direction—X or Y—but not in both directions simultaneously. However, after the devices have been constrained, the placer aligns them in the orthogonal direction.

The Custom Digital Placer honors the following Distance Constraint attributes:

- Max X
- Min X
- Reference X
- Reference Y
- Support for more than two objects

For more information about the Distance constraint, see Distance under **Default Constraint Types** in the *Virtuoso Unified Custom Constraints User Guide*.

Fixed Constraints

The placer honors the fixed constraint that is set to limit the position of one or more components with an allowable set of orientations to a particular x and y location.

The Custom Digital Placer supports the following Fixed Constraint options:

- X
- Y
- Orientation

- Reference
- Edit – Other – Lock Selected

For more information about the Fixed constraint, see Fixed under **Default Constraint Types** in the *Virtuoso Unified Custom Constraints User Guide*.

Symmetry Constraints

The placer interprets the order between the device pairs involved in a symmetry constraint. For example, if Q1 and Q2 are constrained along their horizontal line, then the constraint implies that Q1 lies below the horizontal line and Q2 lies above it. If the devices are constrained along a vertical line, based on the symmetry constraint, Q1 should lie to the left of the vertical line and Q2 to the right.

The placer supports the following types of symmetries:

- line symmetry
- self-symmetry
- pairwise symmetry

Note: The symmetry axis can be floating as well as fixed.

The Virtuoso Custom Digital Placer supports the following Symmetry Constraint options:

- Floating line
- Fixed line
- Horizontal/vertical direction
- Self-symmetry

For more information about the Symmetry constraint, see Symmetry under **Default Constraint Types** in the *Virtuoso Unified Custom Constraints User Guide*.

VCP honors the database cell symmetry set as an OA cell property on the cell master to derive allowed orientations as per the following table:

Database Cell Symmetry	Allowed Orientation of Devices during Placement
dbcXSymmetry "X"	R0, MX
dbcYSymmetry "Y"	R0, MY

dbcXYSymmetry "XY"	R0, R180, MY, MX

Pin Placement Constraints

There are three ways in which you can set placement constraints on pins. These are:

- Using the Constraint Manager Assistant
- Using the Pin Placement Form
- Using SKILL Functions

Setting Pin Placement Constraints using the Constraint Manager Assistant

You can set distance, alignment, grouping, symmetry, and fixed pin constraints by using the Constraint Manager assistant. For information about these constraints and how they are supported by Layout XL and the placer, see Geometric Constraints.

Depending on the requirement of your design, you can set multiple constraints using the Constraint Manager. For example, if you want to constrain pins to an edge of a bounding box with a specified pitch, you must create an alignment constraint and a distance constraint.

Note: Pins that you constrain through the Constraint Manager are moved to their assigned positions in the layout view only when the placer is run.

Important

Cadence recommends that you use any one of the methods for setting all your pin constraints to avoid creating conflicting or redundant constraints.

For more information about setting constraints through the Constraint Manager assistant, see the *Virtuoso Unified Custom Constraints User Guide*.

Setting Pin Placement Constraints using the Pin Placement Form

The advantage of using the Pin Planner form is that it lets you set and apply multiple constraints on multiple pins simultaneously. After you click *Apply*, the specified pins are automatically moved to the specified positions.

You can set the following constraints using the Pin Placement form.

- Distance

- Alignment
- Spacing
- Fixed

The *Pin Placement* form lets you assign pins to a fixed location or to a particular edge and order them along each edge according to your requirement. You can also specify fixed spacing for a group of pins that you have ordered and aligned to a particular edge.

For example, you have a polygon with six edges and you want to place three pins (pin1, pin2, and pin3) on edge number 5. When you specify this in the pin placer form, it creates an alignment constraint as follows.

Members	PR boundary (reference) pin1 pin2 pin3
Constraint parameters	side=edgeNumber

Members parameters	side=edgeNumber
	edge=5

For more information about setting constraints through the Pin Placement form, see the Pin Planner form.

Setting Pin Placement Constraints using SKILL Functions

You can set placement constraints on pins by using SKILL Functions.

Defining Component Types and Placement Parameters

The information about the “type” of each design component is required to specify the chaining and folding parameters for the components, and to define how the various component types are assigned to rows.

Currently, there is no need to define the component types for placing the components by area. In other words, if the component types are not specified, the components are placed by area.

You can define component types by using the *Edit – Component Types* command.

Depending on the requirement of your design, you can define the component types at the library level or at the individual cell level.

Irrespective of their type, the components are placed within the place and route boundary unless:

- The devices have been locked or fixed to a particular location outside the place and route boundary.
- The *Place Selected Only* option in the Auto Placer form is selected.

Note:

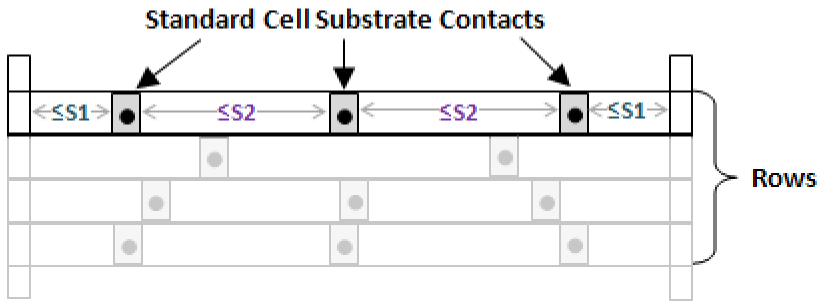
- If the placer does not place a component in a row, use *Edit – Component Types* to verify that the component in question has been correctly assigned to an appropriate component type, and that the row definition allows for that particular component type to be placed.
- If some components in the design are not assigned a component type, the placer ignores these components when calculating the required rows and subsequently ignores them during placement.

Defining a Standard Cell Substrate Contact

Standard cell substrate contacts, also called tap cells, are physical-only filler cells that are inserted between devices to limit resistance and to establish connection between the VDD and GND rails, thereby preventing DRC errors and latchup effects.

Standard cell substrate contacts are neither contacts nor vias; they are standard cells assigned to component types with the component class *STDSUBCONT*.

Standard cell substrate contact insertion is done on a per-row basis. Therefore, the location of standard cell substrate contacts in each row is independent of the placement of standard cell substrate contacts in other rows.



In the above diagram, `S1` represents the offset of the first and last substrates from the respective row edges, and `S2` represents the distance between two consecutive substrates.

You can specify the maximum and minimum spacing values between two consecutive substrates by using one of the following methods:

- Using the *Use Substrate Contacts* field in the Placement Planning form
- Defining the Substrate Contact Spacing Properties

The values for `S1` range from `0` to half of the value of `S2`. For example, if you specify 1 and 4 as the minimum and maximum values for `S2`, the value of `S1` can range from 0 to 2. The Auto Placer respects these values during placement.

Caution

While running the Auto Placer, the positions of the standard cell substrate contacts might be altered to optimize the overall wire length. Such alterations might result in substrate spacing violations. Use the Annotation Browser to view and manage any violation markers that are generated for the design.

Use the Swap Tap Cells feature to detect tap cells with maximum spacing violations. You can also replace them with tap cells that have valid maximum spacing values.

For more information about this feature, see Swapping Tap Cells.

Defining the Substrate Contact Spacing Properties

To add standard cell substrate contacts, you need to define the following properties:

- `lxVcpSubContMaxSpacing`, which specifies the maximum spacing after which the placer must place another standard cell substrate contact in order to avoid a violation. Specify this property on the layout master for the substrate contact.
- `lxVcpSubContMinSpacing`, which specifies the minimum spacing allowed between two adjacent standard cell substrate contacts. Specify this property on the layout master for the substrate contact.

To define these properties:

1. Copy an existing filler cell, which is to be used as the basis for the standard cell substrate contact.
2. Add substrate and well contacts to the cell. You can use `oaVia`, via instance, or polygons to do this.
3. Add the `lxVcpSubContMaxSpacing` and `lxVcpSubContMinSpacing` properties to the cellview.

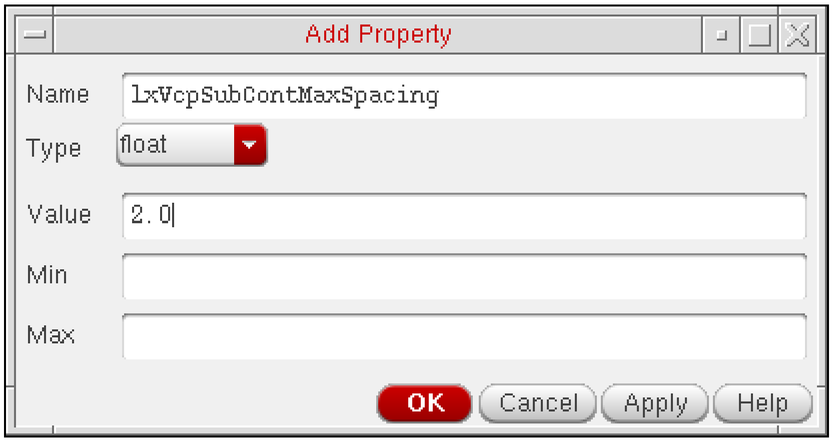
   For example, to set `lxVcpSubContMaxSpacing`, do the following.

   a. From the layout window, choose *Edit – Basic – Properties – Cellview*.

   The Edit Cellview Properties form appears.

   b. Click the *Add* button to display the Add Property form.

   c. Specify the property parameters, as indicated in the figure below, and click *OK* to set the property.



Name	lxVcpSubContMaxSpacing
Type	float
Value	2.0
Min	
Max	

   d. Repeat steps **a** through **c** to add the `lxVcpSubContMinSpacing` property.

  e. Click *OK* to set the properties and close the Edit Cellview Properties form.

4. Choose *Edit – Component Types* to display the Configure Physical Hierarchy window in the Component Types mode.
5. Assign a component type to the new cell that you created, which has its *Component Class* set to *STDSUBCONT*. (Create a new component type, if required.)
6. (Optional) Specify the view name to be used for placing the cells of class *STDSUBCONT*.

   Note: If the view name is not defined, the placer traverses through the *Physical stop view list* in the Global Bindings section of the **Configure Physical Hierarchy** form to determine the view type to be used.
7. Run the automatic placer after selecting the *Insert Standard Cell Substrate Contacts* option selected.

   The placer fills any gaps between the placed cells with the specified standard cell substrate contacts.

Note: You can use the Insert Std Cell Substrate Contacts options in the Auto Placer form to globally override the substrate spacing you had set using the Add Property form. Alternatively, use the subContMaxSpacing or subContMinSpacing environment variables, as required.

Placing Standard Cell Substrate Contacts in Alternate Rows

By default, substrate contacts (tap cells) get placed in consecutive rows. But, if required, you can force them to be placed in alternate rows. However, for this to be supported by the placer, the rows in which the substrate contacts are to be placed must be created in flipped and abutted manner. This means the rail pattern should be PGGP or GPPG and the row-to-row spacing must be zero.

To place substrate contacts in alternate rows:

➡ In the Placement Planning form, select *In Alternate Rows* option on the Component tab.

   Alternatively, set the subContInAlternateRow environment variable.

With substrate contact placement in alternate rows selected, the placer places the substrate contacts in alternate rows, such as the 1st, 3rd, 5th, and so on. But, if the total number of rows created is even, the substrate contacts will also be placed in the top row. For example, if the total number of rows generated is 6, the substrate contacts are placed in the 1st, 3rd, 5th, and the 6th rows.

MOS Transistor Chaining and Folding Parameters

After the devices are assigned a component type, the MOS transistors that need to be chained and folded must have certain parameter values set before you generate a layout.

Troubleshooting Placement

This section discusses common placement problems and some possible solutions.

Room Violations

Room violations are spacing violations that occur when the components placed within a partition are overlapped by another partition or when two partitions overlap.

By default, the VCP engine excludes components that do not belong to the partition to be placed. However, room violations might still be observed if the partition itself is overlapped by another partition.

The markers generated due to room violations in a design are displayed in the Misc tab of the Annotation Browser.

To fix room violations, manually move around the overlapping shapes.

2 Pin Planning

This section explains how you can use the pin planning functionality to set constraints on and plan the placement of the top-level or level-1 pins in your design.

Tip

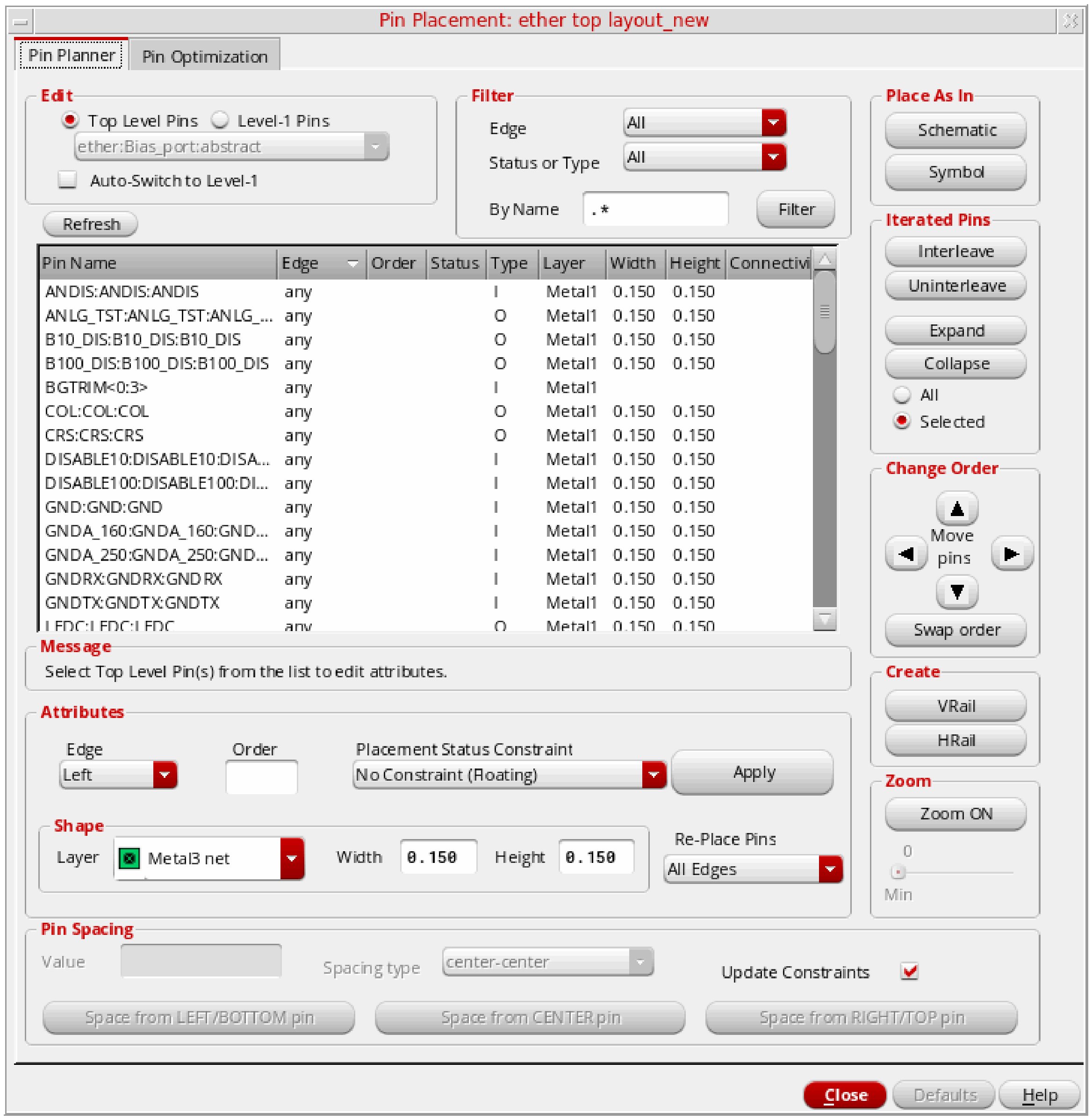
You can use the pin planner functionality only if there is a place and route boundary available in the layout view. To generate a place and route boundary, use the *Connectivity – Generate – All From Source* command.

Using the Pin Placement Form

To use the Pin Placement form:

1. Open the Pin Placement form.

The *Pin Planner* tab is displayed listing all the pins in the current scope in the pin table.



Pin Placement: ether top layout_new

Pin Planner | Pin Optimization

Edit
[✓ Top Level Pins] Level-1 Pins
ether:Bias_port:abstract
Auto-Switch to Level-1
Refresh

Filter
Edge: All
Status or Type: All
By Name: .*
Filter

Pin Name	Edge	Order	Status	Type	Layer	Width	Height	Connectivi
ANDIS:ANDIS:ANDIS	any			I	Metal1	0.150	0.150	
ANLG_TST:ANLG_TST:ANLG_...	any			O	Metal1	0.150	0.150	
B10_DIS:B10_DIS:B10_DIS	any			O	Metal1	0.150	0.150	
B100_DIS:B100_DIS:B100_DIS	any			O	Metal1	0.150	0.150	
BGTRIM<0:3>	any			I	Metal1			
COL:COL:COL	any			O	Metal1	0.150	0.150	
CRS:CRS:CRS	any			O	Metal1	0.150	0.150	
DISABLE10:DISABLE10:DISA...	any			I	Metal1	0.150	0.150	
DISABLE100:DISABLE100:DI...	any			I	Metal1	0.150	0.150	
GND:GND:GND	any			I	Metal1	0.150	0.150	
GNDA_160:GNDA_160:GND...	any			I	Metal1	0.150	0.150	
GNDA_250:GNDA_250:GND...	any			I	Metal1	0.150	0.150	
GNDRX:GNDRX:GNDRX	any			I	Metal1	0.150	0.150	
GNDTX:GNDTX:GNDTX	any			I	Metal1	0.150	0.150	
LFDC:LFDC:LFDC	any			O	Metal1	0.150	0.150	

Message
Select Top Level Pin(s) from the list to edit attributes.

Attributes
Edge: Left
Order:
Placement Status Constraint: No Constraint (Floating)
Apply

Shape
Layer: 🟩 Metal3 net
Width: 0.150
Height: 0.150
Re-Place Pins: All Edges

Place As In
Schematic
Symbol

Iterated Pins
Interleave
Uninterleave
Expand
Collapse
All [✓ Selected]

Change Order
Move pins
Swap order

Create
VRail
HRail

Zoom
Zoom ON
0
Min

Pin Spacing
Value:
Spacing type: center-center
Update Constraints ✓
Space from LEFT/BOTTOM pin | Space from CENTER pin | Space from RIGHT/TOP pin

Close | Defaults | Help

The entries are expressed as *termName:pinName:figName* because the pin figure is the physical entity that you are placing. If a pin has more than two pin figures attached to it, but the pin figures have the same name, they are represented by a single entry in the pin table.

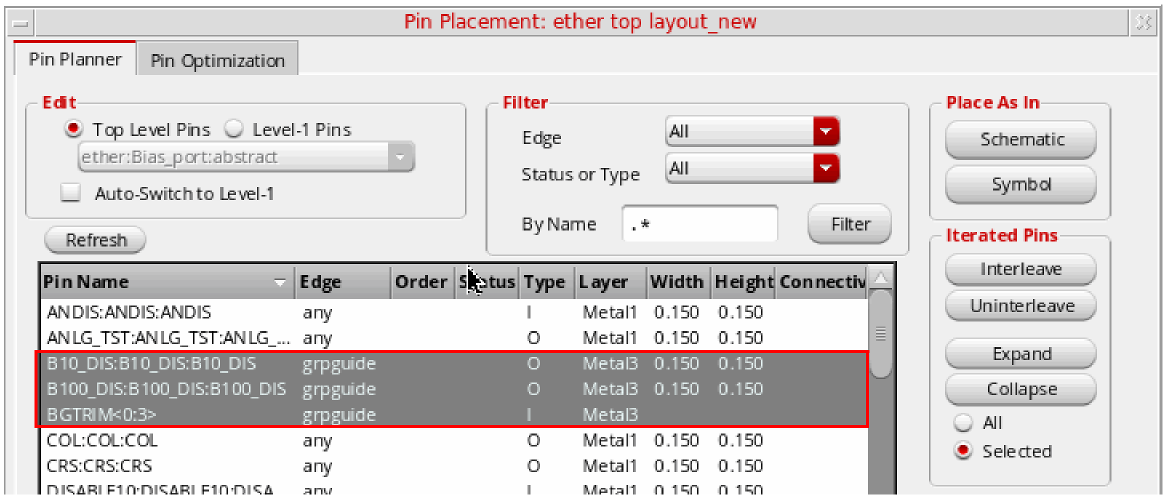
The constraints generated by pin planner also use the *termName:pinName:figName* convention, thus enabling the constraints system to distinguish between pin figures in constraints from pins. This format is used only when the pin has more than one figure. If a pin has only one figure then the old format *termName:pinName* is used when creating the constraint.

You can filter and otherwise manipulate the list using the controls provided. Click the *Pin Name* column heading to view an alphanumeric listing of the pins. For example:

- `APIN` is listed before `BPIN`
- `APIN2` is listed before `APIN19`
- Bus bit `APIN<4>` is listed before bus bit `APIN<18>`

When a pin is selected in the pin table, the pin planner recognizes the corresponding net PRO and displays the values in the *Layer*, *Width* and *Height* fields. You can edit these values manually, while ensuring that the net PRO constraint values are not violated. If the specified pin size is less than the `minWidth` defined in net PRO, then the pin size is reverted to its original value. If the specified pin *Width* or *Height* is equal to or greater than the net PRO value, then the new value is applied.

The pin planner supports the placement and distribution of member pins in pin groups in the specified guide region.



[✓ Top Level Pins] Level-1 Pins
ether:Bias_port:abstract
Auto-Switch to Level-1

Edge: All
Status or Type: All
By Name: .*
Filter

All [✓ Selected]

Pin Name	Edge	Order	Status	Type	Layer	Width	Height	Connectiv
ANDIS:ANDIS:ANDIS	any			I	Metal1	0.150	0.150	
ANLG_TST:ANLG_TST:ANLG_...	any			O	Metal1	0.150	0.150	
B10_DIS:B10_DIS:B10_DIS	grpguide			O	Metal3	0.150	0.150	
B100_DIS:B100_DIS:B100_DIS	grpguide			O	Metal3	0.150	0.150	
BGTRIM<0:3>	grpguide			I	Metal3			
COL:COL:COL	any			O	Metal1	0.150	0.150	
CRS:CRS:CRS	any			O	Metal1	0.150	0.150	
DISABLE10:DISABLE10:DISA	any			I	Metal1	0.150	0.150	

You can continue to use the pin planner to create and display side (alignment) constraints for pins. However, if you assign an alignment constraint to a pin that is already part of a pin group guide constraint, then an error message is displayed in the CIW.

The *Connectivity* column annotates whether pins are connected to buried pins or multiple hard block pins.

2. Use this form to do one or more of the following.

   - Place pins in the same relative positions as in the schematic view. Click *Schematic* in the *Place As In* section.

- ❑ Place pins in the same relative positions as in the symbol view. Click *Symbol* in the *Place As In* section.
- ❑ Place pins in fixed positions on the boundary. See:
  - ❍ Placing a Pin on a Boundary Edge
  - ❍ Placing Ordered Pins on a Boundary Edge
- ❑ Place pins in fixed positions that are not on the boundary. See:
  - ❍ Placing a Pin in a Fixed Position not on the Boundary.
- ❑ Place pins directly on lower-level instance terminals on the same layer. See:
  - ❍ Placing a Pin on a Lower-level Instance Terminal
- ❑ Place soft block or top-level pins that are connected to multiple hard block pins. See:
  - ❍ Placing Pins that Are Connected to Hard Block Pins
- ❑ Place level-1 pins below top-level buried pins. See:
  - ❍ Placing Buried Pins
- ❑ Place pin groups with any given pin pitch spacing. See:
  - ❍ Spacing Pins
- ❑ Create vertical and horizontal rails. See:
  - ❍ Converting a Pin into a Rail
- ❑ Report pins and their assigned locations.
- ❑ Expand and collapse iterated bus pins (`Q<7:0>` = `Q<7>`, `Q<6>`, `Q<5>`, and so on.)
- ❑ Handle multiple pins on the same net, `gnd.1 gnd.2 gnd.3`.

3. When you have finished setting pin constraints, use the *Re-Place Pins* pull-down to specify which pins can be moved by the placer. For more information about these options, see Pin Optimization.

4. Click *Apply* to set the new attributes for the specified pins.

   Note: Pins with placement status as “none” or “unplaced” will not be reinitialized. Instead, a warning message is issued indicating the presence of such pins.

5. Click *Defaults* to return all pins to their default positions.

Important

Using both the Pin Placement form and the Constraint Manager to constrain pins can result in conflicting and redundant constraints. Create pin constraints using *either* the Pin Placement form or the Constraint Manager and its associated SKILL functions.

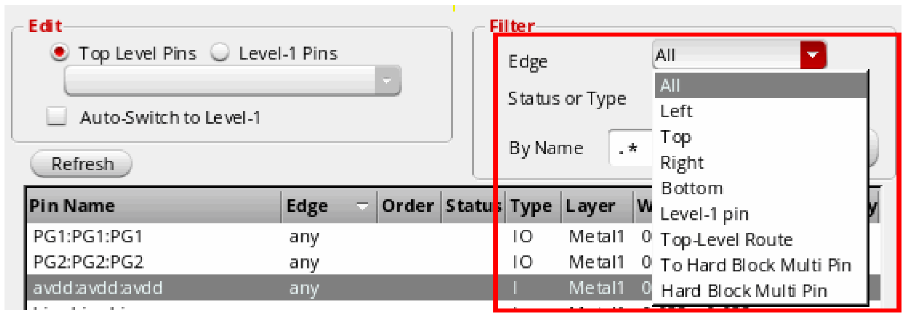
Placing a Pin on a Boundary Edge

To place a pin in a fixed position on the boundary,

1. From the layout window menu bar, choose *Place – Pin Placement*.

   The Pin Placement form is displayed.

2. Choose *Unplaced* from the *Status or Type* drop-down list.

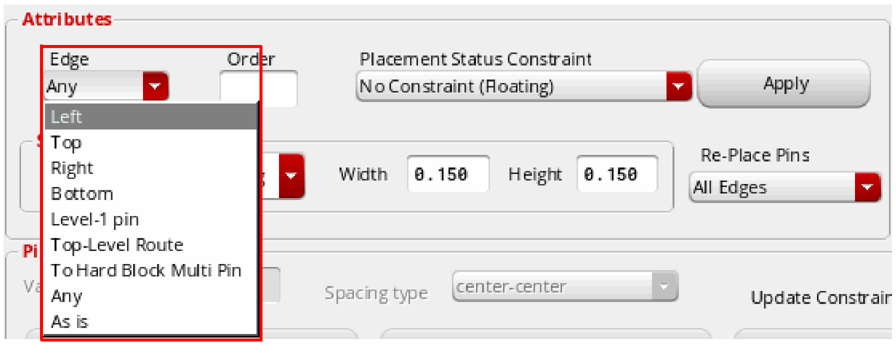


Pin Name	Edge	Order	Status	Type	Layer	W
PG1:PG1:PG1	any			IO	Metal1	0
PG2:PG2:PG2	any			IO	Metal1	0
avdd:avdd:avdd	any			I	Metal1	0

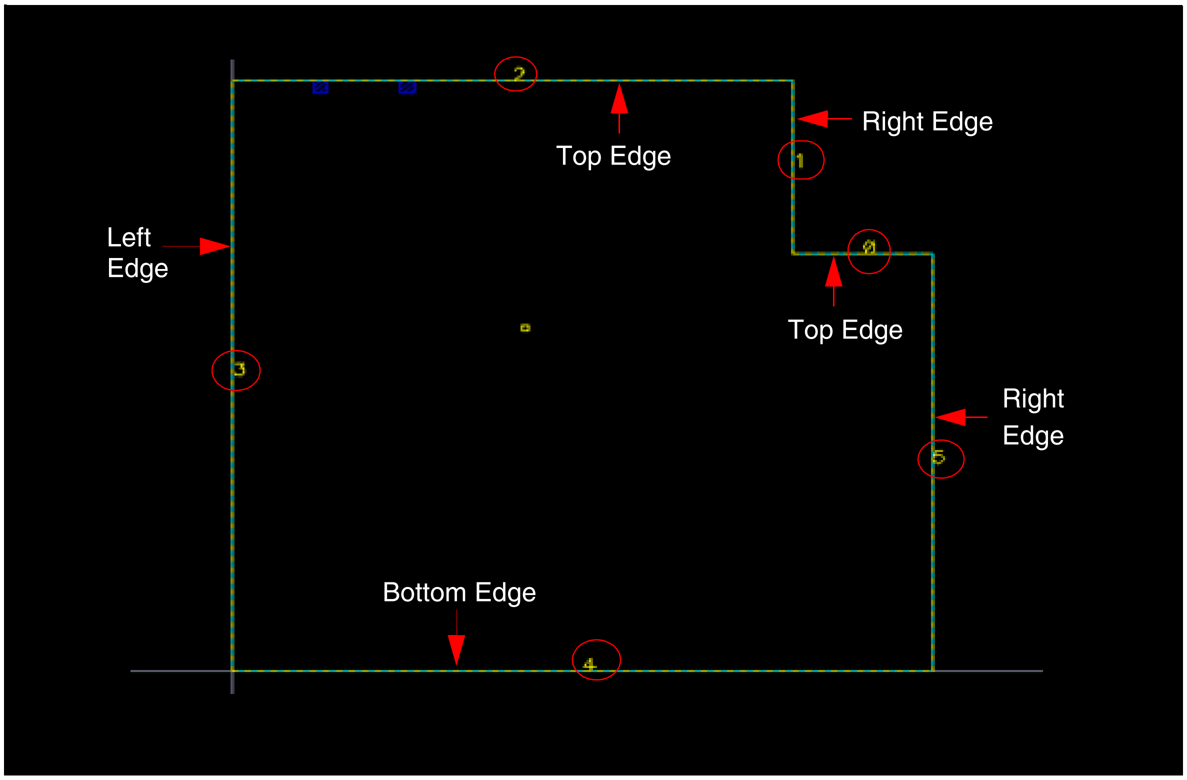
3. Select the pin you want to place from the pin table.

   You can expand iterated bus pins (for example, `A<7:0>`) using the *Expand* button in the *Iterated Pins* group box. This expands the bus into individual pins, which you can then place individually with different constraints.

4. From the *Edge* drop-down list, choose the edge on which the selected pins are to be placed. For rectangular boundaries, the edge sides are listed as *Left*, *Top, Right*, *Bottom, Any, Level-1 pin,* and *As is*.



For rectilinear boundaries, both the edge numbers and the edge sides are listed.



Edge numbers are assigned in the increasing order, starting with zero, which is assigned to the edge at the right-top vertex, followed by the adjoining edge in the anti-clockwise direction.

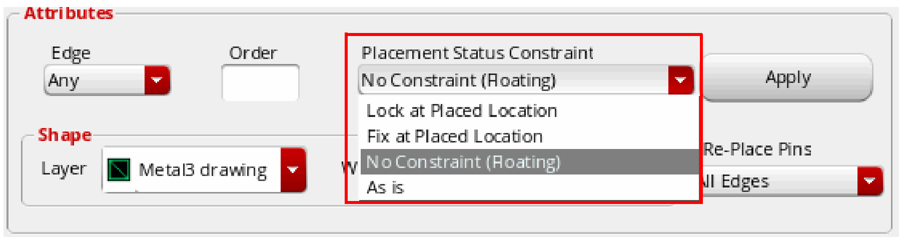
An edge side for rectilinear boundaries might correspond to one or more edges. In the above example, the "`Top`" edge side can correspond to edge numbers 0 or 2. Therefore, the pin planner has a choice of edges.

Note: The alignment Edge/Side constraint is disregarded and a warning message is displayed in the following situations:

- ❑ Situation 1 - Edge number is specified for a cell with a rectangular boundary.
- ❑ Situation 2 - The specified edge does not have enough available slots.
- ❑ Situation 3 - The PR boundary is not a member of the alignment constraint.

For situations 1 and 2, although the alignment Edge/Side constraint is disregarded, the pitch value, if set, is honored. For situation 3, however, the pitch value is not honored.

5. Choose *Fix at Placed Location* from the *Placement Status Constraint* drop-down list.



Fix at Placed Location places the pin at a location on the boundary and then fixes the pin at that location, meaning that it cannot be moved by the automatic placement functions; however, you can move it manually, for example, using the *Edit – Move* command.

Tip

Use *Lock at Placed Location* to lock the selected pins after they are placed. A locked pin cannot be moved either by the automatic placement functions or manually by the user.

6. Click *Apply* to align the pin to the specified boundary edge in the layout.

   Note: If required, use the *Re-Place Pins* pull-down to limit the pins that can be moved by the placer when you click *Apply*.

   The *Status* field is updated in the pin table and the label in the *Placement Status Constraint* field changes to show the coordinates of the center of the pin.

7. Repeat **step 3** through **step 6** for each pin you want to place on the boundary.

8. Click *Close* to close the Pin Placement form.

Placing Ordered Pins on a Boundary Edge

To place a group of pins in a specific order along a boundary edge:

1. From the layout window menu bar, choose *Place – Pin Placement*.

   The Pin Placement form is displayed.

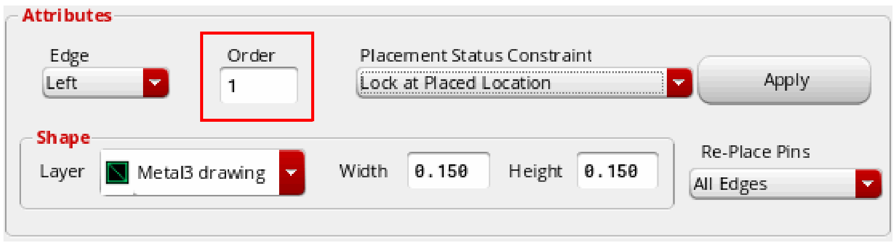
2. Select one or more pins from the pin table.

   Change the order of the pins by selecting a pin in the list and clicking on of the arrows that surround the *Move pins* label. The up and left arrows move items up in the list; the down and right arrows move items down.

   Use the *Swap order* option to swap the position of two selected pins.

3. Choose the edge you want from the *Edge* drop-down list.

4. Type a number in the *Order* field.



Attributes

Edge: Left

Order: 1

Placement Status Constraint: Lock at Placed Location

Shape

Layer: Metal3 drawing

Width: 0.150

Height: 0.150

Re-Place Pins: All Edges

   You can type any number into the field; the selected pins are assigned an integer based on the order in which they are listed in the pin table.

5. Click *Apply* to apply each pin assignment.

   Note: If required, use the *Re-Place Pins* pull-down to limit the pins that can be moved by the placer when you click *Apply*.

   The pins are aligned to the specified boundary edge in the layout in the order you specified.

Placing a Pin in a Fixed Position not on the Boundary

To place a pin in a fixed location not aligned with the boundary,

1. From the layout window menu bar, choose *Edit – Move*.

2. In the layout window, select an unplaced pin and move it to the required location.
3. From the layout window menu bar, choose *Place – Pin Placement*.

   The Pin Placement form is displayed.
4. Select the pin in the pin table.
5. Choose *Fix at Placed Location* from the *Placement Status Constraint* drop-down list.
6. Click *Apply*. The pin is fixed at its current location and is not aligned with any edge.

Note: You can optionally assign the pin to an edge immediately prior to step 5. If you do this, the pin is fixed at the location but the pin width and design rules are projected to the specified edge. This projection leaves open a channel on the aligned edge for the fixed pin to facilitate better routing from outside the boundary.

Placing a Pin on a Lower-level Instance Terminal

To place a pin directly onto its corresponding instance terminal at a lower level of the design hierarchy:

1. From the layout window menu bar, choose *Place – Pin Placement*. The Pin Placement form is displayed.
2. Select the pin from the pin table.
3. Choose *Level-1 pin* from the *Edge* drop-down list.
4. Choose *No Constraint (Floating)* from the *Placement Status Constraint* drop-down list.
5. Click *Apply*. The pin you selected is placed over its corresponding instance terminal in the layout window.

The *Connectivity* column in the pin table annotates whether pins are connected to buried pins or multiple hard block pins. Pins are re-layered and re-sized automatically. Pin labels are updated if the matchLabelLayerWithPin environment variable is set to `t`.

Placing Pins that Are Connected to Hard Block Pins

There might be situations where one top-level or level-1 pin is connected to two or more hard block pins. To place such pins:

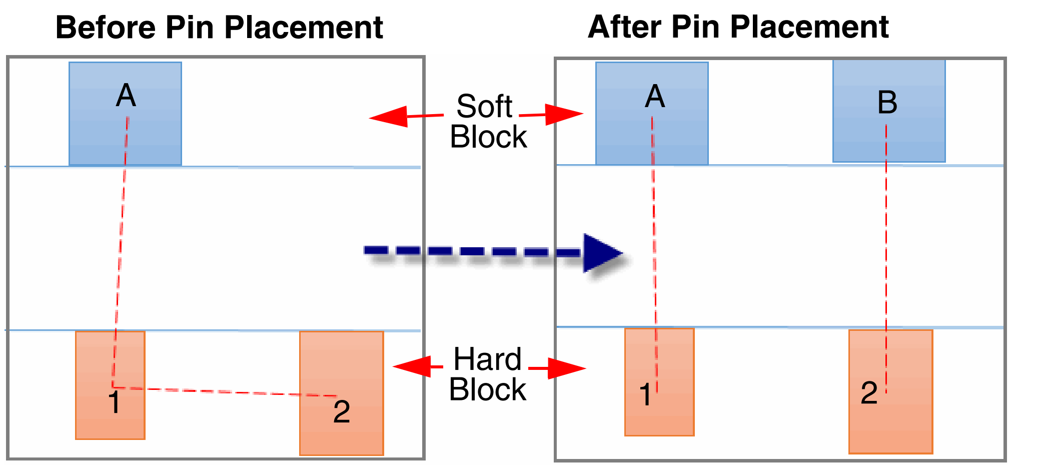
1. From the layout window menu bar, choose *Place – Pin Placement*.

The Pin Placement form is displayed.

2. Select the soft block or top-level pin from the pin table.
3. Choose *To Hard Block Multi Pin* from the *Edge* drop-down list.
4. Choose *No Constraint (Floating)* from the *Placement Status Constraint* drop-down list.
5. Click *Apply*.

The *Connectivity* column in the pin table annotates whether pins are connected to buried pins or multiple hard block pins.

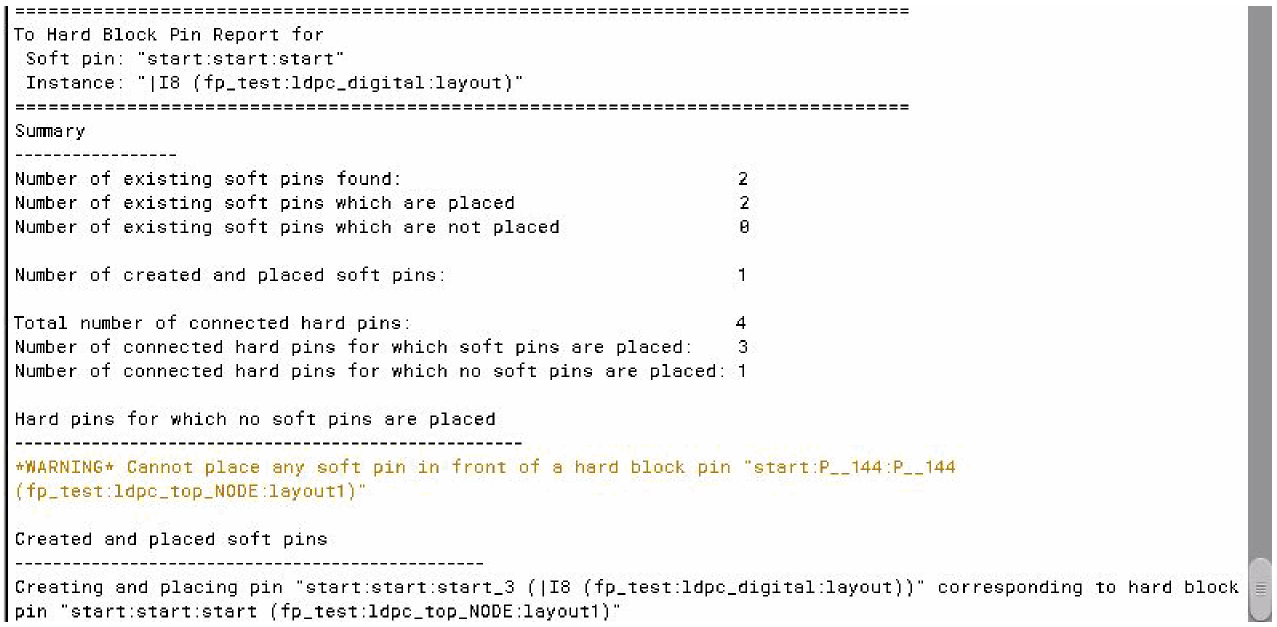
Additional pins are created in the soft block (or at the top-level) to match the number of pins in the hard block. Therefore, each soft block or top-level pin is connected to a corresponding hard block pin, as shown below:



One soft block pin (A) connected to two hard block pins (1 & 2)

An additional soft block pin is created. So there are two soft block pins (A & B) connected to two hard block pins (1 & 2)

A summary report is displayed in the CIW that provides detailed information about hard block pin placement:



```
================================================================================
To Hard Block Pin Report for
 Soft pin: "start:start:start"
 Instance: "|I8 (fp_test:ldpc_digital:layout)"
================================================================================
Summary
----------------
Number of existing soft pins found:                         2
Number of existing soft pins which are placed               2
Number of existing soft pins which are not placed           0

Number of created and placed soft pins:                     1

Total number of connected hard pins:                        4
Number of connected hard pins for which soft pins are placed:    3
Number of connected hard pins for which no soft pins are placed: 1

Hard pins for which no soft pins are placed
----------------------------------------------------
*WARNING* Cannot place any soft pin in front of a hard block pin "start:P__144:P__144
(fp_test:ldpc_top_NODE:layout1)"

Created and placed soft pins
------------------------------------------------
Creating and placing pin "start:start:start_3 (|I8 (fp_test:ldpc_digital:layout))" corresponding to hard block
pin "start:start:start (fp_test:ldpc_top_NODE:layout1)"
```

The summary report includes the following information:

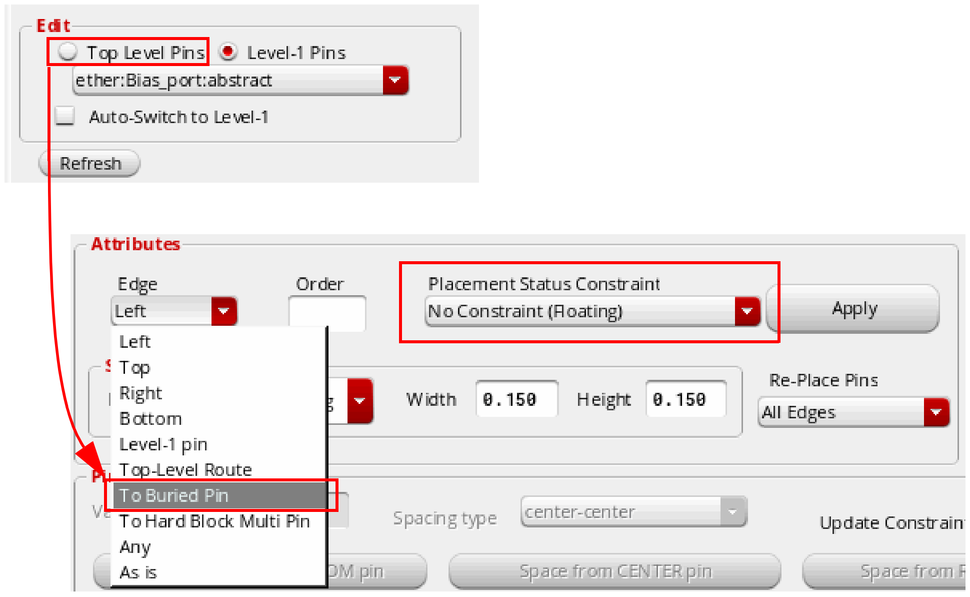
- Number of soft pins found.
- Number of soft block pins that are aligned to hard block pins.
- Number of soft block pins that could not be aligned to hard block pins.
- Number of new soft pins created.
- Number of hard pins found.
- Number of hard block pins to which soft block pins have been aligned.
- Number of hard block pins to which soft block pins could not be aligned.
- A list of messages generated while placing soft block pins that are connected to multiple hard block pin.

Placing Buried Pins

Buried pins are the top-level pins that are also bound by a level-1 PR boundary. To place a level-1 pin below a top-level buried pin:

1. From the layout window menu bar, choose *Place – Pin Placement*.

   The Pin Placement form is displayed.
2. Choose *Level-1 Pins* in the *Edit* region.
3. Select the level-1 pin from the pin table.
4. Choose *To Buried Pin* from the *Edge* drop-down list.



5. Choose *No Constraint (Floating)* from the *Placement Status Constraint* drop-down list.
6. Click *Apply*.

The level-1 pin is moved below the top-level pin. Its size and layer are matched to the buried pin. The status of both the pins (level-1 pin and buried pin) is marked as *Fixed* in the *Status* column of the pin table.

Spacing Pins

Spacing values are enforced only between adjacent pins on an edge, even if the spacing is set between two non-adjacent pins. If you specify a spacing between a pin and an iterated pin, the software applies the spacing between the last pin of the iterated pin and the individual pin. If the specified spacing cannot be satisfied, you see a warning in the CIW and the pins are not moved.

To set the exact distance between two or more ordered pins or between the individual bits of a bus pin,

1. From the layout window menu bar, choose *Place – Pin Placement*.

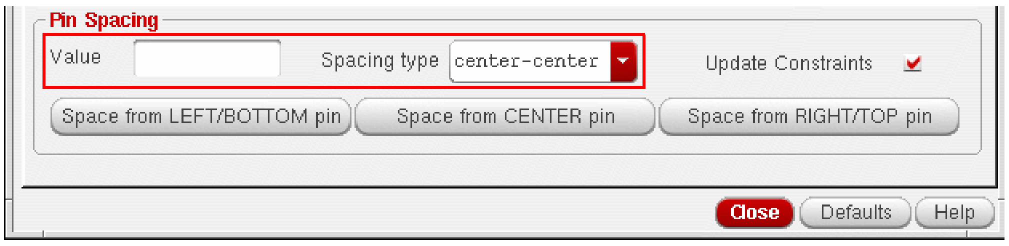
   The Pin Placement form is displayed.

2. Filter the list of pin names as required using the *Display* drop-down list.

3. Select the pins from the pin table.

4. In the *Pin Spacing* group box, type the spacing you require in the *Value* field, choose the type of spacing from the *Spacing type* pull down, and check *Update Constraints* to automatically update any constraints (for example, alignment constraints) associated with the selected pins.

   Note: If a pin to be spaced has the alignment constraint set up, the placer reads the constraint to determine the appropriate access direction for the pin.



5. Click one of the *Space* buttons, depending on which pin is to be the reference pin for the spacing operation.

Important

Pins are spaced and constraints are updated only by clicking one of the *Space* buttons in this section and not by clicking the *Apply* button in the *Attributes* section of the Placement Planning form.

Converting a Pin into a Rail

To convert a pin into a rail,

1. From the layout window menu bar, choose *Place – Pin Placement*.

   The Pin Placement form is displayed.

2. Choose *Unplaced* from the *Status or Type* drop-down list.

3. Select a pin from the pin table.

   Note: To select multiple pins to be converted to a rail, hold down the `Ctrl` key and select the pins.

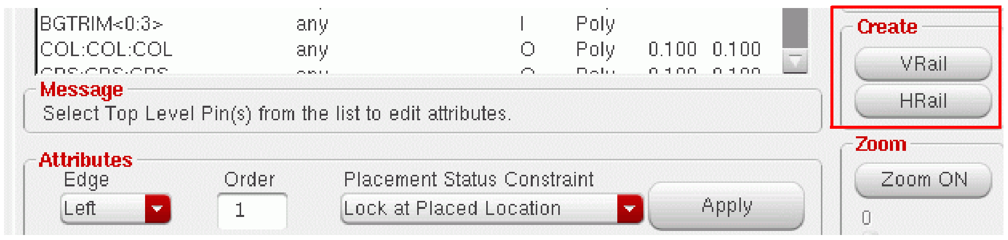
4. Choose an edge from the *Edge* drop-down list.

5. Click *Apply*. The pin is placed on the boundary. If you have selected multiple pins, they will all be placed on the boundary.

6. Select the pin in the layout and do one of the following:

   ❑ Click *HRail* to convert it into a horizontal rail.

   ❑ Click *VRail* to convert it into a vertical rail.


BGTRIM<0:3> any I Poly
COL:COL:COL any O Poly 0.100 0.100
Message
Select Top Level Pin(s) from the list to edit attributes.
Attributes
Edge
Left
Order
1
Placement Status Constraint
Lock at Placed Location
Apply
Create
VRail
HRail
Zoom
Zoom ON
0

When a pin is changed into a rail, it becomes aligned with two edges of the boundary. For a horizontal rail, the alignment is to the left and right edges. For a vertical rail, the alignment is the bottom and top edges. The rail stretches with the boundary in the Constraint-Aware Editing mode.

Important

The Pin Placement form does not distinguish layer information when the pins are placed on the boundary edges. For example, you cannot convert a `metal2` pin to a rail if that pin occupies the same location as a `metal1` pin.

7. (Optional) Use the Constraint Manager Assistant to view the alignment constraints on the rail.

Converting a Rail into a Pin

To convert a rail back into a pin,

1. From the layout window menu bar, choose *Place – Pin Placement*.

   The Pin Placement form is displayed.

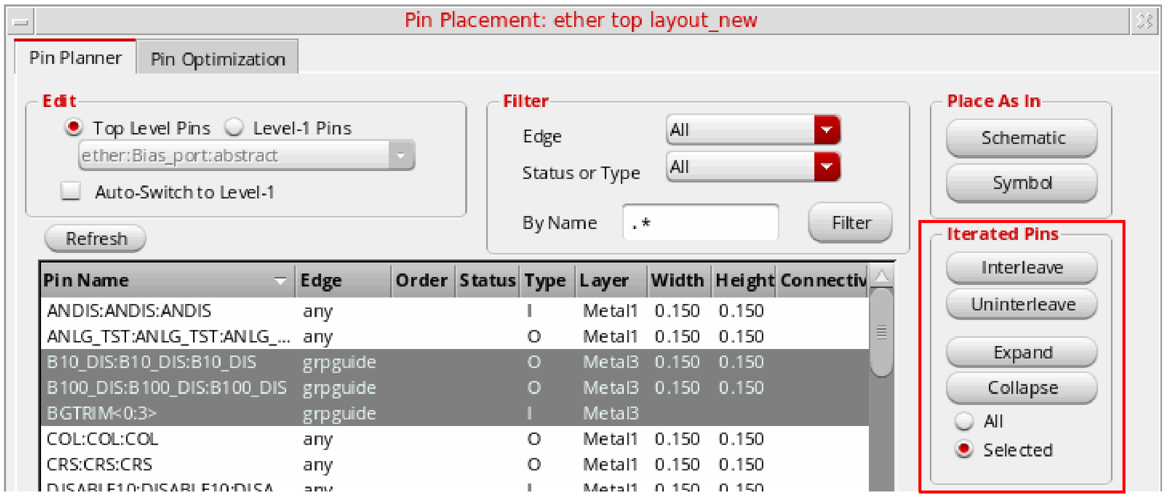
2. In the layout window, select the rail you want to change.

3. In the Pin Placement form, click *unHRail* or *unVRail* as required.

The rail is converted back into a pin and returned to its original size and edge, regardless of the railing direction.

Interleaving the Bits of Iterated Pins

Use the Pin Placement form's *Interleave* button to interleave the bits of the selected iterated pins and automatically apply an order constraint to constrain how the individual bits will be placed.



Edit: [✓ Top Level Pins] Level-1 Pins
ether:Bias_port:abstract
Auto-Switch to Level-1 [ ]

Filter:
Edge: All
Status or Type: All
By Name: .*

Iterated Pins: All [✓ Selected]

Pin Name	Edge	Order	Status	Type	Layer	Width	Height	Connectiv
ANDIS:ANDIS:ANDIS	any			I	Metal1	0.150	0.150	
ANLG_TST:ANLG_TST:ANLG_...	any			O	Metal1	0.150	0.150	
B10_DIS:B10_DIS:B10_DIS	grpguide			O	Metal3	0.150	0.150	
B100_DIS:B100_DIS:B100_DIS	grpguide			O	Metal3	0.150	0.150	
BGTRIM<0:3>	grpguide			I	Metal3			
COL:COL:COL	any			O	Metal1	0.150	0.150	
CRS:CRS:CRS	any			O	Metal1	0.150	0.150	
DISABLE10:DISABLE10:DISA	any			I	Metal1	0.150	0.150	

The selected set must contain at least one pin already assigned to an edge. Any unplaced pins are automatically placed on the same edge and their bits are interleaved. Interleave does not work if the pins in the selected set are assigned to more than one edge or to no edge at all. In this case, the command generates an error message and does nothing.

Important

You can no longer select only unplaced pins, select an *Edge* in the *Attributes* section, and then press *Interleave*. The edge must be specified before you select the pins to be interleaved.

If you select both buses and single pins, interleave processes them in two steps: first it interleaves the buses, then it gets the single pin array, splits it into two parts, and interleaves those also.

For example,

Before Interleave:

```
A<0:2> A1 B<0:2> B1 C<0:1> C1 D1 E1 F1
```

After Interleave:
```
A<0> B<0> C<0> A<1> B<1> C<1> A<2> B<2> A1 D1 B1 E1 C1 F1
```

The *Uninterleave* command works with pins assigned to any number of edges. However, it does not move pins from one edge to another, but instead deals with the pins on each edge separately. *Uninterleave* does nothing with unplaced pins - they are ignored.

3

Placement Planning

This chapter describes the Virtuoso® Custom Digital Placer's placement planning capability. Placement planning lets you define where in the layout view are the transistors and other design elements placed.

Tip

Before you can perform placement planning, you or the library developer must first define the component types for placement. Component types define which components can be placed in which rows.

This chapter covers the following topics:

Row Height Calculation

The row height is determined based on the height of the components supported in the design:

- For designs that contains only devices, the row heights are the height of the *N* and the *P* row.

  For *N* row placement, the row height is calculated based on the maximum height of the Nmos devices. For *P* row placement, the row height is calculated based on the maximum height of the Pmos devices.

- For designs that contains only standard cells, the row height is the height of the standard-cells.

- For designs that contains both devices and standard cells, the row height is the same as the height of the standard-cell unless the device height exceeds the standard-cell height. In this case, the placer issues a warning indicating that the device height is used as row height during placement planning.

  Note: If the design uses FinFET devices—devices of type *NFIN* or *PFIN*—the row height is based on the standard cell height even if the combined height of the *NFIN* and *PFIN* devices exceeds the standard cell height. In this case, the placer always creates the row height based on the standard cell height, but issues a warning to indicate the same, if the combined height of the devices exceeds the standard cell height.

Types of Rows

The custom digital placer handles two different types of rows:

- An OpenAccess row (`oaRow`) is an OpenAccess database object recognized by all tools that support the OpenAccess database. It is represented in Virtuoso as a `figGroup` with type `row`.

  An OpenAccess row requires appropriate site definitions (`siteDefs`) to be present in the technology database for the design. Instances are placed at the fixed `siteDef` locations within the row.

- A custom placement area is an object generated by the placer if there are no appropriate `siteDefs` available with which to generate OpenAccess rows. It is represented as a `figGroup` with type `placeArea`.

  A custom placement area offers more flexibility in the placement of instances because the locations where devices can be placed are not fixed. However, the Virtuoso custom placement solution is guaranteed to recognize a custom placement area as a row object.

  For information on how to manually create rows and custom placement area, see Creating Rows and Create Custom Placement Area.

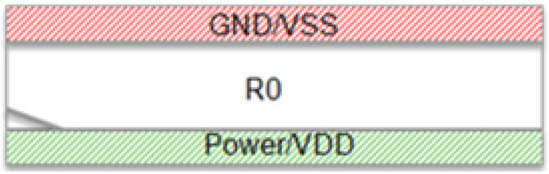
Snapping the Custom Placement Area

When manually creating a custom placement area in the layout environment by using the *Create – PR Boundary – Custom Placement Area* command, the custom placement area is automatically snapped to an appropriate grid depending on the state of the snapCpaToPlacementGrid environment variable. For more information, see Create Custom Placement Area.

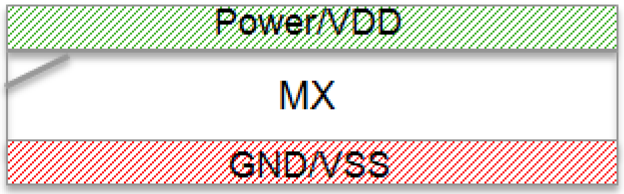
Support for Flipped Row Creation

The Virtuoso Custom Digital Placer automatically flips the rows during creation, if required, to align them with the rail pattern specified. This implies that a horizontal (or vertical) row can have R0 or MX orientation, depending on the rail pattern associated with the row.

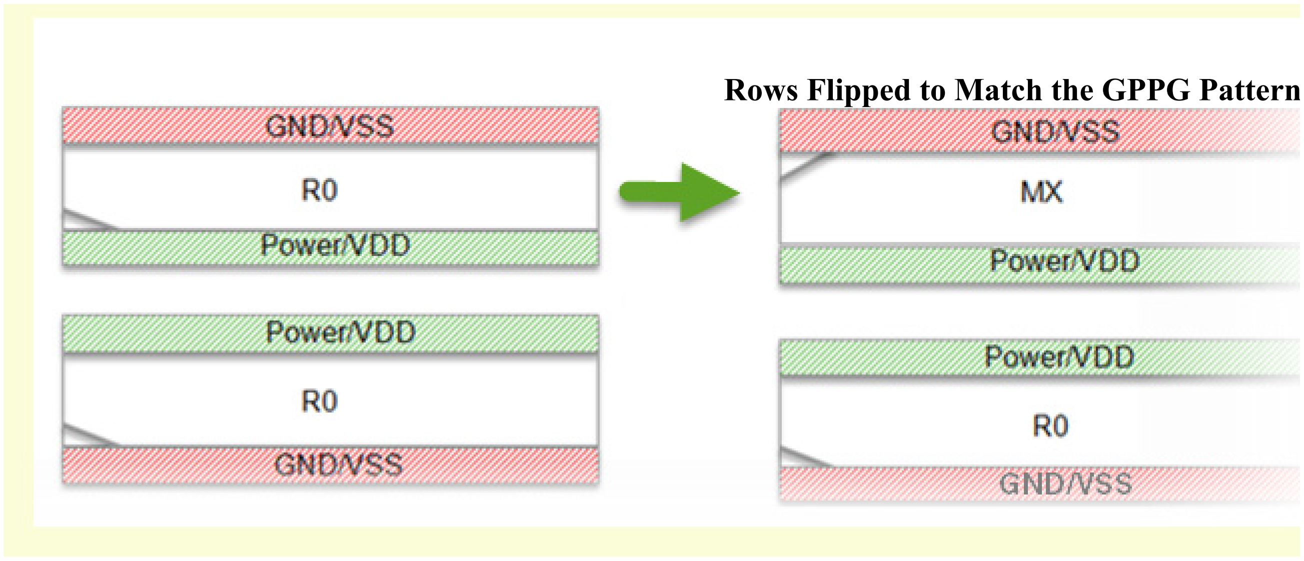
When following a GP rail pattern, the horizontal (or vertical) row can be created with an R0 orientation, as displayed in the figure below, and allowed component orientations set to R0/MY.



When following a PG rail pattern, a horizontal (or vertical) PG row can be created with a row orientation MX, as shown in the figure below, and allowed component orientations set to R0/MY.



The figure below illustrates how the row orientations are now automatically changed to match the specified rail pattern, which is GPPG in this case.



Blockage Support

The custom placer honors the following types of blockages:

- Placement blockages: Components cannot be placed in the specified blockage area. The custom placer also honors partial placement blockages, which specify the density of the instances placed in the area covered by the blockages.

  The density of partial blockages should be greater than 0 and less than 100. For example if the density of a partial placement blockage is 60 percent, the placer tries to keep the density of the instances that are placed in that area to be around 60 percent of the global utilization.

- Routing blockages: Pins cannot be placed in the specified blockage area. Layers of the blockage are not considered. However, standard cells and devices can be placed in the blockage area.
- Pin blockages: Pins cannot be placed in the specified blockage area. Layers of the blockage are not considered. However, standard cells and devices can be placed in the blockage area.

All other types of blockages are not honored.

Placement Restrictions - Designs with Only Devices

The following placement restrictions apply for designs that contain only devices:

- All rows must be horizontal.
- All rows have the same width and span the entire placement region, although the row width can vary if the region is polygonal.
- Each row can contain either NMOS or PMOS devices, but not both and not any other types of components.
- There are equal numbers of NMOS and PMOS rows.
- There is a regular arrangement of alternating power and ground rails between rows:
  - G-P-P-G
  - P-G-G-P
  - G-P
  - P-G
- Device rotation depends on the row and pattern.
- The rows can extend outside of the boundary or cluster.

  Note: The custom placer can place only within the boundary or cluster.

Planning a Placement

Use the Placement Planning form to prepare your design for placement before you run the Virtuoso® Custom Digital Placer. Select *Place – Custom Digital – Placement Planning* to display the Placement Planning form. The options in the form are organized in tabs, which control the following placement settings.

- Adding Boundary Cells
- Generating Tap Cells
- Defining Rows
- Setting Other Placement Parameters

(IC23.1 EXL and Higher Tiers) In the Virtuoso® Design Planner environment, you can use the Placement Planning form to create rows in the PR boundary, cluster boundary, and virtual hierarchy boundary (area boundary).

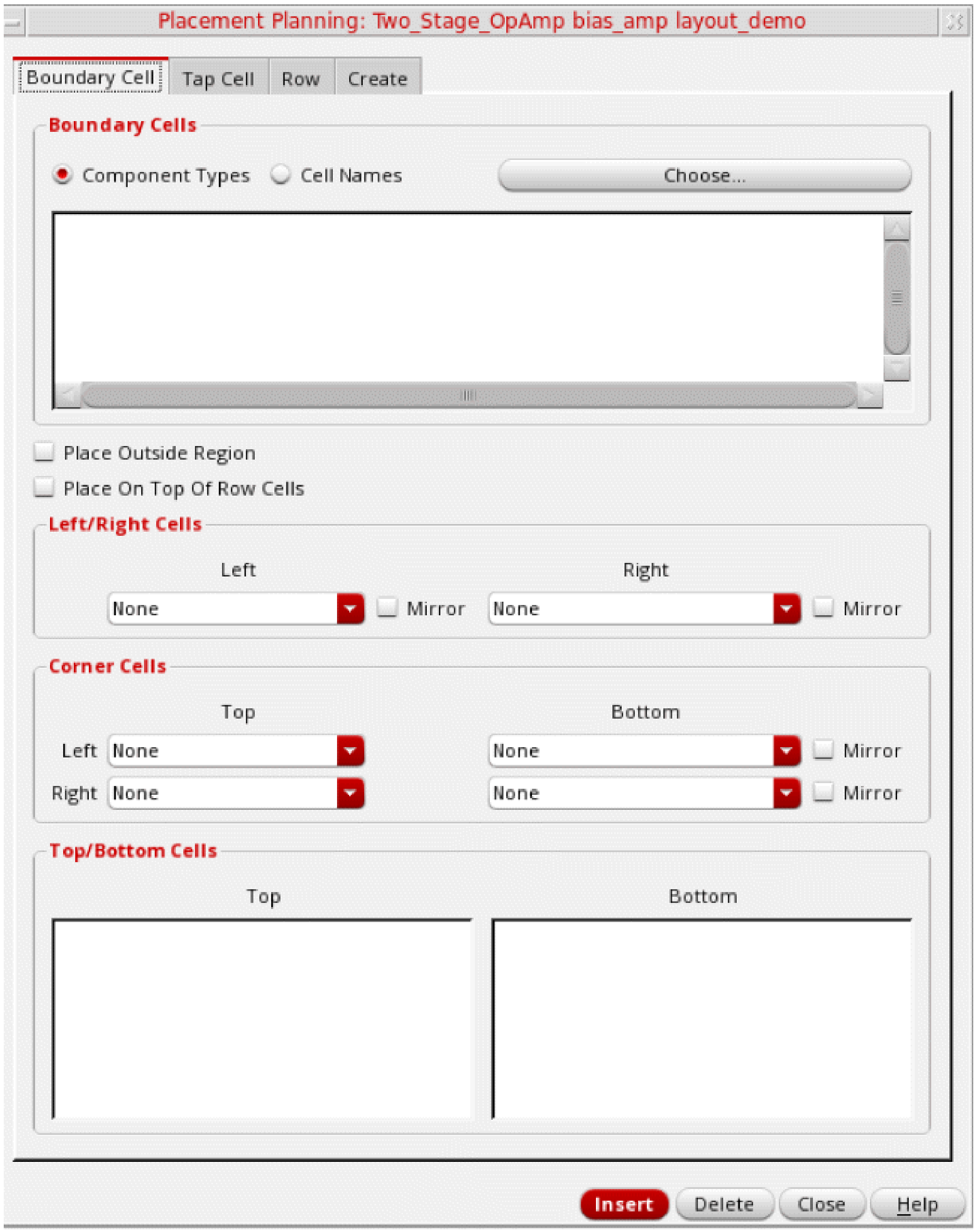
Adding Boundary Cells

At advanced nodes, adjacent cells (core cells) are placed in close proximity, which could lead to shorts and DRC violations. Adding boundary cells around core cells lets you isolate the core cells from each other, and therefore helps prevent such undesired effects.

Boundary cells are represented by cells that have their component class set to `BOUNDARYCELL`. Component types are defined in the *Cells* table of the Configure Physical Hierarchy form - *Component Types* mode.

After ensuring that the boundary cell definitions are in place, you can start inserting them between the core cells in your design.

Use the options on the *Boundary Cell* tab of the Placement Planning form to insert boundary cells.



Placement Planning: Two_Stage_OpAmp bias_amp layout_demo

Boundary Cells

[✓ Component Types] Cell Names Choose...

☐ Place Outside Region
☐ Place On Top Of Row Cells

Left/Right Cells

Left	None ☐ Mirror	Right: None ☐ Mirror

Corner Cells

	Top	Bottom
Left	None	None ☐ Mirror
Right	None	None ☐ Mirror

Top/Bottom Cells

Top Bottom

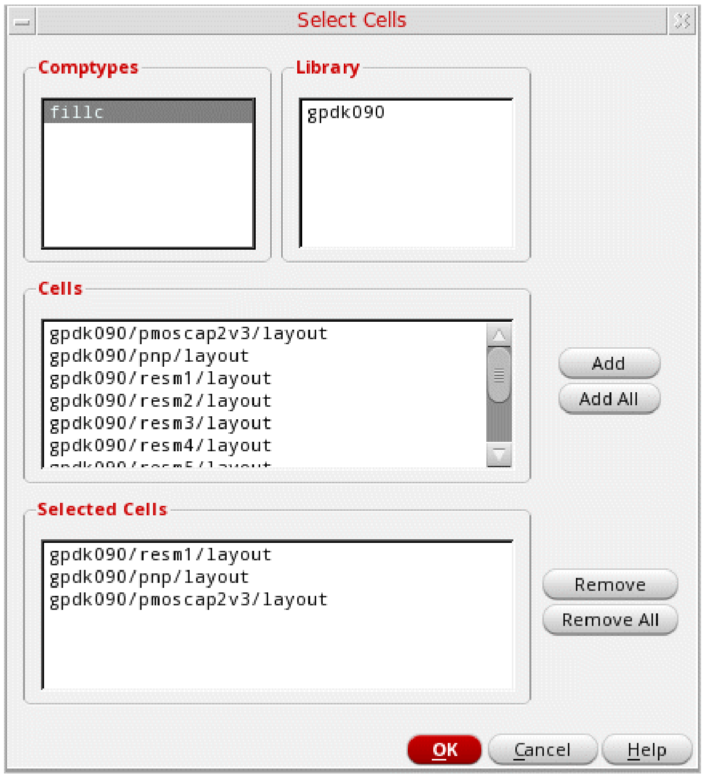
Important

If a row template with boundary cell and tap cell definitions is selected in the *Row* tab, then these definitions are honored. The settings in the *Boundary Cell* and *Tap Cell* tabs are ignored.

On the *Boundary Cell* tab:

1. Specify whether you want to select boundary cells based on their *Component Types* or *Cell Names*.
2. Click *Choose* to display the Select Cells form. The options in the form differ depending on whether *Component Types* or *Cell Names* was selected.

   ❑ When *Component Types* is selected, the following Select Cells form is displayed.



Comptypes: fillc

Library: gpdk090

Cells:
```
gpdk090/pmoscap2v3/layout
gpdk090/pnp/layout
gpdk090/resm1/layout
gpdk090/resm2/layout
gpdk090/resm3/layout
gpdk090/resm4/layout
```

Selected Cells:
```
gpdk090/resm1/layout
gpdk090/pnp/layout
gpdk090/pmoscap2v3/layout
```

In the Select Cells form:

a. Select the required *Comptypes*. The corresponding *Library* and *Cells* entries are listed.

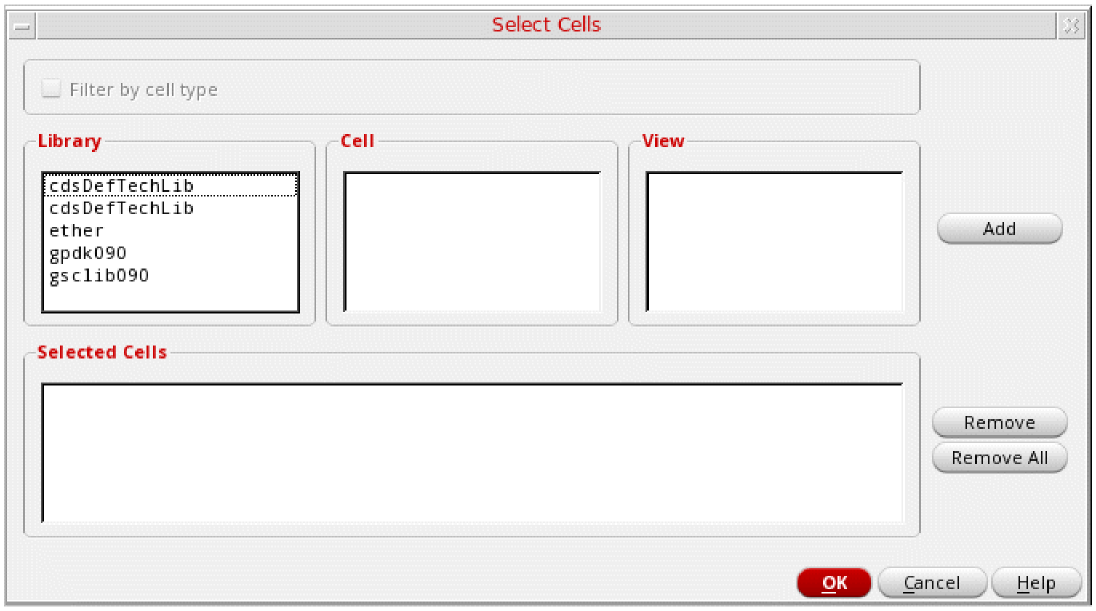
b. Select the required *Library* and *Cells* entries.

c. Click *Add*. The selected cellviews are listed in the *Selected Cells* text box.

Note: Click *Add All* to select all cellviews. Use the *Remove* and *Remove All* buttons to remove cells from the *Selected Cells* list.

d. Click *OK*.

❑ With *Cell Names* selected, the following Select Cells form is displayed.



```
cdsDefTechLib
cdsDefTechLib
ether
gpdk090
gsclib090
```

In the Select Cells form:

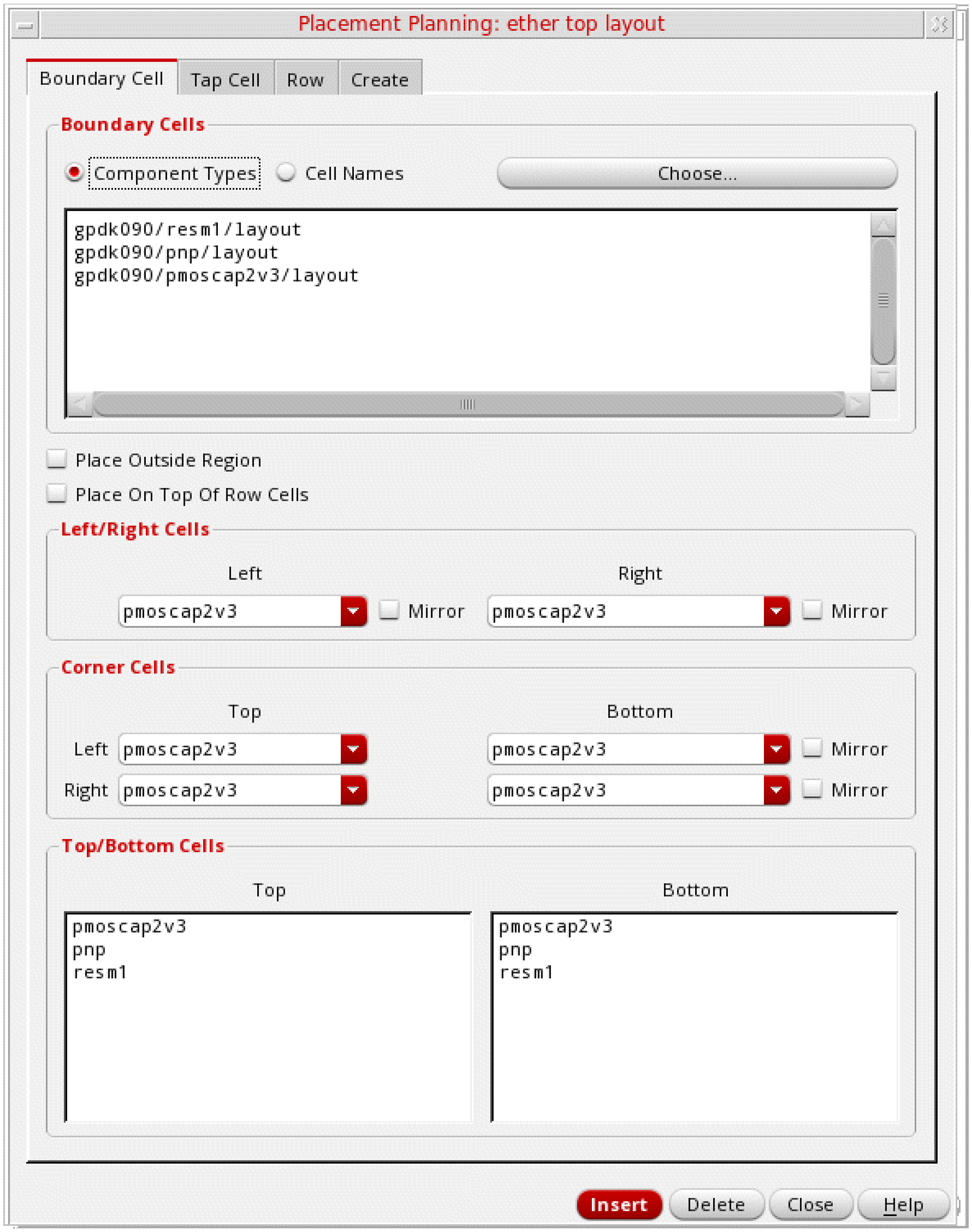
a. Select the required *Library*, *Cell*, and *View*.

b. Click *Add*. The selected cellviews are listed in the *Selected Cells* text box.

c. Click *OK*.

Note: Use the *Remove* and *Remove All* buttons to remove cells from the *Selected Cells* list.

The selected cells are populated in all the applicable fields on the *Boundary Cell* tab of the Placement Planning form.


Placement Planning: ether top layout

Boundary Cells

[✓ Component Types] Cell Names

```
gpdk090/resm1/layout
gpdk090/pnp/layout
gpdk090/pmoscap2v3/layout
```

☐ Place Outside Region
☐ Place On Top Of Row Cells

Left/Right Cells

Left	Mirror	Right	Mirror
pmoscap2v3	☐	pmoscap2v3	☐

Corner Cells

	Top	Bottom	Mirror
Left	pmoscap2v3	pmoscap2v3	☐
Right	pmoscap2v3	pmoscap2v3	☐

Top/Bottom Cells

Top	Bottom
pmoscap2v3	pmoscap2v3
pnp	pnp
resm1	resm1

3. Select *Place Outside Region* to generate boundary cells outside the row region. As a result, the placement region is shrunk by a value equal to the width of the boundary cells.
4. Select *Place On Top Of Row Cells* to allow the placement of boundary cells over any existing standard cells on the left, right, top, and bottom rows.
5. Select the boundary cells to be inserted in the *Left* and *Right* edges. Select *Mirror* to apply a mirrored orientation to both the cells.
6. Select the boundary cells to be inserted in the corners – *Top Left* and *Bottom Left*. Select *Mirror* to apply a mirrored orientation to both cells.
7. Select the boundary cells to be inserted in the corners – *Top Right* and *Bottom Right*. Select *Mirror* to apply a mirrored orientation to both cells.

   Note: The *Mirror* setting is applicable only to horizontal (MY) orientation of instances. There is no need to specify mirror setting for vertical (MX) orientation. However, vertical orientation of boundary cells is controlled by the corresponding row orientation.
8. Select the boundary cells to be inserted in the *Top* and *Bottom* edges.
9. Before inserting boundary cells, navigate to the other tabs to define the tap cells and rows that need to be added.
10. Open the *Create* tab to specify the placement region and other generic settings.
11. Ensure that *Boundary Cells* is selected.
12. Click *Insert*.

Important

For more information about the boundary cell types, depending on the position of the boundary cell relative to the standard cell row, see Valid edge assignments in CPH window.

Related Environment Variables

You can use environment variables to define the boundary cell component types or cell names that are to be loaded when the form is invoked.

The following example populates cells based on the boundary cell component types defined in the CPH form.

```
envSetVal("layoutXL.placement" "selectBoundaryCellsFrom" 'cyclic "Component 
Types")
envSetVal("layoutXL.placement" "boundaryCellType"  'string "comptye1 comptype2")
```

The following example gets cells from a selected boundary cell component type.

```
envSetVal("layoutXL.placement" "selectBoundaryCellsFrom" 'cyclic "Component
Types")
envSetVal("layoutXL.placement" "boundaryCellType"  'string "comptye1")
```

The following example gets cells from a list of selected cells.

```
envSetVal("layoutXL.placement" "selectBoundaryCellsFrom" 'cyclic "Cell Names")
envSetVal("layoutXL.placement" "selectedBoundaryCells" 'string "(mylib fillc21
layout)(mylib fillc41 layout)")
```

List of Environment Variables

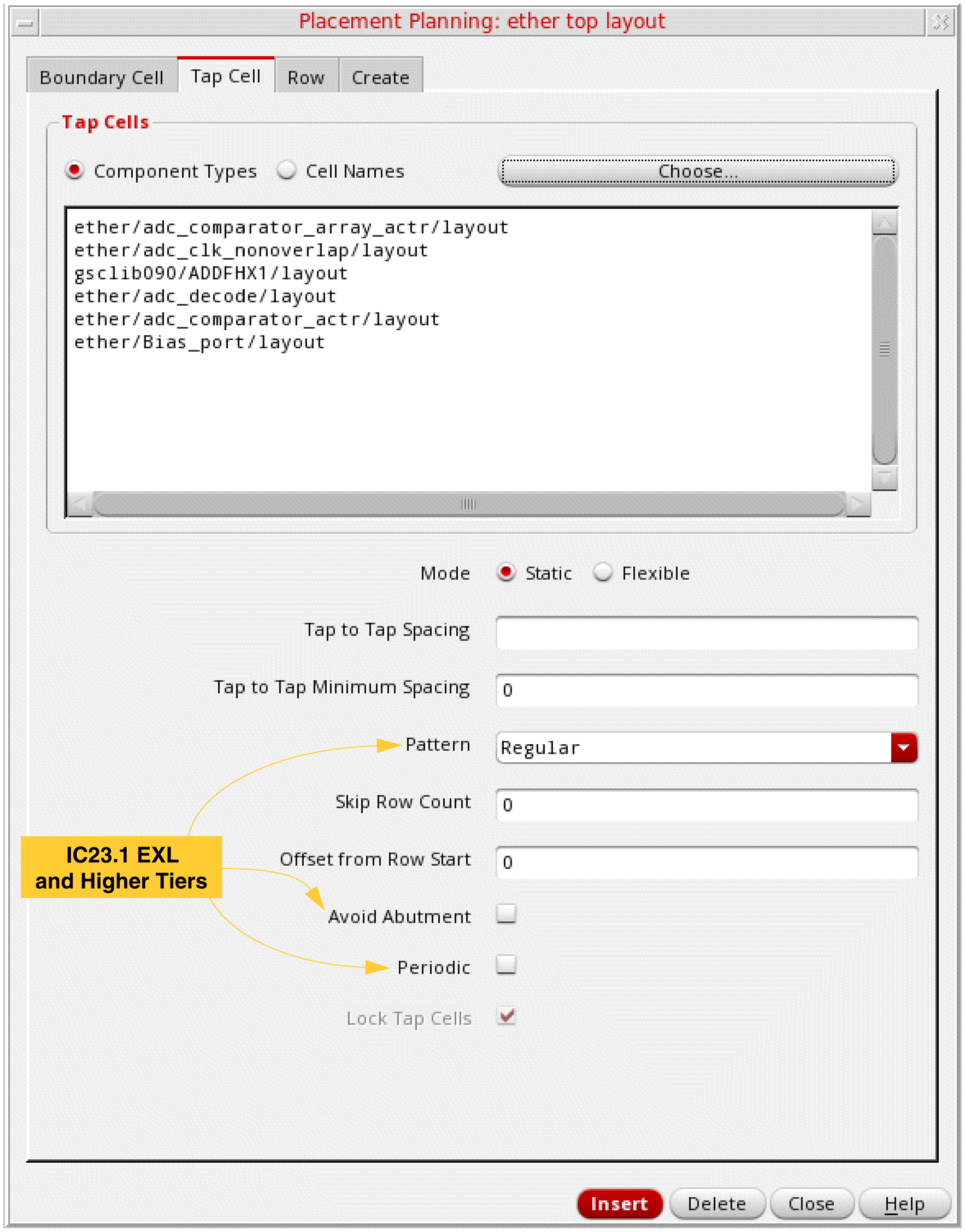
- selectBoundaryCellsFrom
- selectedBoundaryCells
- boundaryCellType
- insertBdryCells
- topRowBdryCell
- topTapBdryCell
- botRowBdryCell
- botTapBdryCell
- leftBotCornerBdryCell
- leftEdgeBdryCell
- leftTopCornerBdryCell
- rightBotCornerBdryCell
- rightEdgeBdryCell
- rightTopCornerBdryCell
- topBotRowPlaceable
- leftBotEdgeBdryCell

Generating Tap Cells

After specifying boundary cell parameters, you can (optionally) insert either single-height or multi-height tap cells in the empty spaces between the standard cells. Tap cells are a set of contacts that are used to reduce latch-up effects between power and ground connections and the connections with wells or substrate contacts.

Tap cells are represented by cells that have their component class set to `STDSUBCONT`. Component types are defined in the *Cells* table of the Configure Physical Hierarchy form - *Component Types* mode. After ensuring that the tap cell definitions are in place, you can start inserting them between the core cells in your design.

Use the options on the *Tap Cell* tab of the Placement Planning form to insert tap cells before running Custom Digital Placer.



[● Component Types] ○ Cell Names

```
ether/adc_comparator_array_actr/layout
ether/adc_clk_nonoverlap/layout
gsclib090/ADDFHX1/layout
ether/adc_decode/layout
ether/adc_comparator_actr/layout
ether/Bias_port/layout
```

Mode	[● Static] ○ Flexible
Tap to Tap Spacing	
Tap to Tap Minimum Spacing	0
Pattern	Regular
Skip Row Count	0
Offset from Row Start	0
Avoid Abutment	☐
Periodic	☐
Lock Tap Cells	✓

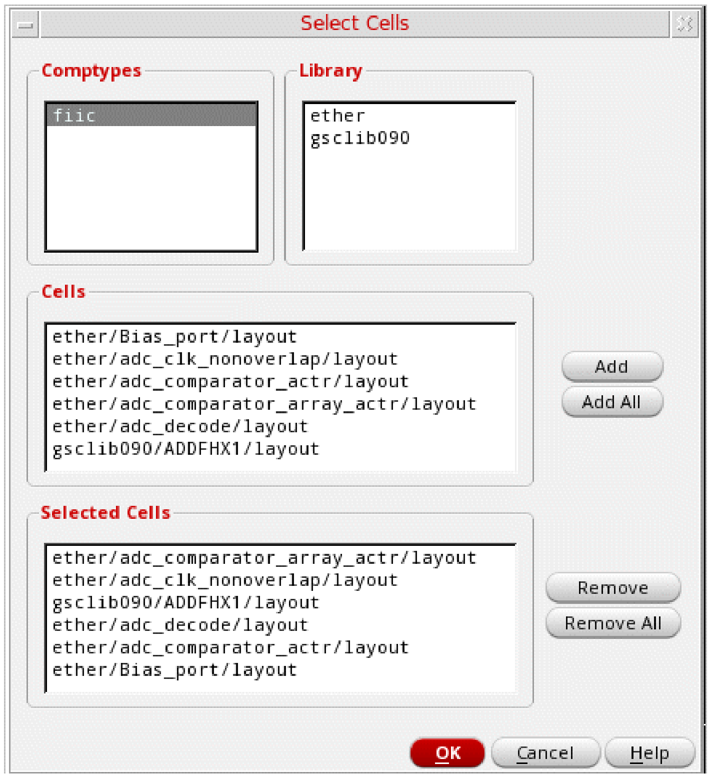
Corresponding SKILL function: vcpfePlaceTapCells

Important

If a row template with boundary cell and tap cell definitions is selected in the *Row* tab, then these definitions are honored. The settings in the *Boundary Cell* and *Tap Cell* tabs are ignored.

On the *Tap Cell* tab:

1. Select either *Component Types* or *Cell Names* to specify the tap cells. With *Component Types* selected, all valid tap cells, which have their component type set to `STDSUBCONT`, are automatically listed in the *Tap Cells* text box. You can edit this list.
2. Click *Choose* to display the Select Cells form. The options in the form differ depending on whether *Component Types* or *Cell Names* is selected.
   - ❑ When *Component Types* is selected, the following Select Cells form is displayed.



Select Cells

Comptypes

```
fiic
```

Library

```
ether
gsclib090
```

Cells

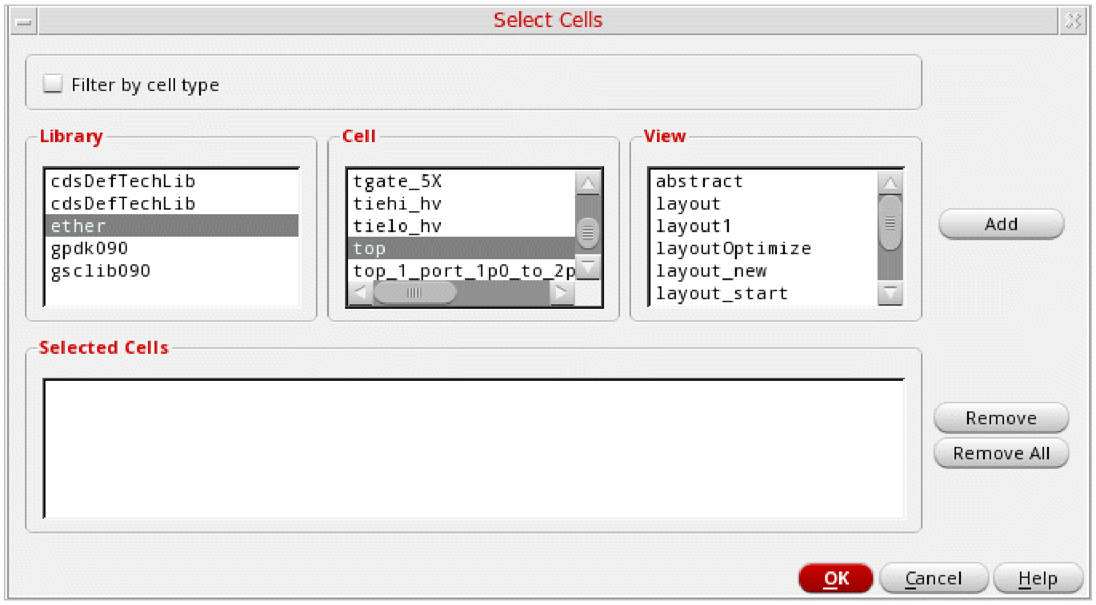
```
ether/Bias_port/layout
ether/adc_clk_nonoverlap/layout
ether/adc_comparator_actr/layout
ether/adc_comparator_array_actr/layout
ether/adc_decode/layout
gsclib090/ADDFHX1/layout
```

Selected Cells

```
ether/adc_comparator_array_actr/layout
ether/adc_clk_nonoverlap/layout
gsclib090/ADDFHX1/layout
ether/adc_decode/layout
ether/adc_comparator_actr/layout
ether/Bias_port/layout
```

In the Select Cells form, all valid cells are automatically listed in the *Selected Cells* text box. You can add or remove cells based on your requirement.

❑ With *Cell Names* selected, the following Select Cells form is displayed.



In the Select Cells form:

a. Select the required *Library*, *Cell*, and *View*.

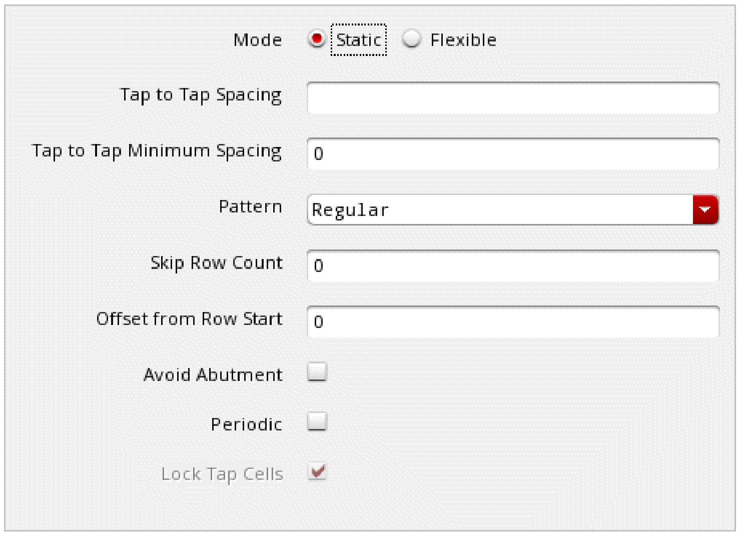
b. Click *Add*. The selected cellviews are listed in the *Selected Cells* text box.

c. Click *OK*.

The selected cells are populated in all the applicable fields on the *Tap Cell* tab of the Placement Planning form.

3. Select a *Mode* to specify whether the tap cells must be *Static* or *Flexible*.

- ❑ **Static:** Fixes the positions of tap cells as per the exact spacing values. Therefore, any change to the core cell placement does not affect the placement of the static tap cells.

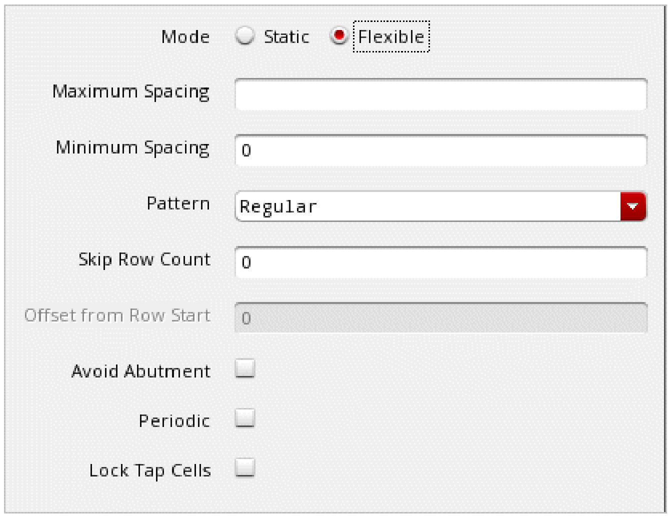


Mode	[✓ Static] Flexible
Tap to Tap Spacing	
Tap to Tap Minimum Spacing	0
Pattern	Regular
Skip Row Count	0
Offset from Row Start	0
Avoid Abutment	☐
Periodic	☐
Lock Tap Cells	✓

The following options are available only in the *Static* mode:

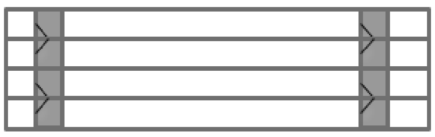
- ❍ *Tap to Tap Spacing* specifies the spacing to be maintained between adjacent tap cells.
- ❍ *Tap to Tap Minimum Spacing* is applicable in situations where the tap cell placement needs to back track, for example when there is an existing blockage.

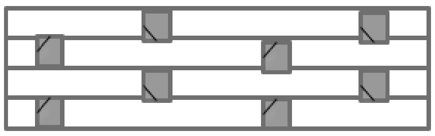
- ❑ **Flexible:** Does not fix the positions of tap cells. Therefore, changes to core cell placement alter the tap cell placement. Use the *Maximum Spacing* and *Minimum Spacing* options to define a spacing range in the *Flexible* mode.



Mode	Static [✓ Flexible]
Maximum Spacing	
Minimum Spacing	0
Pattern	Regular
Skip Row Count	0
Offset from Row Start	0
Avoid Abutment	☐
Periodic	☐
Lock Tap Cells	☐

4. Specify the *Pattern* in which the tap cells must be inserted. You can select between the *Regular* and *Checker Board* patterns.

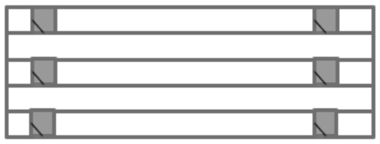




   Note: The *Skip Row Count* and *Avoid Abutment* settings are not available when the *Checker Board* pattern is selected.

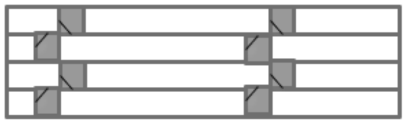
   Note: The *Skip Row Count* and *Checker Board* options are ignored when multi-height tap cells are used.

5. (optional) Specify the *Skip Row Count*, which specifies the rows that must be skipped during tap cell placement. The top row cannot be skipped. In the following example, alternate rows are skipped.

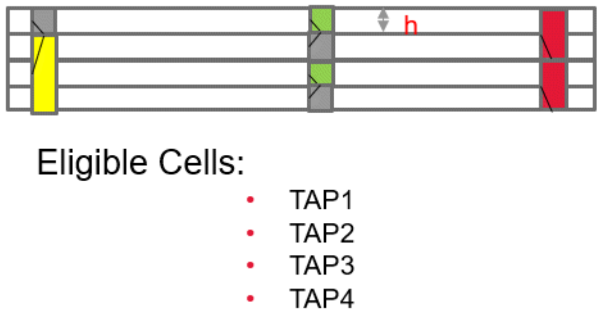


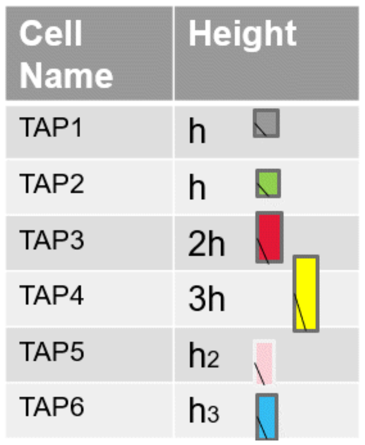
6. Specify the *Offset from Row Start*, which is the initial offset for the first tap cell in each row. This option is available only when *Mode* is set to *Static*.

7. Select *Avoid Abutment* to specify that the tap cells must not be abutted vertically. By default, tap cells are abutted.



8. Select *Periodic* to insert tap cells periodically from the list of available cells starting from the bottom lower left.





Cell Name	Height
TAP1	h ▨
TAP2	h 🟩
TAP3	2h 🟥
TAP4	3h 🟨
TAP5	h_2 ▯
TAP6	h_3 🟦

9. Select *Lock Tap Cells* to apply the settings of *Flexible* tap cells and then lock the tap cells to their positions. This option is not available in the *Static* mode.

Related Environment Variables:

You can use environment variables to define the tap cell component types or cell names that are to be loaded when the form is invoked.

The following example populates cells based on the tap cell component types defined in the CPH form.

```
envSetVal("layoutXL.placement" "selectTapCellsFrom" 'cyclic "Component Types")
envSetVal("layoutXL.placement" "substrateContactType"  'string "comptye1
comptype2")
```

The following example gets cells from a selected boundary cell component type.

```
envSetVal("layoutXL.placement" "selectTapCellsFrom" 'cyclic "Component Types")
envSetVal("layoutXL.placement" "substrateContactType"  'string "comptye1")
```

The following example gets cells from a list of selected cells.

```
envSetVal("layoutXL.placement" "selectTapCellsFrom" 'cyclic "Cell Names")
envSetVal("layoutXL.placement" "selectedTapCells" 'string "(mylib fillc21
layout)(mylib fillc41 layout)")
```

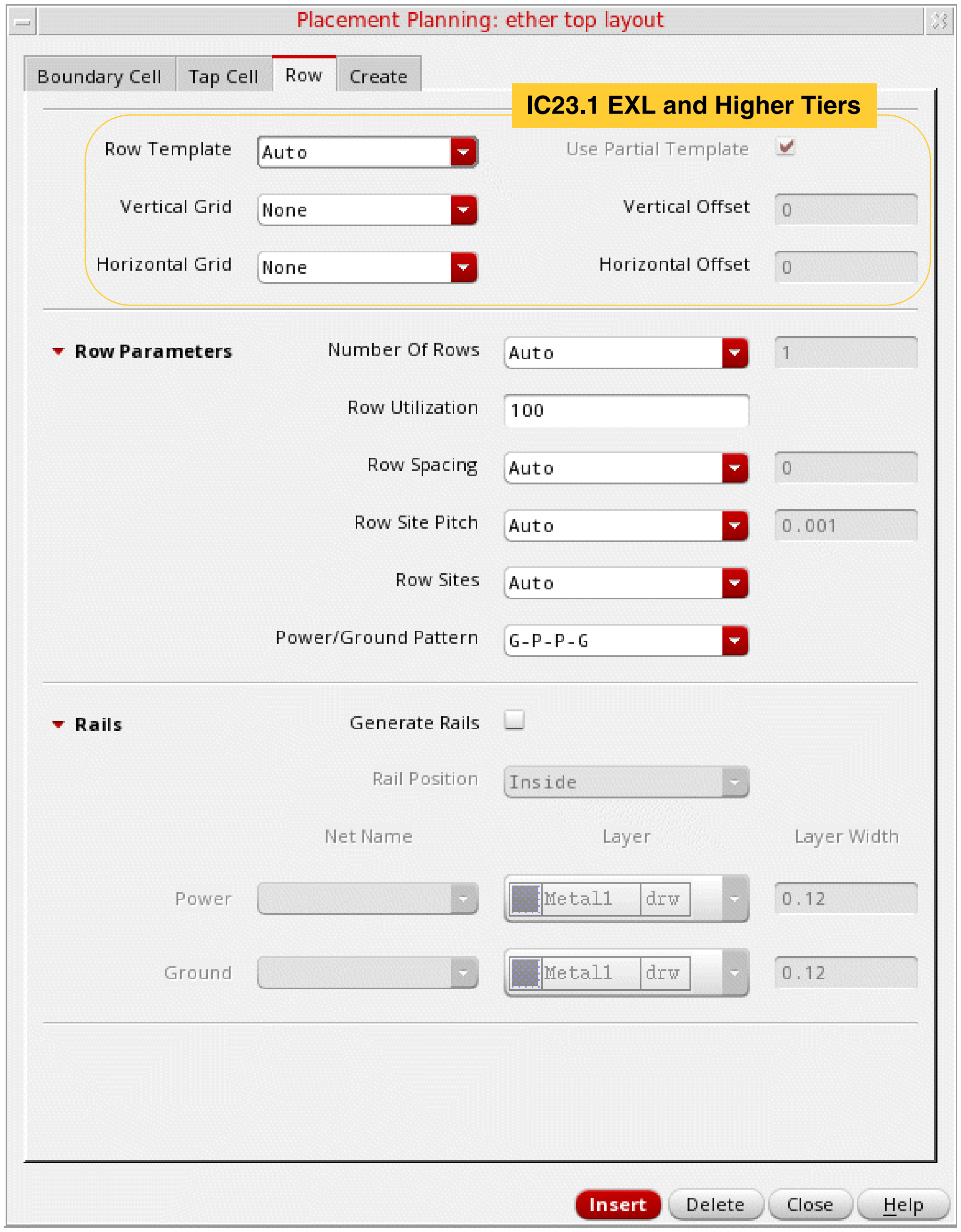
List of Environment Variables

- selectedTapCells
- selectTapCellsFrom

- substrateContactType
- topTapBdryCell
- avoidAbutment
- insertSubstrateContacts
- lockTap
- packedPlacement
- periodicTap
- rowTapEndOffset
- subContInAlternateRow
- subContMaxSpacing
- subContMinSpacing
- substrateContactType

Defining Rows

Row-based placement provides tremendous advantage over the conventional placement methods. Row-based placement helps improve routability, achieve better wire length after routing, and resolve illegal placements by balancing the number of instances placed in each row. It also helps resolve overlaps between the devices in each row. Use the options on the *Row* tab to define the row settings before generating them in the placement region.


Placement Planning: ether top layout

Boundary Cell | Tap Cell | Row | Create

IC23.1 EXL and Higher Tiers

Row Template	Auto	Use Partial Template	✓
Vertical Grid	None	Vertical Offset	0
Horizontal Grid	None	Horizontal Offset	0

Row Parameters

Number Of Rows	Auto	1
Row Utilization	100	
Row Spacing	Auto	0
Row Site Pitch	Auto	0.001
Row Sites	Auto	
Power/Ground Pattern	G-P-P-G	

Rails

Generate Rails ☐

Rail Position: Inside

	Net Name	Layer	Layer Width
Power		Metal1 drw	0.12
Ground		Metal1 drw	0.12

1. Select the *Row Template* to be used to generate rows in the layout canvas. Use the Row Template Manager to define row templates.

If you select a row template, the *Row Parameters* and *Rails* sections are collapsed. *Number of Rows* is the only option available in the *Row Parameters* section, and its value is reset to *Maximum*.

If the selected row template has boundary cell and tap cell definitions, then these definitions are honored. In that case the settings in the *Boundary Cell* and *Tap Cell* tabs are ignored.

2. Select *Use Partial Template* to allow partial row templates while filling the placement region. Therefore, the selected row template is repeated multiple times, and the remaining space at the top, which is less than the height of a row template, is filled with a partial row template.

   When *Use Partial Template* is not selected, only full row templates are allowed. Therefore, any remaining space at the top, which is less than the height of a row template, is left empty.

   Note: *Use Partial Template* is available only when you select a row template in the previous step.

3. Specify the reference grids (*Vertical Grid* and *Horizontal Grid*) for snapping the devices and rows during placement.

4. Specify the offsets (*Vertical Offset* and *Horizontal Offset*) that must be applied to rows after they are snapped to the reference grids.

5. Expand the *Row Parameters* section to define the row parameters.

6. Specify the *Number Of Rows*. Options are:

   - **Auto:** Automatically calculates the number of rows depending on the number of devices to be placed and the specified *Row Utilization (%)*.
   - **Maximum:** Accommodates the maximum number or rows in the placement region.
   - **Specify:** Lets you specify the number of rows to be generated. If the specified number of rows cannot be accommodated within the boundary, for example if the row height is greater than the region height, then an appropriate warning is displayed.

7. Specify a *Row Utilization (%)* value to indicate the space available for placing components. The remaining space is used for routing. The higher the utilization value, the lesser the space available for the router to route the design. The lower the utilization value, the more free space is made available for the router.

8. Specify the *Row Spacing* between rows. If a value is not specified, then the placer automatically calculates a value.

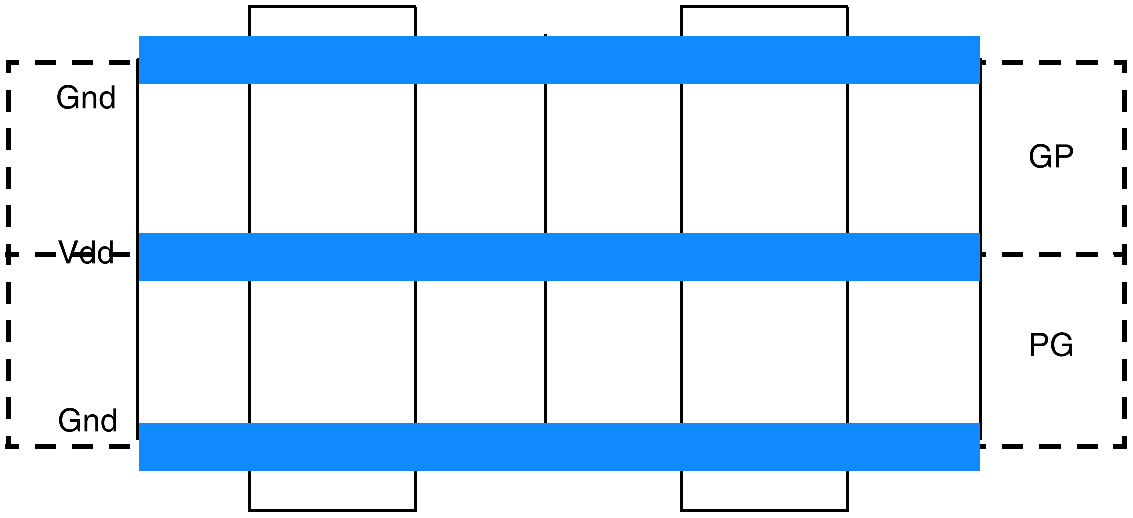
9. Specify whether the *Row Site Pitch* value will be calculated automatically or defined manually.
10. Specify whether the value calculated for the number of *Row Sites* must be used as is or rolled down to the nearest *Odd* or *Even* number.
11. Specify the *Power/Ground Pattern* for alternating rails between rows. Available patterns are G-P, P-G, G-P-P-G, and P-G-G-P. This option lets you flip rows and save space by sharing power or ground rails between the rows.
12. Expand the *Rails* section to define rail parameters.
13. Select *Generate Rails* to create rails.
14. Specify the *Rail Position* with respect to the rows. Available options are *Inside*, *Outside*, and *Center*.
15. Specify the *Power Net* to which the rail belongs.
16. Specify the *Ground Net* to which the rail belongs.
17. Specify the *Layer* on which the rail must be drawn. If rails are not created for any reason, then a suitable warning message is displayed.
18. Specify the *Width* of the rails.

Example 1:

The following parameters are specified:

- *Row Spacing*: *2*
- *Power/Ground Pattern*: *GPPG*
- *Generate Rails—Rail Position*: *Inside*

Here, the power rails are not shared because the *Row Spacing* is not `0`.

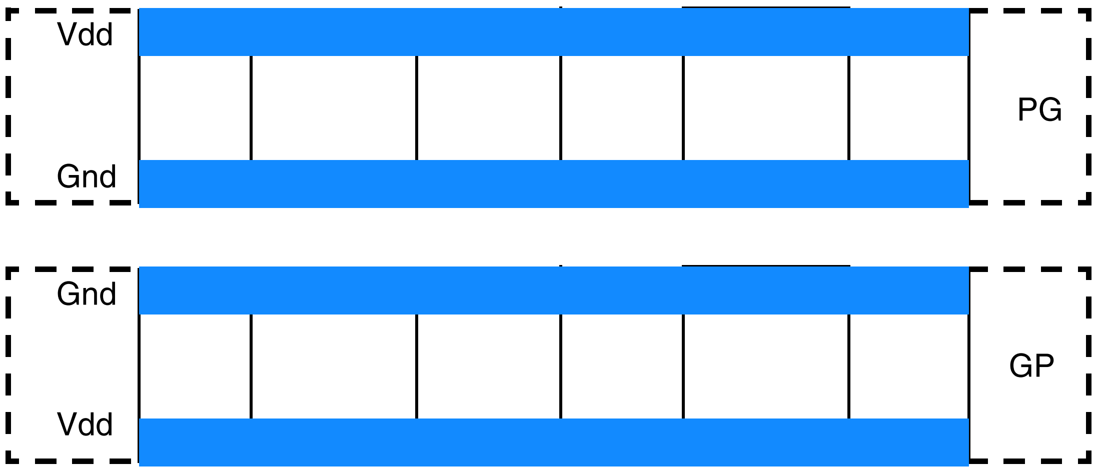


Example 2:

The following parameters are specified:

- *Row Spacing*: *0*
- *Power/Ground Pattern*: *PGGP*
- *Generate Rails—Rail Position*: *Center*

Here, ground rails are shared because the *Row Spacing* is `0`.



Related Environment Variables:

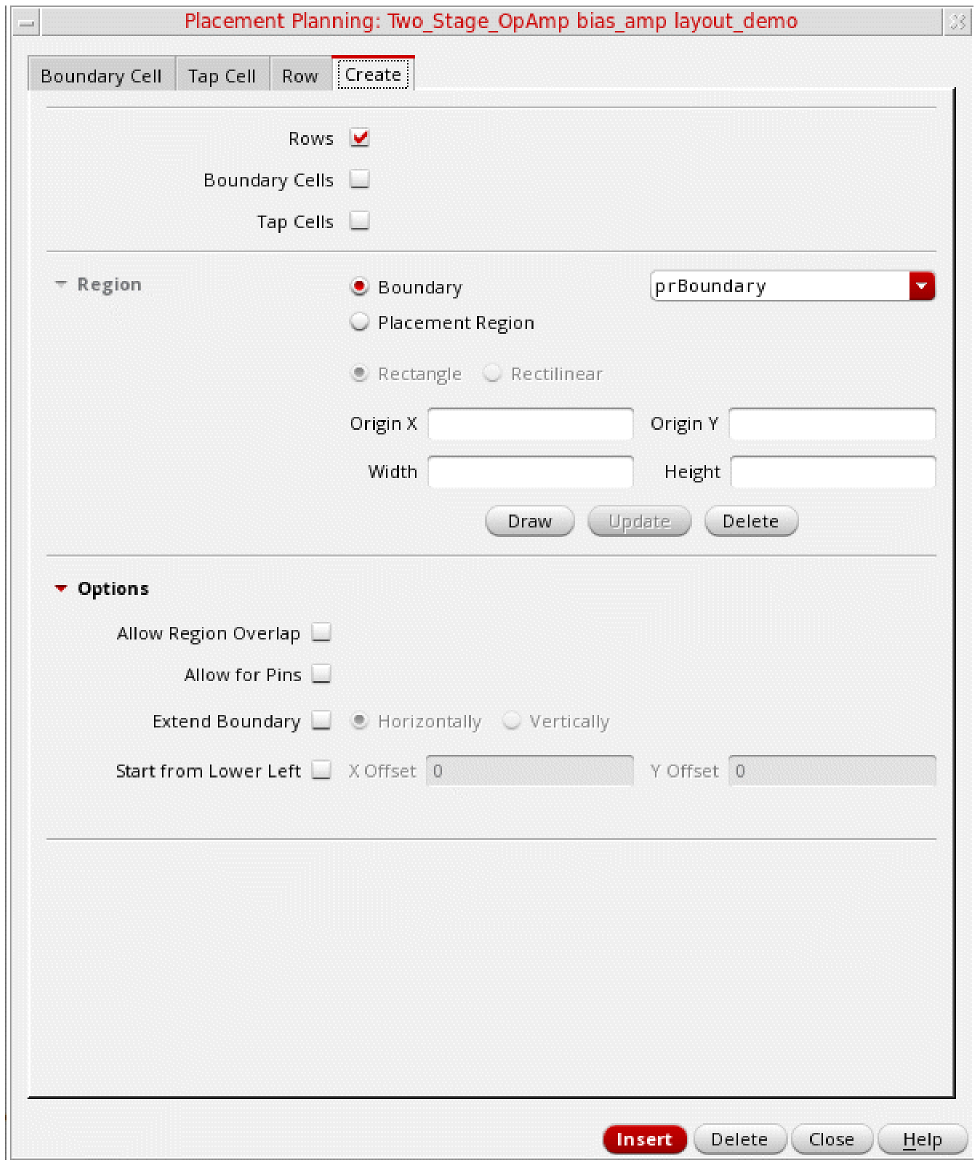
Rows:

- allowRowsBeyondRegion
- generateRails
- interRowSpacer
- intraRowSpacer
- numRowOption
- numRowSites
- numTemplateRowOption
- rowCount
- rowsFromLowerLeft
- rowGroundLayer
- rowGroundName
- rowGroundWidth
- rowOffsetX
- rowOffsetY
- rowPowerLayer
- rowPowerName
- rowPowerWidth
- rowSpacingOption
- rowTapEndOffset
- rowTemplateName
- rowUtilization
- row2RowSpacing
- rowPitchOption
- rowPitchValue

- templateRowCount
- topBotRowPlaceable
- usePartialTemplate

Setting Other Placement Parameters

After defining the *Boundary Cell*, *Tap Cell*, and *Row* parameters, the final step is to define the generic placement settings before running the command. Use the options on the *Create* tab to define the generic settings.


Placement Planning: Two_Stage_OpAmp bias_amp layout_demo

Boundary Cell | Tap Cell | Row | Create

Rows [✓]
Boundary Cells [ ]
Tap Cells [ ]

Region
[● Boundary] prBoundary
Placement Region
[● Rectangle] Rectilinear
Origin X
Origin Y
Width
Height

Options
Allow Region Overlap [ ]
Allow for Pins [ ]
Extend Boundary [ ] [● Horizontally] Vertically
Start from Lower Left [ ] X Offset 0 Y Offset 0

1. Select whether you want to generate *Rows*, *Boundary Cells*, and *Tap Cells*.
2. Select a placement *Region*. You can either select *Boundary* and select an existing boundary (default is PR Boundary) or select *Placement Region* to define a custom placement region.

   When *Boundary* is set to `rowRegion`, the row template selected on the *Row* tab is considered. A row region,as defined in the row template is created. If the row template has boundary cell and tap cell definitions, then the settings in the *Boundary Cell* and *Tap Cell* tabs are ignored.

   With *Placement Region* selected, you can define a custom placement region by one of the following methods:

   - By defining a rectangular region:

     a. Select *Rectangle*.

     b. Type the X and Y coordinates of the lower-left corner of the region in the *Origin X* and *Origin Y* fields. Notice the X and Y coordinates of the cursor position in the status line at the top of the layout window. If you do not type any coordinates, the new row is placed at the lower-left corner of the region.

     c. Type the *Width* and *Height* dimensions for the region.

     d. Click *Update*.

   - By defining a rectilinear region:

     a. Specify the coordinates in the *Points* text box.

     b. Click *Update*.

     Note: Use *Delete* to remove an existing region.

   - By drawing in the canvas (manually)

     a. Select *Placement Region*.

     b. Click *Draw.*

     c. Specify the placement region manually in the layout window. You are prompted to point at the first corner of the region's rectangle.

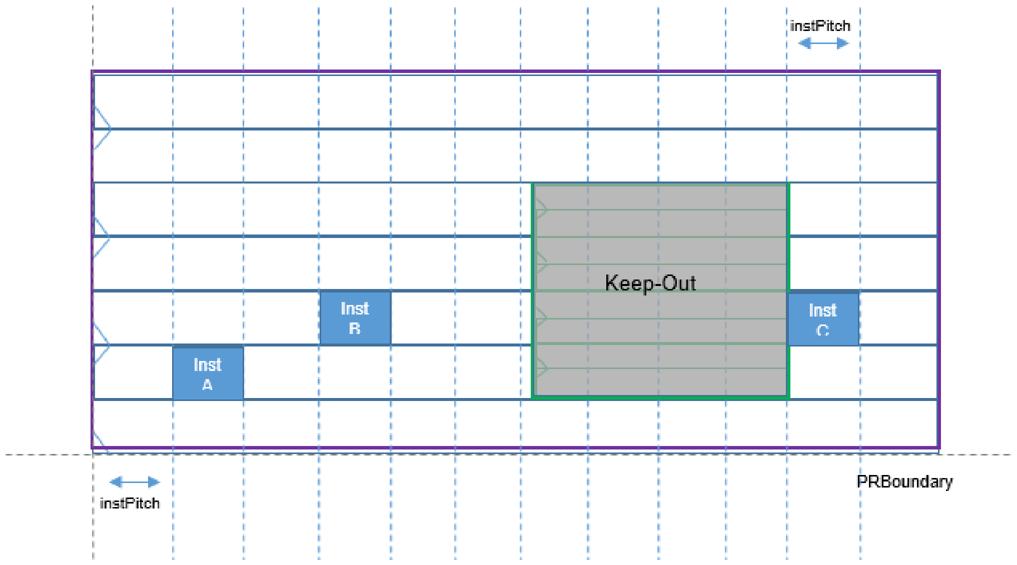
     Note: Rectilinear row regions in the U-shape and concave polygons are not supported.

3. Expand the *Options* section.

4. (IC23.1 EXL and Higher Tiers) Select *Allow Region Overlap* to allow the presence of overlapping row regions in the placement region. Existing row region is maintained and a new row region is generated over it. You can use this option to create partial overlapping row regions.

   When running the placer, placement of instances depends on the selected placement region as follows:

   - Row region: Instances are placed only in the selected row region. All other overlapping row regions are considered as blockages.
   - PR boundary: Instances are placed on all overlapping row regions that exist within the PR boundary. Instances are placed in row regions according to their allowed component types. When an instance is placed in the outer row region, the inner row region is considered blockage. In case of partial overlap between two row regions, nothing is placed in the overlapping area.



5. Select *Allow for Pins* to reserve space for pins. The budgeted pin space equals the size of the pin plus the spacing rule inside the boundary edge, which helps avoid pin overlaps. This option is available only when *Region* is set to *Boundary*.

6. Choose *Extend Boundary* to expand the row and the PR boundary beyond the specified region, wherever needed. Also specify whether rows are to be stretched *Horizontally* or *Vertically*. This option is available only when *Region* is set to *Boundary*.

7. Select *Start from Lower Left* to create rows from the lower-left side of the PR boundary or region. Specify the corresponding row *X Offset X* and *Y Offset* values.

   Note: This option is not available when you select a *Row Template* on the *Row* tab.

8. Click *Insert* to generate the specified boundary cells, tap cells, and rows in the layout canvas.

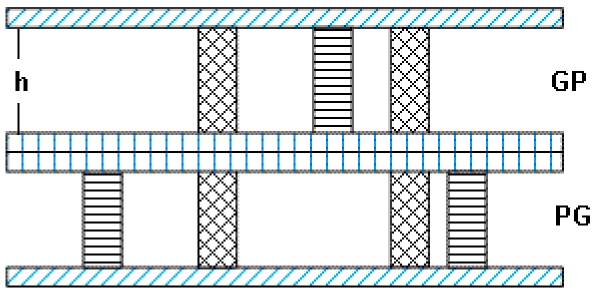
   Note: Click *Delete* to remove boundary cells, tap cells, and rows from the layout canvas.

Planning Placement of Multi-Height Cells

The Virtuoso Custom Placement solution supports the placement of multi-height standard cells. Multi-height cells are standard cells that are double, triple, or other multiples of the height of a single-height standard cell. This implies that during row estimation, if the placer encounters double-height (2x), triple height (3x), or any multi-height (Hx) cells in the design, it still creates rows with the height of a standard, single-height cell (x). But, for placing these multi-height cells, the placer budgets more space.

In a design with double-height cells, each row is half the height (x) of a double-height (2x) cell. Therefore, the double-height cells along with the specified rail pattern occupy two rows each.

Both PGGP and GPPG rail patterns are supported and these patterns can co-exist in a design to ensure uninterrupted power and ground connectivity to the cells, as shown in the figure below.



If the design had cells that were triple or any other integer multiple of the height of a standard cell, the placer would budget for them similarly, by considering their height as an integer multiple of the standard single height cell. However, the placer creates single-height rows for a multi-height design only if all the cells in the design have heights that are integer multiples of the single-height cell.

For example, if the cell heights are 3.0, 6.0, 9.0, and 12.0; the placer creates rows of height `3.0` because that is the height of the single-height cell.

If the height of even a single cell in the design is not an integer multiple of the single-height cell, the placer creates rows that are the height of the largest cell. For example, if the cell heights are 3.0, 4.0, 6.0, 9.0, and 12.0; the placer creates rows of height `12.0`.

Note:

- To ensure that multi-height cells are accommodated within the rows and the cells are not placed outside, the placer automatically sets the row spacing to `0`.

- To provide optimal placement results with designs that carry multi-height cells, the placer might perform several iterations before generating the final placement. Therefore, designs that have multi-height cells might take longer to place.

If you notice that in spite of running several iterations, the placement results are not as desired, you can choose to run the placer again.

Note: In the mixed mode, the Placement Planner creates only single-height rows, irrespective of whether the design has single, double, or any *n*-height cells, where *n* is a valid integer.

Planning Placement of Standard Cells

If standard cells are placed using the regular placement method, they can get placed too close together, resulting in spacing violations. The litho and stress effects due to neighboring layout cells at lower nodes can further affect cell delay and leakage. Therefore, placement planning should pre-empt possible placement challenges and aim for context-aware row-based placement.

To achieve context-aware placement, VCP honors the cell-to-cell boundary conditions called cell-edge characterizations. The cell-edge characterization defines that a specific *edge type,* such as *edge type 1 or 2*, that is assigned to the *LEFT* or *RIGHT* edge of a standard cell be placed with a specific edge type, such as 2 or 3, that is assigned to the *LEFT* or *RIGHT* edge of another standard cell such that the cells be placed adjacent to each other, within acceptable spacing values as defined using the `LEF58_CELLEDGESPACINGTABLE` property.

To define cell-edge characterizations, the custom placer uses the following LEF 5.8 edge properties defined in the standard cell masters:

- `LEF58_EDGETYPE`
- `LEF58_CELLEDGESPACINGTABLE`

Important

The LEF5.8 edge properties must be appropriately set in the technology LEF and the Macro LEF. Else, you must provide an XML file to specify the edge type and the edge spacing table for the standard cells to be abutted.

`LEF58_EDGETYPE`

Use the `LEF58_EDGETYPE` property to assign edge types for standard cell boundaries. For example:

```
EDGETYPE {RIGHT | LEFT | TOP | BOTTOM}
```

The acceptable spacing between the edges is defined in the `LEF58_CELLEDGESPACINGTABLE` library definition by referring to the above edge types.

Note: These properties can be specified either on the current cellview or in the `techLib`. The property directly on the cellview is given preference.

For information about the `LEF58_EDGETYPE` property, see the Edge Type Rule in the LEF/ DEF Language Reference.

`LEF58_CELLEDGESPACINGTABLE`

`LEF58_CELLEDGESPACINGTABLE` is a spacing rule table that defines the minimum acceptable spacing that must be maintained between two abutting edges of different edge types, `EDGETYPE1` and `EDGETYPE2`.

An example of the edge spacing table is given below:

Edge Type 1	Edge Type 2	Spacing (in micron)
1	1	1.0
1	2	0.5
1	3	2.7
2	3	1.3

To achieve context-aware placement, the adjacent edges of standard cells should be placed such that they meet the spacing constraints defined in the table above.

Note: If no spacing constraint is defined between two edge types, the edges are considered abuttable.

Virtuoso Custom Digital Placer honors the `LEF58_CELLEDGESPACINGTABLE` property. However, the following properties are partially honored. The following table describes VCP behavior when these properties are set:

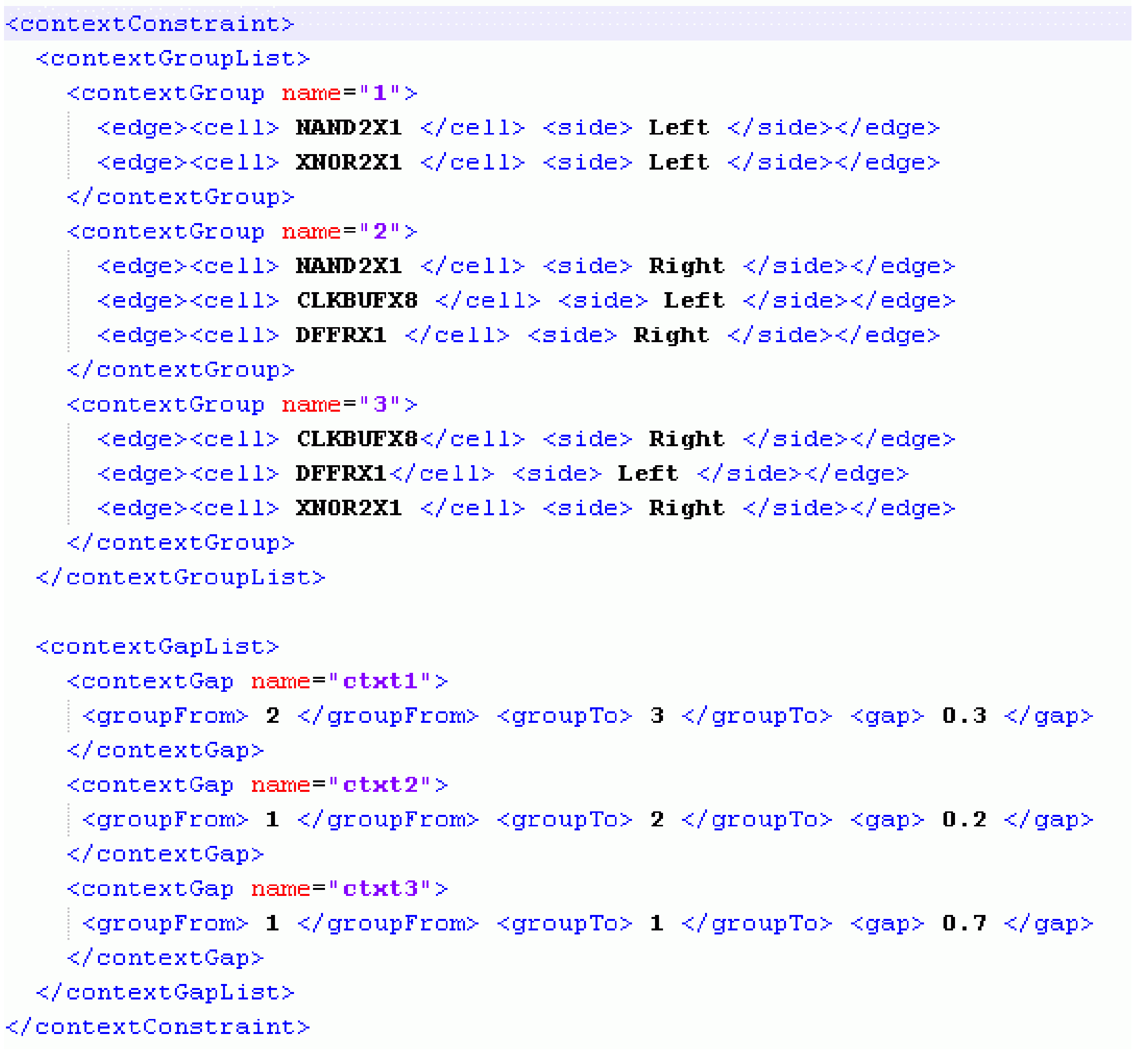
Property	VCP Behavior
`NODEFAULT`	Works as desired
`EXCEPTABUTTED`	Specifies that two abutted macros of the given edge types are allowed
`OPTIONAL`	Considers these groups only if the `considerOptionalEdges` environment variable is set to `t`
`SOFT`	Ignores these groups

For more information about `LEF58_CELLEDGESPACINGTABLE`, see the Cell Edge Spacing Table Rule in the LEF/DEF Language Reference.

Context-Aware Placement

To support context-aware placement planning of standard cells, Virtuoso Custom Placement Solution provides a Context Aware Placement option on the Component tab of the Placement Planning form. The option can also be controlled using the contextAwarePlacement environment variable.

If placement planning needs to be context-aware, define the cell edge and spacing constraints in the technology LEF and the macro LEF. Also, select the Context Aware Placement option on the Placement Planning form. Alternatively, if the constraint information is not defined in the LEF files, provide an XML file that contains the required constraint information. A sample XML file carrying the context information is displayed below.



```
<contextConstraint>
  <contextGroupList>
    <contextGroup name="1">
      <edge><cell> NAND2X1 </cell> <side> Left </side></edge>
      <edge><cell> XNOR2X1 </cell> <side> Left </side></edge>
    </contextGroup>
    <contextGroup name="2">
      <edge><cell> NAND2X1 </cell> <side> Right </side></edge>
      <edge><cell> CLKBUFX8 </cell> <side> Left </side></edge>
      <edge><cell> DFFRX1 </cell> <side> Right </side></edge>
    </contextGroup>
    <contextGroup name="3">
      <edge><cell> CLKBUFX8</cell> <side> Right </side></edge>
      <edge><cell> DFFRX1</cell> <side> Left </side></edge>
      <edge><cell> XNOR2X1 </cell> <side> Right </side></edge>
    </contextGroup>
  </contextGroupList>

  <contextGapList>
    <contextGap name="ctxt1">
     <groupFrom> 2 </groupFrom> <groupTo> 3 </groupTo> <gap> 0.3 </gap>
    </contextGap>
    <contextGap name="ctxt2">
     <groupFrom> 1 </groupFrom> <groupTo> 2 </groupTo> <gap> 0.2 </gap>
    </contextGap>
    <contextGap name="ctxt3">
     <groupFrom> 1 </groupFrom> <groupTo> 1 </groupTo> <gap> 0.7 </gap>
    </contextGap>
  </contextGapList>
</contextConstraint>
```

Note: The XML file can also be provided by using the contextConstraintFile environment variable.

If the XML file is unavailable or an invalid file name is specified, the placer looks for the required information in the technology LEF and the Macro LEF.

Tip

If the constraint information is not available in the LEF file, a warning is issued and context-aware planning fails.

The placer performs placement planning in accordance with the specified `LEF58_EDGETYPE` and `LEF58_CELLEDGESPACINGTABLE` properties, budgeting extra rows to accommodate the context-aware standard cells.

The placer looks for the `LEF58_EDGETYPE` property on the cell masters that are instantiated in the current cellview. If not found, the placer looks for the property in the abstract views of the cell masters.

The placer looks for `LEF58_CELLEDGESPACINGTABLE` property in the following order:

- In the user-specified library, using the `cdsEnv ("layoutXL.placement", "cellEdgeSpacingTableLib")` command.
- In the current cellview.
- In the current design library.
- In the attached technology library and all the referenced technology libraries.
- In all the reference libraries (of the master cellviews).
- In the technology file of the attached technology library.

Assigning Edges for Boundary Cells

The edge assignments for boundary cells types are done in the Configure Physical Hierarchy (CPH) window - *Component Types* mode. Depending on these assignments, a filtered list of cells is displayed in the various edge drop-down lists on the *Defaults* tab.

Example: In the CPH window, the *Edge type* for a set of boundary cells (with their *Component class* `BOUNDARYCELL`) is set to *Left* and *Right*. This setting is automatically propagated to the Boundary Cells Placement form. When the corresponding *Component Type* is selected, the boundary cells are listed only in the *Left Edge* and *Right Edge* drop-down lists.

Valid edge assignments in CPH window

Boundary Cell Placement Form - Edges	**CPH Window - Valid Edge Assignments (Case insensitive)**
Left Edge	`left, leftedge`
Right Edge	`right, rightedge`
Left Top Corner	`lefttopcorner, ltc`
Left Bottom Corner	`leftbottomcorner, leftbotcorner, lbc`
Right Top Corner	`righttopcorner, rbc`
Right Bottom Corner	`rightbottomcorner, rightbotcorner, rbc`
Top	`top, topedge`
Bottom	`bottom, bot, bottomedge, botedge`
Top Tap	`toptap`
Bottom Tap	`bottomtap, bottap`

4

Running the Placer

This chapter describes the Virtuoso® Custom Digital Placer's placement capability.

This chapter covers the following topics:

You launch the Virtuoso® Custom Digital Placer from the Auto Placer form. The placer can run in foreground, background, or Load Balancing Service (LBS) mode, reporting its status in the Placement Status window.

Important

While the placer is running on a cellview, the cellview is temporarily switched to Read mode and you cannot perform any other edits on it. When the placer has finished, the cellview is switched back to Edit mode.

After the placer finishes, the placed cellview appears if you are updating the source view with the placed view, or if you are creating a new view and chose the *Open Window* option in the Auto Placer form.

After placement, you can iteratively

- Lock the position of components whose placement is good.
- Refine the constraints on other components to better guide the placer.
- Rerun the placer as needed to improve the placement results.

Prerequisites to Placement

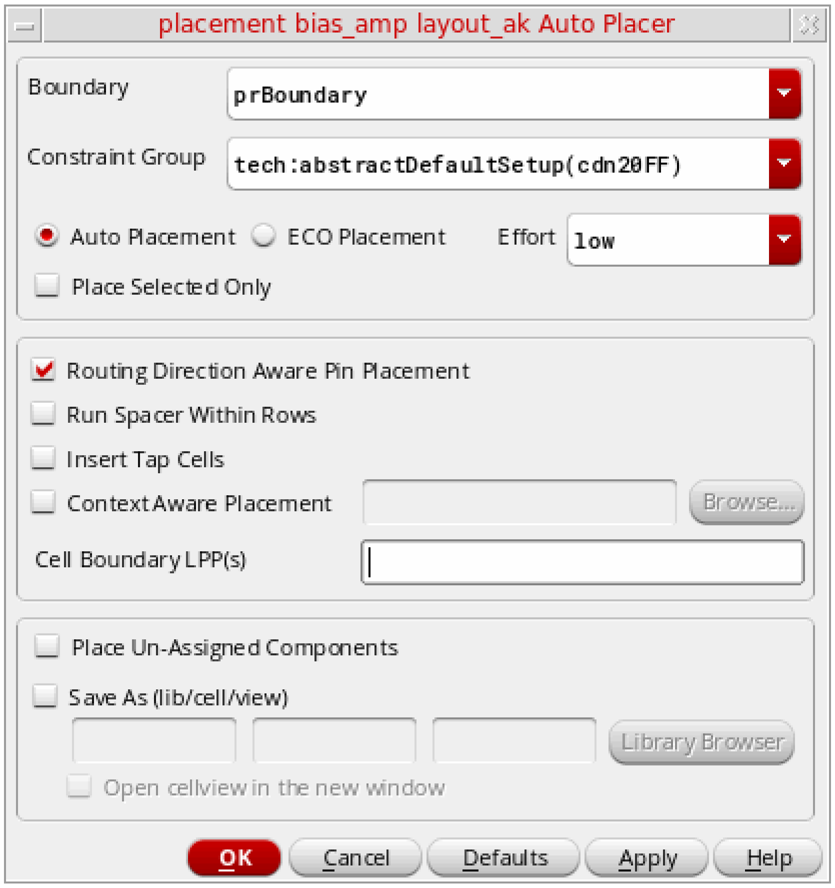
To prepare for placement:

1. Set the required environment variables before launching the Cadence software. For more information, see Setting Up Layout XL for Placement.
2. If you want to place only some of the components, select the components you want to place in the layout window, and check the *Place Selected Only* option.

Running the Placer

To perform placement of the selected components:

1. Choose *Place – Custom Digital – Placer*. The Auto Placer form is displayed.



placement bias_amp layout_ak Auto Placer

Boundary: prBoundary

Constraint Group: tech:abstractDefaultSetup(cdn20FF)

[✓ Auto Placement] ECO Placement   Effort: low

☐ Place Selected Only

☑ Routing Direction Aware Pin Placement

☐ Run Spacer Within Rows

☐ Insert Tap Cells

☐ Context Aware Placement

Cell Boundary LPP(s):

☐ Place Un-Assigned Components

☐ Save As (lib/cell/view)

☐ Open cellview in the new window

2. Select the placement options that you want to specify.

3. Select a suitable option from the *Effort* drop-down list. The run time of the placer and quality of results differ based on the effort chosen. The selected *Effort* level determines the number of passes of the placer. An increased number passes results in better (optimized) placement results. For example, the *High* effort setting generates better placement results than *Low*. However the run time of the placer differs based on the effort. For example, the placer takes more time to run when *Effort* set to *High* than when set to *Low*.

4. To save the placed cellview with a different view name,

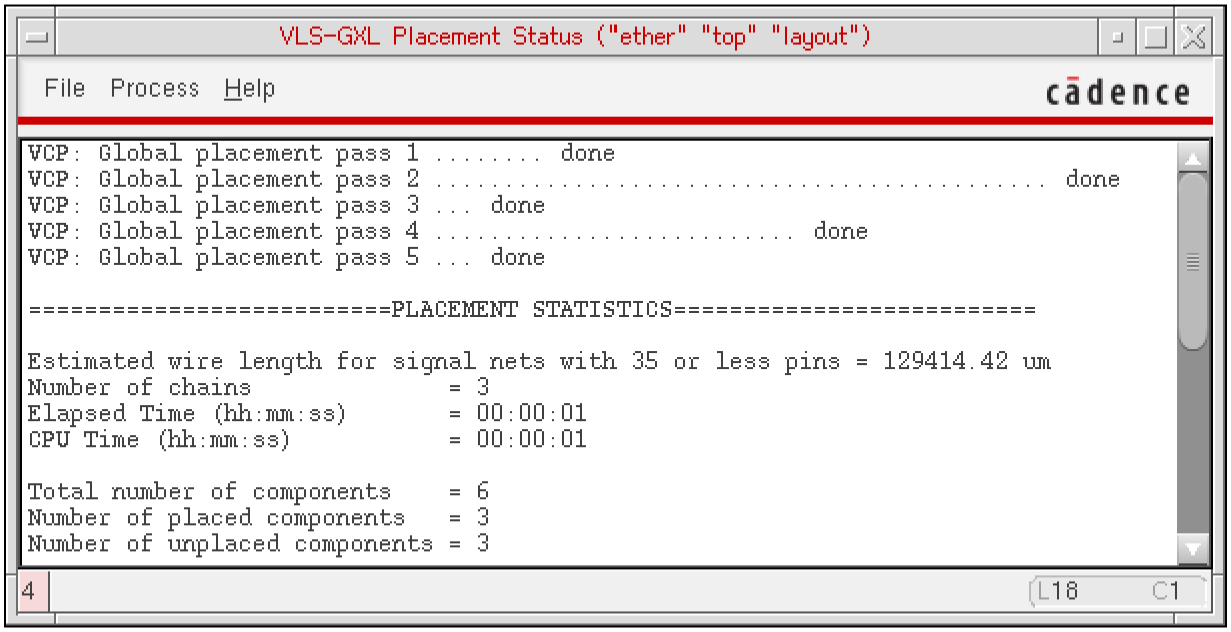
a. Check the *Save As* option. Type in the *Library*, *Cell*, and *View* names for the new cellview. By default the view name is *layout.plc*. You can change the library or cell name, or browse to choose the cellview name.

Note: Do not use the view name `placerScratch`. This name is reserved by the placer.

b. Check the *Open cellview in the new window* option to open the placed cellview in a new session window.

5. Click *OK* or *Apply*.

The placer initializes and the Placement Status window appears.



```
VCP: Global placement pass 1 ........ done
VCP: Global placement pass 2 ............................................. done
VCP: Global placement pass 3 ... done
VCP: Global placement pass 4 .......................... done
VCP: Global placement pass 5 ... done

==========================PLACEMENT STATISTICS=========================

Estimated wire length for signal nets with 35 or less pins = 129414.42 um
Number of chains               = 3
Elapsed Time (hh:mm:ss)        = 00:00:01
CPU Time (hh:mm:ss)            = 00:00:01

Total number of components     = 6
Number of placed components    = 3
Number of unplaced components  = 3
```

You can specify the level of detail that appears in the status window by setting the vcpVerboseLevel environmental variable. The text columns in the form do not align if a variable font is used. Use a fixed text font instead.

The placer moves pins that were not assigned to placement information inside the design boundary and aligns to any boundary edge.

The placer honors the `explicitColoredPurposes` environment variable. When set, only the shapes on the specified purposes are considered by the placer for snapping instances. When this environment variable is not set, the tool honors the colored purposes defined in the technology file.

Important

You can use the *Edit – Undo* command to discard the placement results.

Tip

At any point, press the `Ctrl` key and then press `C` to stop the placer.

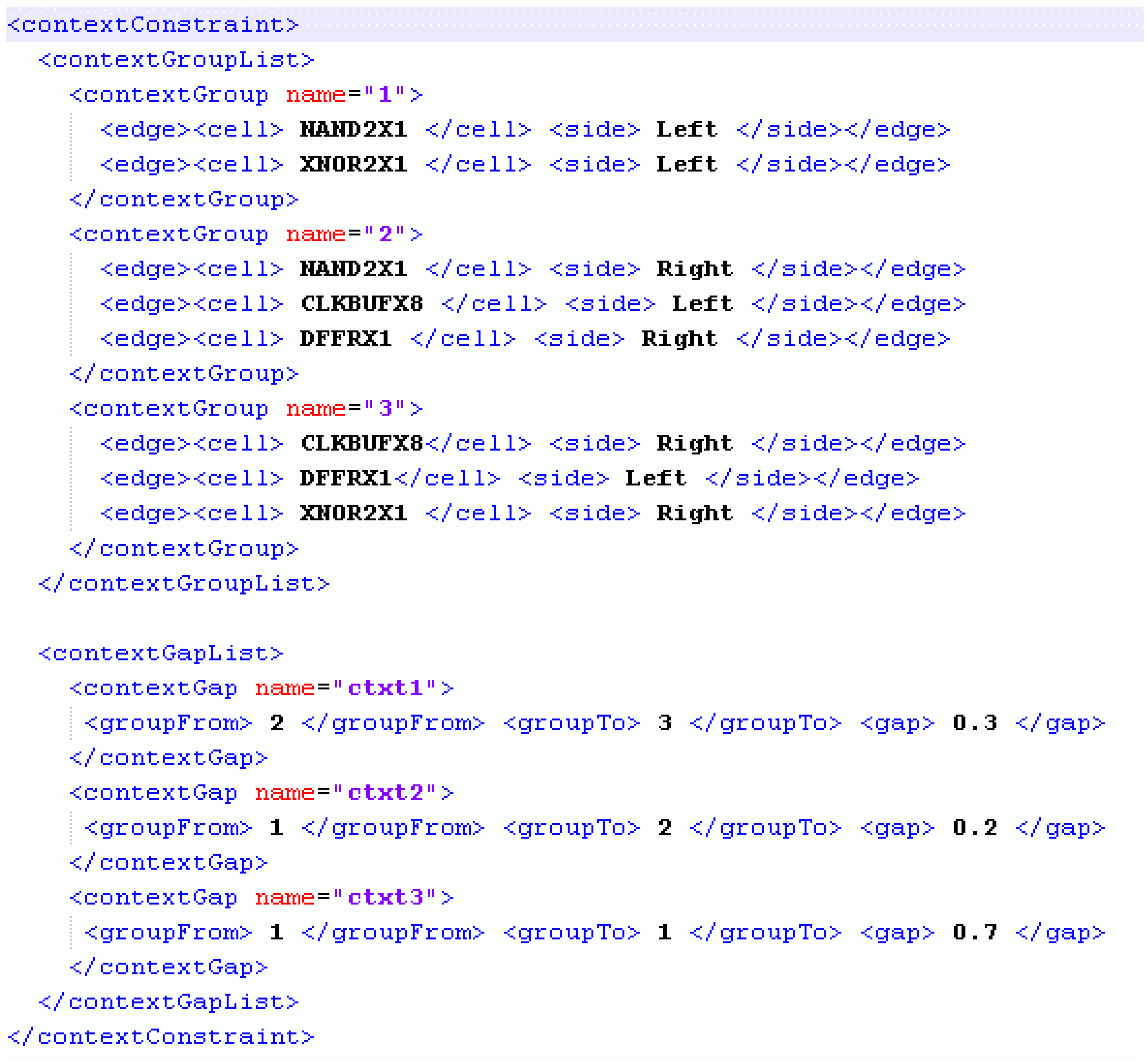
Context-Aware Placement

At advanced nodes, the key placement goal is to achieve maximum compaction. But achieving maximum compaction at such small geometries poses its own set of challenges, such as litho and stress effects. To overcome such issues and to achieve context-aware placement at small nodes, the Virtuoso Custom Digital Placer honors the `LEF58_EDGETYPE` and `LEF58_CELLEDGESPACINGTABLE` LEF5.8 constraints. For more information, see Planning Placement of Standard Cells.

To enable context-aware placement, the Auto Placer form provides a *Context Aware Placement* option, which can be controlled using the contextAwarePlacement environment variable. However, for the option to be available in the form, the advanced node *VISNG200* license must be available.

For the standard-cell placement to be context-aware, the cell edge and spacing constraints should be defined in the technology LEF and the macro LEF and the *Context Aware Placement* option on the Auto Placer form should be selected. Alternatively, if the constraint information is not defined in the LEF files, an XML file, which contains the required constraint information, should be provided.

A sample of the XML file carrying the context information is shown below.



```xml
<contextConstraint>
  <contextGroupList>
    <contextGroup name="1">
      <edge><cell> NAND2X1 </cell> <side> Left </side></edge>
      <edge><cell> XNOR2X1 </cell> <side> Left </side></edge>
    </contextGroup>
    <contextGroup name="2">
      <edge><cell> NAND2X1 </cell> <side> Right </side></edge>
      <edge><cell> CLKBUFX8 </cell> <side> Left </side></edge>
      <edge><cell> DFFRX1 </cell> <side> Right </side></edge>
    </contextGroup>
    <contextGroup name="3">
      <edge><cell> CLKBUFX8</cell> <side> Right </side></edge>
      <edge><cell> DFFRX1</cell> <side> Left </side></edge>
      <edge><cell> XNOR2X1 </cell> <side> Right </side></edge>
    </contextGroup>
  </contextGroupList>

  <contextGapList>
    <contextGap name="ctxt1">
     <groupFrom> 2 </groupFrom> <groupTo> 3 </groupTo> <gap> 0.3 </gap>
    </contextGap>
    <contextGap name="ctxt2">
     <groupFrom> 1 </groupFrom> <groupTo> 2 </groupTo> <gap> 0.2 </gap>
    </contextGap>
    <contextGap name="ctxt3">
     <groupFrom> 1 </groupFrom> <groupTo> 1 </groupTo> <gap> 0.7 </gap>
    </contextGap>
  </contextGapList>
</contextConstraint>
```

The XML file can also be provided by using the contextConstraintFile environment variable.

However, if the XML file specified through the form or by using the environment variable is invalid, or the file is not provided, the placer reads the technology and the macro LEF for relevant placement information and places the standard cells according to the specified spacing constraints. If the spacing information is also unavailable in the LEF files, the placer issues an error message indicating context-aware placement cannot be performed. For more information, see the Auto Placer form.

To achieve optimum compaction, the placer might run several iterations before generating the placement results. Although the placer honors the spacing constraints that are defined, if any spacing constraints are violated, the markers indicating the violations are displayed in the Misc tab of the Annotation Browser.

For information about providing the XML file using the Auto Placer form, see Auto Placer.

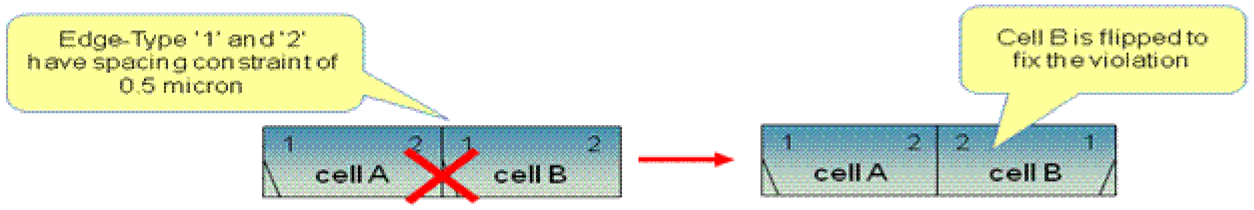
Resolving Spacing Violations During Placement

To achieve context-aware placement, the placer uses the edge type and acceptable spacing information available in the Edge Spacing Table.

Let us consider an example to see how the placer can use the cell edge spacing information, as given in the table below, to resolve any spacing violations and to generate optimal placement results.

Edge Type 1	Edge Type 2	Spacing (in micron)
1	1	1.0
1	2	0.5
1	3	2.7
2	3	1.3

Let us consider cells A and B, as given in the figure below, are placed such that edge type `2` of cell A is placed adjacent to edge type `1` of cell B. Since the edge types `1` and `2` have a spacing constraint of `0.5` micron, as defined in the cell edge spacing table above, a spacing violation is generated if the cells are placed adjacent to each other. To fix the violation, the placer might flip, move, or flip and move cell B such that edge type `2` of both the cells is placed without generating any spacing violations.


Edge-Type '1' and '2' have spacing constraint of 0.5 micron
1
2
cell A
1
2
cell B
Cell B is flipped to fix the violation
1
2
cell A
2
1
cell B

Placement of Multi-Voltage Cells

Standard cells can be characterized into different voltage (VT) cells based on the VT layer shape that they include. At advanced nodes, Virtuoso Custom Digital Placer supports placement of multi-voltage cells.

Note:

- ❑ To specify VT layers, use the multiVTLayers environment variable. The default value is `(VTS_N VTL_N VTUL_N)`.
- ❑ To enable placement of multi-voltage cells, set the multiVTPlacement environment variable to `t`.

During placement of multi-voltage cells, Virtuoso Custom Digital Placer honors the following multi-voltage placement rules:

- The same VT-type standard cells must maintain a minimum width that equals four times the poly pitch value. This value can be specified using the minVTLength environment variable. The default value is 0.`228`.
- For standard cells with different VT types that are placed next to each other, an appropriate gap should be maintained to ensure that the minimum width rule is followed.

Note: Standard cells with no voltage layers specifications are grouped internally and placed normally.

Support for Virtual Hierarchies in the Design Planner Environment

Virtuoso® Design Planner is a connectivity-based tool in Layout EXL and higher tiers that supports automatic placement and congestion analysis capabilities. In the Design Planner environment, Placement Planning lets you create rows in the PR boundary, cluster boundary, and virtual hierarchy boundary (area boundary).

For more information about virtual hierarchies, see *Virtuoso Design Planner User Guide*.

Running the Placer Using SKILL

You can also use the vcpfeRunCustomDigitalPlacer public SKILL function to run the custom digital placer without opening the graphical user interface.

5

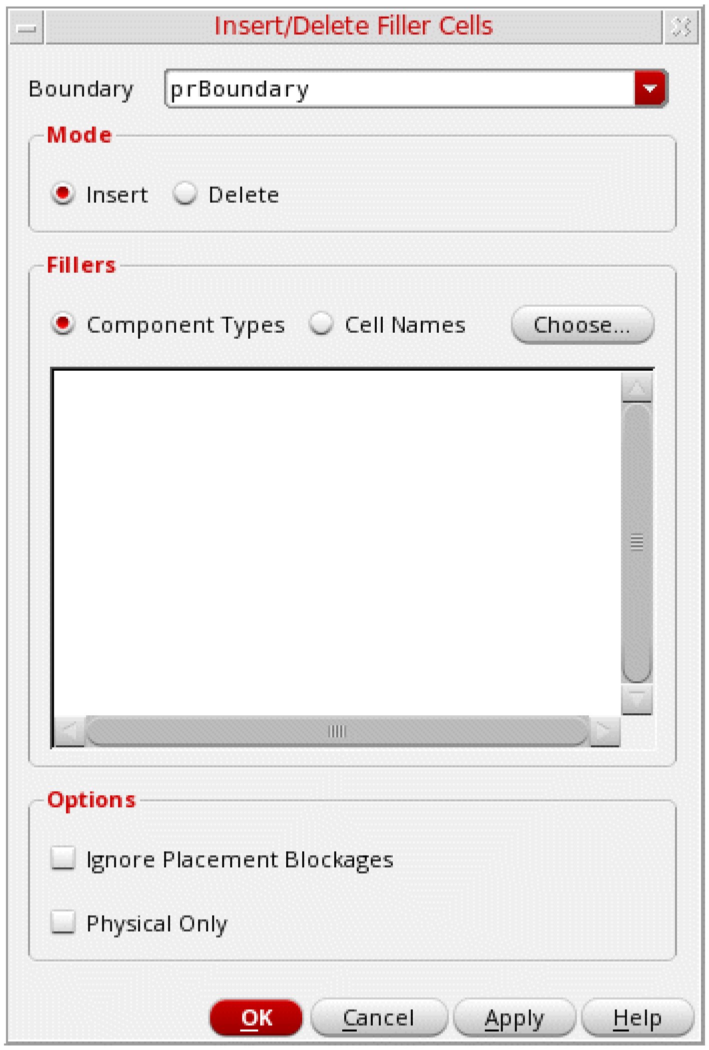
Finishing Tasks

This section describes the following tasks performed after running the Virtuoso Custom Digital Placer:

- Inserting and Deleting Filler Cells
- Swapping Tap Cells

Inserting and Deleting Filler Cells

From the layout window menu bar, choose *Place – Custom Digital – Insert/Delete Fillers* to display the Insert/Delete Filler Cells Form.



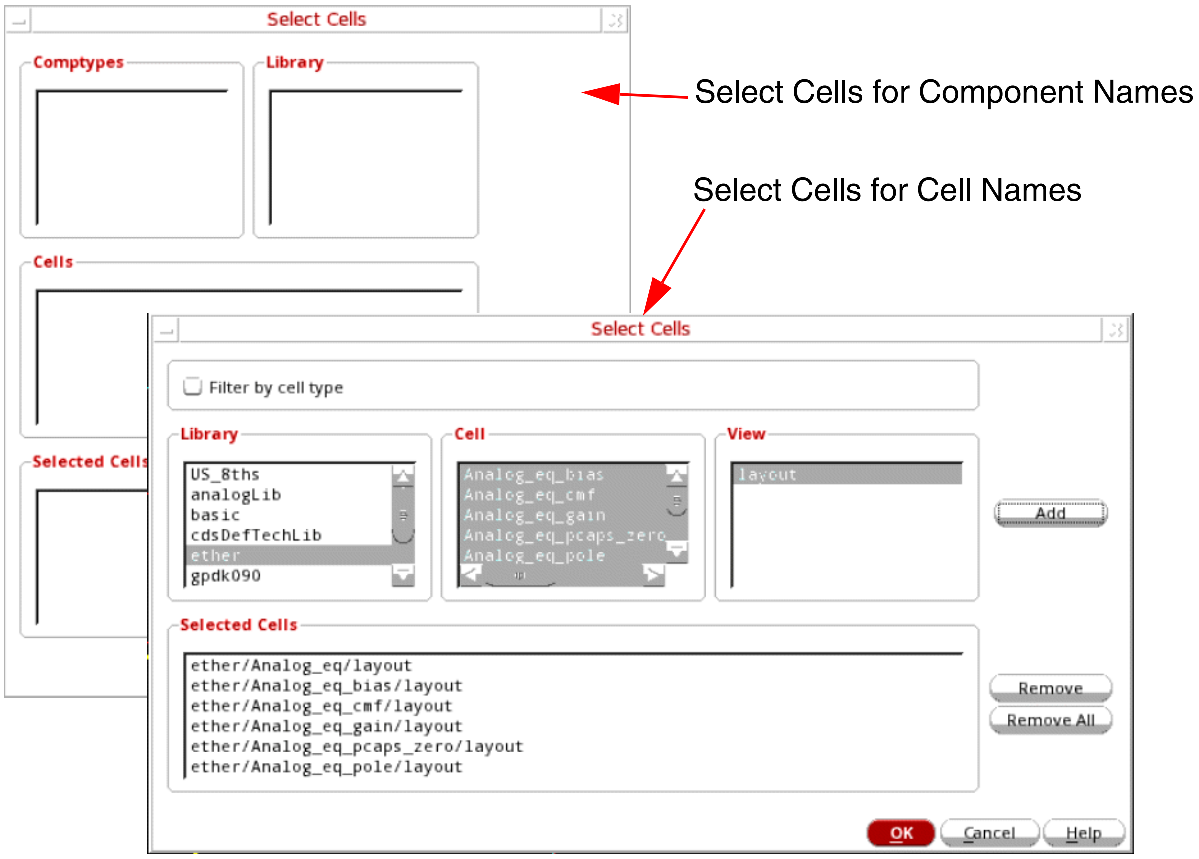
Use the options in the form to insert or delete filler cells. During insertion of filler cells, all the existing filler cells in the design are deleted, and new filler cells are inserted according to the specified component type/cell names. For example, filler cells of component types A and B are present in a design. If you specify that filler cells of component type B need to be inserted, then filler cells of both type A and B are deleted, and new filler cells of type B are inserted.

To insert filler cells:

1. Select the *Boundary* within which filler cells must be added. The default is prBoundary. You can specify any custom boundary that is defined in the design.

2. Select the *Insert* Mode.
3. Select whether you want to filter the filler cells based on the *Component Types* or *Cell Names*.
4. Click *Choose*. The Select Cells form is displayed.
   - ❑ With *Component Types* selected, the Select Cells form lets you select the required *Comptypes* and *Library*. Filler cells that are assigned to a component type with the component class *FILLER* are listed. For more information on component types, see Defining a Component Type and Defining a Standard Cell Substrate Contact.
   - ❑ With *Cell Names* selected, the Select Cells form lets you select the required *Library*, *Cell*, and *View* names. You can select multiple cells. *View* displays only the common layout views.

     In this mode, you can select *Filter by celltype* to restrict the *Cell* list to only those cells that belong to the coreSpacer *cellType*.



```
US_8ths
analogLib
basic
cdsDefTechLib
ether
gpdk090
```

```
Analog_eq_bias
Analog_eq_cmf
Analog_eq_gain
Analog_eq_pcaps_zero
Analog_eq_pole
```

```
layout
```

```
ether/Analog_eq/layout
ether/Analog_eq_bias/layout
ether/Analog_eq_cmf/layout
ether/Analog_eq_gain/layout
ether/Analog_eq_pcaps_zero/layout
ether/Analog_eq_pole/layout
```

5. Click *Add* to add the selections to the *Selected Cells* section.

Remove lets you delete individual cells from the list, whereas *Remove All* clears all entries from the list.

6. Click *OK* to return to the Insert/Delete Filler Cells form.
7. Select *Ignore Placement Blockages* in the *Options* section to ignore blockages during filler cell creation.
8. Select *Physical Only* to set the physOnly attribute for filler cells.
9. Click *OK* to insert filler cells.

Filler cells are added into the empty spaces between the standard cells in each row without impacting the row size.

The same options and procedure apply to the *Delete* mode. Filler cells are deleted from the empty spaces between standard cells in each row without impacting the row size. All filler cells of the specified component type or cell names are deleted from the design. For example, if the filler cells of component types A and B are present in a design, and you want to replace the filler cells of component type B with the new filler cells. In this case, if you do an Insert, then all filler cells are deleted and replaced. To retain the filler cells of component types A, first delete the existing filler cells of component type B, and then insert the new filler cells of component type B. For more information, see Insert/Delete Filler Cells Form.

Related Environment Variables

You can use environment variables to define the filler cell component types or cell names that are to be loaded when the form is invoked.

The following example populates cells based on the filler cell component types defined in the CPH form.

```
envSetVal("layoutXL.placement" "selectFillerCellsFrom" 'cyclic "Component Types")
envSetVal("layoutXL.placement" "fillerCompTypes"  'string "comptye1 comptype2")
```

The following example gets cells from a selected filler cell component type.

```
envSetVal("layoutXL.placement" "selectFillerCellsFrom" 'cyclic "Component Types")
envSetVal("layoutXL.placement" "insertFillerCells"  'string "comptype1")
```

The following example gets cells from a list of selected cells.

```
envSetVal("layoutXL.placement" "selectFillerCellsFrom" 'cyclic "Cell Names")
envSetVal("layoutXL.placement" "selectFillerCells" 'string "(mylib fillc21
layout)(mylib fillc41 layout)")
```

List of Environment Variables

fillerCompTypes

insertFillerCells

insertFillers

selectFillerCellsFrom

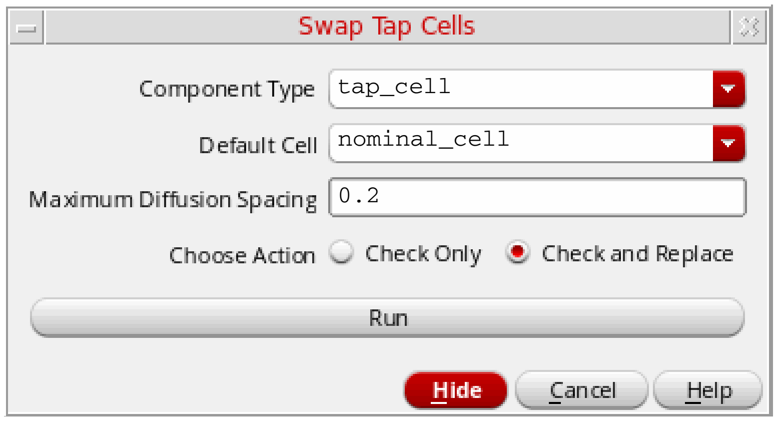
selectedFillerCells

Swapping Tap Cells

The Swap Tap Cells feature helps you to detect tap cells with maximum spacing violations. You can also replace them with tap cells that have valid maximum spacing values.

To run this command:

1. Select *Place – Swap Tap Cells* to view the Swap Tap Cells form.



Component Type: tap_cell
Default Cell: nominal_cell
Maximum Diffusion Spacing: 0.2
Choose Action: Check Only [✓ Check and Replace]

2. Specify the *Component Type* for which you need to perform the check.

Important

Ensure that the same component type is assigned to both, the existing tap cells and the tap cells to be swapped.

3. Select the *Default Cell* with which the tap cells in the designs need to be replaced.
4. Specify the permitted *Maximum Diffusion Spacing* for validation.
5. Select the action that needs to be performed:
    - ❑ Check Only: Checks and reports maximum diffusion spacing violations in the current design. All tap cells in the design are highlighted in white and the tap cells with violations are highlighted in yellow.

      Note: You can click *Hide* at any point to hide the form to review the violations.
    - ❑ Check and Replace: This is a two-step process. First, all tap cells in the design are replaced with the selected *Default Cell*. Next, all maximum diffusion spacing violations are detected and their tap cells are replaced with the alternate best-matched tap cells that are available in the design.
6. Click *Run*.

For information about inserting tap cells, see Defining a Standard Cell Substrate Contact.

6

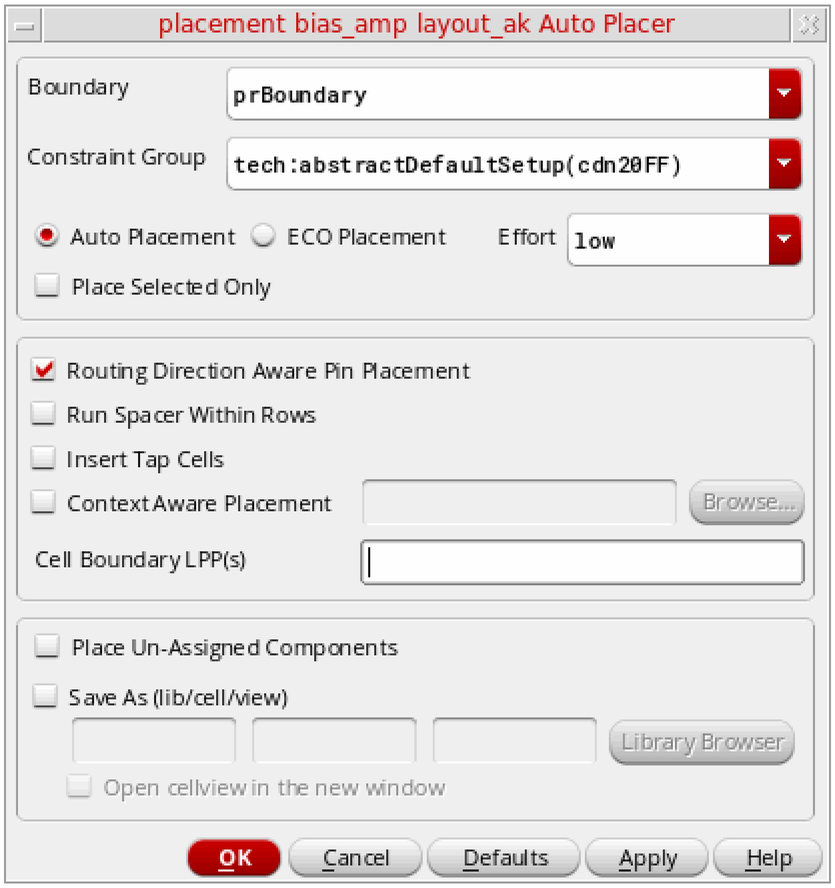
Custom Digital Placer Forms

This section lists the controls in the Virtuoso® custom digital placer forms.

Note: Many of the options described in this section have a corresponding environment variable. For more information, see Appendix 7, "Custom Digital Placer Environment Variables."

Auto Placer

Use the **Auto Placer** form to define the type of placement to be run, the technology rules to be used, any automatic spacing adjustments to be done, and how the modified cellview is saved.


placement bias_amp layout_ak Auto Placer

Boundary: prBoundary
Constraint Group: tech:abstractDefaultSetup(cdn20FF)
[✓ Auto Placement] ECO Placement   Effort: low
☐ Place Selected Only

☑ Routing Direction Aware Pin Placement
☐ Run Spacer Within Rows
☐ Insert Tap Cells
☐ Context Aware Placement
Cell Boundary LPP(s):

☐ Place Un-Assigned Components
☐ Save As (lib/cell/view)
☐ Open cellview in the new window

Boundary provides a list of available boundaries in the current design. Choose the boundary to be considered for placement.

Constraint Group lets you choose the constraint group containing the placement rules for the current technology.

Auto Placement places components without taking into consideration any initial placement. You would typically use this option the first time you place a design or to discard previous unsatisfactory placement results.

ECO Placement places all unplaced components based on their connectivity with the placed components.

Effort lets you choose from the *Low*, *Medium*, and *High* effort levels. The run time of the placer and quality of results differ based on the effort chosen. The time for each effort level increases in roughly linear increments.

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

Routing Direction Aware Pin Placement considers the routing direction of layers while placing unconstrained pins. For example, if M2 has a preferred routing direction as horizontal, then pins on M2 are placed on the left and right edges.

Run Spacer Within Rows distributes components evenly in the rows.

Insert Tap Cells inserts tap cells (standard cell substrate contacts) in the empty spaces between the standard cells in a row based on the specified minimum and maximum contact spacing value.

Environment Variable: insertSubstrateContacts

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 `LEF58_EDGETYPE` and `LEF58_CELLEDGESPACINGTABLE` LEF5.8 constraints that are either defined in the technology and macro LEF or provided using an XML file. Click *Browse* to select a context file, which is an XML file containing edge spacing constraint information for performing context-aware placement.

For more information, see Context-Aware Placement.

By default, the **Context Aware Placement** option is OFF.

Environment variables: contextAwarePlacement, contextConstraintFile

Cell Boundary LPP(s) specifies the layer-purpose pairs (LPPs) that the placer uses to derive the cell boundary for standard cells.

Environment variable: vcpCellBoundaryLPPs

Place Un-Assigned Components allows floating components to be moved when using the Assisted Mixed CMOS/Standard Cell placement mode.

Caution

Enabling this option can adversely affect the performance of the placer.

Save As lets you specify a new **Library**, **Cell**, and **View** name for the placed cellview. By default, the placed cellview is saved under the current cellview name. Click *Library Browser* to choose a different name.

Tip

The cellview is not saved automatically before placement is run. Therefore, to preserve any edits that are still in memory, be sure to save it to disk before you start.

Open cellview in the new window displays the placed view in a new window after placement if it is not already displayed. Otherwise, the system updates the output view in the existing session window.

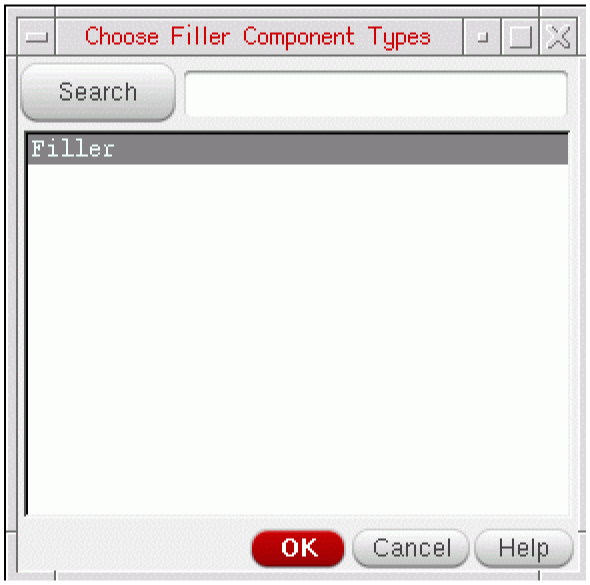
Related Topics

Custom Digital Placer Forms

Running the Placer

Choose Types

Use the **Choose Filler Component Types** form to choose the component type to which filler cells (or standard cell substrate contacts) are assigned.



Search scrolls the list to the component type name you type into the field. The name must match.

Related Topics

Custom Digital Placer Forms

Dressing Template Editor

Use the **Dressing Template Editor** to define the rails and allowed component types for a specific custom placement area. You access it by clicking the button with the same name on the Create Custom Placement Area Form or on the Placement Planning form.

The form is split into two tabs.

- Rail Definition Tab
- Component Type Set Tab

Related Topics

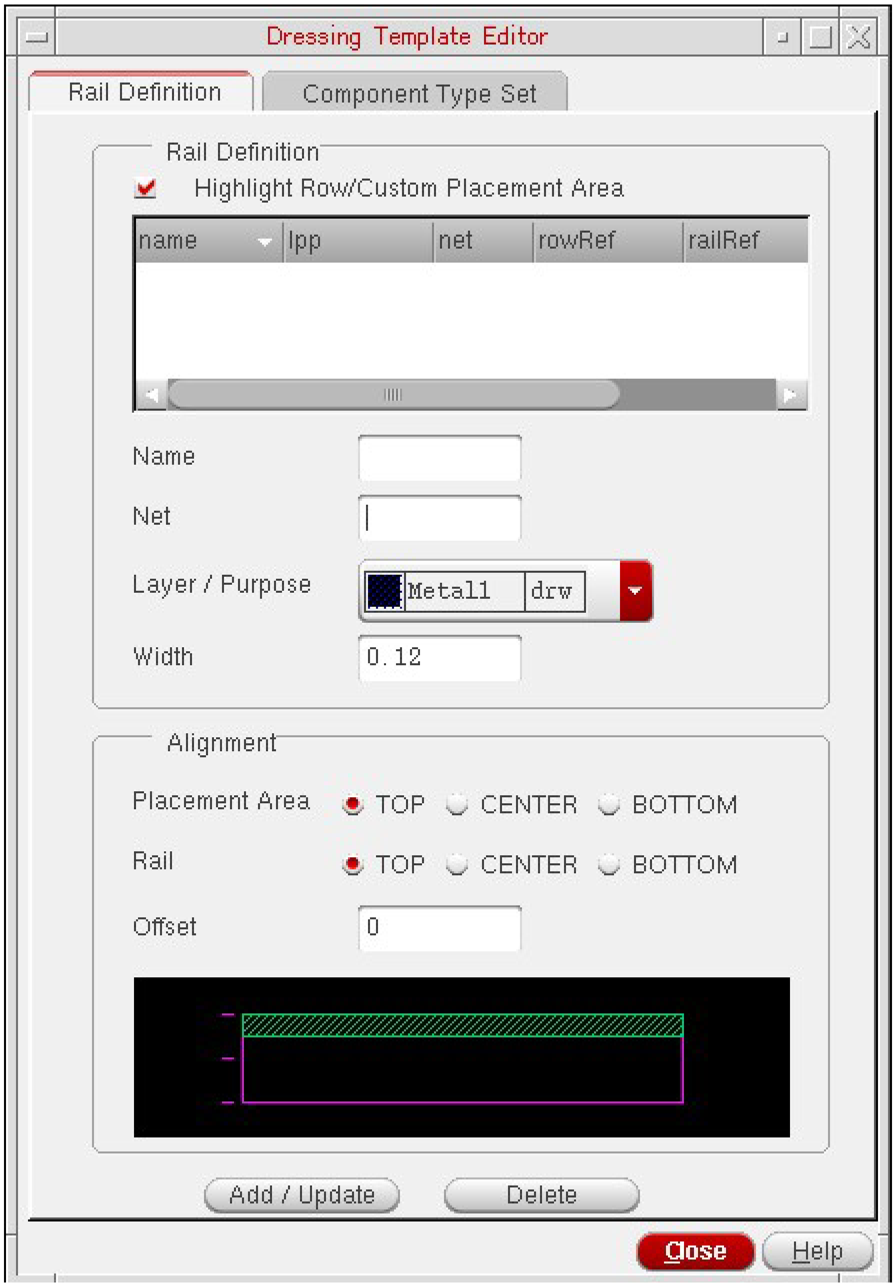
Creating a Custom Placement Area

Auto Placer

Custom Digital Placer Forms

Rail Definition Tab

Use the *Rail Definition* tab to define the rails for a custom placement area.



[✓] Highlight Row/Custom Placement Area

name	lpp	net	rowRef	railRef

Name	
Net	
Layer / Purpose	Metal1 drw
Width	0.12
Placement Area	[✓ TOP] CENTER BOTTOM
Rail	[✓ TOP] CENTER BOTTOM
Offset	0

Rail Definition lets you set the basic attributes of a rail; its *Name*, the *Net* to which it is attached, the *Layer/Purpose* pair on which it is drawn, and its *Width*. If you select the *Highlight Row/Custom Placement Area* check box and then select one or more rail definitions, then the corresponding rows that use the selected definitions are highlighted in the layout canvas. Alternatively, use environment variable highlightRowFromDTE.

Alignment lets you specify the position of the rail in a custom placement area. The picture at the bottom of the form lets you visualize the selected alignment.

> **Placement Area** specifies whether the rail is to be aligned with the *TOP*, *CENTER*, or *BOTTOM* of the custom placement area.
>
> **Rail** specifies which part of the rail is used as the reference point for the alignment; its *TOP* edge, *CENTER* line, or *BOTTOM* edge.
>
> **Offset** defines an offset from the specified alignment. This lets you leave a channel around the edge of the custom placement area, or align the rail slightly off-center. Both, positive and negative offset values are honored.

When you are done, click *Add/Update* to add to add the rail definition to the list at the top of the form and to the *Predefined* list in the Create Custom Placement Area form.

Use *Delete* to remove a selected rail definition from the list.

Related Topics

Component Type Set Tab

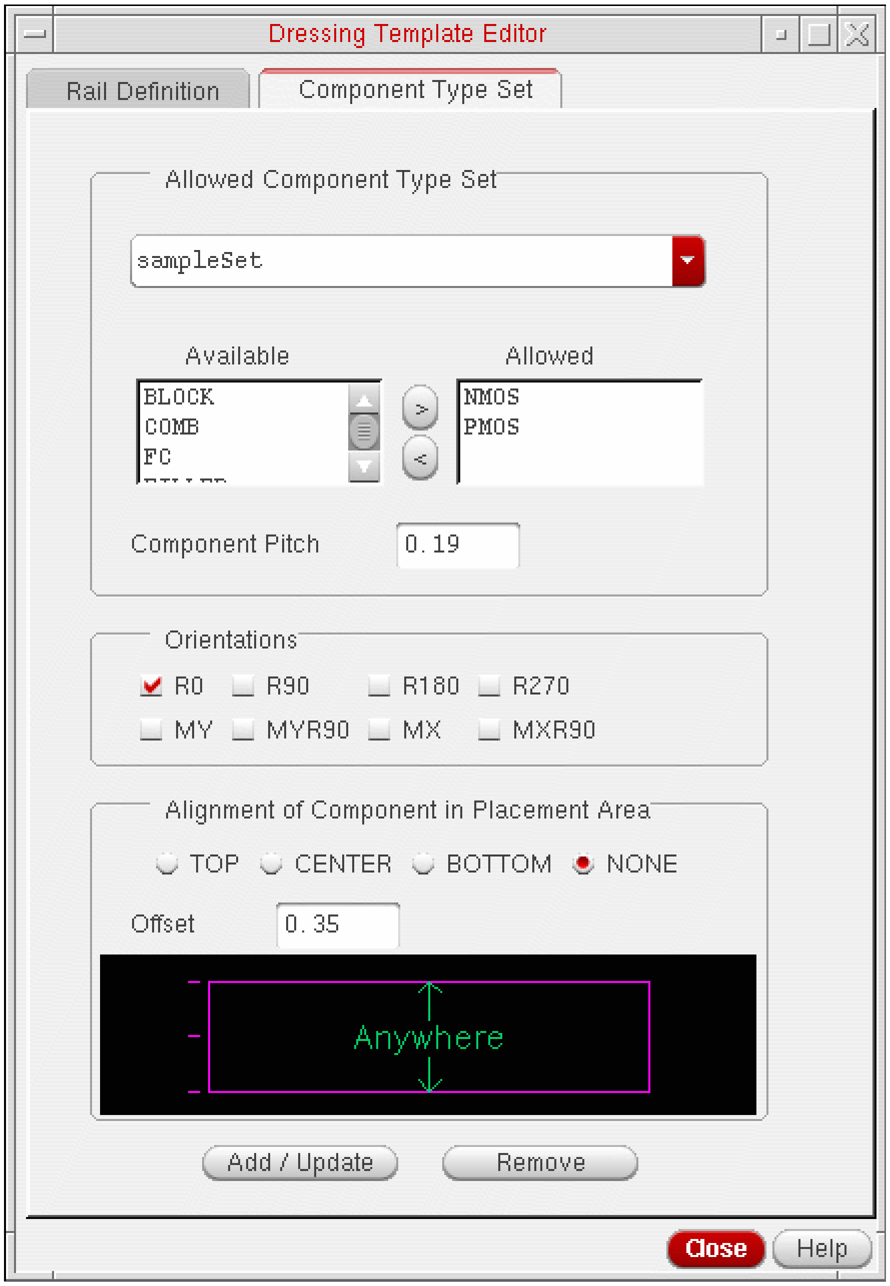
Creating a Custom Placement Area

Auto Placer

Custom Digital Placer Forms

Component Type Set Tab

Use this *Component Type Set* tab to specify the component types that can be placed in a custom placement area.


Dressing Template Editor

Rail Definition | Component Type Set

Allowed Component Type Set: sampleSet

Available: BLOCK, COMB, FC

Allowed: NMOS, PMOS

Component Pitch: 0.19

Orientations: [✓ R0] R90 R180 R270 MY MYR90 MX MXR90

Alignment of Component in Placement Area: TOP CENTER BOTTOM [✓ NONE]

Offset: 0.35

Allowed Component Type Set lets you create a component type set and specify which of the component types defined in your design can be placed in it.

The **combo-box** at the top of the form lets you define the name for the component type set.

Available lists all the component types defined in the current design. Use the right arrow button to move a selected component type into the *Allowed* list.

Allowed lists the component types allowed in the current component type set. Use the left arrow button to move a selected component type into the *Available* list.

Component Pitch specifies the minimum spacing between individual components placed in the custom placement area.

Orientations specifies how components can be flipped and rotated by the placer's *Allow Rotation* option. Check the orientations you want to allow. Orientations specified here are relative to the custom placement area.

Note: To prevent components from being rotated or flipped relative to the custom placement area, check only *R0*. Components will automatically follow the orientation of the custom placement area.

`R0`	Components can be placed in the same orientation as the custom placement area.
`R90`	Components can be rotated 90 degrees clockwise relative to the custom placement area.
`R180`	Components can be rotated 180 degrees clockwise relative to the custom placement area.
`R270`	Components can be rotated 270 degrees clockwise relative to the custom placement area.
`MY`	Components can be mirrored over the Y axis relative to the custom placement area.
`MYR90`	Components can be mirrored over the Y axis and rotated 90 degrees clockwise relative to the custom placement area.
`MX`	Components can be mirrored over the X axis relative to the custom placement area.
`MXR90`	Components can be mirrored over the X axis and rotated by 90 degrees clockwise relative to the custom placement area.

Alignment of Component in Placement Area specifies how components are aligned in the custom placement area; *TOP*, *CENTER*, *BOTTOM*, or *NONE*. The picture at the bottom of the form lets you visualize the selected alignment.

Offset defines an offset from the specified alignment. This lets you leave a channel around the edge of the custom placement area, or align the components slightly off-center.

When you are done, click the *Add/Update* button to add the component type set to the *Predefined* list in the Create Custom Placement Area form. Use *Remove* to remove a selected component type set from the list.

Related Topics

Rail Definition Tab

Creating a Custom Placement Area

Auto Placer

Custom Digital Placer Forms

Pin Placement

Use the **Pin Placement** form to:

- Specify pin location constraints and to pre-place pins independently from the main placement run. See Pin Planner for more information.
- Optimize your pins to achieve the shortest possible net lengths at a particular level. See "Pin Optimization" in *Virtuoso Floorplanner User Guide*.

The Pin Placement form is available from both the *Place* menu and the *Floorplan* menu.

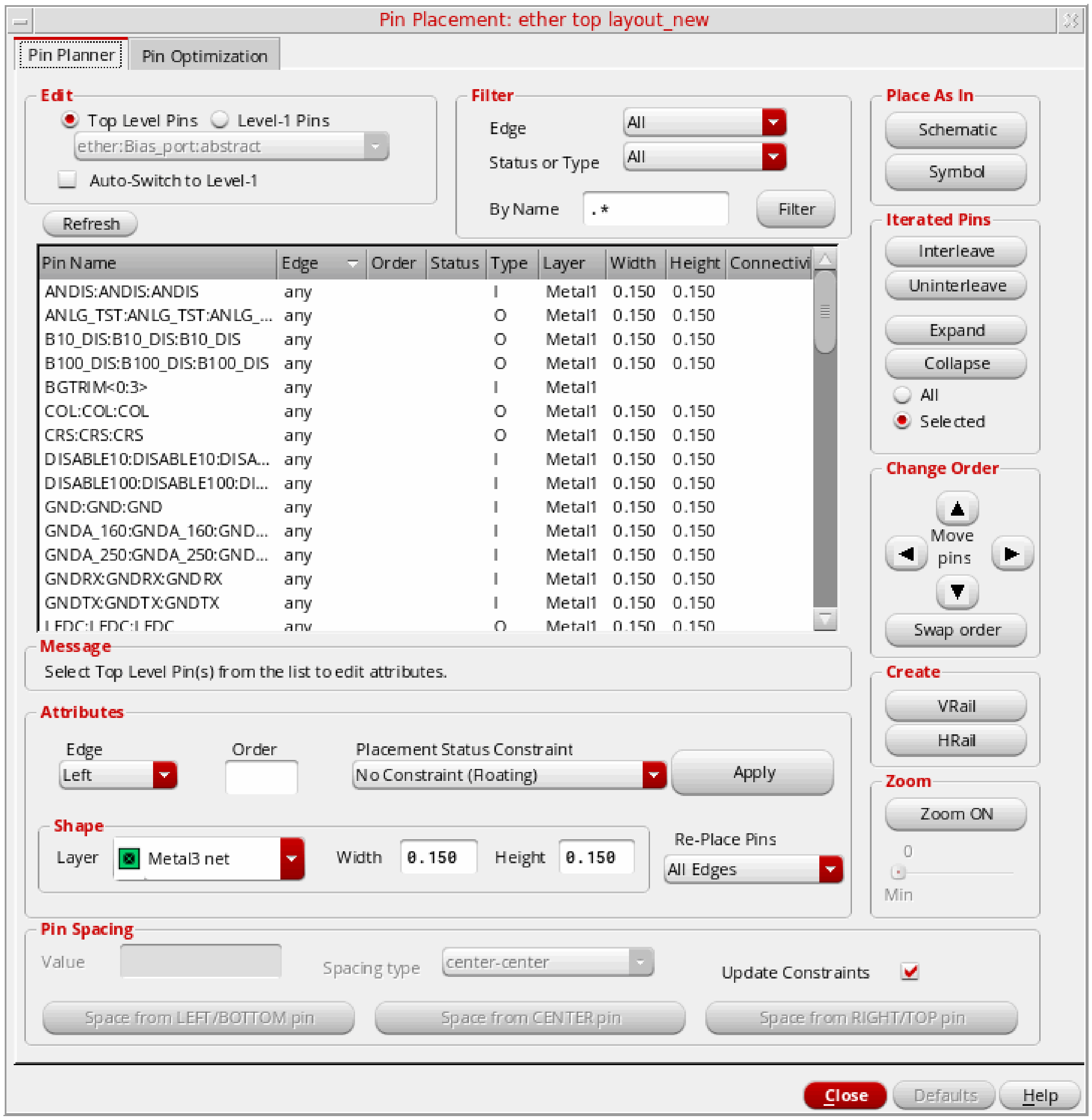
- *Place – Pin Placement* opens the Pin Placement form with the *Pin Planner* tab at the front.
- *Floorplan – Pin Optimization* opens the Pin Placement form with the *Pin Optimization* tab at the front.

Related Topics

Custom Digital Placer Forms

Pin Planner

Use the **Pin Planner** tab to specify how pins are placed in your design.



Pin Placement: ether top layout_new

Pin Planner | Pin Optimization

Edit
[✓ Top Level Pins] Level-1 Pins
ether:Bias_port:abstract
Auto-Switch to Level-1

Refresh

Filter
Edge: All
Status or Type: All
By Name: .*
Filter

Pin Name	Edge	Order	Status	Type	Layer	Width	Height	Connectivi
ANDIS:ANDIS:ANDIS	any			I	Metal1	0.150	0.150	
ANLG_TST:ANLG_TST:ANLG_...	any			O	Metal1	0.150	0.150	
B10_DIS:B10_DIS:B10_DIS	any			O	Metal1	0.150	0.150	
B100_DIS:B100_DIS:B100_DIS	any			O	Metal1	0.150	0.150	
BGTRIM<0:3>	any			I	Metal1			
COL:COL:COL	any			O	Metal1	0.150	0.150	
CRS:CRS:CRS	any			O	Metal1	0.150	0.150	
DISABLE10:DISABLE10:DISA...	any			I	Metal1	0.150	0.150	
DISABLE100:DISABLE100:DI...	any			I	Metal1	0.150	0.150	
GND:GND:GND	any			I	Metal1	0.150	0.150	
GNDA_160:GNDA_160:GND...	any			I	Metal1	0.150	0.150	
GNDA_250:GNDA_250:GND...	any			I	Metal1	0.150	0.150	
GNDRX:GNDRX:GNDRX	any			I	Metal1	0.150	0.150	
GNDTX:GNDTX:GNDTX	any			I	Metal1	0.150	0.150	
LEDC:LEDC:LEDC	any			O	Metal1	0.150	0.150	

Message
Select Top Level Pin(s) from the list to edit attributes.

Attributes
Edge: Left
Order:
Placement Status Constraint: No Constraint (Floating)
Apply

Shape
Layer: Metal3 net
Width: 0.150
Height: 0.150
Re-Place Pins: All Edges

Pin Spacing
Value:
Spacing type: center-center
Update Constraints ✓
Space from LEFT/BOTTOM pin | Space from CENTER pin | Space from RIGHT/TOP pin

Place As In
Schematic
Symbol

Iterated Pins
Interleave
Uninterleave
Expand
Collapse
All [✓ Selected]

Change Order
Move pins
Swap order

Create
VRail
HRail

Zoom
Zoom ON
0
Min

Close | Defaults | Help

Edit lets you specify the scope of the pin planning.

Top Level Pins applies the settings on the form only to the top-level pins.

Auto-Switch to Level-1 automatically zooms to the selected soft block and switches to Level-1 Pins mode when you click a soft block in the design canvas. This option is off by default, which means that when you click a soft block in Top Level Pins mode, the soft block is selected but you remain at the top level.

Level-1 Pins applies the settings on the form to the pins of a selected soft block in the design. Choose the soft block whose pins you want to place from the drop-down list.

Note: If you have already selected a block in the canvas, that block is automatically selected in the *Pin Planner* tab. If you have more than one block selected in the canvas, the first block you selected is the one selected in the form.

Filter lets you specify the pins that are listed in the form. Use a combination of the two pull-downs to filter the pins that are displayed. Alternatively, you can use the *By Name* field to filter the pins by name.

Edge lets you filter by the edge to which a pin is assigned. Choose one of *All*, *Left*, *Top*, *Right*, or *Bottom*.

Status or Type lets you filter by placement status or pin type. Choose one of *All*, *Placed*, *Unplaced*, *Locked*, *Fixed*, *Ordered*, *Unordered*, *I*, *O*, *IO* pins.

By Name lets you filter the Top level or Level-1 pins by name. The default regular expression for this filter is ".*" (dot asterisk), which implies that a name filter is not applied. Therefore, all the pins in the design are displayed. Other valid regular expressions are:

- " " (Blank) — Is equivalent to the default regular expression and it implies that the name filter is not being applied. Displays all the pins.

- "in.*" — Displays all the pins that have names starting with "in".

- ".*in.*" — Displays all the pins that contain "in" in their name.

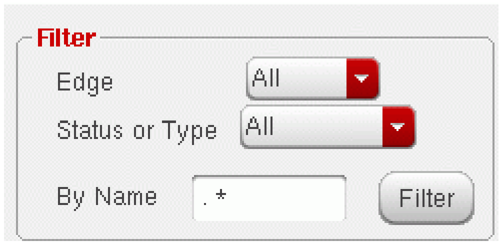
- ".*in" — Displays all the pins that have names ending in "in".

The "in" in the valid regular expressions above can be replaced with any other expression, such as "VDD".

The filter string "*.*" is not a valid regular expression. Use a valid regular expression instead.

The *By Name* field is case-sensitive.

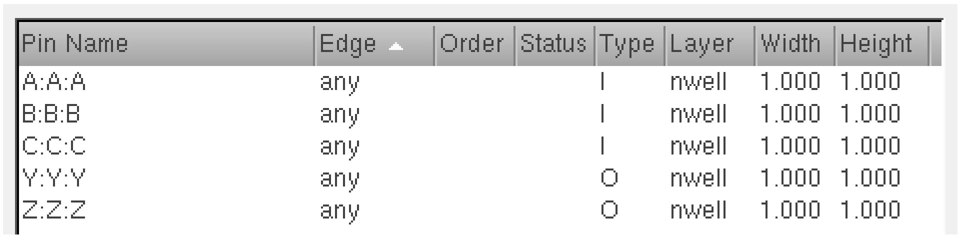
To list all the I/O pins on the left edge of the design, you can set the options, as illustrated below.



Filter	
Edge	All
Status or Type	All
By Name	.*
	Filter

The **pin table** lists all the shape pins found in the current scope. It does not list pins that are vias. The entries are expressed as *termName:pinName:figName* because the pin figure is the physical entity that you are placing. If a pin has more than two pin figures attached to it, but the pin figures have the same name, they are represented by a single entry in the pin table.

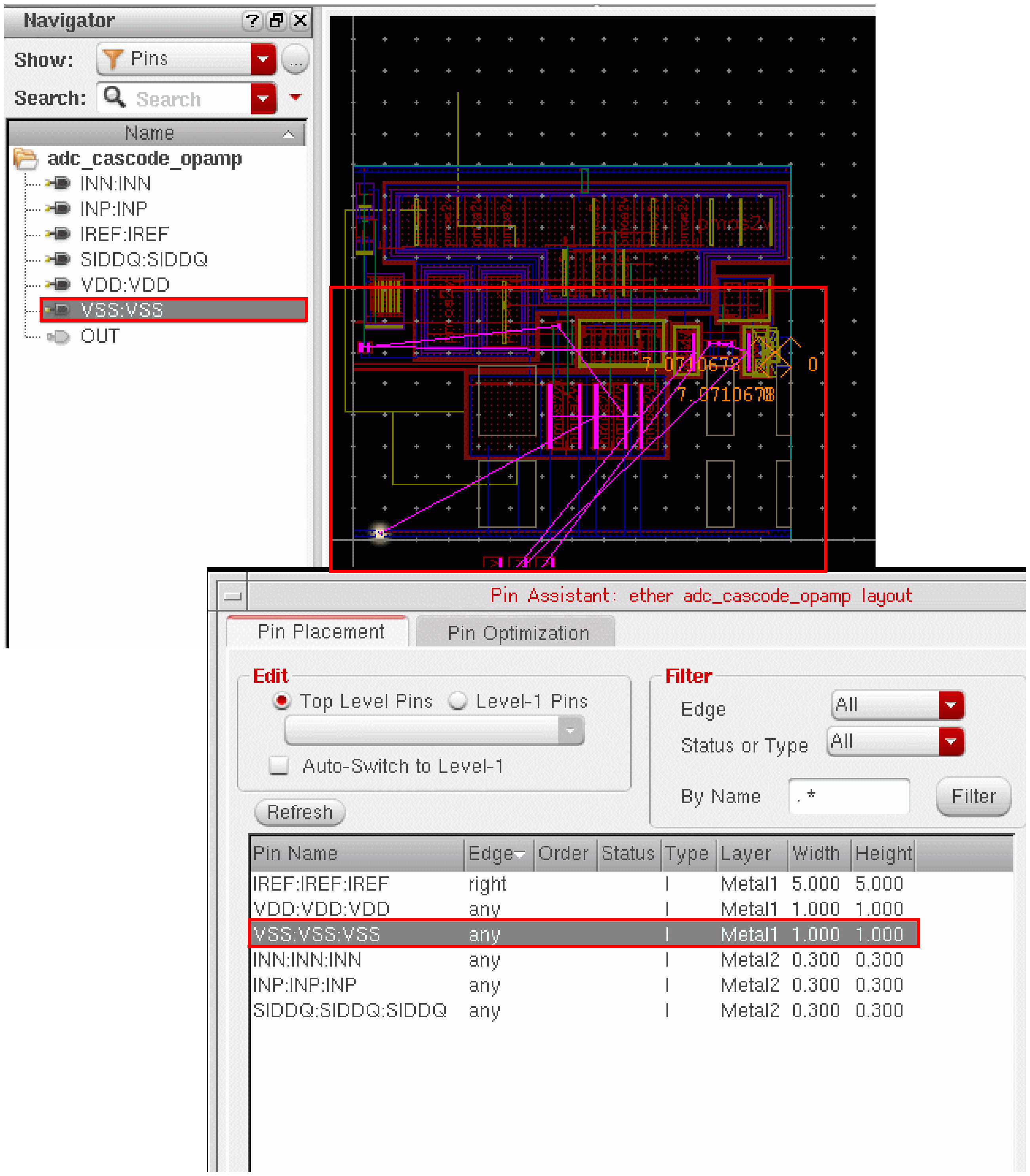
The other columns in the pin table list the *Edge* to which the pin is assigned, the *Order* in which the pins are placed, and the *Pitch*, *Status*, *Type*, *Layer*, *Width*, and *Height* of the pins. You can sort the table by any column by clicking the relevant column header; click again to reverse the sort order.



Pin Name	Edge ▲	Order	Status	Type	Layer	Width	Height
A:A:A	any			I	nwell	1.000	1.000
B:B:B	any			I	nwell	1.000	1.000
C:C:C	any			I	nwell	1.000	1.000
Y:Y:Y	any			O	nwell	1.000	1.000
Z:Z:Z	any			O	nwell	1.000	1.000

Use Model for Cross-Selection of Pins

Any pin that you select in the pin table, as shown below, also gets selected in the navigator assistant and the layout canvas, or vice versa.



Edit: [✓ Top Level Pins] Level-1 Pins; Auto-Switch to Level-1 (unchecked)

Filter: Edge All; Status or Type All; By Name .*

Pin Name	Edge	Order	Status	Type	Layer	Width	Height
IREF:IREF:IREF	right			I	Metal1	5.000	5.000
VDD:VDD:VDD	any			I	Metal1	1.000	1.000
VSS:VSS:VSS	any			I	Metal1	1.000	1.000
INN:INN:INN	any			I	Metal2	0.300	0.300
INP:INP:INP	any			I	Metal2	0.300	0.300
SIDDQ:SIDDQ:SIDDQ	any			I	Metal2	0.300	0.300

Important

The pin selection use-model has been enhanced such that any pins selected in the pin table are now **highlighted and selected** in the layout canvas. This enables directly selecting the pins from the pin table and editing them in the layout canvas.

If you select a top-level pin in the layout canvas and the *Pin Planner* tab has the **Top Level Pins** edit mode selected, the pin table displays the selected pin as highlighted. However, if the edit mode selected in the *Pin Planner* tab is **Level-1 Pins**, the form automatically switches to the **Top Level Pins** mode to show the cross-selection. The automatic mode switching works as long as the selected layout pins belong to the same level—top or level-1. If you select a mix of top-level and level-1 pins in the layout canvas, the *Pin Planner* tab continues to be in the current selected mode and displays only those pins as highlighted in the pin table that belong to the current edit mode.

The table below summarizes how the *Pin Planner* tab handles the cross-selection depending on the layout pins selected. Let us assume here the default edit mode selected is **Top Level Pins**:

Selected Pin	**Pin Planner Tab Behavior**
`Only top pins`	Shows the selected top pins in the pin table.
`Only soft block instance pins`	Automatically switches to the Level-1 Pins mode and shows the selected instance pins in the pin table.
`Top pins + Soft block instance pins`	No automatic mode switching. The Top Level Pins mode shows the selected top pins and the Level-1 Pins mode shows the selected instance pins.
`Only soft block instance`	Automatically switches to the Level-1 Pins mode but shows no selection in the pin table as the instance “pin” is not selected.
`Soft block instance + Soft block instance pins`	Automatically switches to the Level-1 Pins mode and shows the selected instance pins in the pin table.
`Top pins + Soft block instance`	No automatic mode switching. The Top Level Pins mode shows the selected top pins.

`Top pins + Soft block instance + Soft block instance pins`	No automatic mode switching. The **Top Level Pins** mode shows the selected top pins and the **Level-1 Pins** mode shows the selected instance pins.
`Top pins + Soft block instance + Soft block instance pins + Pins of another softblock`	No automatic mode switching. Each level shows the selected pins at that level.

Note:

- To add to your selection set in the pin table, incrementally select the pins using the `Ctrl` or `Shift` key, as appropriate.
- To nullify your existing selection:
  - In the pin table, click any pin. This nullifies the previous selection and shows only the clicked pin as selected.
  - In the layout canvas, click anywhere. This nullifies the previous selection and shows no pins as selected. However, if you click a pin in the layout canvas, the previous selection is nullified and the clicked pin appears as selected.

Messages provides a hint on your next action.

Attributes lets you specify the attributes and *Apply* them to the specified pins in the pin table.

Edge specifies the edge on which the pin is to be placed. The options are *Left*, *Top*, *Right*, *Bottom*, *Any*, *None*, and *As is*. The *As is* setting leaves the edge settings of pins intact when changing the *Order* or *Placement Status Constraint* properties.

Note: Depending on the block type, the pin edges are snapped to the manufacturing or the routing grid. If the block type is custom, the pin edges are snapped to the manufacturing grid. If the block type is digital, the pin edges are snapped to the routing grid.

Order specifies the order in which the selected pins are to be placed. You can specify an order only if you have assigned the pins in question to an edge.

Note: To place the pins unordered, leave the field blank or type “-”. To specify an order for the pins, enter a digit.

Placement Status Constraint specifies the status assigned to the pins after they are placed.

Lock at Placed Location places the pin at the appropriate location on the boundary and then locks the pin at that location. A locked pin cannot be moved either by the automatic placement functions or manually by the user.

Fix at Placed Location places the pin at the appropriate location on the boundary and then fixes the pin at that location. A fixed pin cannot be moved by the automatic placement functions; however, you can move it manually, for example, using the *Edit – Move* command.

No Constraint (Floating) means that the selected pins are neither locked nor fixed after they are placed. A *no constraint*, floating pin can be moved freely by the automatic placement functions and using manual editing functions.

As is does not change the status of the selected pins after placement. Use this option to preserve differing status settings on the selected pins.

Shape lets you specify the *Width* and *Height* of the pin shape to be generated and the *Layer* on which it is drawn. By default, the *Layer*, *Width*, and *Height* of the pin selected in the pins table is displayed. When you change the *Layer*, then the pin placer checks whether the pin *Width* and *Height* values are valid. The pin *Width* and *Height* values cannot be lesser than the layer `minwidth`. So, when you select a different layer:

If the pin *Width* and/or *Height* values fetched are/is equal to or greater than the layer `minwidth` - The values are displayed in the form

If the pin *Width* and/or *Height* values fetched are/is less than the layer `minwidth` - The layer `minwidth` is displayed in the corresponding field

Re-Place Pins specifies which pins are updated with the new attributes when you click *Apply*.

All Edges means that all pins on all edges can be moved by the placer when you click *Apply*.

Selected Edge means that only the pins assigned to the selected edge can be moved when you click *Apply*. All other pins are fixed in their current position.

Selected Pins means that only the selected pins can be moved by the placer when you click *Apply*. All other pins are fixed in their current position.

Pin Spacing lets you specify how the selected pins are spaced. Choose the pins you want to space, set the spacing attributes, and click one of the *Space* buttons depending on which of the selected pins is to be the reference pin. Pins are spaced only by clicking a *Space* button in this section and not by clicking the *Apply* button in the **Attributes** section.

Value specifies the distance by which the selected pins are to be spaced.

Spacing type specifies how the *Value* is applied.

center-center applies the spacing value to the center lines of each of the pin shapes to be spaced.

edge-edge applies the spacing value to the edges of the each of the pin shapes to be spaced.

Note: When spacing pins, the Custom Digital Placer now honors any table spacing rules that are defined in the technology file for metal layers.

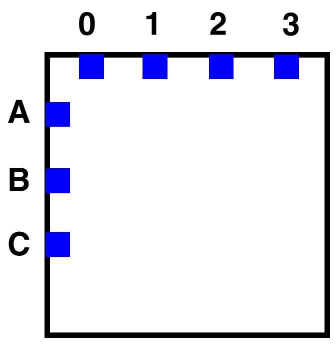
Update Constraints automatically updates any constraints that are applicable for the design (for example, alignment constraints on the selected pins) based on the settings in the **Pin Spacing** section.

Note:

The constraints are updated only by clicking the *Space* button in this section and not by clicking the *Apply* button in the **Attributes** section.

If a pin to be spaced has the alignment constraint set up, the placer reads the constraint to determine the appropriate access direction for the pin.

Space from LEFT/BOTTOM pin uses as its reference pin the leftmost pin on the top and bottom edges (**0** in the example below) and the bottom pin on the side edges (**C**).



Space from CENTER pin uses the center pins on all edges. On the left edge in the example above, this is pin **B**. On the top edge, the pin spacer divides the total number of pins by 2 and uses the result as the reference pin (**2**).

Space from RIGHT/TOP pin uses the rightmost pin on the top and bottom edges (**3**) and the top pin on the side edges (**A**).

Place As In Schematic places the pins in the same relative positions as they have in the schematic view.

Iterated Pins lets you changed the way that iterated pins are listed.

> **Interleave** interleaves the bits of the selected iterated pins, automatically applying an order constraint to specify how the individual bits will be placed.
>
> **Uninterleave** reverses the interleaving of the selected iterated pin bits.
>
> **Expand** shows all the individual bits of one or more iterated pins.
>
> **Collapse** shows bus notation for one or more iterated pins.
>
> **All** performs the *Expand* or *Collapse* function on all the pins in the list box.
>
> **Selected** performs the *Expand* or *Collapse* function only on the selected pins.

Change Order lets you change the order of ordered pins in the pin table of the same layer.

> **Arrow left/down** moves the selected pins down in the ordered list
>
> **Arrow right/up** moves the selected pins up in the ordered list.
>
> **Swap order** inverts the order of two selected pins in the list.

Create lets you create horizontal and vertical rail pins.

> **HRail** elongates the selected pins horizontally from the left edge of the boundary to the right.
>
> **VRail** elongates the selected pins vertically from the top edge of the boundary to the bottom.

Zoom lets you zoom the display in the canvas to the currently-selected pins. To enable the zoom functionality, click the *Zoom ON* button. This makes the slider active. You can then use the slider to set the desired zoom level. The acceptable zoom range is between 0 and 4. Level 0 provides the minimum zoom and level 4 provides the maximum zoom.

Alternatively, you can use the environment variable, vpaAutoZoomEnable, to enable the auto zoom functionality. This ensures that the zoom functionality is automatically enabled when you open the layout view. To set the desired zoom level, you can use the environment variable, vpaAutoZoomLevel.

Note: If you set the auto zoom level to a value less than 0 when using the vpaAutoZoomLevel environment variable, the minimum zoom level will automatically default to 0. Likewise, if you set a maximum value higher than 4 using the environment variable, the maximum zoom value will automatically default to 4. The zoom range lies between 0 to 4 irrespective of the method you use to set the zoom level.

Related Topics

Pin Optimization

Pin Placement

Custom Digital Placer Forms

Placement Planning

Use the **Placement Planning** form to define the boundary cells, tap cells, and rows to be generated before running the Custom Digital Placer.

Use the *Boundary Cell* tab to specify the information about the boundary cells to be placed around a set of rows.

Note: The boundary cells are placed on top of the standard cells. Therefore, additional space must be budgeted for these cells during placement planning.

Component Types specifies the component types that contain the boundary cells. Either type the component types or click *Choose* to select from a list.

Environment Variable: boundaryCellType

Cell Names specifies the names of the boundary cells. Either type the cell name or click *Choose* to select from a list.

Place Outside Region generates the boundary cells outside the row region.

Place On Top Of Row Cells lets you place standard cells over the boundary cells that are placed on the top and bottom rows. By default, the top and bottom rows, which contain boundary cells, are not usable for standard cells.

Environment variable:

Left/Right Cells provides options to select the boundary cells to be inserted in the left and right edges.

Environment variables: leftBotCornerBdryCell, leftEdgeBdryCell, leftTopCornerBdryCell, rightBotCornerBdryCell, rightEdgeBdryCell, rightTopCornerBdryCell, leftBotEdgeBdryCell

Mirror indicates flipped orientation of the selected left and right cells.

Corner Cells provides options to select the boundary cells to be inserted in the *Top*, *Bottom*, *Left*, and *Right* corners of the selected placement region.

Mirror indicates that the selected top, bottom, left, and right cells are to be used with a flipped orientation.

Top/Bottom Cells provides options to select the boundary cells to be inserted in the *Top* and *Bottom* edges of the placement region.

Environment variables: topTapBdryCell, botTapBdryCell, topRowBdryCell, botRowBdryCell

Use the *Tap Cell* tab to specify the information about the tap cells to be inserted in the design.

Component Types specifies the component type that contains the tap cells to be used in the field. You can type the value or click *Choose* to select from a list. The specified component type must have the component class *STDSUBCONT*.

Environment variable: topTapBdryCell

Cell Names specifies the names of the tap cells. Either type the cell names or click *Choose* to select from a list.

Mode specifies whether the tap cells must be *Static* or *Flexible*. Environment variables: subContMaxSpacing, subContMinSpacing, substrateContactType

Static fixes the positions of tap cells as per the specified exact spacing values. Therefore, any change to the core cell placement does not affect the placement of the static tap cells. Specify the *Tap to Tap Spacing* and *Tap to Tap Minimum Spacing* values.

Flexible does not fix the positions of tap cells. Therefore, changes to core cell placement alter the tap cell placement. Use the *Maximum Spacing* and *Minimum Spacing* options to define a spacing range in the *Flexible* mode.

Pattern lets you select a pattern to place the tap cells. Options are: *Regular* and *Checker Board*.

Skip Row Count specifies the rows that must be skipped during tap cell placement. Environment variable: subContInAlternateRow

Offset from Row Start specifies the initial offset of the first tap cell in each row. This option is available only when *Mode* is set to *Static*.

Avoid Abutment specifies whether the tap cells must not be abutted vertically. Environment variable: avoidAbutment

Periodic inserts tap cells periodically from the list of available cells starting from bottom lower left. Environment variable: periodicTap

Lock Tap Cells applies the settings of *Flexible* tap cells and then locks the tap cells to their positions. Environment variable: lockTap

Use the *Row* tab to specify the information about the rows to be generated in the placement region.

Row Template lists all the row templates defined using the Row Template Manager. Select the row template based on which rows need to be generated in the layout canvas.

Use Partial Template allows use of partial row templates to fill the placement region. Environment Variable: usePartialTemplate

Vertical Grid and **Horizontal Grid** specify the reference grids (along each axis) for snapping the devices and rows during placement.

Vertical Offset and **Horizontal Offset** specify the offsets (along each axis) that need to be applied to rows after they are snapped to the reference grids (*Ref X Grid* and *Ref Y Grid*).

Number Of Rows is available only when you select a *Row Template* based on which rows need to be generated. You can either select *Maximum* to generate the maximum number of rows in the row region or *Specify* the number of rows to be generated within the region. Environment Variables: numRowOption, rowCount

Row Utilization specifies how much of a row is used by the components and how much is left free for routing. The default value is `100`, which indicates that all available row space is reserved for placement of the components. Environment Variable: rowUtilization

Row Spacing specifies the spacing between the rows. To abut rows, specify the spacing as `0`. If not specified, the spacing value is auto-computed by the tool. Environment Variables: row2RowSpacing, rowSpacingOption

Note: *Row Spacing* cannot be honored if multi-height standard cells exist in the design.

Row Site Pitch is a non-zero value that indicates the minimum spacing between instances in each row. When specified, the value overrides the calculated value. Environment Variables: rowPitchOption, rowPitchValue

Row Site specifies whether the number of row sites should be calculated automatically (*Auto*) or whether the value needs to be rolled down to the nearest *Odd* or *Even* number. Environment Variable: numRowSites

Power/Ground Pattern specifies how power and ground rails alternate between rows. Environment Variable: supplyPattern

Generate Rails creates rails based on the power and ground rail specifications that you provide. Environment Variable: generateRails

Rail Position controls the position of rails relative to rows. The available values are: *Inside*, *Outside*, and *Center*.

Power Net (and **Ground Net**) specifies the *Net* to which the rail belongs, the *Layer* on which the power (or ground) rail is drawn, and the *Layer Width* of the rail. Environment Variables: generateRails, supplyPosition, rowGroundLayer, rowGroundName, rowGroundWidth, rowPowerLayer, rowPowerName, rowPowerWidth

Use the options on the *Create* tab to define the generic settings.

Rows specifies whether rows must be generated.

Boundary Cells specifies whether boundary cells must be generated.

Tap Cells specifies whether tap cells must be inserted.

Boundary lets you choose whether the boundary for defining the region should be the PR boundary or a custom boundary defined in the design. Environment Variable: placeCluster

Placement Region lets you choose a shape for the placement region.

Rectangle creates a rectangular region.

> **Origin X** sets the X coordinate of the lower-left corner of the lowest numbered row to be generated.
>
> **Origin Y** sets the Y coordinate of the lower-left corner of the lowest numbered row to be generated.
>
> **Width** specifies the width of the region. Make this greater than the height if you want to generate horizontal rows.
>
> **Height** specifies the height of the region. Make this greater than the width if you want to generate vertical rows.

Rectilinear creates a rectilinear region.

> **Points** specifies the coordinates that define the required rectilinear shape. Environment Variable: placementRegion

Draw lets you use the pointer to draw the required shape in the layout window. When creating a *Rectangle*, the shape you draw sets the placement region, and the *X, Y, Width*, and *Height* fields are filled with appropriate values. When creating a *Rectilinear shape*, the coordinates of the individual vertices are updated in the *Points* text field as you draw.

Update updates an existing region in the layout window.

Delete removes an existing region from the layout window.

Allow for Pins reserves a space equal to the size of the pin plus the spacing rule inside the boundary edge. This prevents rows from overlapping pins. This option is not available when you select a *Row Template*. Environment Variable: allowPins

Extend Boundary automatically expands the row and the PR boundary beyond the region, if needed. Specify whether rows are to be stretched *Horizontally* or *Vertically*.

Note: This option is not available when you select a *Row Template*. Environment Variables: extendBoundary, extendBoundaryDirection

Start from Lower Left lets you create rows from the lower-left side of the PR boundary or region. Use the *X Offset* and *Y Offset* fields to specify the corresponding row offset values.

Note: This option is not available when you select a *Row Template*. Environment Variables: rowsFromLowerLeft, rowOffsetX, rowOffsetY

Insert generates the specified components in the design canvas.

Delete deletes the specified components from the design canvas.

Related Topics

Placement Planning

Custom Digital Placer Forms

Custom Digital Placer Environment Variables

Swap Tap Cells

Use the **Swap Tap Cells** form to detect tap cells with maximum spacing violations and replace them with tap cells that have valid maximum spacing values.

Component Type lists all available component types that have their the component class set to *STDSUBCONT*.

Default Cell is the tap cell with which the existing tap cells in the design need to be replaced.

Maximum Diffusion Spacing is the spacing value against which the tap cells in the design need to be checked.

Choose Action includes the following options:

> **Check Only** checks and reports maximum diffusion spacing violations in the current design.
>
> **Check and Replace** replaces all tap cells in the design with the selected Default Cell; and then checks for maximum diffusion spacing violations are replaces erroneous tap cells with the alternate valid tap cells.

Run runs the Swap Tap Cells command.

Insert/Delete Filler Cells Form

Use the **Insert/Delete Filler Cells** form to either insert filler cells into a design or delete filler cells from a design.

Placement Region lets you define the region within which filler cells need to be added or deleted. Choose either prBoundary or any custom boundary defined in the design.

Mode section provides the following options:

Insert specifies that filler cells need to be inserted in the design.

Environment variable: insertFillers

Delete specifies that filler cells need to be deleted from the design.

Fillers section lets you define the following for the filler cells to be inserted or deleted. You can choose either one or both options.

Component Types specifies the component types that contain filler cells. Click **Choose** to display the Select Cells form.Select the required comptypes and libraries from a list. You can select multiple filler cell component types.

Environment variable: fillerCompTypes

Cell Names specifies the names of the cells that are configured as `coreSpacer` using the `cellType` attribute. Click **Choose** and select the cellviews from a list.

Environment variables: fillerCompTypes

Tip

Component Types support the use of the * character, which is default value. All the cells from the filler comptypes are listed in the selected cells field.

Options section provides the following options:

Ignore Placement Blockages specifies whether blockages need to be ignored during filler cell creation.

Environment variable: ignoreBlockage

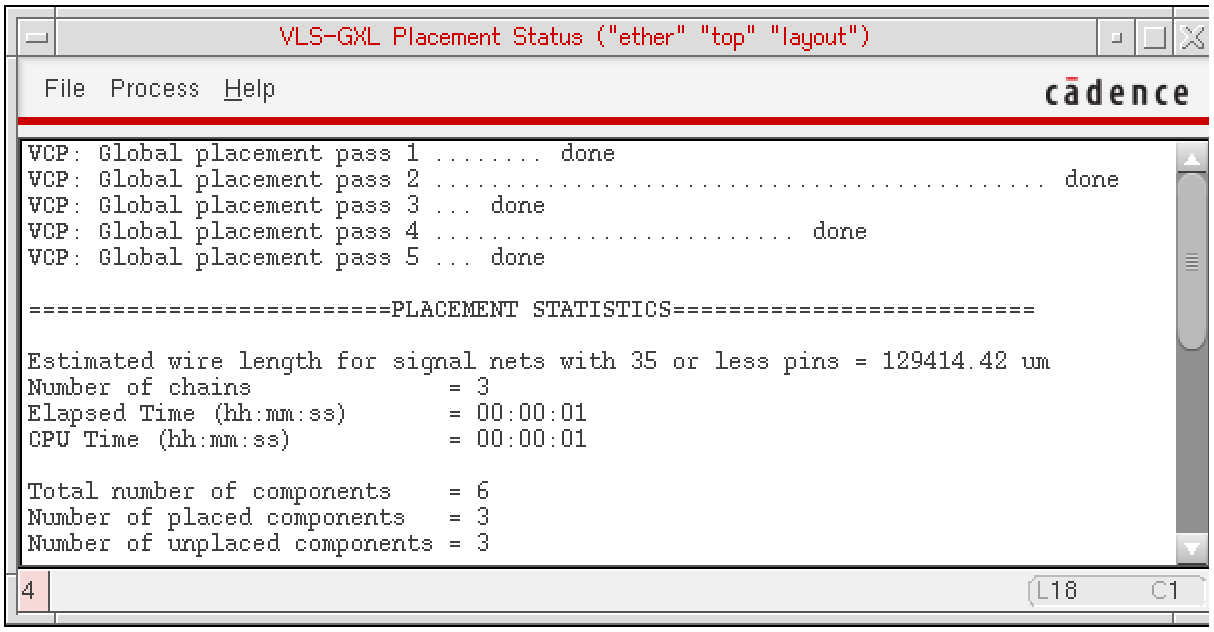
Physical Only specifies whether the `physOnly` attribute needs to be set for filler cells. For more information about this attribute, see `physOnly`.

Environment variable: createPhysOnlyFillers

For more information, see Inserting and Deleting Filler Cells.

Placement Status

The **Placement Status** window reports the progress of the Auto Placer and presents detailed statistics on the results of the placement run. You display this window by checking the **Show Placement Progress** option in the Auto Placer form.



```
VCP: Global placement pass 1 ........ done
VCP: Global placement pass 2 ............................................ done
VCP: Global placement pass 3 ... done
VCP: Global placement pass 4 .......................... done
VCP: Global placement pass 5 ... done

==========================PLACEMENT STATISTICS==========================

Estimated wire length for signal nets with 35 or less pins = 129414.42 um
Number of chains               = 3
Elapsed Time (hh:mm:ss)        = 00:00:01
CPU Time (hh:mm:ss)            = 00:00:01

Total number of components     = 6
Number of placed components    = 3
Number of unplaced components = 3
```

File menu lets you pick from the following functions:

Save As lets you save the log file under the name of your choice.

Search lets you search the log file for a specified string.

Close Window closes the Placement Status window.

Process menu lets you pause the placement run using the **Stop Placer** menu pick.

Related Topics

Custom Digital Placer Forms

7

Custom Digital Placer Environment Variables

This appendix provides information on the names, descriptions, and graphical user interface equivalents for the Virtuoso® custom digital placer (the placer).

Note: Only the environment variables documented in this chapter are supported for public use. All other placer environment variables, regardless of their name or prefix, and undocumented aspects of the environment variables described below, are private and are subject to change at any time.

Inherited Environment Variable Settings

Many of the environment variables honored by the placer are set in Layout XL and Layout L. Information on these environment variables is not duplicated in this section.

For more information, see Environment Variables in the Virtuoso Layout Suite documentation.

Related Topics

List of Custom Digital Placer Environment Variables

List of Custom Digital Placer Environment Variables

- adjustBdy
- allowPins
- allowRotation
- allowRowsBeyondRegion
- avoidAbutment
- botRowBdryCell
- botTapBdryCell
- boundaryCellType
- colorRemasteringCellMapFiles
- colorRemasteringCellMapSuffix
- componentEdge
- considerOptionalEdges
- contextAwarePlacement
- contextConstraintFile
- createPhysOnlyFillers
- ecoModeImpact
- enableColorRemastering
- endCapCellName
- endCapLibName
- endCapViewName
- extendBoundary
- extendBoundaryDirection
- fillerCompTypes
- highlightRowFromDTE
- generateRails

- groupCMOSPairs
- groupMFactors
- hierVHPlace
- ignoreBlockage
- ignoreBlockageAndInsts
- insertBdryCells
- insertCapCells
- insertFillerCells
- insertFillers
- insertSubstrateContacts
- interRowSpacer
- intraRowSpacer
- leftBotCornerBdryCell
- leftBotEdgeBdryCell
- leftCellMirror
- leftCornerCellMirror
- leftEdgeBdryCell
- leftTopCornerBdryCell
- leftTopEdgeBdryCell
- lockTap
- lxGroundNetNames
- lxSupplyNetNames
- maxPinsPerNet
- minBoundaryOffset
- mixedMode
- multiVTLayers
- minVTLength

- multiVTPlacement
- nDiffSpacing
- numRowOption
- numRowSites
- numTemplateRowOption
- openWindow
- optimization
- packedPlacement
- pDiffSpacing
- periodicTap
- pinPlacementXGrid
- pinPlacementYGrid
- placeCluster
- placementMode
- placementRegion
- preserveChains
- refXGridName
- refYGridName
- refXOffset
- refYOffset
- reserveTracksForRouting
- rightBotCornerBdryCell
- rightCellMirror
- rightCornerCellMirror
- rightEdgeBdryCell
- rightTopCornerBdryCell
- rightTopEdgeBdryCell

- routingAwarePinPlc
- rowCount
- rowsFromLowerLeft
- rowGroundLayer
- rowGroundName
- rowGroundWidth
- rowOffsetX
- rowOffsetY
- rowPowerLayer
- rowPowerName
- rowPowerWidth
- rowSpacingOption
- rowTapEndOffset
- rowTemplateName
- rowUtilization
- row2RowSpacing
- rowPitchOption
- rowPitchValue
- saveAs
- saveAsCellName
- saveAsLibName
- saveAsViewName
- selectedFillerCells
- selectFillerCellsFrom
- selectedOnly
- showProgress
-

- startCapCellName
- startCapLibName
- startCapViewName
- subContInAlternateRow
- subContMaxSpacing
- subContMinSpacing
- substrateContactType
- supplyPattern
- supplyPosition
- templateRowCount
- topBotRowPlaceable
- topRowBdryCell
- topTapBdryCell
- usePartialTemplate
- vcpCellBoundaryLPPs
- vcpCellBoundaryUI
- vcpEnableMultiVoltagePlacement
- vcpGlobalPathScript
- vcpKeepoutDepth
- vcpRulesConstraintGroup
- vcpVerboseLevel
- vcpWriteToCIW
- vpaAutoLevel1Switch
- vpaAutoZoomEnable
- vpaAutoZoomLevel
- vpaUpdateConstraints

adjustBdy

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

Description

Automatically recalculates the place and route boundary to take into account the spacing options set on the Auto Placer and Placement Planning forms.

The default is `nil`.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

allowPins

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

Description

Prevents rows from overlapping pins by reserving inside the boundary edge a space equal to the size of the pin plus the spacing rule.

The default is `nil`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Boundary Cell* tab
Field	*Place On Top Of Row Cells* (Placement Planning)

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

allowRotation

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

Description

Allows the placer to rotate components as part of its optimization. When set to `nil`, the placer can move components but not rotate them.

The default is `t`.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

allowRowsBeyondRegion

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

Description

Automatically expands the row beyond the region if needed.

The default is `nil`.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

avoidAbutment

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

Description

Controls the abutment of tap cells that are inserted by running the `vcpfePlaceTapCells` SKILL function. This setting is ignored when `tapPattern` is set to `Checker Board`.

The default value is `nil`, and therefore the tap cells are abutted vertically.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Tap Cell* tab
Field	*Avoid Abutment* (Placement Planning)

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

botRowBdryCell

```
layoutXL.placement botRowBdryCell string { "" | "string1 string2 ..." }
```

Description

Specifies the row cell to be inserted at the bottom of the standard cell.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Boundary Cell* tab
Field	*Top/Bottom Cells* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "botRowBdryCell")
envSetVal("layoutXL.placement" "botRowBdryCell" 'string "ROW_CELL_BOTTOM")
```

Related Topics

List of Custom Digital Placer Environment Variables

botTapBdryCell

```
layoutXL.placement botTapBdryCell string { "" | "string" }
```

Description

Specifies the tap cell to be inserted at the bottom of the standard cell.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "botTapBdryCell")
envSetVal("layoutXL.placement" "botTapBdryCell" 'string "FILLER_CELL_BT")
```

Related Topics

List of Custom Digital Placer Environment Variables

boundaryCellType

```
layoutXL.placement boundaryCellType string { "" | "string1 string2 ..." }
```

Description

Specifies the component type that contains the boundary cells. This setting is valid only if selectBoundaryCellsFrom is set to `Component Types`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Boundary Cell* tab
Field	*Component Types* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "boundaryCellType")
envSetVal("layoutXL.placement" "boundaryCellType" 'string "BounCell")
```

Related Topics

List of Custom Digital Placer Environment Variables

colorRemasteringCellMapFiles

```
layoutXL.placement endCapCellName string "cellMapFile"
```

Description

Specifies the name of the cell map file that stores a list of color variants for instances. This information is used during color re-mastering of instances.

The default value is "", which indicates that the cell map file is either empty or not specified. Use the enableColorRemastering environment variable to turn on color re-mastering.

Important

This environment variable is available only for 3nm process nodes.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "endCapCellName")
envSetVal("layoutXL.placement" "endCapCellName" 'string
"MyFile_cell_color_mapping.lst")
```

Related Topics

List of Custom Digital Placer Environment Variables

colorRemasteringCellMapSuffix

```
layoutXL.placement colorRemasteringCellMapSuffix string "suffix"
```

Description

Specifies the suffix to be added to the name of the cell map file defined by colorRemasteringCellMapFiles.

The default value is `"_cell_color_mapping.lst"`.

Important

This environment variable is available only for 3nm process nodes.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "colorRemasteringCellMapSuffix")
envSetVal("layoutXL.placement" "colorRemasteringCellMapSuffix" 'string
"_custom_color_mapping.lst")
```

Related Topics

List of Custom Digital Placer Environment Variables

componentEdge

```
layoutXL.placement componentEdge cyclic { "Boundary" | "Bounding Box" }
```

Description

Specifies which edge is used when placing a component against the placement boundary. This environment variable can take two values. Default is `Boundary`.

- `Boundary`: Places the component's boundary against the placement boundary. This does not include any well spacing defined around the component, which consequently might lie outside the boundary after placement.
- `Bounding Box`: Places the component's bounding box against the placement boundary. This includes any well spacing defined around the component.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "componentEdge")
envSetVal("layoutXL.placement" "componentEdge" 'cyclic "Boundary")
envSetVal("layoutXL.placement" "componentEdge" 'cyclic "Bounding Box")
```

Related Topics

List of Custom Digital Placer Environment Variables

considerOptionalEdges

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

Description

When set to `t`, VCP honors the `OPTIONAL CELLEDGESPACINGTABLE` property.

The default is `nil`.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

contextAwarePlacement

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

Description

Specifies that the placement should be context-aware.

The default is `nil`, which means that no extra rows are budgeted for or created for context-aware placement.

GUI Equivalent

None

Examples

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

Related Topics

contextConstraintFile

List of Custom Digital Placer Environment Variables

contextConstraintFile

```
layoutXL.placement contextConstraintFile string {"" "string"}
```

Description

Specifies the XML file containing edge spacing constraint information for performing context-aware placement.

If the specified file name is invalid, or the file is not specified at all, a warning is issued.

Note: If a valid context constraint file is provided, packedPlacement is ignored.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "contextConstraintFile")
envSetVal("layoutXL.placement" "contextConstraintFile" 'string "")
envSetVal("layoutXL.placement" "contextConstraintFile" 'string "xmlFile")
```

Related Topics

contextAwarePlacement

List of Custom Digital Placer Environment Variables

createPhysOnlyFillers

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

Description

Specifies whether the `physOnly` attribute needs to be set for filler cells.

The default is `nil`.

GUI Equivalent

Command	*Place – Insert/Delete Filler Cells*
Field	*Options – Physical Only* (Placement Planning)

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

ecoModeImpact

```
layoutXL.placement ecoModeImpact cyclic { "none" | "localized" | "maximized" }
```

Description

Specifies the extent to which the placed components within the design can be moved from their original positions.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "ecoModeImpact")
envSetVal("layoutXL.placement" "ecoModeImpact" 'cyclic "none")
envSetVal("layoutXL.placement" "ecoModeImpact" 'cyclic "localized")
envSetVal("layoutXL.placement" "ecoModeImpact" 'cyclic "maximized")
```

Related Topics

List of Custom Digital Placer Environment Variables

enableColorRemastering

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

Description

Enables re-mastering of instances to their color variants, when required, to prevent color shorts due to cell structure. Color variants are stored in a color map file, which is defined by the colorRemasteringCellMapFiles environment variable.

The default value is `nil`, in which case color re-mastering is turned off.

Important

This environment variable is available only for 3nm process nodes.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

endCapCellName

```
layoutXL.placement endCapCellName string "cell_name"
```

Description

Specifies the cell name of the cellview to be placed as an end cap.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "endCapCellName")
envSetVal("layoutXL.placement" "endCapCellName" 'string "ENDCAP")
```

Related Topics

List of Custom Digital Placer Environment Variables

insertCapCells

startCapLibName

startCapCellName

startCapViewName

endCapLibName

endCapViewName

endCapLibName

```
layoutXL.placement endCapLibName 'string "library_name"
```

Description

Specifies the library name of the cellview to be placed as an end cap.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "endCapLibName")
envSetVal("layoutXL.placement" "endCapLibName" 'string "TESTLIB")
```

Related Topics

List of Custom Digital Placer Environment Variables

insertCapCells

startCapLibName

startCapCellName

startCapViewName

endCapCellName

endCapViewName

endCapViewName

```
layoutXL.placement endCapViewName string "view_name"
```

Description

Specifies the view name of the cellview to be placed as an end cap.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "endCapViewName")
envSetVal("layoutXL.placement" "endCapViewName" 'string "layout")
```

Related Topics

List of Custom Digital Placer Environment Variables

insertCapCells

startCapLibName

startCapCellName

startCapViewName

endCapLibName

endCapCellName

extendBoundary

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

Description

When set to `t`, automatically expands the row and the PR boundary beyond the region, if needed. Use the extendBoundaryDirection environment variable to specify whether rows are to be stretched horizontally or vertically.

Default is `nil`, in which case expansion of rows and boundaries is not possible.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Create* tab
Field	*Extend Boundary* (Placement Planning)

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

extendBoundaryDirection

```
layoutXL.placement extendBoundaryDirection cyclic { "Horizontally" | "Vertically"
     }
```

Description

Specifies whether rows are to be stretched horizontally or vertically. This environment variable can be used only when extendBoundary is set to `t`.

The default is `Horizontally`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Create* tab
Field	*Extend Boundary* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "extendBoundaryDirection")
envSetVal("layoutXL.placement" "extendBoundaryDirection" 'cyclic "Horizontally")
envSetVal("layoutXL.placement" "extendBoundaryDirection" 'cyclic "Vertically")
```

Related Topics

List of Custom Digital Placer Environment Variables

fillerCompTypes

```
layoutXL.placement fillerCompTypes string "fillerCompName"
```

Description

Specifies the names of the component types that contain filler cells that you want to insert in the spaces between the standard cells in a row.

GUI Equivalent

Command	*Place – Custom Digital – Insert/Delete Fillers*
Field	*Component Types* (Insert/Delete Filler Cells Form)

Examples

```
envGetVal("layoutXL.placement" "fillerCompTypes")
envSetVal("layoutXL.placement" "fillerCompTypes" 'string "FILLER_2")
```

Related Topics

List of Custom Digital Placer Environment Variables

highlightRowFromDTE

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

Description

Highlights those rows in the layout canvas that use the rail definitions that are selected on the *Rail Definition* tab of the Dressing Template Editor.

The default is `nil`.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

generateRails

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

Description

Creates rails based on the power and ground rail specifications that you provide using environment variables rowGroundLayer, rowGroundName, rowGroundWidth, rowPowerLayer, rowPowerName, rowPowerWidth.

The default is `nil`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Row* tab
Field	*Generate Rails* (Placement Planning)

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

groupCMOSPairs

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

Description

Groups pairs of CMOS devices in designs that have been generated with chaining switched on. (This environment variable does not work with devices that have been abutted manually because clustering needs chaining information.)

The default is `nil`.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

groupMFactors

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

Description

Automatically adds a grouping constraint for complementary MOS devices that have a multiplication factor in the schematic.

The default is `t`.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

hierVHPlace

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

Description

Specifies whether the placer must descend into the hierarchy and place the instances inside it while placing virtual figGroups. The default is `t`.

Note: Virtual figGroups that are frozen are not modified. Instead, the entire figGroup is placed as a unit.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

ignoreBlockage

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

Description

Specifies whether blockages need to be ignored during filler cell creation.

The default is `nil`.

GUI Equivalent

Command	*Place – Custom Digital – Insert/Delete Filler Cells*
Field	*Ignore Placement Blockages* [Insert/Delete Filler Cells Form]

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

ignoreBlockageAndInsts

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

Description

Ignores existing blockages and instances while generating rows. As a result, rows are and created over the existing blockages and instances.

The default is `nil`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning*
Field	*Region – Ignore Blockages and Instances* [Placement Planning]

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

insertBdryCells

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

Description

When set to `t`, inserts boundary cells around core cells. When set to `nil`, deletes existing boundary cells from the current design.

The default is `t`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Create* tab
Field	*Boundary Cells* (Placement Planning)

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

insertCapCells

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

Description

Inserts cap cells at the start and end of each row when placing components using the Assisted Standard-Cell placement mode.

The default is `nil`.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

startCapLibName

startCapCellName

startCapViewName

endCapLibName

endCapCellName

endCapViewName

insertFillerCells

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

Description

Inserts filler cells into the empty spaces between standard cells in a row without impacting the row size.

The default is `nil`.

Note: You can choose to insert filler cells during a placement run or separately after the initial placement is complete.

You define cells as filler cells by assigning them to a component type with component class *FILLER*. See Defining a Component Type in the *Virtuoso Layout Suite XL User Guide* for more information.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

insertFillers

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

Description

Inserts filler cells into the empty spaces between standard cells in a row without impacting the row size. Specify the component type that contains filler cells by using the fillerCompTypes environment variable. Specify the filler cell types by using the fillerCompTypes environment variable.

The default is `t`.

You define cells as filler cells by assigning them to a component type with component class *FILLER*. See Defining a Component Type in the *Virtuoso Layout Suite XL User Guide* for more information.

GUI Equivalent

Command	*Place – Custom Digital – Insert/Delete Fillers*
Field	*Insert* (Insert/Delete Filler Cells Form)

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

insertSubstrateContacts

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

Description

Inserts tap cells (standard cell substrate contacts) in the empty spaces between the standard cells in a row based on a specified minimum and maximum contact spacing value. Specify the name of the component type that contains the substrate contacts by using the substrateContactType environment variable.

Note: Using this option does not change the row size.

You define cells as substrate contacts by assigning them to a component type with component class *STDSUBCONT*. See Defining a Design-Level Component Type in the *Virtuoso Layout Suite XL User Guide* and Defining a Standard Cell Substrate Contact for more information.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Create* tab
Field	*Rows* (Placement Planning)

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

interRowSpacer

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

Description

Adds or removes space between each row to improve routability. Specify the number of tracks to add between each row using the reserveTracksForRouting environment variable.

The default is `nil`.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

intraRowSpacer

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

Description

Adds or removes space between the components within each row.

The default is `nil`.

GUI Equivalent

Command	*Place – Custom Digital – Auto Placement*
Field	*Run Spacer Within Rows* (Auto Placer)

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

leftBotCornerBdryCell

```
layoutXL.placement leftBotCornerBdryCell string { "" | "string" }
```

Description

Specifies the cut poly cell to be inserted at the left bottom corner of the standard cell.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Boundary Cell* tab
Field	*Top/Bottom Cells* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "leftBotCornerBdryCell")
envSetVal("layoutXL.placement" "leftBotCornerBdryCell" 'string "FILLER_CELL_LB")
```

Related Topics

List of Custom Digital Placer Environment Variables

leftBotEdgeBdryCell

```
layoutXL.placement leftBotEdgeBdryCell string { "" | "string" }
```

Description

Specifies the boundary cell to be inserted at the left bottom edge.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Boundary Cell* tab
Field	*Top/Bottom Cells* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "leftBotEdgeBdryCell")
envSetVal("layoutXL.placement" "leftBotEdgeBdryCell" 'string "BOUNDARY_CELL_LB")
```

Related Topics

List of Custom Digital Placer Environment Variables

leftCellMirror

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

Description

Applies a mirrored orientation to the boundary cells to be inserted in the left edge.

The default is `nil`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Boundary Cell* tab
Field	*Left/Right Cells – Left – Mirror* (Placement Planning)

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

leftCornerCellMirror

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

Description

Applies a mirrored orientation to the boundary cells to be inserted in the left to and bottom corner.

The default is `nil`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Boundary Cell* tab
Field	*Corner Cells – Left – Mirror* (Placement Planning)

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

leftEdgeBdryCell

```
layoutXL.placement leftEdgeBdryCell string { "" | "string" }
```

Description

Specifies the boundary cell that needs to be inserted at the left edge.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Boundary Cell* tab
Field	*Left/Right Cells* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "leftEdgeBdryCell")
envSetVal("layoutXL.placement" "leftEdgeBdryCell" 'string "FILLER_CELL_L")
```

Related Topics

List of Custom Digital Placer Environment Variables

leftTopCornerBdryCell

```
layoutXL.placement leftTopCornerBdryCell string { "" | "string" }
```

Description

Specifies the cut poly cell to be inserted at the upper-left corner of the standard cell.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Boundary Cell* tab
Field	*Left/Right Cells* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "leftTopCornerBdryCell")
envSetVal("layoutXL.placement" "leftTopCornerBdryCell" 'string "FILLER_CELL_TL")
```

Related Topics

List of Custom Digital Placer Environment Variables

leftTopEdgeBdryCell

```
layoutXL.placement leftTopEdgeBdryCell string { "" | "string" }
```

Description

Specifies the boundary cell to be inserted at the left top edges of standard cells.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Boundary Cell* tab
Field	*Left/Right Cells* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "leftTopEdgeBdryCell")
envSetVal("layoutXL.placement" "leftTopEdgeBdryCell" 'string "BOUNDARY_CELL_LTE")
```

Related Topics

List of Custom Digital Placer Environment Variables

lockTap

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

Description

Specifies whether the tap cells inserted by running the `vcpfePlaceTapCells` SKILL function must be locked.

The default value is `t`.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

lxGroundNetNames

```
layoutXL lxGroundNetNames string "list_of_ground_net_names"
```

Description

Specifies a list of ground net names to be excluded from wire length optimization. The list must be enclosed in quotation marks, and each ground net name must be separated by a space.

The default list of names is: "`gnd gnd! gnd: vss vss! vss:`".

Note: This environment variable is also used to achieve optimized chaining results in Layout XL device-level schematics.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lxGroundNetNames")
envSetVal("layoutXL" "lxGroundNetNames" 'string "gnd gnd! gnd: vss vss! vss:")
```

Related Topics

List of Custom Digital Placer Environment Variables

lxSupplyNetNames

`layoutXL lxSupplyNetNames string "`*`list_of_supply_net_names`*`"`

Description

Specifies a list of power net names to be excluded from wire length optimization. The list must be enclosed in quotation marks, and each supply net name must be separated by a space.

The default list of names is: "`vcc vcc! vcc: vdd vdd! vdd:`".

Note: This environment variable is also used to achieve optimized chaining results in Layout XL device-level schematics.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "lxSupplyNetNames")
envSetVal("layoutXL" "lxSupplyNetNames" 'string "vcc vcc! vcc: vdd vdd! vdd:")
```

Related Topics

List of Custom Digital Placer Environment Variables

maxPinsPerNet

`layoutXL.placement maxPinsPerNet int` *`integer_number`*

Description

Specifies the maximum number of pins a net can have for its cost to be considered during placement. Nets with more than the specified number of pins are not considered.

The default is `35`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "maxPinsPerNet")
envSetVal("layoutXL.placement" "maxPinsPerNet" 'int 25)
```

Related Topics

List of Custom Digital Placer Environment Variables

minBoundaryOffset

```
layoutXL.placement minBoundaryOffset float float_number
```

Description

Specifies the minimum spacing allowed between the boundary edge and the nearest component. The default value is half of the maximum value of all the spacing rules for the layers specified in the applicable technology file.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "minBoundaryOffset")
envSetVal("layoutXL.placement" "minBoundaryOffset" 'float 0.5)
```

Related Topics

List of Custom Digital Placer Environment Variables

mixedMode

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

Description

Allows floating components to be moved when using the Assisted Mixed CMOS/Standard Cell placement mode.

The default is `nil`.

Note: Enabling this environment variable can adversely affect the performance of the placer.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

multiVTLayers

`layoutXL.placement multiVTLayers string "`*`layerNames`*`"`

Description

Specifies the multi-voltage layer names to be used for multi-voltage placement.

The default value is `"VTS_N VTL_N VTUL_N"`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "multiVTLayers")
envSetVal("layoutXL.placement" "multiVTLayers" 'string "VTS_N VTL_N VTUL_N")
```

Related Topics

List of Custom Digital Placer Environment Variables

minVTLength

`layoutXL.placement minVTLength float` *`float_number`*

Description

Specifies the minimum length of multi-voltage layers.

The default is `0.228`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "minVTLength")
envSetVal("layoutXL.placement" "minVTLength" 'float 0.30)
```

Related Topics

List of Custom Digital Placer Environment Variables

multiVTPlacement

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

Description

Enables multi-voltage placement.

The default is `nil`.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

nDiffSpacing

```
layoutXL.placement nDiffSpacing float float_number
```

Description

Sets the minimum spacing required between adjacent NMOS chains. This value affects the estimation of the number of rows required.

Note: The spacing value specified by using this environment variable can override the value already specified in the Placement Planning form. Therefore, this environment variable should not be used to specify a new spacing value unless the intent is to replan placement.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "nDiffSpacing")
envSetVal("layoutXL.placement" "nDiffSpacing" 'float 5.0)
```

Related Topics

List of Custom Digital Placer Environment Variables

numRowOption

```
layoutXL.placement numRowOption cyclic { "Auto" | "Maximum" | "Specify" }
```

Description

Lets you *Specify* the number of rows to be created or leave it to the placer to create the *Maximum* number of rows possible in the space available.

The default is `Auto`, where the placer automatically creates rows based on other row parameters.

- `Auto`: Lets the placer calculate the number of rows to be generated based on other row parameters.
- `Maximum`: Specifies that the placer to generate the maximum number of rows possible in the space available.
- `Specify`: Specifies the number of rows to be generated. Use the rowCount environment variable to specify the required value.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Row* tab
Field	*Number Of Rows* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "numRowOption")
envSetVal("layoutXL.placement" "numRowOption" 'cyclic "Auto")
envSetVal("layoutXL.placement" "numRowOption" 'cyclic "Maximum")
envSetVal("layoutXL.placement" "numRowOption" 'cyclic "Specify")
```

Related Topics

List of Custom Digital Placer Environment Variables

numRowSites

```
layoutXL.placement numRowSites cyclic { "Auto" | "Odd" | "Even" }
```

Description

Specifies whether the number of row sites that is calculated automatically (`Auto`) should be used or whether the value needs to be rolled down to the nearest `Odd` or `Even` number.

The default is `Auto`.

- `Auto`: The calculated number of row sites is used.
- `Odd`: The calculated number of row sites is rolled down to the nearest odd number. For example, if the calculated number of row sites is 76, then it is rolled down to 75.
- `Even`: The calculated number of row sites is rolled down to the nearest even number. For example, if the calculated number of row sites is 77, then it is rolled down to 76.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "numRowSites")
envSetVal("layoutXL.placement" "numRowSites" 'cyclic "Auto")
envSetVal("layoutXL.placement" "numRowSites" 'cyclic "Odd")
envSetVal("layoutXL.placement" "numRowSites" 'cyclic "Even")
```

Related Topics

List of Custom Digital Placer Environment Variables

numTemplateRowOption

```
layoutXL.placement numTemplateRowOption cyclic { "Maximum" | "Specify" }
```

Description

Specifies the number of rows to be generated in the row region. Valid options are:

- `Maximum` (default) - Generates the maximum number rows in the row region.
- `Specify` - Lets you specify the exact number of rows to be generated using the `templateRowCount` environment variable.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Row* tab
Field	*Number Of Rows* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "numTemplateRowOption")
envSetVal("layoutXL.placement" "numTemplateRowOption" 'cyclic "Specify")
envSetVal("layoutXL.placement" "numTemplateRowOption" 'cyclic "Maximum")
```

Related Topics

List of Custom Digital Placer Environment Variables

openWindow

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

Description

Displays the placed view in a new window after placement. Otherwise, the system updates the output view in an existing window.

The default is `nil`.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

optimization

```
layoutXL.placement optimization cyclic { "none" | "basic" | "moderate" |
     "optimized" }
```

Description

Controls the time that the placer spends trying to achieve the best possible results.

The default is `basic`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "optimization")
envSetVal("layoutXL.placement" "optimization" 'cyclic "none")
envSetVal("layoutXL.placement" "optimization" 'cyclic "basic")
envSetVal("layoutXL.placement" "optimization" 'cyclic "moderate")
envSetVal("layoutXL.placement" "optimization" 'cyclic "optimized")
```

Related Topics

List of Custom Digital Placer Environment Variables

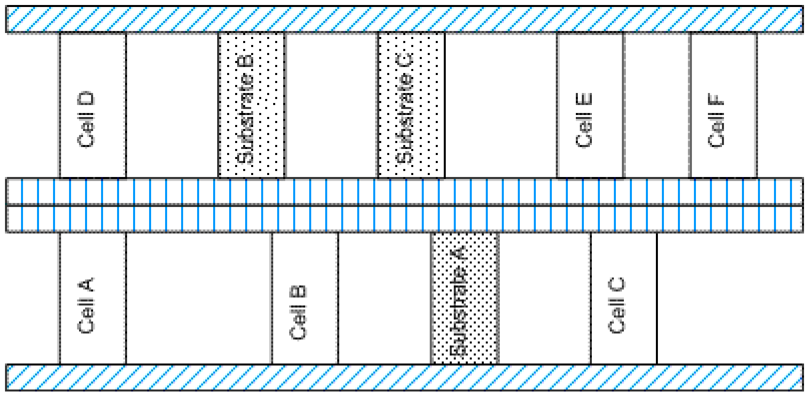
packedPlacement

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

Description

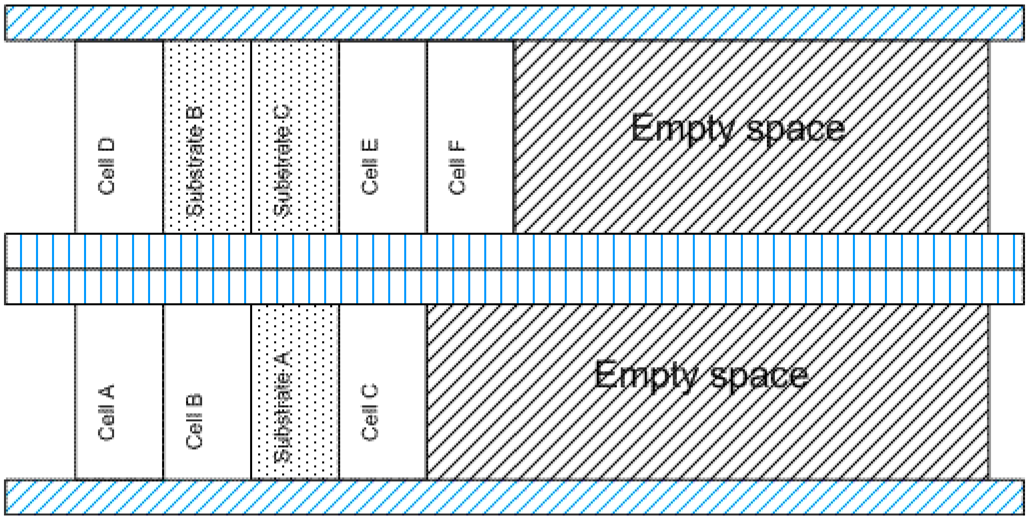
Places the standard cells and tap cells abutted together. However, after abutment, you might notice empty spaces at the end of the row because the environment variable does not adjust the existing prBoundary based on the new placement. Also, if a design has any predefined constraints, the environment variable might not honor them, resulting in constraint violations.

Below is a design for which the `packedPlacement` environment variable has not been set.



After the `packedPlacement` environment variable is set for the design, the cells in the design, as shown below, appear abutted together and an empty space is created at the end of the row.

Note: If the rows are vertical, the direction of abutment is downwards.



GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

pDiffSpacing

```
layoutXL.placement pDiffSpacing float float_number
```

Description

Sets the minimum spacing required between adjacent PMOS chains. This value affects the estimation of the number of rows required.

Note: The spacing value specified by using this environment variable can override the value already specified in the Placement Planning form. Therefore, this environment variable should not be used to specify a new spacing value unless the intent is to replan placement.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "pDiffSpacing")
envSetVal("layoutXL.placement" "pDiffSpacing" 'float 5.0)
```

Related Topics

List of Custom Digital Placer Environment Variables

periodicTap

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

Description

Inserts all the available tap cells periodically starting from bottom lower-left corner of the boundary in a circular pattern. The default value is `nil`.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

pinPlacementXGrid

```
layoutXL.placement pinPlacementXGrid float float_number
```

Description

Specifies the X-coordinate of the placement grid on which the Custom Digital placer places pins. When not set, the default value, `0.0`, is used to indicate that the placer places pins on the manufacturing grid.

Set the pinPlacementYGrid environment variable to specify the Y-coordinate of the placement grid.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "pinPlacementXGrid")
envSetVal("layoutXL.placement" "pinPlacementXGrid" 'float 0.02)
```

Related Topics

List of Custom Digital Placer Environment Variables

pinPlacementYGrid

```
layoutXL.placement pinPlacementYGrid float float_number
```

Description

Specifies the Y-coordinate of the placement grid on which the Custom Digital placer places pins. When not set, the default value, `0.0`, is used to indicate that the placer places pins on the manufacturing grid.

Set the pinPlacementXGrid environment variable to specify the X-coordinate of the placement grid.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "pinPlacementYGrid")
envSetVal("layoutXL.placement" "pinPlacementYGrid" 'float 0.005)
```

Related Topics

List of Custom Digital Placer Environment Variables

placeCluster

```
layoutXL.placement placeCluster string "clusterName"
```

Description

Places only the components assigned to the specified cluster name.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "placeCluster")
envSetVal("layoutXL.placement" "placeCluster" 'string "myCluster")
```

Related Topics

List of Custom Digital Placer Environment Variables

placementMode

```
layoutXL.placement placementMode cyclic { "Auto Placement" | "ECO Placement" }
```

Description

Specifies the mode to be used to place components. If `ECO Placement` is specified, then all unplaced components are placed based on their connectivity with the placed components. If `Auto Placement` is specified, then components are placed without any regard to their initial placement.

The default is `Auto Placement`.

GUI Equivalent

Command	*Place – Custom Digital – Auto Placement*
Field	*Auto Placement* and *ECO Placement* (Auto Placer)

Examples

```
envGetVal("layoutXL.placement" "placementMode")
envSetVal("layoutXL.placement" "placementMode" 'cyclic "Auto Placement")
envSetVal("layoutXL.placement" "placementMode" 'cyclic "ECO Placement")
```

Related Topics

List of Custom Digital Placer Environment Variables

placementRegion

```
layoutXL.placement placementRegion cyclic { "Boundary" | "Placement Region" }
```

Description

Specifies the region or boundary within which components need to be placed.

The default is `Boundary`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Create* tab
Field	*Placement Region* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "placementRegion")
envSetVal("layoutXL.placement" "placementRegion" 'cyclic "Boundary")
envSetVal("layoutXL.placement" "placementRegion" 'cyclic "Placement Region")
```

Related Topics

List of Custom Digital Placer Environment Variables

preserveChains

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

Description

Preserves chains that were generated automatically by the *Generate All From Source* or *Placement Planning* commands. When switched off, the placer is free to break chains to share diffusion on individual transistors, which can reduce the wire length of polysilicon gate connections and result in better alignment.

The default is `t`.

Cadence recommends that you leave this environment variable set to `t` and inspect the results obtained by the automatic chaining function to see if they are satisfactory. If they are not, you can switch off the option and let the placer try to generate a better result.

When disabling this option, consider the following:

- It is more efficient for the placer to place transistor chains generated by the *Generate All From Source* or *Placement Planning* commands than to place individual transistors.
- Diffusion sharing during automatic placement is not guaranteed because the placer is geared primarily toward reducing overall wire length. If the area within the rows is limited, the placer attempts to share diffusion; if there is excess area within the rows, it does not.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

refXGridName

```
layoutXL.placement refXGridName string "GridName"
```

Description

Specifies the reference grids along the x-axis for snapping the devices during placement. The value must be enclosed in quotation marks.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "refXGridName")
envSetVal("layoutXL.placement" "refXGridName" 'string "GridX_1")
```

Related Topics

List of Custom Digital Placer Environment Variables

refYGridName

```
layoutXL.placement refYGridName string "GridName"
```

Description

Specifies the reference grids along the y-axis for snapping the devices during placement. The value must be enclosed in quotation marks.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "refYGridName")
envSetVal("layoutXL.placement" "refYGridName" 'string "GridY_1")
```

Related Topics

List of Custom Digital Placer Environment Variables

refXOffset

```
layoutXL.placement refXOffset float float_number
```

Description

Specifies the horizontal offset for creating a row region.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Create* tab
Field	*Placement Region – Rectangle – Origin X* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "refXOffset")
envSetVal("layoutXL.placement" "refXOffset" 'float 3)
```

Related Topics

List of Custom Digital Placer Environment Variables

refYOffset

```
layoutXL.placement refYOffset float float_number
```

Description

Specifies the vertical offset for creating a row region.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Create* tab
Field	*Placement Region – Rectangle – Origin Y* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "refYOffset")
envSetVal("layoutXL.placement" "refYOffset" 'float 3)
```

Related Topics

List of Custom Digital Placer Environment Variables

reserveTracksForRouting

```
layoutXL.placement reserveTracksForRouting cyclic { -1 | 0 | 1 ... 10 }
```

Description

Specifies the additional number of routing tracks to be reserved between each row when running the interRowSpacer. The number you specify is added to the number of tracks the placer estimates are required to perform the routing to give the total number of tracks required. This is then compared to the number of tracks available between rows, and the space available is increased or reduced accordingly.

For example,

> Placer estimates required tracks for routing: 10
> reserveTracksForRouting: 3
> Total number of tracks required: 13

If there are currently 8 tracks available between rows, the placer adds space for 5 more.
If there are currently 15 tracks available between rows, the placer removes 2.

The default is `2`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "reserveTracksForRouting")
envSetVal("layoutXL.placement" "reserveTracksForRouting" 'cyclic "2")
```

Related Topics

List of Custom Digital Placer Environment Variables

rightBotCornerBdryCell

```
layoutXL.placement rightBotCornerBdryCell string { "" | "string" }
```

Description

Specifies the cut poly cell to be inserted at the right bottom corner of the standard cell.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Boundary Cell* tab
Field	*Left/Right Cells* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "rightBotCornerBdryCell")
envSetVal("layoutXL.placement" "rightBotCornerBdryCell" 'string "FILLER_CELL_RB")
```

Related Topics

List of Custom Digital Placer Environment Variables

rightCellMirror

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

Description

Applies a mirrored orientation to the boundary cells to be inserted in the right edge.

The default is `nil`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Boundary Cell* tab
Field	*Left/Right Cells – Right – Mirror* (Placement Planning)

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

rightCornerCellMirror

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

Description

Applies a mirrored orientation to the boundary cells to be inserted in the right corner.

The default is `nil`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Boundary Cell* tab
Field	*Corner Cells – Right – Mirror* (Placement Planning)

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

rightEdgeBdryCell

```
layoutXL.placement rightEdgeBdryCell string { "" | "string" }
```

Description

Specifies the boundary cell that needs to be inserted at the right edge.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Boundary Cell* tab
Field	*Left/Right Cells* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "rightEdgeBdryCell")
envSetVal("layoutXL.placement" "rightEdgeBdryCell" 'string "FILLER_CELL_RE")
```

Related Topics

List of Custom Digital Placer Environment Variables

rightTopCornerBdryCell

```
layoutXL.placement rightTopCornerBdryCell string { "" | "string" }
```

Description

Specifies the cut poly cell to be inserted at the upper-right corner of the standard cell.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Boundary Cell* tab
Field	*Left/Right Cells* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "rightTopCornerBdryCell")
envSetVal("layoutXL.placement" "rightTopCornerBdryCell" 'string "FILLER_CELL_TR")
```

Related Topics

List of Custom Digital Placer Environment Variables

rightTopEdgeBdryCell

```
layoutXL.placement rightTopEdgeBdryCell string { "" | "string" }
```

Description

Specifies the boundary cell to be inserted at the right top edges of standard cells.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Boundary Cell* tab
Field	*Left/Right Cells* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "rightTopEdgeBdryCell")
envSetVal("layoutXL.placement" "rightTopEdgeBdryCell" 'string
"BOUNDARY_CELL_RTE")
```

Related Topics

List of Custom Digital Placer Environment Variables

routingAwarePinPlc

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

Description

Specifies whether the placer must consider the WSP direction while placing unconstrained pins.

The default is `t`.

GUI Equivalent

Command	*Place – Custom Digital – Auto Placement*
Field	*Routing Direction Aware Pin Placement* (Auto Placer)

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

rowCount

`layoutXL.placement rowCount int "`*`integer_number`*`"`

Description

Specifies the exact number of rows to be generated by placement planning. The default is `1`.

This environment variable is honored only when the numRowOption environment variable is set to `Specify`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Row* tab
Field	*Number Of Rows: Specify* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "rowCount")
envSetVal("layoutXL.placement" "rowCount" 'int 10)
```

Related Topics

List of Custom Digital Placer Environment Variables

rowsFromLowerLeft

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

Description

Creates rows from the lower-left side of the PR boundary or region. Use the rowOffsetX and rowOffsetY environment variables to specify the corresponding row offset values.

The default is `nil`.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

rowGroundLayer

`layoutXL.placement rowGroundLayer string "`*`layer_name`*`"`

Description

Specifies the name of the layer on which ground rails are drawn during row generation.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Row* tab
Field	*Generate Rails – Ground – Layer* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "rowGroundLayer")
envSetVal("layoutXL.placement" "rowGroundLayer" 'string "Metal1")
```

Related Topics

List of Custom Digital Placer Environment Variables

rowGroundName

`layoutXL.placement rowGroundName string "`*`net_name`*`"`

Description

Specifies the name of the net to which ground rails are assigned during row generation.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Row* tab
Field	*Rails – Ground – Net Name* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "rowGroundName")
envSetVal("layoutXL.placement" "rowGroundName" 'string "vss!")
```

Related Topics

List of Custom Digital Placer Environment Variables

rowGroundWidth

```
layoutXL.placement rowGroundWidth float float_number
```

Description

Specifies the width of ground rails created during row generation.

The default is `1.0`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Row* tab
Field	*Rails – Ground – Layer Width* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "rowGroundWidth")
envSetVal("layoutXL.placement" "rowGroundWidth" 'float 1.25)
```

Related Topics

List of Custom Digital Placer Environment Variables

rowOffsetX

```
layoutXL.placement rowOffsetX float float_number
```

Description

Specifies the x-offset value for creating rows from the lower-left side of the PR boundary or region. This environment variable can be used only when rowsFromLowerLeft is set to `t`.

The default is `0`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "rowOffsetX")
envSetVal("layoutXL.placement" "rowOffsetX" 'float 9.0)
```

Related Topics

List of Custom Digital Placer Environment Variables

rowOffsetY

`layoutXL.placement rowOffsetY float` *`float_number`*

Description

Specifies the y-offset value for creating rows from the lower-left side of the PR boundary or region. This environment variable can be used only when rowsFromLowerLeft is set to `t`.

The default is `0`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "rowOffsetY")
envSetVal("layoutXL.placement" "rowOffsetY" 'float 9.0)
```

Related Topics

List of Custom Digital Placer Environment Variables

rowPowerLayer

```
layoutXL.placement rowPowerLayer string "layer_name"
```

Description

Specifies the name of the layer on which power rails are drawn during row generation.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Row* tab
Field	*Generate Rails – Power – Layer* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "rowPowerLayer")
envSetVal("layoutXL.placement" "rowPowerLayer" 'string "Metal1")
```

Related Topics

List of Custom Digital Placer Environment Variables

rowPowerName

```
layoutXL.placement rowPowerName string "net_name"
```

Description

Specifies the name of the net to which power rails are assigned during row generation.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Row* tab
Field	*Generate Rails – Power – Net Name* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "rowPowerName")
envSetVal("layoutXL.placement" "rowPowerName" 'string "vdd!")
```

Related Topics

List of Custom Digital Placer Environment Variables

rowPowerWidth

`layoutXL.placement rowPowerWidth float` *`float_number`*

Description

Specifies the width of power rails created during row generation.

The default is `1.0`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Row* tab
Field	*Generate Rails – Power – Layer Width* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "rowPowerWidth")
envSetVal("layoutXL.placement" "rowPowerWidth" 'float 1.25)
```

Related Topics

List of Custom Digital Placer Environment Variables

rowSpacingOption

```
layoutXL.placement rowSpacingOption cyclic { "Auto" | "Specify" }
```

Description

Specifies whether the spacing between rows needs to be calculated automatically or specified manually.

The default is `Auto`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Row* tab
Field	*Row Spacing* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "rowSpacingOption")
envSetVal("layoutXL.placement" "rowSpacingOption" 'cyclic "Auto")
envSetVal("layoutXL.placement" "rowSpacingOption" 'cyclic "Specify")
```

Related Topics

List of Custom Digital Placer Environment Variables

rowTapEndOffset

`layoutXL.placement rowTapEndOffset float` *`float_number`*

Description

Specifies the distance from the right edge at which the last tap cell must be placed in each row. The value is applicable when the `vcpfePlaceTapCells` SKILL function is run to insert tap cells in the empty spaces between standard cells.

The default is `0.0`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "rowTapEndOffset")
envSetVal("layoutXL.placement" "rowTapEndOffset" 'float 0.1)
```

Related Topics

List of Custom Digital Placer Environment Variables

rowTemplateName

```
layoutXL.placement rowTemplateName string "templateName"
```

Description

Specifies the name of the row templates based on which rows need to be inserted in the layout canvas. The value must be enclosed in quotation marks. Use the Row Template Manager to define a new row template.

The default is `None`.

GUI Equivalent

Command	*Place – Placement Planning – Row* tab
Field	*Row Template* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "rowTemplateName")
envSetVal("layoutXL.placement" "rowTemplateName" 'string "Template1")
```

Related Topics

List of Custom Digital Placer Environment Variables

rowUtilization

```
layoutXL.placement rowUtilization int integer_number
```

Description

Specifies how much space is allowed for the components inside each generated row and how much is reserved for routing.

The default is `100%`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Row* tab
Field	*Row Utilization* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "rowUtilization")
envSetVal("layoutXL.placement" "rowUtilization" 'int 75)
```

Related Topics

List of Custom Digital Placer Environment Variables

row2RowSpacing

```
layoutXL.placement row2RowSpacing float float_number
```

Description

Specifies the spacing between rows. To abut rows, specify the spacing as `0`. If not specified, then the value is auto-computed by the tool.

The default is `0`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Row* tab
Field	*Row Spacing: Specify* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "row2RowSpacing")
envSetVal("layoutXL.placement" "row2RowSpacing" 'float 1.0)
```

Related Topics

List of Custom Digital Placer Environment Variables

rowPitchOption

```
layoutXL.placement rowPitchOption cyclic { "Auto" | "Specify" }
```

Description

Specifies whether the value for minimum spacing between instances in each row needs to be calculated automatically or specified manually.

The default is `Auto`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Row* tab
Field	*Row Site Pitch* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "rowPitchOption")
envSetVal("layoutXL.placement" "rowPitchOption" 'cyclic "Auto")
envSetVal("layoutXL.placement" "rowPitchOption" 'cyclic "Specify")
```

Related Topics

List of Custom Digital Placer Environment Variables

rowPitchValue

```
layoutXL.placement rowPitchValue float float_number
```

Description

Indicates the minimum spacing between instances in each row.

The default is `0.001`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Row* tab
Field	*Row Site Pitch: Specify* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "rowPitchValue")
envSetVal("layoutXL.placement" "rowPitchValue" 'float 0.002)
```

Related Topics

List of Custom Digital Placer Environment Variables

saveAs

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

Description

Specifies that the placement results are saved under a cellview name different from the current cellview.

Tip

Specify the name of the cellview using the saveAsLibName, saveAsCellName, and saveAsViewName environment variables.

The default is `nil`.

GUI Equivalent

Command	*Place – Custom Digital – Auto Placement*
Field	*Save As* (Auto Placer)

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

saveAsCellName

```
layoutXL.placement saveAsCellName string "cellName"
```

Description

Specifies the cell name of the saveAs cellview.

The default is the same name as the source cell.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "saveAsCellName")
envSetVal("layoutXL.placement" "saveAsCellName" 'string "myCell")
```

Related Topics

List of Custom Digital Placer Environment Variables

saveAsLibName

```
layoutXL.placement saveAsLibName string "libraryName"
```

Description

Specifies the library in which a saveAs cellview is saved.

The default is the same name as the source library.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "saveAsLibName")
envSetVal("layoutXL.placement" "saveAsLibName" 'string "myLib")
```

Related Topics

List of Custom Digital Placer Environment Variables

saveAsViewName

```
layoutXL.placement saveAsViewName string "viewName"
```

Description

Specifies the view name for a saveAs cellview.

The default is the same name as the source view.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "saveAsViewName")
envSetVal("layoutXL.placement" "saveAsViewName" 'string "myView")
```

Related Topics

List of Custom Digital Placer Environment Variables

selectBoundaryCellsFrom

```
layoutXL.placement selectBoundaryCellsFrom cyclic { "Component Types" | "Cell
     Names" }
```

Description

Specifies whether boundary cells are to be selected based on their component types or cell names.

The default is `Component Types`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning*
Field	*Boundary Cell – Component Types, Cell Names* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "selectBoundaryCellsFrom")
envSetVal("layoutXL.placement" "selectBoundaryCellsFrom" 'cyclic "Cell Names")
```

Related Topics

List of Custom Digital Placer Environment Variables

selectedBoundaryCells

```
layoutXL.placement selectedBoundaryCells string "list_of_LCVs"
```

Description

Specifies a list of `lib:cell:view` values of the cells to be used as boundary cells. This setting is valid only if selectBoundaryCellsFrom is set to `Cell Names`.

The default value is blank, indicating the absence of boundary cells.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning*
Field	*Boundary Cell – Cell Names* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "selectedBoundaryCells")
envSetVal("layoutXL.placement" "selectedBoundaryCells" 'string "(L1 C1 V1) (L2 C2
V2)")
```

Related Topics

List of Custom Digital Placer Environment Variables

selectedFillerCells

```
layoutXL.placement selectedFillerCells string "list_of_LCVs"
```

Description

Specifies a list of `lib:cell:view` values of the cells to be used as filler cells.

The default value is blank, indicating the absence of filler cells.

GUI Equivalent

Command	*Place – Custom Digital – Insert/Delete Fillers*
Field	*Fillers – Cell Names* (Insert/Delete Filler Cells Form)

Examples

```
envGetVal("layoutXL.placement" "selectedFillerCells")
envSetVal("layoutXL.placement" "selectedFillerCells" 'string "(L1 C1 V1) (L2 C2
V2)")
```

Related Topics

List of Custom Digital Placer Environment Variables

selectFillerCellsFrom

```
layoutXL.placement selectFillerCellsFrom cyclic { "Component Types" | "Cell Names"
     }
```

Description

Specifies whether filler cells are to be selected based on their component types or cell names.

The default is `Component Types`.

GUI Equivalent

Command	*Place – Custom Digital – Insert/Delete Fillers*
Field	*Fillers – Component Types, Cell Names* (Insert/Delete Filler Cells Form)

Examples

```
envGetVal("layoutXL.placement" "selectFillerCellsFrom")
envSetVal("layoutXL.placement" "selectFillerCellsFrom" 'cyclic "Cell Names")
```

Related Topics

List of Custom Digital Placer Environment Variables

selectedTapCells

```
layoutXL.placement selectedTapCells string "list_of_LCVs"
```

Description

Specifies a list of `lib:cell:view` values of the cells to be used as tap cells. This setting is valid only if selectTapCellsFrom is set to `Cell Names`.

The default value is blank, indicating the absence of tap cells.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning*
Field	*Tap Cell – Cell Names* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "selectedTapCells")
envSetVal("layoutXL.placement" "selectedTapCells" 'string "(L1 C1 V1) (L2 C2 V2)")
```

Related Topics

List of Custom Digital Placer Environment Variables

selectTapCellsFrom

```
layoutXL.placement selectTapCellsFrom cyclic { "Component Types" | "Cell Names" }
```

Description

Specifies whether tap cells are to be selected based on their component types or cell names.

The default is `Component Types`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning*
Field	*Tap Cell – Component Types, Cell Names* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "selectTapCellsFrom")
envSetVal("layoutXL.placement" "selectTapCellsFrom" 'cyclic "Cell Names")
```

Related Topics

List of Custom Digital Placer Environment Variables

selectedOnly

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

Description

Places only the currently selected objects. When switched off, the placer automatically places all the objects in the cellview.

The default is `nil`.

GUI Equivalent

Command	*Place – Custom Digital – Auto Placement*
Field	*Place Selected Only* (Auto Placer)

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

showProgress

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

Description

Displays the Placement Status window, which indicates the placer's progress and provides statistics on the placement run.

The default is `nil`.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

skipCellRailPattern

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

Description

Ignores rail patterns in masters while running the Custom Digital Placer. Therefore, the cells are not flipped.

Set this environment variable when cell placement must follow the orientation set on the row, instead of individual cells.

The default value is `nil`.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

startCapCellName

```
layoutXL.placement startCapCellName string "cell_name"
```

Description

Specifies the cell name of the cellview to be placed as a start cap.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "startCapCellName")
envSetVal("layoutXL.placement" "startCapCellName" 'string "STARTCAP")
```

Related Topics

List of Custom Digital Placer Environment Variables

startCapLibName

startCapViewName

endCapLibName

endCapCellName

endCapViewName

startCapLibName

```
layoutXL.placement startCapLibName string "library_name"
```

Description

Specifies the library name of the cellview to be placed as a start cap.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "startCapLibName")
envSetVal("layoutXL.placement" "startCapLibName" 'string "TESTLIB")
```

Related Topics

List of Custom Digital Placer Environment Variables

insertCapCells

startCapCellName

startCapViewName

endCapLibName

endCapCellName

endCapViewName

startCapViewName

```
layoutXL.placement startCapViewName string "view_name"
```

Description

Specifies the view name of the cellview to be placed as a start cap.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "startCapViewName")
envSetVal("layoutXL.placement" "startCapViewName" 'string "layout")
```

Related Topics

List of Custom Digital Placer Environment Variables

startCapLibName

startCapCellName

endCapLibName

endCapCellName

endCapViewName

subContInAlternateRow

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

Description

Specifies if the standard cell substrate contacts (tap cells) must be placed in alternate rows, such as 1st, 3rd, and so on.

The default is `nil`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning– Tap Cell* tab
Field	*Skip Row Count* (Placement Planning)

Examples

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

Related Topics

Placing Standard Cell Substrate Contacts in Alternate Rows

List of Custom Digital Placer Environment Variables

subContMaxSpacing

```
layoutXL.placement subContMaxSpacing float float_number
```

Description

Specifies the maximum spacing allowed between the standard cell substrate contacts in a row.

By default, no maximum contact spacing value is specified.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Tap Cell* tab
Field	*Flexible: Maximum Spacing* (Pin Placement)

Examples

```
envGetVal("layoutXL.placement" "subContMaxSpacing")
envSetVal("layoutXL.placement" "subContMaxSpacing" 'float 16.5)
```

Related Topics

List of Custom Digital Placer Environment Variables

subContMinSpacing

`layoutXL.placement subContMinSpacing float` *`float_number`*

Description

Specifies the minimum spacing allowed between the tap cells (standard cell substrate contacts) in a row.

By default, no minimum contact spacing value is specified.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Tap Cell* tab
Field	*Flexible: Minimum Spacing* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "subContMinSpacing")
envSetVal("layoutXL.placement" "subContMinSpacing" 'float 12.0)
```

Related Topics

List of Custom Digital Placer Environment Variables

substrateContactType

`layoutXL.placement substrateContactType string "`*`compTypeName`*`"`

Description

Specifies the name of the component type that contains the tap cells (standard cell substrate contacts) you want to insert in the spaces between the standard cells in a row. This setting is valid only if selectTapCellsFrom is set to `Component Types`.

Note: This environment variable is considered only when insertSubstrateContacts is set to `t`. The component type you specify must have component class *STDSUBCONT*.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Tap Cell* tab
Field	*Component Types* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "substrateContactType")
envSetVal("layoutXL.placement" "substrateContactType" 'string "myTapCell")
```

Related Topics

List of Custom Digital Placer Environment Variables

supplyPattern

```
layoutXL.placement supplyPattern cyclic { "G-P-P-G" | "P-G-G-P" | "G-P" | "P-G" }
```

Description

Specifies how power and ground rails alternate between rows.

The default is `G-P-P-G`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Row* tab
Field	*Power/Ground Pattern* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "supplyPattern")
envSetVal("layoutXL.placement" "supplyPattern" 'cyclic "G-P-P-G")
envSetVal("layoutXL.placement" "supplyPattern" 'cyclic "P-G-G-P")
envSetVal("layoutXL.placement" "supplyPattern" 'cyclic "G-P")
envSetVal("layoutXL.placement" "supplyPattern" 'cyclic "P-G")
```

Related Topics

List of Custom Digital Placer Environment Variables

supplyPosition

```
layoutXL.placement supplyPosition cyclic { "Inside" | "Outside" | "Center" }
```

Description

Specifies the position of rails relative to rows. This option can be used only when generateRails is set to `t`.

The default is `Inside`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Row* tab
Field	*Rail Position* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "supplyPosition")
envSetVal("layoutXL.placement" "supplyPosition" 'cyclic "Inside")
envSetVal("layoutXL.placement" "supplyPosition" 'cyclic "Outside")
envSetVal("layoutXL.placement" "supplyPosition" 'cyclic "Center")
```

Related Topics

List of Custom Digital Placer Environment Variables

templateRowCount

```
layoutXL.placement templateRowCount int integer_number
```

Description

Specifies the exact number of rows to be generated in the given row region when the `numTemplateRowOption` environment variable is set to `Specify`.

The default is `1`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Row* tab
Field	*Number Of Rows – Specify* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "templateRowCount")
envSetVal("layoutXL.placement" "templateRowCount" 'int 3)
```

Related Topics

List of Custom Digital Placer Environment Variables

topBotRowPlaceable

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

Description

Specifies whether standard cells can be placed over the boundary cells that are placed on the top and bottom rows.

The default is `nil`.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

topRowBdryCell

`layoutXL.placement topRowBdryCell string "`*`rowCell_name`*`"`

Description

Specifies the row cell to be inserted at the top of standard cells.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Boundary Cell* tab
Field	*Top/Bottom Cells* (Placement Planning)

Examples

```
envGetVal("layoutXL.placement" "topRowBdryCell")
envSetVal("layoutXL.placement" "topRowBdryCell" 'string "ROW_CELL_TOP")
```

Related Topics

List of Custom Digital Placer Environment Variables

topTapBdryCell

```
layoutXL.placement topTapBdryCell string "tapCell_name"
```

Description

Specifies the tap cell to be inserted at the top of the standard cell.

GUI Equivalent

Examples

```
envGetVal("layoutXL.placement" "topTapBdryCell")
envSetVal("layoutXL.placement" "topTapBdryCell" 'string "FILLER_CELL_TT")
```

Related Topics

List of Custom Digital Placer Environment Variables

usePartialTemplate

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

Description

Allows use of partial row templates to fill the placement region. Therefore, the selected row template is repeated multiple times, and the remaining space at the top is filled with a partial row template. When set to `nil`, the remaining space is left empty. This setting is valid only when `numRowOption` is set to `Maximum`.

The default is `t`.

GUI Equivalent

Command	*Place – Custom Digital – Placement Planning – Row tab*
Field	*Use Partial Template* (Placement Planning)

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

vcpCellBoundaryLPPs

```
layoutXL.placement vcpCellBoundaryLPPs string "(layer purpose)..."
```

Description

Specifies the layer-purpose pairs (LPPs) the placer uses to derive the cell boundary for standard cells. Use this environment variable to specify a cell boundary using LPPs that are not included in the default layers.

The cell boundary is derived from the available shapes inside the standard cells in following order:

1. The list of layer-purpose pairs from `vcpCellBoundaryLPPs`. If multiple LPPs are specified, the placer uses only the first LPP encountered in the cells.
2. The PR Boundary object. If the PR Boundary is rectilinear and a `snapBoundary` exists, the `snapBoundary` is used.
3. The `snapBoundary` object.
4. The `(prboundary drawing)` layer.
5. The `(prboundary boundary)` layer.
6. The `(instance drawing)` layer.
7. The `(cellBoundary drawing)` layer.
8. The overall bounding box (excluding labels).

If none of the above LPPs exists in the design, the placer draws the boundary such that it encloses all the objects within the cell.

GUI Equivalent

Command	*Place – Custom Digital – Auto Placement*
Field	*Cell Boundary LPP(s)*

Note: To hide the GUI field, the vcpCellBoundaryUI environment variable should be set to `nil`.

Examples

```
envGetVal("layoutXL.placement" "vcpCellBoundaryLPPs")
```

```
envSetVal("layoutXL.placement" "vcpCellBoundaryLPPs" 'string "(prboundary
drawing)")
```

Related Topics

List of Custom Digital Placer Environment Variables

vcpCellBoundaryUI

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

Description

Controls the display of the *Cell Boundary LPPs* field on the Auto Placer form, which specifies the layers that have been used to derive the place and route boundary.

The default is `t`.

Tip

You can also use the vcpCellBoundaryLPPs environment variable to specify these LPPs.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

vcpConductorDepth

`layoutXL.placement vcpConductorDepth int` *`integer_number`*

Description

Specifies the conductor depth for the Design Framework II (DFII) to Virtuoso Custom Digital Placer translation.

Tip

The default value is `2` and is recommended for device level placement. For block and standard cell placements, Cadence recommends that you set the `vcpConductorDepth` to `0`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "vcpConductorDepth")
envSetVal("layoutXL.placement" "vcpConductorDepth" 'int 3)
```

Related Topics

List of Custom Digital Placer Environment Variables

vcpEnableMultiVoltagePlacement

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

Description

Enables and disables the placer's multi-voltage placement capability.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

vcpGlobalPathScript

```
layoutXL.placement vcpGlobalPathScript string "fileName"
```

Description

Specifies a script that returns the full network path for the current working directory. This path is required if you are running the placement job on your Load Balancing Service.

If there is no script specified, the placer uses the current path (`/net/hostname/currentPath`.)

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "vcpGlobalPathScript")
envSetVal("layoutXL.placement" "vcpGlobalPathScript" 'string "myScript.txt")
```

Related Topics

List of Custom Digital Placer Environment Variables

vcpKeepoutDepth

`layoutXL.placement vcpKeepoutDepth int` *`integer_number`*

Description

Specifies the keepout depth for the DFII to Virtuoso Custom Digital Placer translation.

Tip

The default value is `2` and is recommended for device level placement. For block and standard cell placements, Cadence recommends that you set the `vcpKeepoutDepth` to `0`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "vcpKeepoutDepth")
envSetVal("layoutXL.placement" "vcpKeepoutDepth" 'int 3)
```

Related Topics

List of Custom Digital Placer Environment Variables

vcpRulesConstraintGroup

```
layoutXL.placement vcpRulesConstraintGroup string "constraint_group_names"
```

Description

Specifies the constraint group that contains the placement rules for the current technology.

GUI Equivalent

Command	*Place – Custom Digital – Auto Placement*
Field	*Constraint Group* (Auto Placer)

Examples

```
envGetVal("layoutXL.placement" "vcpRulesConstraintGroup")
envSetVal("layoutXL.placement" "vcpRulesConstraintGroup" 'string "myCG")
```

Related Topics

List of Custom Digital Placer Environment Variables

vcpVerboseLevel

```
layoutXL.placement vcpVerboseLevel int { 0 | 1 | 2 | 3 }
```

Description

Specifies the verbosity of the placer's message output.

For example, `0` specifies that the only status reported is the final set of placement statistics; `3` specifies full messaging.

The default is `1`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL.placement" "vcpVerboseLevel")
envSetVal("layoutXL.placement" "vcpVerboseLevel" 'int 3)
```

Related Topics

List of Custom Digital Placer Environment Variables

vcpWriteToCIW

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

Description

Specifies that placer output is written to the command interpreter window (CIW).

The default is `nil`.

GUI Equivalent

None

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables

vpaAutoLevel1Switch

```
layoutXL.placement vpaAutoLevel1Switch boolean { t | nil }
```

Description

Automatically zooms to the selected soft block and switches to Level-1 Pins mode when you click a soft block while using the Pin Planner tab in Top Level Pins mode.

The default is `nil`, which means that when you click a soft block in Top Level Pins mode, the soft block is selected but you remain at the top level.

GUI Equivalent

Command	*Place – Pin Placement*
Field	*Auto-Switch to Level-1* (Pin Placement)

Examples

```
envGetVal("layoutXL.placement" "vpaAutoLevel1Switch")
envSetVal("layoutXL.placement" "vpaAutoLevel1Switch" 'boolean t)
envSetVal("layoutXL.placement" "vpaAutoLevel1Switch" 'boolean nil)
```

Related Topics

List of Custom Digital Placer Environment Variables

vpaAutoZoomEnable

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

Description

Automatically enables the zoom functionality when the layout view is opened. When set to `t`, the layout canvas zooms to the selected pins.

The default is `nil`.

GUI Equivalent

Command	*Place – Pin Placement*
Field	*Zoom* (Pin Placement)

Examples

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

Related Topics

vpaAutoZoomLevel

List of Custom Digital Placer Environment Variables

vpaAutoZoomLevel

`layoutXL.placement vpaAutoZoomLevel int` *`integer_number`*`}`

Description

Sets the automatic zoom level for the selected pins in the layout canvas to an integer value. The acceptable zoom levels are between `0` and `4`.

The default is `0`.

Note: If you set a zoom level of value less than `0`, the minimum zoom value automatically defaults to `0`. If you set a zoom level of value more than `4`, the maximum zoom value automatically defaults to `4`.

GUI Equivalent

Command	*Place – Pin Placement*
Tool name	*Zoom slider* (Pin Placement)

Examples

```
envGetVal("layoutXL.placement" "vpaAutoZoomLevel")
envSetVal("layoutXL.placement" "vpaAutoZoomLevel" 'int 3)
```

Related Topics

vpaAutoZoomEnable

List of Custom Digital Placer Environment Variables

vpaUpdateConstraints

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

Description

Automatically updates any constraints that are applicable for the design based on the settings in the *Pin Spacing* section of the Pin Planner tab. Constraints are updated when you click the *Space* button in the form (and not by clicking the *Apply* button in the *Attributes* section).

The default is `t`.

GUI Equivalent

Command	*Place – Pin Placement*
Field	*Update Constraints* (Pin Placement)

Examples

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

Related Topics

List of Custom Digital Placer Environment Variables