cādence®

Virtuoso Module Generator User Guide

Product Version IC23.1
November 2023

Printed in the United States of America.

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

Contents

1

Module Generators (Modgens)

Virtuoso® Module Generators (Modgens) provide a way generate multiple Pcell instances into a complex, highly matched, structured array. You can use the Modgen tool to specify the devices to be arrayed, specify an interdigitation pattern, and insert dummy devices, body contacts, and guard rings. You can also control the routing style and generate internal routing geometry.

Modgens can be created from either the schematic or the layout cellview. When you create a Modgen from schematic, the tool creates a Modgen constraint on the schematic, which can then be generated in the layout using the *Generate All From Source* or *Generate Selected From Source* commands.

When run from schematic, by default, a Modgen creates a temporary physConfig. You can use the `modgenPhysConfigs` environment variable to specify a different physconfig for the Modgen.

Related Topics

modgenPhysConfigs

Modgen Rapid Adoption Kit

Technology File Requirements for Modgens

Creating a Modgen

The Modgen Workspace

Technology File Requirements for Modgens

The following Modgen features have specific technology file requirements:

Modgen Feature	Technology File Requirement
Routing	Routing in Modgens requires a complete layout function section for metal and poly layers and basic width and spacing rules on the metal and poly layers to be defined in the technology file.
Body Contacts	To use body contacts in Modgens, ensure that the technology file includes either a standard or a custom via definition that defines connections to well or active layers. The well and active layers must also be defined in the Layer Function section of the technology file.
Guard Rings	To use guard rings in Modgens, define the required Multiple Part Paths (MPPs) in the technology file. See multipartPathTemplates.

Technology Group and Constraint Groups

Modgens read rules from the user-defined constraint groups and the foundry technology groups in the technology file.

The default lookup path for these groups is in the following sequence:

1. Specified tool constraint group stored either in the design or in the technology library.
2. Design-level default constraint group.
3. Foundry constraint group stored in the technology library.

The tool constraint group is stored outside the foundry constraint group. Before running the tool, specify the required tool constraint group using the modgenPlacementConstraintGroup environment variable.

If the string in the environment variable is empty, the foundry constraint group is used. If the string has an invalid value, a warning is displayed and the foundry constraint group is used.

Modgens also read from the default wire editing constraint group, if set.

See Constraint Groups.

Placement Grid Support

Placement grid support is a method that keeps all shapes on specific grids per layer. The placement grid rule is obeyed at a higher precedence than custom spacing or detailed spacing rules. So, even if, according to the custom spacing rule, an instance is placed off the instance grid, it is automatically snapped to the next nearest placement grid. In addition, the origin of the Modgen figGroup is placed on the placement grid. This ensures correct placement of instances on their respective placement grids when the entire Modgen is placed by the placer.

Use environment variable chainPermutePins to control the application of the placement grid rule.

If `aapUsePlacementGrid` is set to `t`, you must set the following placement grid rules either in the techfile or in a custom Constraint Group that the Modgen refers to.

- **horizontalPlacementOffset** – Specifies the horizontal offset from the origin for the vertical placement grid
- **horizontalPlacementPitch** – Specifies the horizontal spacing between each vertical placement grid line
- **verticalPlacementOffset** – Specifies the vertical offset from the origin for the horizontal placement grid.
- **verticalPlacementPitch** – Specifies the vertical space between each horizontal placement grid line.

Example:

```
(placementGrids
    (verticalPitch 1.0)
    (verticalOffset 0.0)
    (horizontalPitch 1.5)
    (horizontalOffset 0.5)
)
```

Situation 1: The placement grid values exist and are on the grid with respect to the manufacturing grid.

During placement, the origin of each instance is snapped to the nearest placement grid horizontally and vertically. For the Modgen, the instances are snapped up.

Situation 2: The placement grid values exist, but are not on the grid with respect to the manufacturing grid.

A warning is displayed and placement continues using only the manufacturing grid.

See Placement and Alignment Constraints.

Related Topics

Technology File Organization

Technology File Via Definitions and Via Specifications

Creating a Modgen

The Modgen Workspace

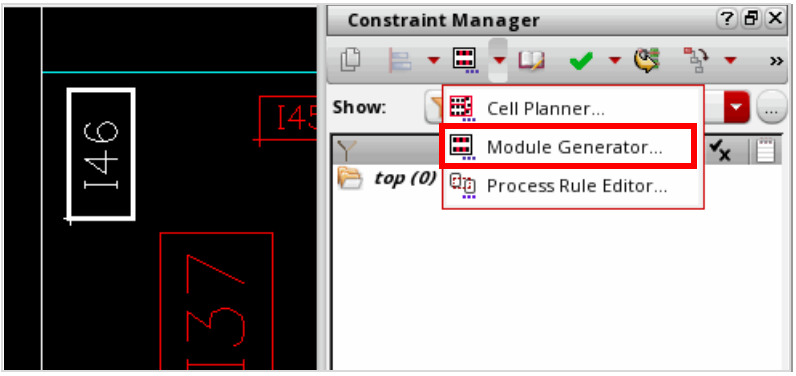
Creating a Modgen

A Modgen is a placeable object that combines multiple device instances, dummy instances, Modgens, figGroups, or mosaics. Modgens can be created from both, schematic and layout designs. There are several ways to create a Modgen:

- Using the Constraint Manager Toolbar
- Using the Place Menu
- Using Convert to Modgen
- Using SKILL Functions

Using the Constraint Manager Toolbar

1. Select the required devices from the schematic or layout design.
2. Choose *Module Generator* from the Constraint Manager toolbar.



A new Modgen constraint is created and the Modgen Editor is invoked.

Using the Place Menu

1. Select the instances, Modgens, figGroups, or mosaics from the schematic or layout design.
2. Select *Place—Modgen—Create/Edit Modgen*.

A new Modgen constraint is created. However, the Modgen Editor is not invoked. Use the Modgen on-canvas commands to edit the Modgen directly in the layout canvas.

Using Convert to Modgen

A mosaic is a compact array of instances that belong to the same mfactored instance, and therefore have the same master and connectivity. You can convert a mosaic in the layout design to a Modgen. To do this:

1. Select the mosaic to be converted into a Modgen.
2. Select *Edit—Convert—To Modgen*. Alternatively, right-click the mosaic in the canvas and choose *Convert to Modgen* from the shortcut menu.

A new Modgen constraint is created. However, the Modgen Editor is not invoked. Use the Modgen on-canvas commands to edit the Modgen directly in the layout canvas.

Using SKILL Functions

Use one of the following SKILL Functions:

- ciConCreate: Creates the Modgen constraint.
- gpeCreateSandbox: Creates a Modgen sandbox object that includes the specified instances.
- mgCreateModgenAsLayout**:** Creates a Modgen from the list of selected instances and keeps the instance locations inside the Modgen, as close as possible to their original locations in the layout.
- mgCreateModgenConstraintAsLayout**:** Creates a Modgen constraint that uses the current layout to drive the initial Modgen constraint and row/column assignments.

Related Topics

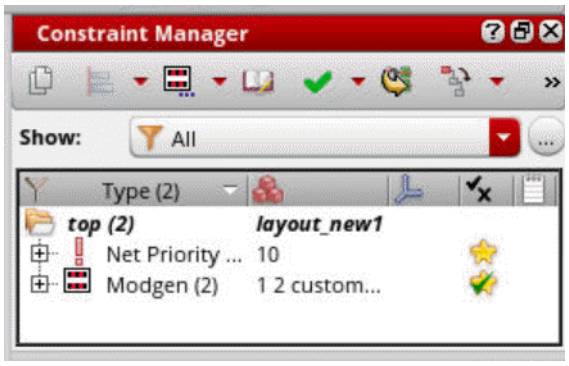
Modgen On-Canvas Commands

Creating Modgens (Video)

Open Modgens in the Modgen Editor

Use one of the following methods to open an existing Modgen in the Modgen editor:

- Double-click the Modgen in the design.
- Double-click the Modgen constraint in the Constraint Manager toolbar.

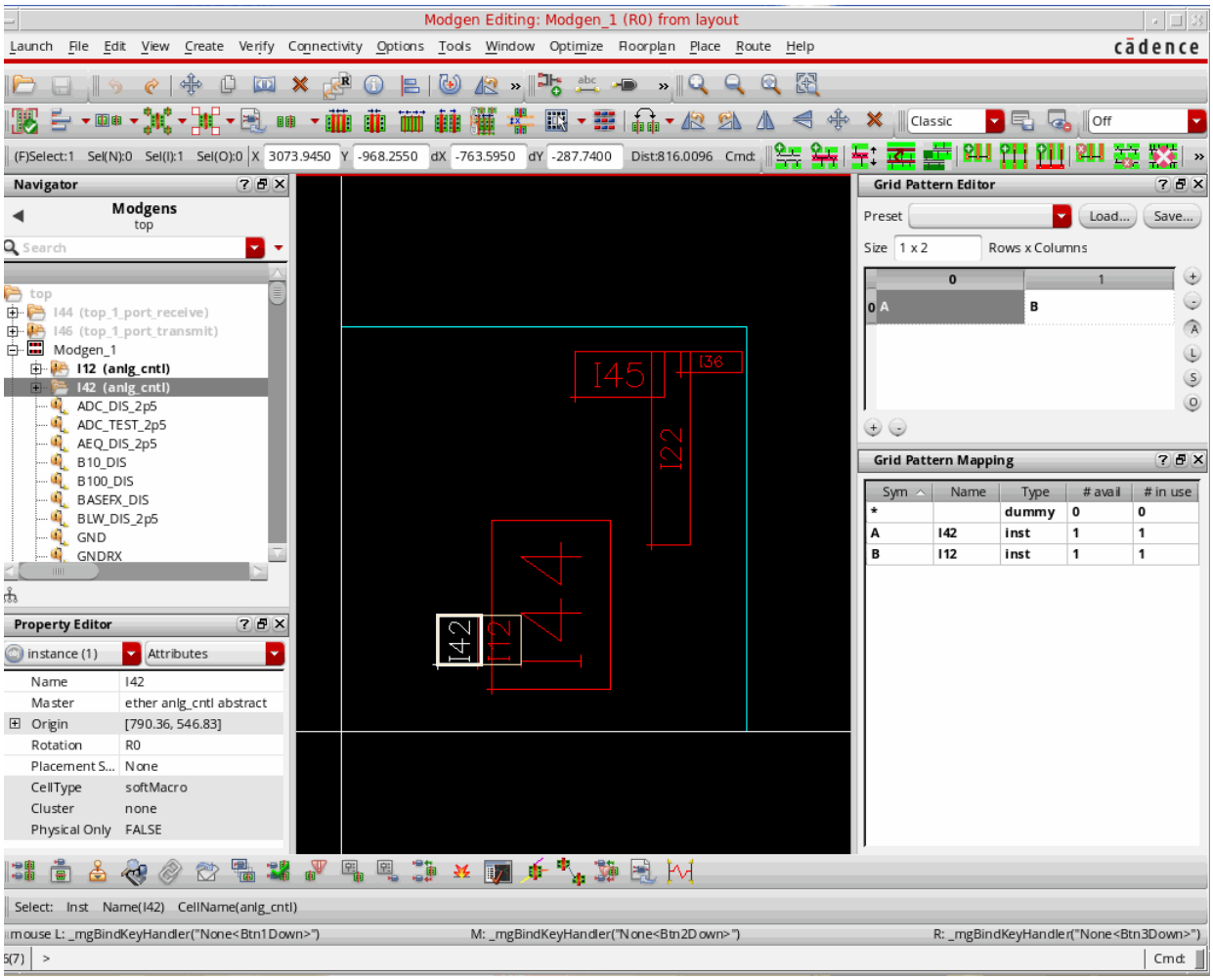


If the Constraint Manager isn't visible, choose *Windows – Assistants – Constraint Manager*. If the *Modgen* icon isn't visible, click the >> button.

- Select the Modgen and choose *Place—Modgen—Create/Edit Modgen*.
- Select the Modgen and choose the Module Generator icon in the Constraint Manager assistant toolbar.
- Select the Modgen figGroups in the layout canvas, right click, and select *Edit Modgen*.

  (Layout only) If you have launched Modgen from the layout, place the module in the layout.

The module appears in the layout window and the layout window moves into Module Generator Edit mode. The Modgen Editor is displayed. The default workspace is displayed.



You can either use the existing default workspace or define a different workspace and set it as the default workspace.

Modgen commands are invoked from the Modgen Editor, pop-up menus, and the *Modgen Placement* toolbar.

Typical Virtuoso commands such as *Edit—Delete* are not supported in the Modgen Editor mode.

If you want to perform some interactive customizations, such as moving devices inside a Modgen, you need to first disable the Modgen constraint by selecting it in the Constraint Manager and setting the Enabled parameter to False. This will turn the Modgen into a

standard database figGroup. Then you need to do an Edit in Place to customize the Modgen. Any operation, such as UCN, should not affect any custom edits in this state.

This operation should only after completing the final layout and configuration of the Modgen. After making these modifications, if you re-enable the Modgen constraint by setting the Enabled parameter to True, all modifications performed outside the Modgen environment will be lost.

When a Modgen is invoked from a schematic cellView, it is displayed in the Modgen Previewer window. Changes made to the Modgen are saved in the constraint view for that schematic cellView. Later, you can transfer these changes from the schematic to the layout cellView.

Related Topics

mgCreateOrEdit (SKILL function)

Specifying Modgen Parameters

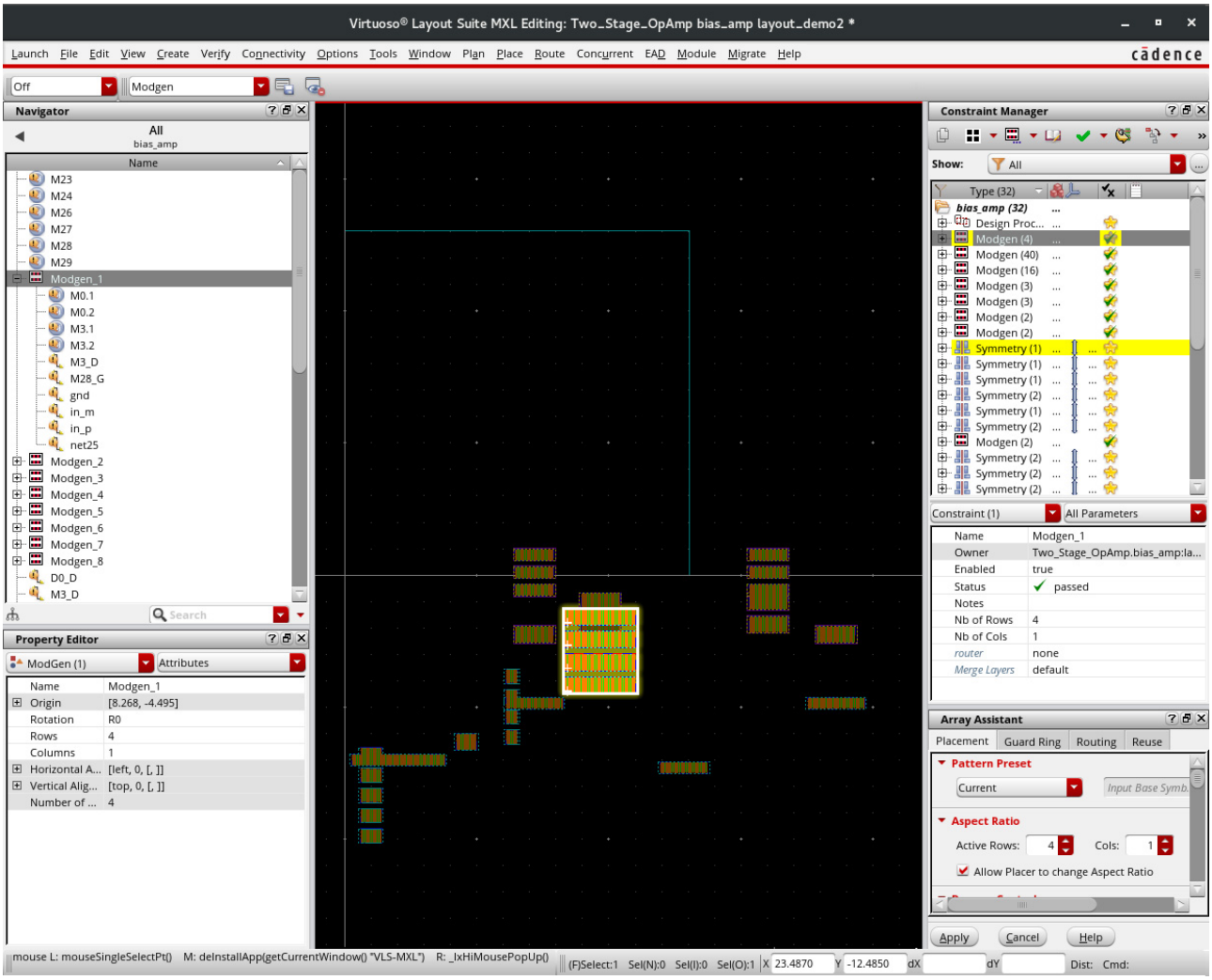
Modgen On-Canvas Commands

Creating a Modgen

The Modgen Workspace

Use the `modgenWindowConfigFile` environment variable to specify the default workspace to be loaded when Modgen is invoked. If the workspace is invalid, a warning message is displayed and the default Modgen Edition workspace is loaded.

The following image shows the Modgen default workspace:



Related Topics

Open Modgens in the Modgen Editor

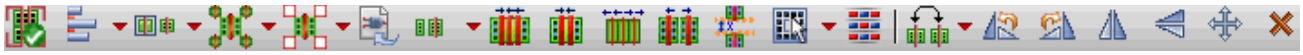
Creating a Modgen

Modgen Placement Toolbar

Row Regions in Modgens

Modgen Placement Toolbar

The Modgen Editor includes the *Modgen Placement* toolbar:



As with all other toolbars, you can enable or disable this toolbar by right-clicking anywhere in the main toolbar area and selecting the required toolbar name. You can use the handle on the left side of the toolbar to reposition it anywhere within the Modgen Editor window.

You can use the Toolbar Manager to make incremental and local changes to the toolbar. For example, you can add, edit, or remove toolbar items. You can also create a workspace for the current Modgen session.

Related Topics

Getting Started with Workspaces

Working with Workspaces

Using Toolbar Manager

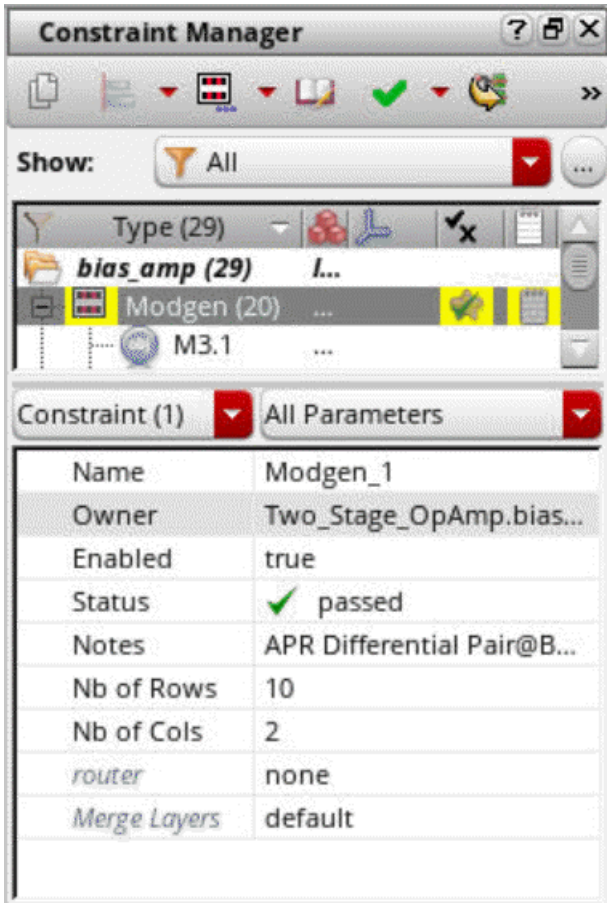
modgenWindowConfigFile (Environment Variable)

Specifying Modgen Parameters

You can specify Modgen parameters, such as the number of rows and columns, the type of router, and the interdigitation pattern, in the Constraints Manager.

1. Select the required Modgen in the Constraint Manager.

   The Modgen parameters are listed, as shown below.

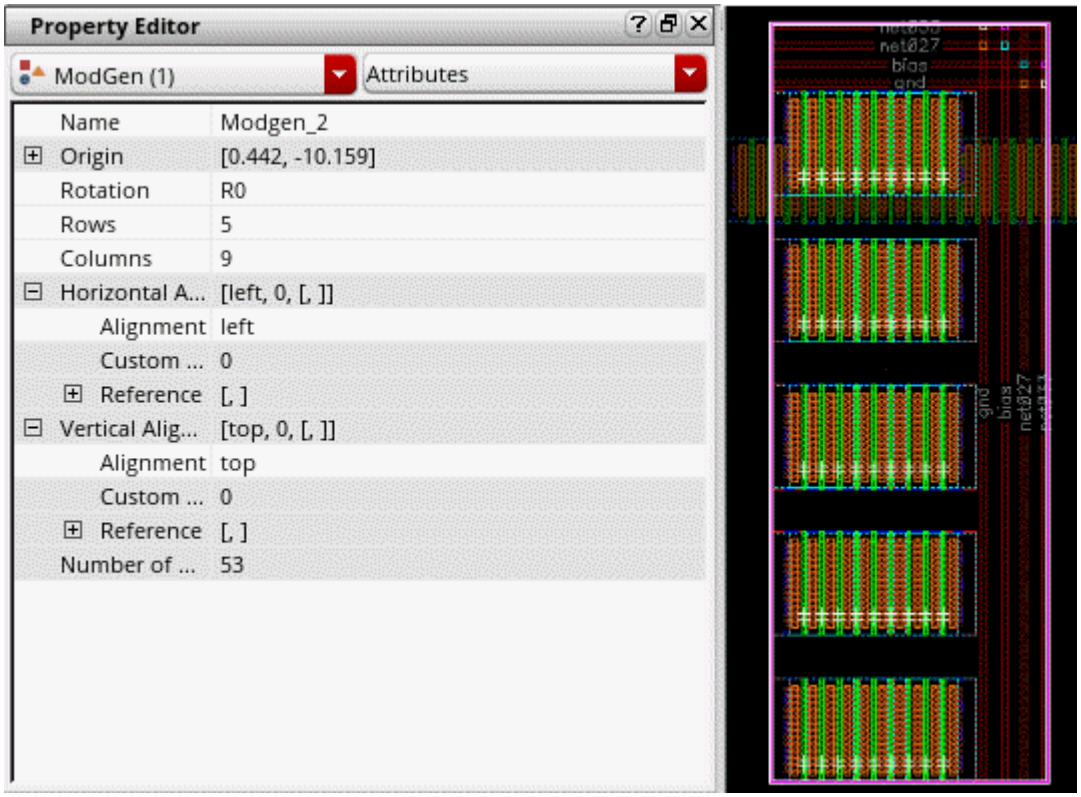


2. Edit the required parameters: *Name*, *Status*, and *Notes*.
3. Specify the number of rows for the array in the *Nb of Rows* field.
4. Specify the number of columns for the array in the *Nb of Cols* field.
5. Select a router to be used to route the Modgen.
6. Use the *Merge Layers* option to specify the layers to be merged in the Modgen.

Tip

While other parameters are listed in the Module Generator panel, there are easier graphical interfaces to update these parameters available in the *Modgen Placement* toolbar.

You can use the Property Editor assistant to interactively update the properties of individual Modgen instances and Modgen arrays containing stack-type members. Any changes you make to the instance properties are immediately applied to the Modgen in the layout canvas, as displayed in the figure below.



Related Topics

Specifying Modgen Parameters

Adding and Deleting Dummies in the Modgen Editor

Adding and Removing Modgen Body Contacts

Modgen Guard Rings

Modgen Device Abutment

Row Regions in Modgens

(Layout EXL and Higher Tiers) All Modgen creation, modification, and regeneration commands, including the Modgen on-canvas commands, recognize row regions. Therefore, a Modgen, when created or regenerated, automatically fits into an existing row region that is valid for the members in the Modgen configuration. If invalid, the row region is ignored during Modgen generation.

When a Modgen is created interactively from the layout design, the orientations of the Modgen members may be altered to ensure compact placement of the Modgen members within the placement rows.

(Virtuoso Advanced Node for Layout Only) To support FinFETs, the layout editor automatically snaps the Pcells in Modgens to the snap pattern. The entire Modgen block is snapped to the top-level snap pattern shape.

Related Topics

Object Snapping to Local Snap Pattern Shapes

Creating Rows

Row-based Functions

Placement Area Definition Functions

Closing and Regenerating a Modgen

After making the required changes to a Modgen, use one of the following methods to quit the Modgen Editor mode:

- Click the *Close* button in the upper-right corner of the window. Use the `modgenSaveOnClosePreviewWindow` environment variable to control the behavior when using the Close button.
- Click *Exit* on the toolbar.
- Press `Shift` + `B`.
- Right-click and select *Exit Modgen*.

To customize the Modgen geometry after the Modgen has been created and edited using the Modgen editor in the Layout Editor, disable the Modgen constraint. That will change the Modgen FigGroup type to *none*, which will then allow you to utilize *Hierarchy — Edit In Place* to customize any of the Modgen geometry, or add any geometry to the FigGroup.

After performing certain tasks, such as moving a Modgen from one placement area to another, you may want to regenerate the Modgen. Here are the steps to regenerate a Modgen:

1. Select one or more Modgen-type figGroups in the layout canvas.
2. Right-click to display the shortcut menu.
3. Select *Regenerate Modgen*.

Any geometry that was created or customized inside the figGroup is discarded and replaced with the regenerated Modgen geometry. Other edits or customizations in the cellview, such as constraint addition, deletion, editing, or instance parameter changes, are retained.

Related Topics

Open Modgens in the Modgen Editor

Specifying Modgen Parameters

mgRegenerateModgen

2

Automatic Generation of Modgens using the Array Assistant

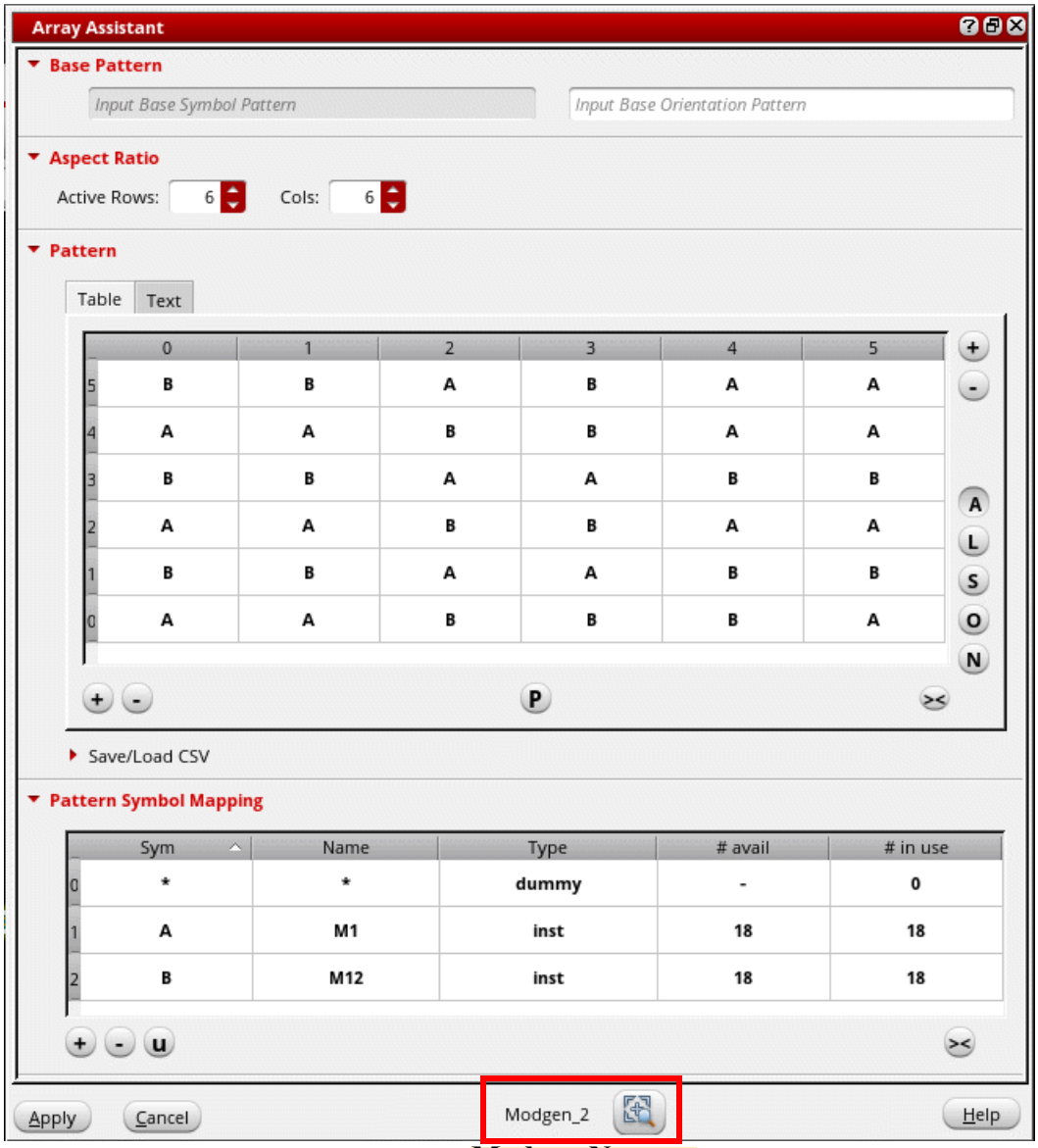
The Modgen Array Assistant is a modeless, dockable assistant that facilitates the visualization and editing of Modgen device arrays in an abstracted manner. It is a unified interface that allows you to quickly create and edit Modgen device arrays through its use model, which batches edits together before applying them to the layout canvas.

When you select a Modgen in the layout design, its settings are automatically loaded in the Array Assistant. The name of the Modgen is displayed at the bottom of the Array Assistant.

On closing and re-opening the Array Assistant, settings of the last viewed Modgen are automatically loaded.

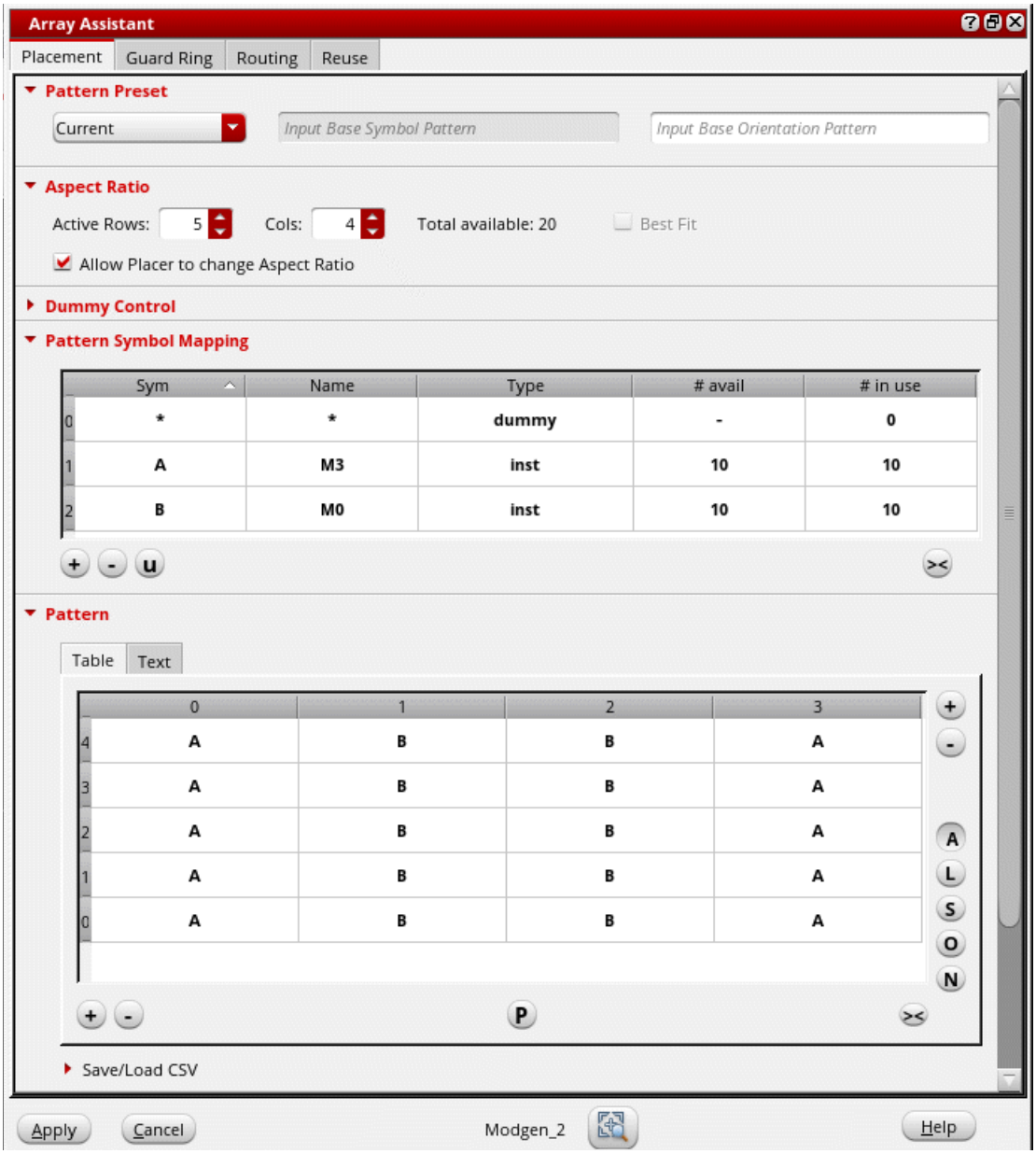
When launched from Layout XL, the Array Assistant checks out two Virtuoso Layout Suite GXL tokens (95321). When launched from higher tiers, no additional licenses are required.

- In Layout XL, the Array Assistant lets you specify the Modgen placement settings.



Modgen Name

- In Layout EXL and higher tiers, the Array Assistant lets you specify the Modgen placement settings, the guard ring options, the topology and routing settings, and the reuse options.



The options in the assistant are organized in four tabs that let you perform the following tasks:

- Define Modgen Placement Settings: Use the *Placement* tab of the Array Assistant to specify the Modgen pattern settings.

- Create Guard Rings: Use the *Guard Ring* tab of the Array Assistant to create guard rings around the Modgens.
- Define Modgen Topology Settings: Use the *Routing* tab of the Array Assistant to define Modgen topology patterns and set routing preferences.

  The *Routing* tab is available only when the Array Assistant is launched from the Constraint Manager assistant. In the Virtuoso automated device placement and routing flow, the tree router is used for routing. Therefore, the *Routing* tab is not available when the Array Assistant is launched from the Auto P&R assistant.
- Reuse Modgen Templates: Use the *Reuse* tab of the Array Assistant to load and save settings to Modgen template files.

Select the instances that you want to include in the Modgen and use the options in the Array Assistant to generate a placed and routed Modgen.

You can directly access the Array Assistant from the layout canvas without opening the Modgen Editor. The Array Assistant is accessible from both the Constraint Manager assistant and the Auto P&R assistant (Layout EXL and Higher Tiers).

Related Topics

Array Assistant

Opening Array Assistant From the Constraint Manager

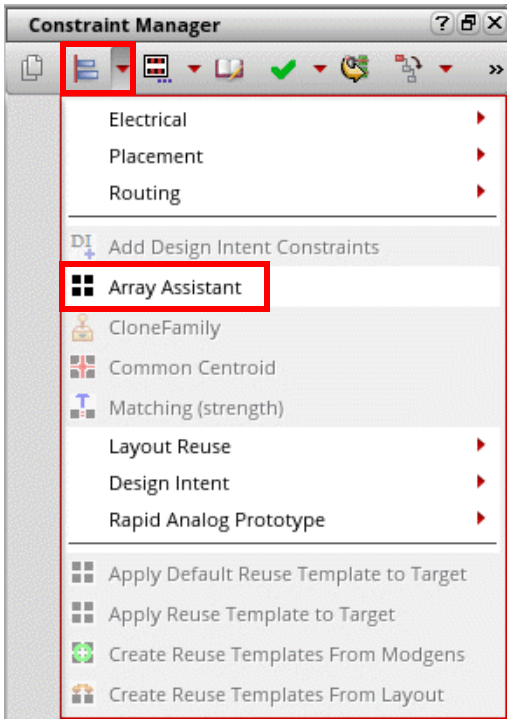
Opening Array Assistant During Device Placement

Opening the Array Assistant During Interactive Placement

Opening Array Assistant From the Constraint Manager

To open the Array Assistant from the Constraint Manager:

1. Select either a Modgen in the Constraint Manager assistant or the required devices in the layout canvas.
2. Open the constraint list, which is the second icon in the Constraint Manager assistant.



3. Choose *Array Assistant*. The Array Assistant appears.

Alternatively, you can select a Modgen or the required instances and press `Control + M`. This bindkey may not open the Array Assistant if the `bindKeys.il` file is loaded. The bindkey assignments in the file take precedence. The `Control + M` bindkey can then be used to swap instances.

Related Topics

Array Assistant

Automatic Generation of Modgens using the Array Assistant

Opening Array Assistant During Device Placement

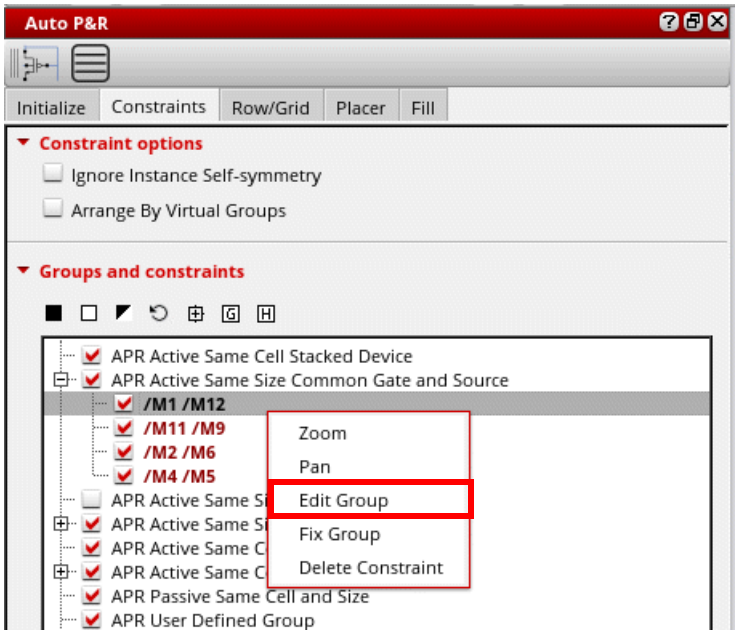
Opening the Array Assistant During Interactive Placement

Opening Array Assistant During Device Placement

(Layout EXL and Higher Tiers) You can open the Array Assistant from the Auto P&R assistant either in the constraint generation step or after running the placer. To invoke the Array Assistant during constraint generation:

1. Ensure that `aprCreateModgens` is set to `t` (default setting). In this mode, grouped devices are generated as Modgens in the automated device placement and routing flow.
2. Initialize the layout.
3. Generate the required device groups.
4. Right-click the required device group in the *View and Edit Constraints* pane.
5. Choose *Edit Group*.

   The Array Assistant is displayed.



Related Topics

Initializing a Layout

Generating Constraints and Constraint Groups

Array Assistant

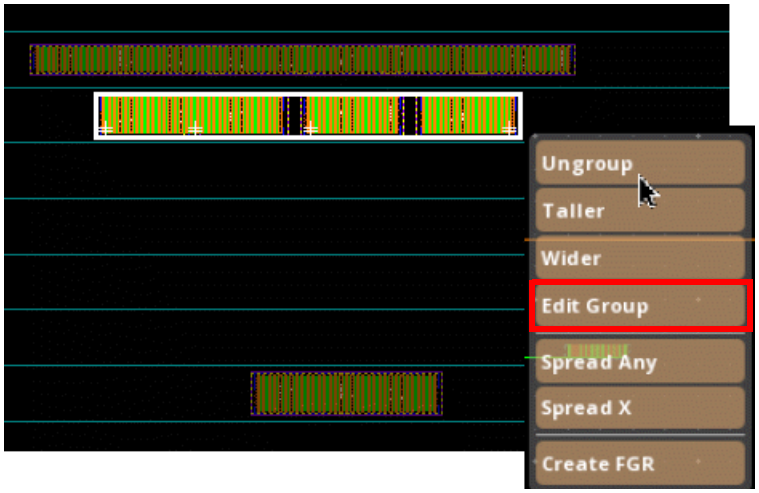
Opening Array Assistant From the Constraint Manager

Opening the Array Assistant During Interactive Placement

Opening the Array Assistant During Interactive Placement

(Layout EXL and Higher Tiers) The Auto P&R assistant lets you create Modgens after running the placer—as part of interactive placement. To do this:

1. Select the Context Menu icon on the toolbar of the Auto P&R assistant.
2. Select the required Modgens or devices in the layout canvas. A context menu is displayed.
3. If you have selected devices in the layout canvas, select *Group* to create a group that includes the selected devices. If you have selected Modgens, skip this step.
4. Select *Edit Group* from the context menu. The Array Assistant appears.



Related Topics

Array Assistant

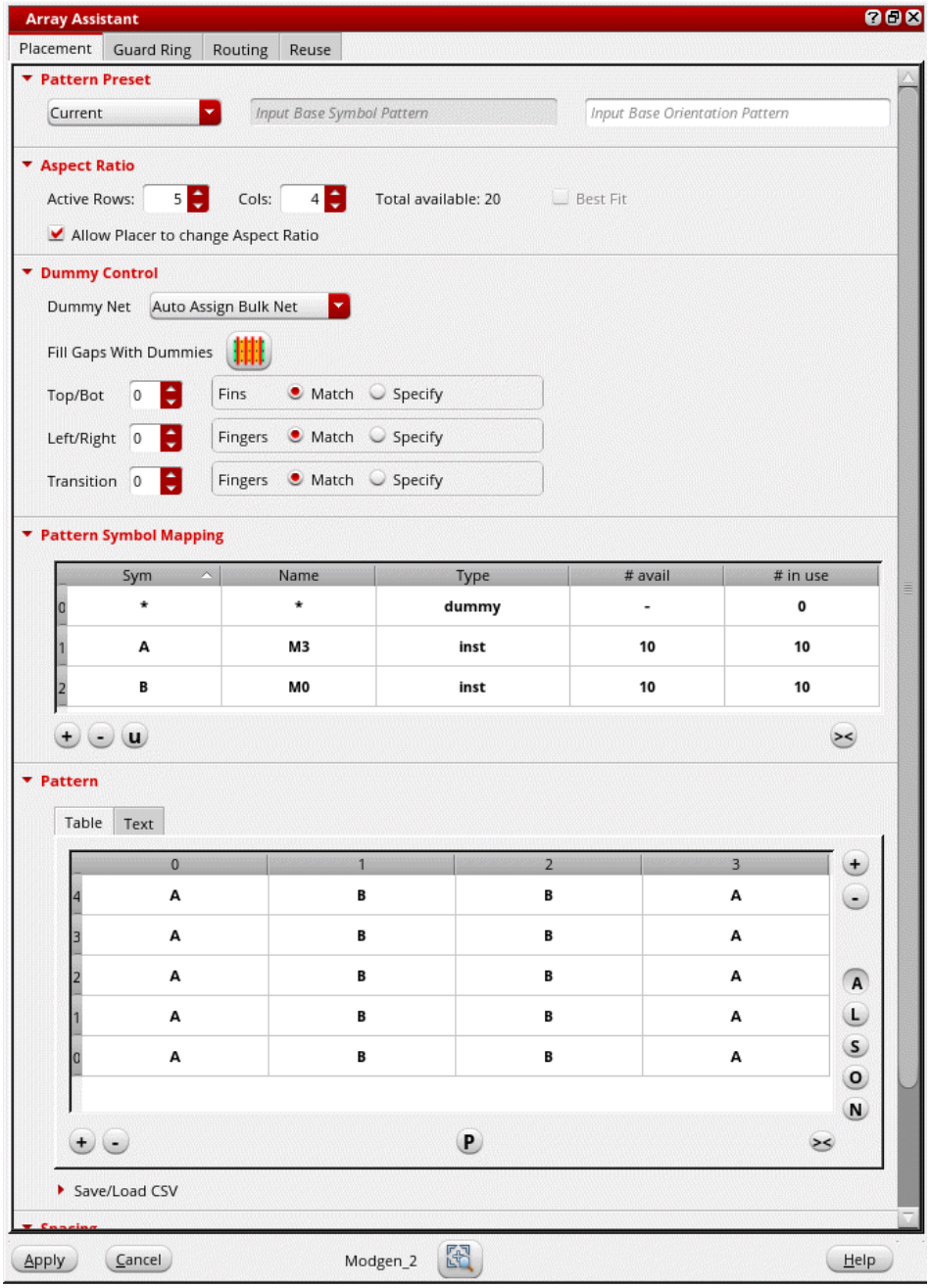
Automatic Generation of Modgens using the Array Assistant

Opening Array Assistant From the Constraint Manager

Opening Array Assistant During Device Placement

Modgen Placement Settings in the Array Assistant

When you open the Array Assistant, the *Placement* tab is displayed by default. The initial device placement is defined by the current placement in the target Modgen. Use the options on the *Placement* tab to refine the placement as per your requirements.

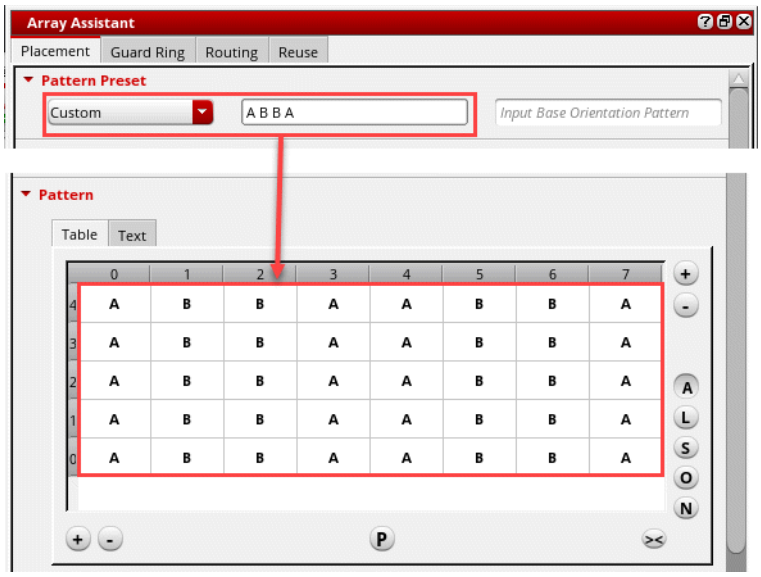


Use one of the following methods to specify a grid pattern:

- (Layout EXL and Higher Tiers) Select a *Pattern Preset* from the drop-down list.
- Specify the grid dimensions in the *Aspect Ratio* section.

  If you specify both *Active Rows* and *Cols*, then a grid of the specified dimensions is generated. The engine calculates and adjusts the number of columns to accommodate the instances and creates a suitable grid.

  Devices that are placed on the grid are automatically interdigitated.
- Specify a base pattern and press `Enter`. In Layout EXL and higher tiers, set *Pattern Preset* to *Custom* before specifying the base pattern.



You can generate the resulting pattern and modify it, if needed.

Base patterns begin at the lower left corner of the grid and proceed left to right until all rows are filled. For example, if you have two devices with an m-factor of four, and you specify two rows with the base pattern of ABBA, you would create the following interdigitation pattern:

```
A B B A
A B B A
```

The base pattern does not have to cover an entire row. If, instead, you specified a base pattern of ABB, the resulting pattern would be as follows:

```
BBAB
ABBA
```

The base pattern supports the following:

- **Repetition Factor for Base Pattern:** You can specify a repetition factor to repeat a pattern symbol multiple times in the *Base Pattern* text using the following syntax:

  ```
  <pattern symbol>:<repetition count>
  ```

  For example, if you type `A B:8 C` in the *Base Pattern* text box, then the resultant value is displayed in the following pattern:

  ```
  A B B B B B B B B C
  ```

  The repetition factor can also be used in the *Base Orientation* field. For example, entering `R0 MY:8 R0` would result in:

  ```
  R0 MY MY MY MY MY MY MY MY R0
  ```

  You can use brackets to specify the repetition factor for a group of devices or orientations. For example, `(A B):2 (B A):2` would result in:

  ```
  A B A B B A B A
  ```

  Similarly, `(R0 MY):2 (MX R180):2` would result in:

  ```
  R0 MY R0 MY MX R180 MX R180
  ```

- **Repetition Sequences of Pattern Symbols:** The pattern notation specifies repeating sequences of pattern symbols. This is useful for more complicated patterns required to generate the device configurations, such as common centroid.

  A pattern symbol is a single letter, `'*'` or `'-'`, which is optionally followed by a number. A pattern sequence contains one or more pattern symbols or pattern expressions concatenated together. You can also use a shorthand notation for filling out the rows of a pattern. If the pattern text has less lines than what is specified in the *Rows* field of the form, then the lines entered so far are repeated to fill out the remaining rows.

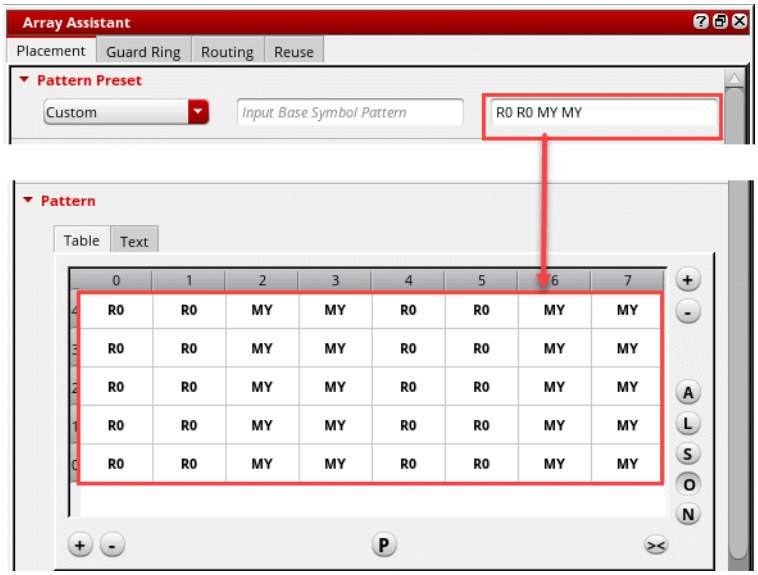
  The shorthand notation can only be used if the number of schematic devices in the Modgen are less than or equal to 25. If there are more than 25 schematic devices, Modgens generate symbols with numbers concatenated to them, such as `A1` or `B1` to make them unique. Therefore, the syntax for the shorthand repetition factor and the syntax to make unique symbols are ambiguous.

  You can still specify a repetition factor if there are more than 25 schematic devices, but it must be delimited with a ':'. For example, `a2` is interpreted as the pattern symbol `A2` whereas, `a:2` is interpreted as the pattern symbol `A` repeated `2` times.If you want to generate the pattern for `(b2a2)2` and `(a2b2)2`, where a Modgen has `4` rows, then the following pattern is generated.

  ```
  B B A A B B A A

  A A B B A A B B
  ```

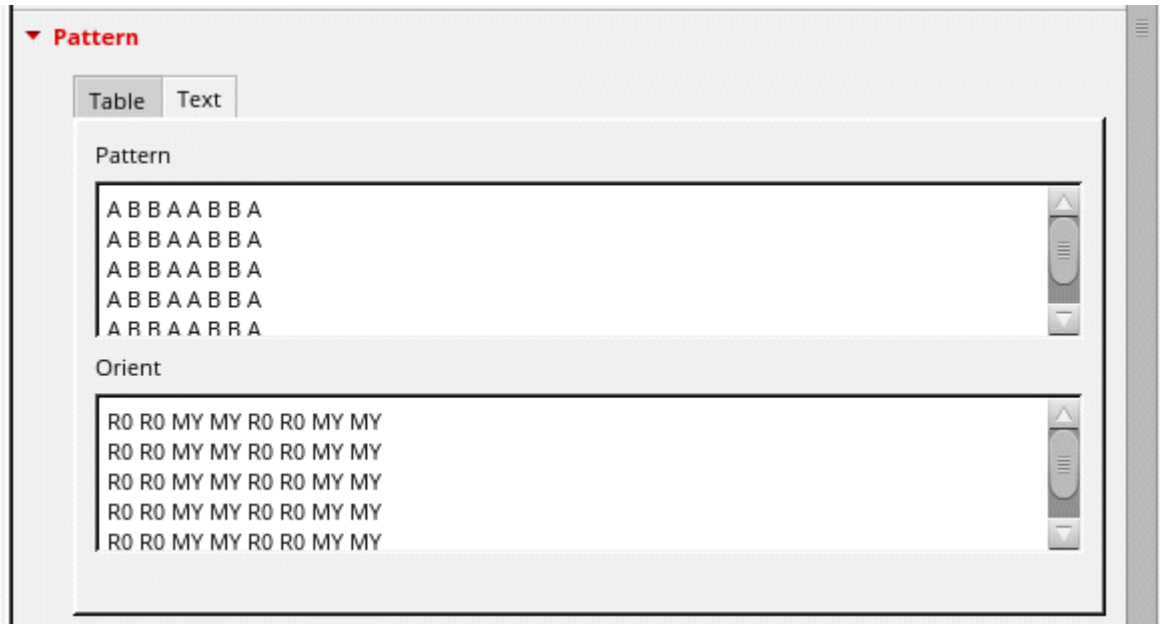
```
B B A A B B A A

A A B B A A B B
```

- Specify a base orientation in the *Base Orientation* box and press `Enter`. In Layout EXL and higher tiers, set *Pattern Preset* to *Custom* before specifying the base orientation.



You can generate the resulting pattern and modify it, if needed.

- Enter a textual pattern for the grid on the *Text* tab in the *Pattern* section. Click *Apply* to generate the pattern in the layout canvas.



Each row in the *Pattern* box corresponds to the corresponding row in the Modgen. You can use shortcuts to specify the pattern, for example AA, A2 or (A)2 — all indicate two instances of device A. *Orient* indicates the orientation of devices in each row.

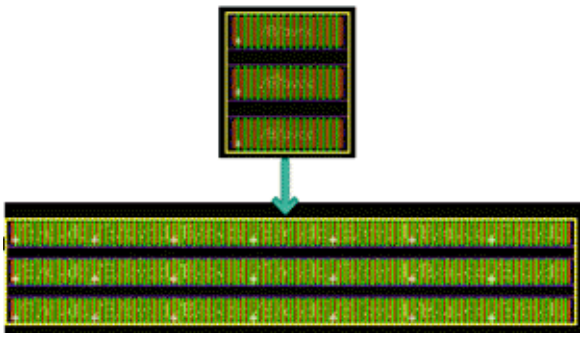
Note: Multi-character symbols such as `P1` and `MN` are not supported. Each symbol must be represented by a single character, such as `A`, `a`, `B`, and `b`.

When you select a device instance in the *Pattern* section, it is highlighted in the layout canvas.

When you modify a Modgen grid, the existing grid member object orientations are overridden when all of the following conditions are met:

- The regenModgenPostProcess environment variable is set to `t`.
- The target position of the array results in an overlapping placement rows or row-like placement grids.
- The placement rows or row-like placement grids specify a limited set of orientations for compliance or snapping.
- The current grid member orientation configuration does not result in the most compact array-row-to-placement-row mapping.

The following images show how the placement settings are applied to a Modgen.

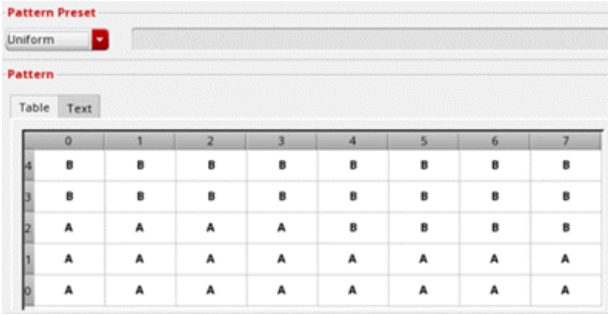
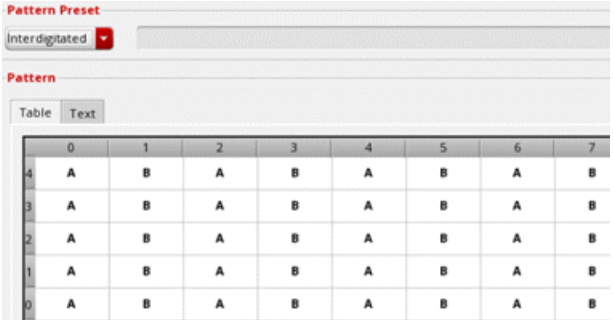
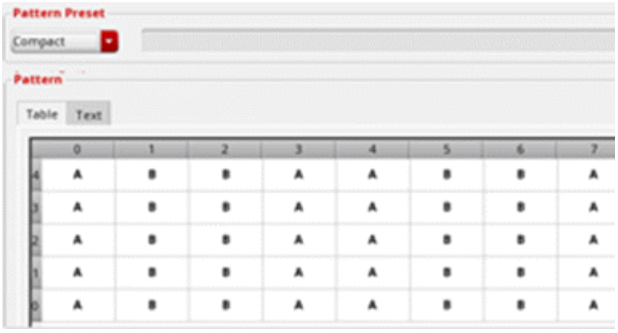


Modgen Pattern Presets

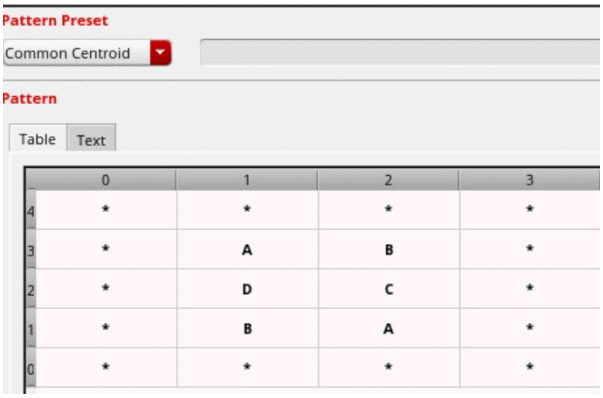
(Layout EXL and Higher Tiers) You can specify the pattern in which devices are to be placed in the Modgen. The *Pattern Preset* list includes the presets listed in the table below. Select the required preset. The output pattern is displayed in the *Pattern* box.

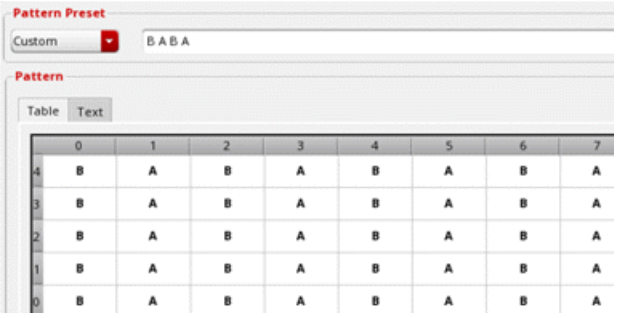
Pattern presets such as *Current*, *Custom*, and *Clustered* respect the specified rows and columns fields. However, certain pattern presets, for example *Compact* and *Common*

Centroid, define specific array dimensions, and therefore they do not respect the specified rows and columns.

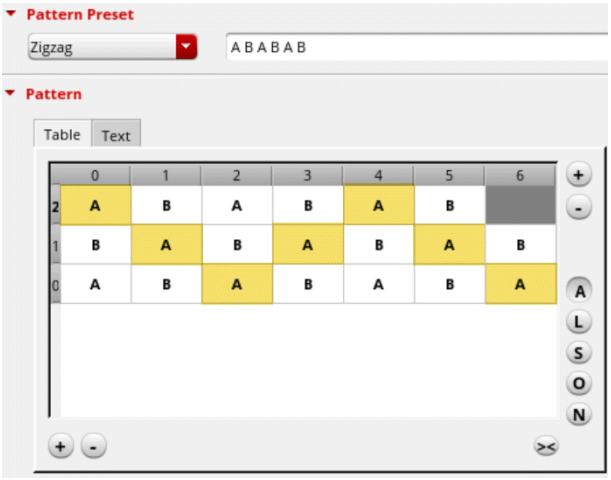
Pattern Preset	Description	Pattern Generated
Current	Uses the current Modgen pattern.	
Clustered	Displays a pattern based on the bottom-up approach with row-based split.	
Interdigitated	Applies an interdigitation of 1 to the devices.	
Compact	Adjusts the pattern to achieve the maximum abutment of devices.	

Pattern Preset	Description	Pattern Generated

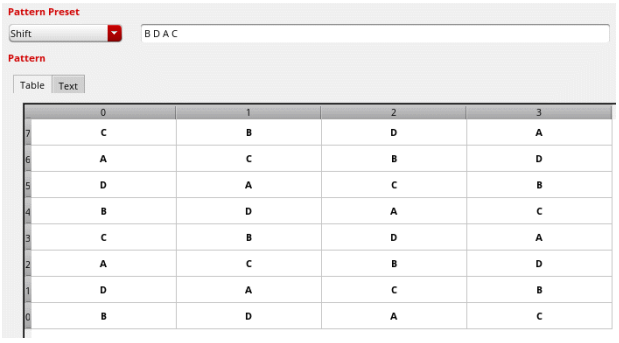
Common Centroid	Places devices in a grid following a common centroid pattern. Symbols are ordered in precedence, outward from the center.
	
	The priority order for placement follows the sequence of available instances to use starting from the center of the array and moving outward in a circular arraignment. The priority is determined by symbols in the ascending order by their number of instances. Symbols with same number of instances are in ascending order alphabetically.
	
	Consider an array with symbols `A(3) B(8) C(2) D(2) E`. The placement is generated using priority `E C(2) D(2) A(3) B(8)`.
	
	Instance `E` is located at the center of the array followed outward by `C`, `D`, `A`, and `B` in that order.
	
	For number of rows of 4, the resulting pattern is:
	
	`B B B B`
	
	`D E C A`
	
	`A D C A`
	
	`B B B B`
		

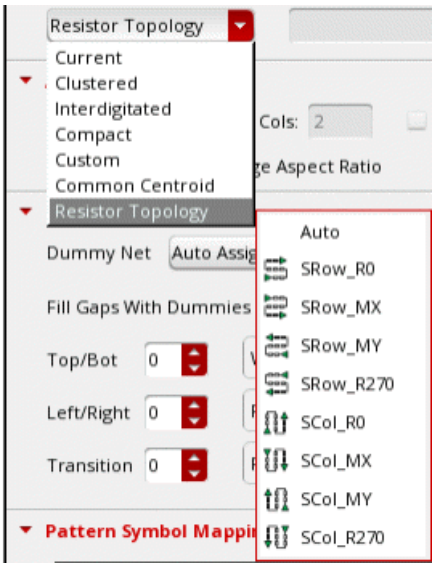
Pattern Preset	Description	Pattern Generated
Custom	Lets you specify a base pattern for the Modgen in the adjoining text box. In the example, a custom pattern `B A B A` is applied. The repeat pattern of Custom patterns in each row starts from left to right. The following example shows pattern `ABCD` in four rows: `A B C D A B C D` `A B C D A B C D` `A B C D A B C D`	

Pattern Preset	Description	Pattern Generated

Zigzag	Applies the base symbol pattern in the opposite horizontal direction for alternating the array rows, starting with left-to-right for row index 0. In the example, a custom pattern `A B A B` is applied in a zigzag manner.
	
	Zigzag pattern starts from left to right for odd number of rows and from right to left for even number of rows. For example, a four-row array with base symbol pattern `A B C D` is generated as:
	
	`A B C D A B C D`
	`D C B A D C B A`
	`A B C D A B C D`
	
	A three-row array with the base symbol pattern `C B A` is generated as:
	
	`C B A`
	`A B C`
	`C B A`
		

Pattern Preset	Description	Pattern Generated

Shift	Shifts the pattern by one position for each consecutive row, starting from the bottom-most row. For example, a three-row array with base symbol pattern `C B A` is generated as:
	`A C B`
	`B A C`
	`C B A`
	
	In pattern `A B C D`, the bottom-row has pattern `A B C D`, the row immediately above it has pattern `B C D A`, followed by `C D A B`:
	`C D A B C D A B`
	`B C D A B C D A`
	`A B C D A B C D`
		

Pattern Preset	Description	Pattern Generated
Resistor Topology	Lets you select a pattern from a list of predefined array topologies. *Auto* selects the most suitable array topology pattern type based on the interconnect analysis. This option is available only in certain advanced node flows for Modgen arrays with all the Modgen members registered as resistors, capacitors, or inductors. For example: `ciRegisterDevice("resistor" '(("passiveLib" "resnsppoly" nil)))` Also, ensure that the `modgenPassiveCreateOnConnectivity` environment variable is set to `zigzag`.	

You can use the following preset generator SKILL functions to control the presets that are displayed in the *Preset* drop-down list.

- gpeRegisterPresetGen: Registers preset generator functions. Once registered, these functions are displayed in the *Preset* drop-down list.
- gpeIsRegisteredPresetGen: Checks whether a preset generator function is registered.
- gpeRunPresetGen: Invokes a preset generator function on the active figGroup. The current figGroup is updated according to the preset generator logic.
- gpeIsPresetGenDisplayable: Checks whether a preset generator should be displayed in the *Preset* drop-down list of the Grid Pattern Editor.
- gpeClearPresetGenerators: Deletes all the registered preset generator functions from the system.

- gpeUnregisterPresetGen: Unregisters the given preset generator function from the Grid Pattern Editor.

Setting the Placement Objective

To specify the placement objective, do the following in the *Aspect Ratio* section:

1. Specify the number of rows in the Modgen in the *Active Rows* field.

   The number of columns is automatically calculated and displayed in *Cols*.

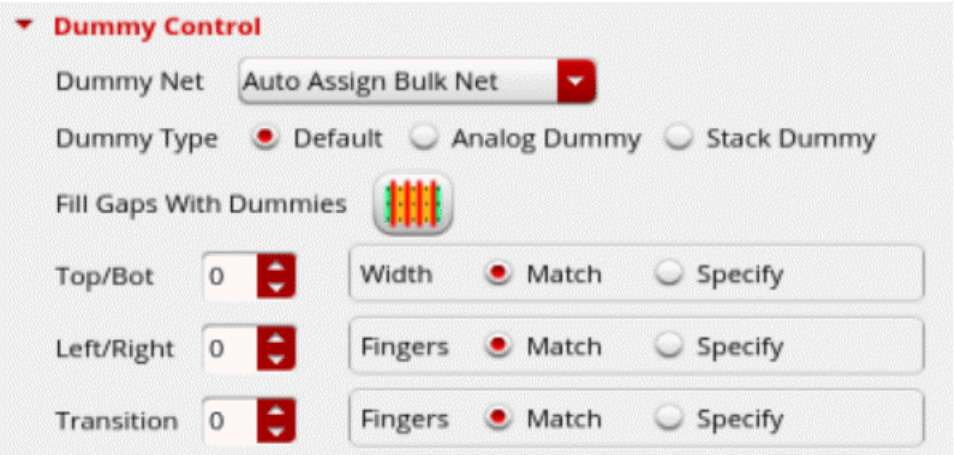
2. Select *Best Fit* to ensure that an optimal placement of the Modgen devices is achieved according to the specified pattern.

   This option is available only when *Pattern Preset* is set to *Custom* (Layout EXL and Higher Tiers) and a base pattern is specified.

3. Select *Allow Placer to change Aspect Ratio* to let the Virtuoso device-level automatic placer adjust the aspect ratio of the Modgen to achieve optimized placement for the given floorplan.

Adding Dummies Around Modgens

(Layout EXL and Higher Tiers) Use the options in the *Dummy Control* section to add dummies around the Modgen.



1. From the *Dummy Net* list, choose the net to which the pins of the dummies are to be connected.
2. Select a *Dummy Type*.

   This option is available only in certain advanced node flows.
3. Click *Fill Gaps With Dummies* to fill gaps between devices with dummies and add dummy columns where there are abutment breaks. Dummies are inserted when you click *Apply* in the Array Assistant.
4. Specify the number of dummy rows and columns in the *Top/Bot* and *Left/Right* fields.

   To add surround dummies, specify both *Top/Bot* and *Left/Right* values.

   - Specify the number of fins or fingers as either *Match* or *Specify*.
   - Specify whether *Transition* dummy columns are to be inserted.

Editing the Grid Pattern

You can edit the mapping of instances and dummies to symbols in the *Pattern Symbol Mapping* table.

You can edit the Modgen devices in the *Pattern* grid directly or using the options in the shortcut menu.

Saving and Loading Array Settings from a CSV File

The *Save/Load CSV* section lets you save the current array settings to a comma-separated values (CSV) file. You can also load array settings from a CSV file.

As the name suggests, a CSV file comprises character strings that are separated by commas. A CSV file allows data to be saved in a tabular format.

Here is a sample CSV file:

A CSV File

```
C, C, C, C, C, C, D, D, D, D, D, D

C, C, C, C, C, C, D, D, D, D, D, D

C, C, C, C, C, C, D, D, D, D, D, D

C, C, C, C, C, C, D, D, D, D, D, D

A, A, A, A, A, A, B, B, B, B, B, B

A, A, A, A, A, A, B, B, B, B, B, B

A, A, A, A, A, A, B, B, B, B, B, B

A, A, A, A, A, A, B, B, B, B, B, B
```

First Section:

- Represents the placement of devices
- Each row is followed by a blank line
- Lists symbol names, as mapped in the third section

```
R0, MY, R0, MY, R0, MY, R0, MY, R0, MY, R0, MY

R0, MY, R0, MY, R0, MY, R0, MY, R0, MY, R0, MY

R0, MY, R0, MY, R0, MY, R0, MY, R0, MY, R0, MY

R0, MY, R0, MY, R0, MY, R0, MY, R0, MY, R0, MY

R0, MY, R0, MY, R0, MY, R0, MY, R0, MY, R0, MY

R0, MY, R0, MY, R0, MY, R0, MY, R0, MY, R0, MY

R0, MY, R0, MY, R0, MY, R0, MY, R0, MY, R0, MY

R0, MY, R0, MY, R0, MY, R0, MY, R0, MY, R0, MY
```

Second Section

- The keyword “`dummy`” represents the device orientations
- Each row is followed by a blank line

```
dummy, *

P1, C

P0, D

N0, A

N1, B
```

Third Section

- Represents a dummy device
- Specifies the mapping of devices

To save the current array settings to a CSV file:

1. Specify the path to the CSV file in the *File Name* field.

   Alternatively, click the *Browse* button to open a file browser, where you can select the path from the hierarchy and specify a file name.

2. Set *Pattern View* to one of the following:

- *Symbol:* Prints the letter that is mapped to that instance in the Modgen Pattern Map.
- *Schematic Name*: Prints instance names from schematic.

3. Click *Save*.

The array settings are saved to the specified CSV file.

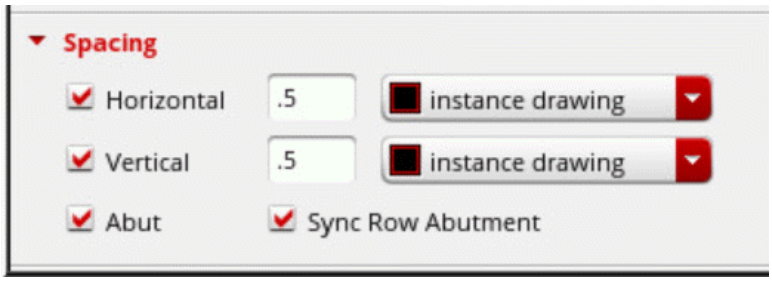
To load array settings from a CSV file:

1. Specify the CSV file name in the *File Name* field.
2. Click *Load*.
3. Click *OK*.

The preset is loaded from the specified preset file.

Specifying the Spacing Between Modgens

(Layout EXL and Higher Tiers) Use the options in the *Spacing* section to specify the spacing between Modgen devices.



1. Specify a *Horizontal* value, which indicates the spacing between Modgens in a row and select a reference layer for calculating the horizontal spacing.
2. Specify a *Vertical* value, which indicates the spacing between Modgens in columns, and select a reference layer for calculating the vertical spacing.
3. Select *Abut* to abut all Modgen devices and dummies to get a compact placement.
4. Select *Sync Row Abutment* to abut all rows in a Modgen in a synchronized operation along a column.

An unselected state for the spacing, alignment, and abut fields implies that if the current Array Assistant settings for these properties are customized (their state is unreachable by the spacing, alignment, or abut the Array Assistant fields), the custom settings remain untouched during an Array Assistant *Apply* operation.

To customize the Array Assistant spacing, alignment, and abut settings without using these fields, you can load an existing customized Modgen array into the Array Assistant.

Deselecting the spacing, alignment, and abut fields from a selected state always resets these values to their defaults, align left for *Horizontal*, align bottom for *Vertical*, and unabut all.

Related Topics

Array Assistant

Modgen Pattern Presets

Setting the Placement Objective

Adding Dummies Around Modgens

Editing the Grid Pattern

Saving and Loading Array Settings from a CSV File

Specifying the Spacing Between Modgens

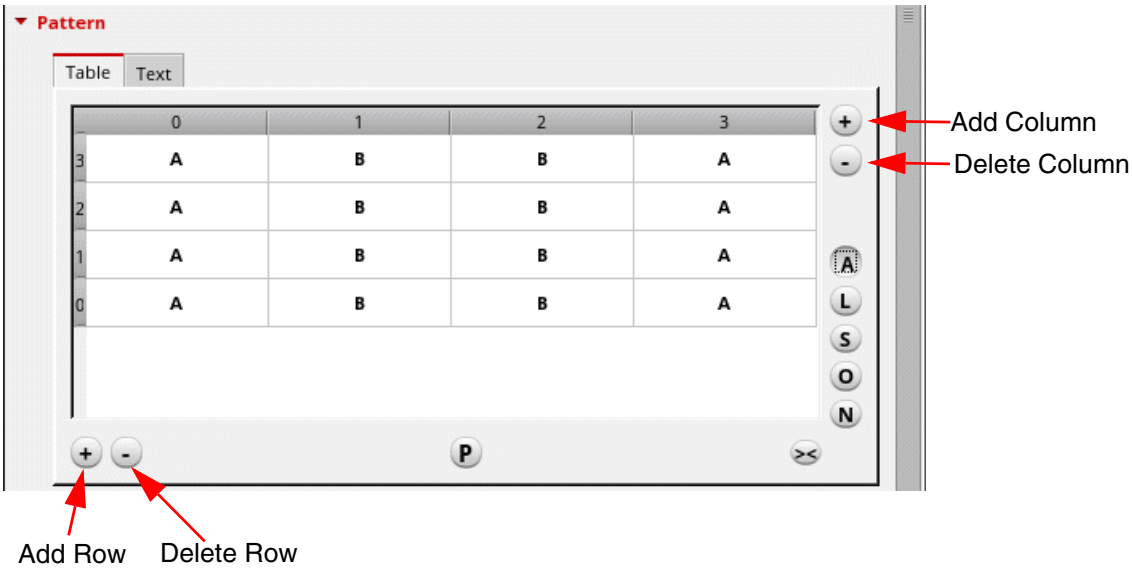
Display Names in Array Assistant

Methods to Add and Delete Rows and Columns in the Array Assistant

Use one of the following methods to add new rows or columns to Modgen arrays in the Array Assistant:

- Using the Add buttons in the Array Assistant.
  - To add a new column at the bottom of the grid, click the Add button at the top-right corner of the Array Assistant.

- To add a new row at the right of the grid, click the Add button at the bottom-left corner of the Array Assistant.



- Using the shortcut menu.
  - **a.** Select the required row or column.
  - **b.** Right-click to display the shortcut menu.
  - **c.** Choose *Insert Row(s)* or *Insert Column(s)*.

You can delete rows or columns using one of the following methods:

- Using the Delete button in the Array Assistant.
  - To delete the bottom most column, click the Delete button at the top-right corner of the Array Assistant.
  - To delete the right most column, click the Delete button at the bottom-left corner of the Array Assistant.
- Using the shortcut menu.
  - **a.** Select the required row or column.
  - **b.** Right-click to display the shortcut menu.
  - **c.** Choose *Delete Row(s)* or *Delete Column(s)*.

If you alter the size of any column in the *Patten* or *Pattern Mapping* sections, click  to auto-fit the columns.

Related Topics

Creating Rows

Array Assistant

Modgen Placement Settings in the Array Assistant

Editing the Grid Pattern

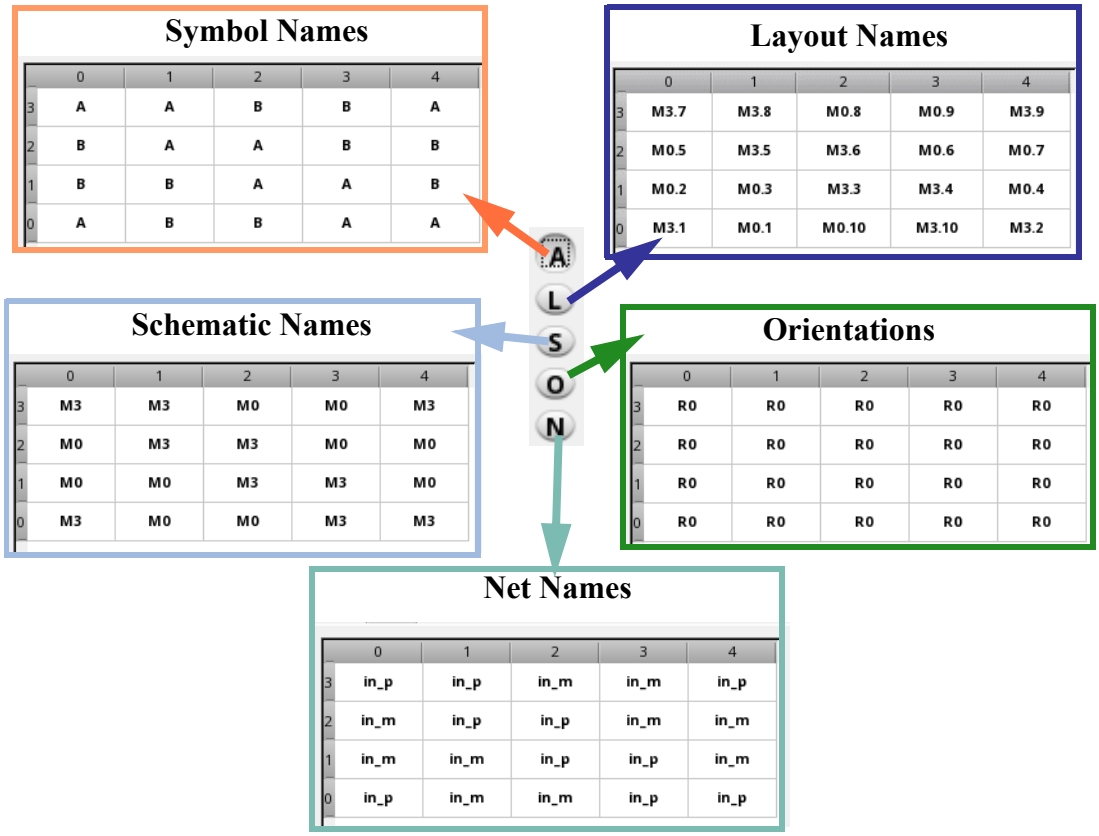
Specifying the Spacing Between Modgens

Display Names in Array Assistant

Display Names in Array Assistant

By default, the cells in the *Pattern* section of Array Assistant display the symbols that are mapped to instances in the *Pattern Symbol Mapping* section.

Use the following buttons to toggle between symbol names, layout names, schematic names, device orientations, and net names.



The above buttons are available on the *Table* tab of *Pattern* section. The *A*, *L*, and *S* settings are also applied to the *Text* tab.

Click P to bring all power and ground-connected terminals to their nearest row edges. The orientations of the related instances are updated based on their locations and the pattern preset is set to *Custom*. This command does not run on dummies added to the grid.

Related Topics

Array Assistant

Modgen Placement Settings in the Array Assistant

Editing the Grid Pattern

The Move Command in the Array Assistant

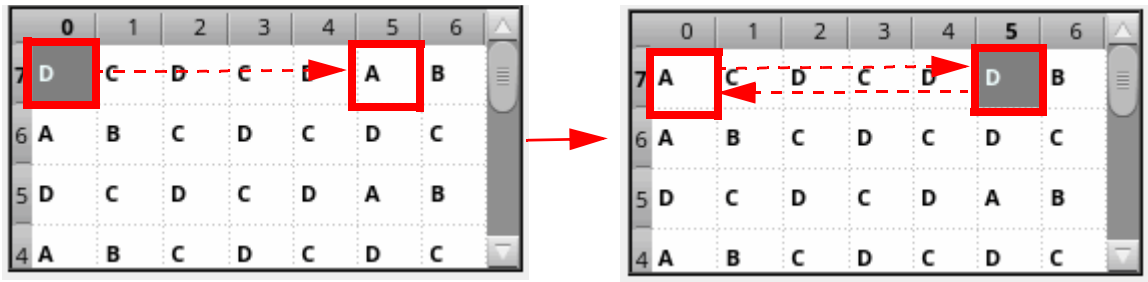
Methods to Add and Delete Rows and Columns in the Array Assistant

The Move Command in the Array Assistant

You can move one or more instances by dragging and dropping them to their new locations either in the *Pattern* section of the Array Assistant or on the layout canvas.

The behavior of the move command depends on the mode set using the `moveAsSwap` environment variable.

- When set to `t`, Move as Swap mode is enabled, which is the default mode. When an instance is moved in this mode, the instances in the source and target cells are swapped, as shown below.



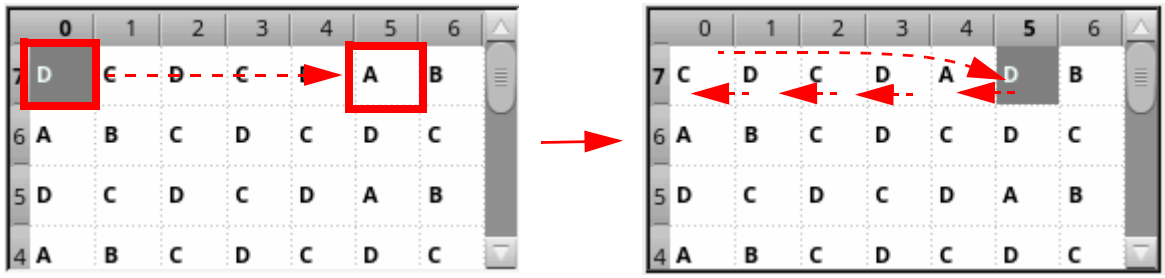
  Instances that are selected for movement are highlighted in the layout canvas.

  A similar behavior is observed while moving rows and columns in this mode.

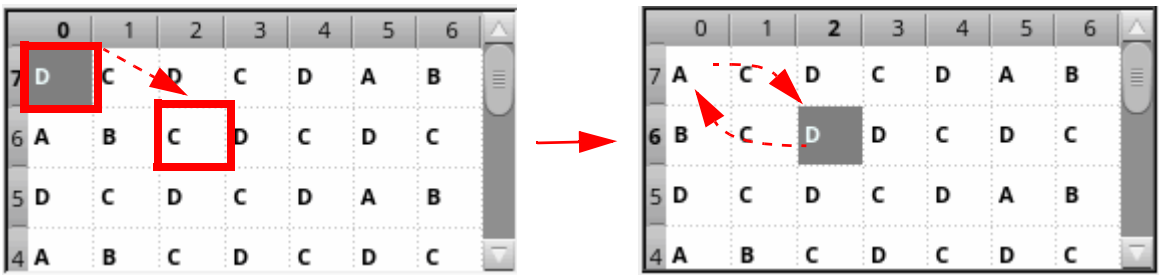
- When set to `nil`, Move as Insert mode is enabled. When an instance is moved in this mode, the instance first shifts horizontally, and then vertically, until the source location is back-filled.

  Instances that are selected for movement in the Array Assistant are highlighted in the layout canvas.

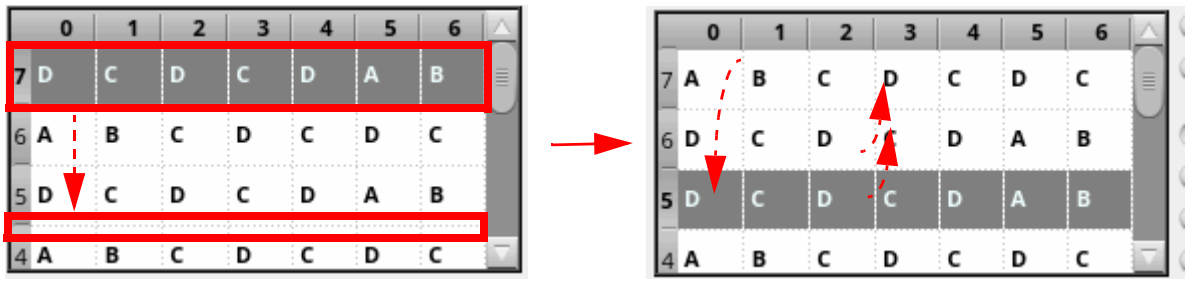
  In the following example, the source instance is moved horizontally to the target location. The other instances are shifted horizontally to fill the empty cell as shown below.



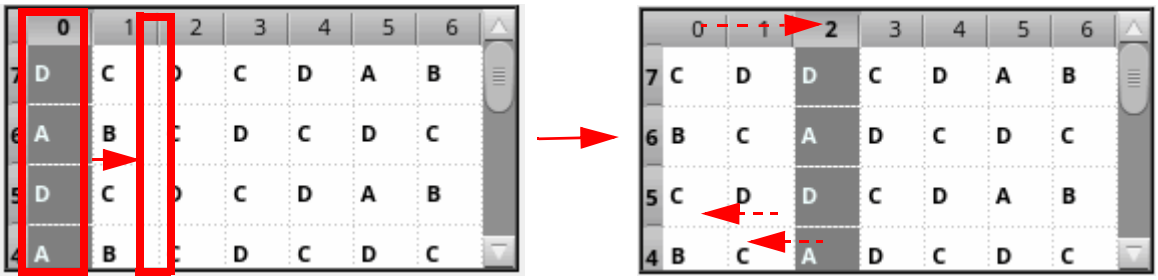
In the following example, the source instance is moved diagonally to the target location. The other instances are shifted horizontally and then vertically to fill the empty cell as shown below.



In the following example, instances in the source row are moved to the target location. The remaining rows are moved up one step at a time to fill the empty row as shown below.



In the following example, the instances in the source column are moved to the target location. The remaining columns are moved horizontally one step at a time to fill the empty column as shown below.



Related Topics

moveAsSwap (Environment variable)

Swapping Cells, Rows, and Columns in the Array Assistant

Display Names in Array Assistant

Swapping Cells, Rows, and Columns in the Array Assistant

To swap two cells in the Array Assistant:

1. Select the cells.
2. Right-click and choose *Swap Cells*.

To swap two rows in the Array Assistant:

1. Select the rows.

   Use `Ctrl +` click to select two non-adjacent rows or columns.
2. Right-click and choose *Swap Rows*.

To swap two columns in the Array Assistant:

1. Select the columns.

   Use `Ctrl +` click to select two non-adjacent rows or columns.
2. Right-click and choose *Swap Columns*.

Related Topics

moveAsSwap (Environment variable)

The Move Command in the Array Assistant

Display Names in Array Assistant

Copying and Deleting Instances in the Array Assistant

Array Assistant

Modgen Placement Settings in the Array Assistant

Copying and Deleting Instances in the Array Assistant

The Array Assistant supports copying and deleting instances.

1. Use the `Ctrl` + `C` key combination to copy instances in a pattern.
2. Use the `Ctrl` + `V` key combination to paste instances in a pattern.

You can delete individual cells or entire rows or columns. The corresponding instances are highlighted in the layout canvas.

- Either press `Delete` or right-click and choose *Clear Cell(s)*.

The deleted instances are highlighted in the *Pattern Symbol Mapping* section or the Array Assistant.

Related Topics

Array Assistant

Modgen Placement Settings in the Array Assistant

Methods to Add and Delete Rows and Columns in the Array Assistant

Display Names in Array Assistant

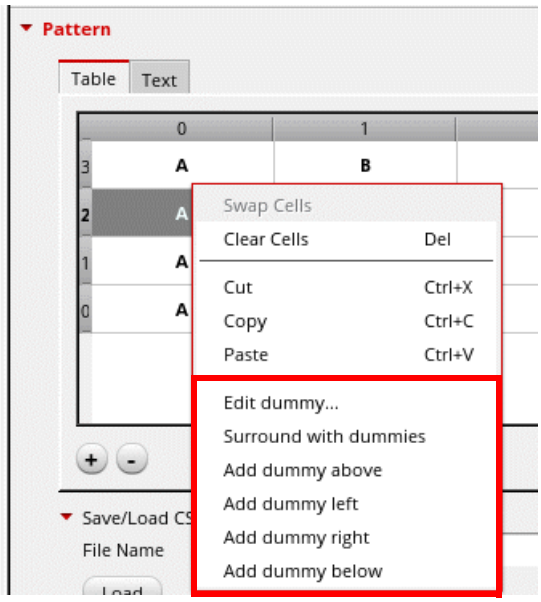
Adding Dummies around Modgen Instances using the Array Assistant

Adding Dummies around Modgen Instances using the Array Assistant

The Array Assistant lets you add dummies along one or more edges of a Modgen grid pattern.

To add dummies:

1. Select the required instances in the *Pattern* region of the Array Assistant.
2. Right-click to display the shortcut menu.
3. Select a direction to insert dummies.



Dummies are added in the specified direction. The dummies created using this method are backward-compatible with the Modgen dummies creation options.

Dummy properties are automatically reset to defaults when you change the pattern of the entire Modgen, for example, when you apply a base pattern, a different pattern using the *Text* tab, or a preset pattern. However, pattern edits that are localized to a portion of the pattern grid, for example, editing or adding dummies, rows, or columns, do not impact the dummy configurations.

Related Topics

Array Assistant

Modgen Placement Settings in the Array Assistant

Editing Modgen Dummy Properties in Array Assistant

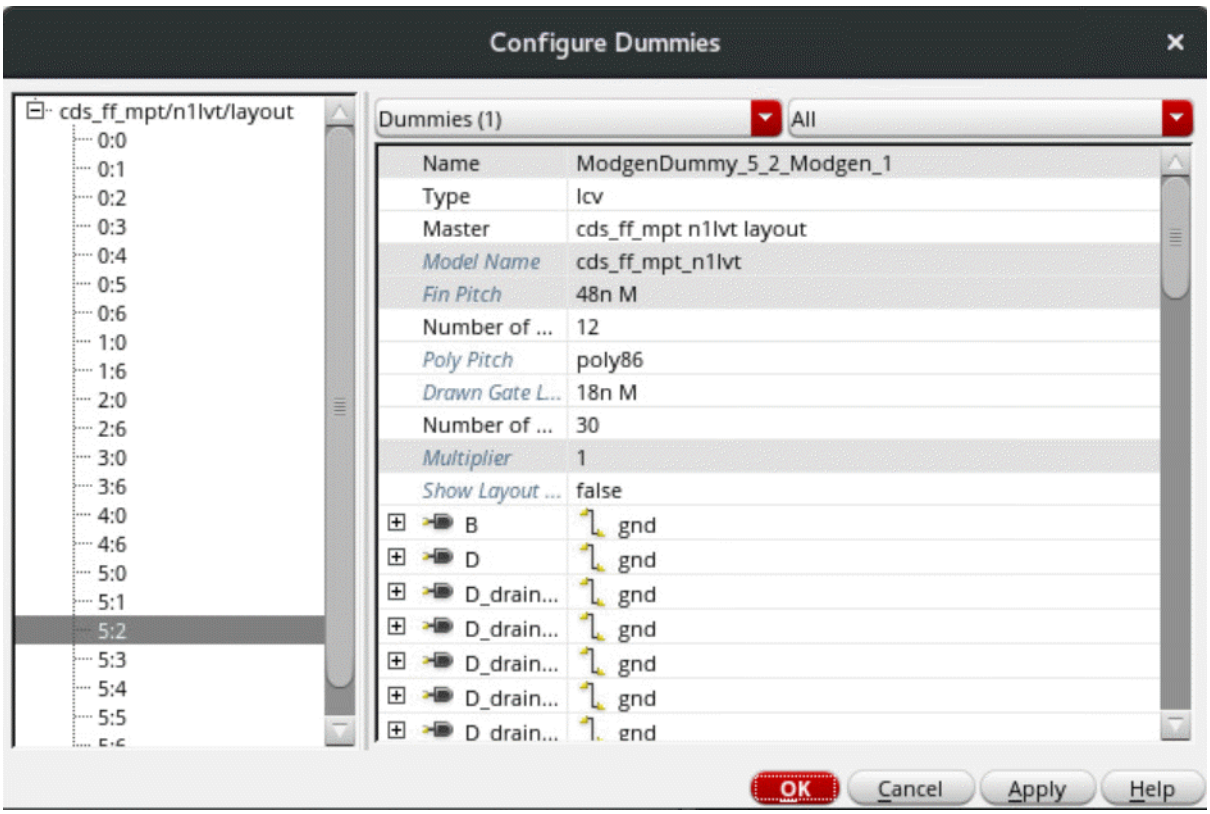
Editing Modgen Dummy Properties in Array Assistant

You can use either the Property Editor assistant or the Configure Dummies form to edit the properties of the dummies that are added around Modgen instances.

To edit dummy properties using the Configure Dummies form:

1. Select the dummies in the Array Assistant
2. Right-click and select *Edit Dummy* from the shortcut menu.

   The Configure Dummies form appears.

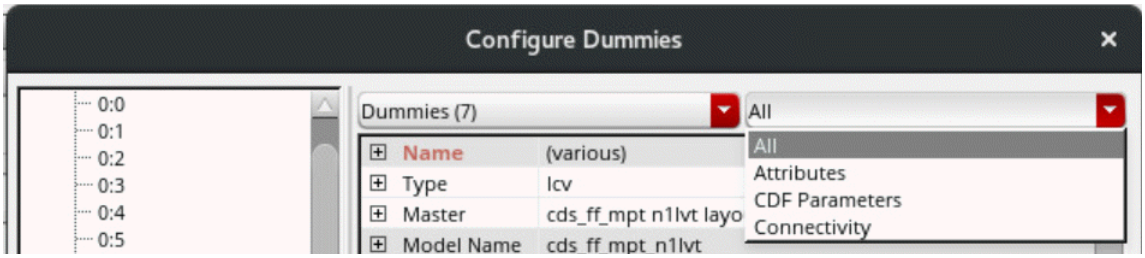


3. In the left pane, select the dummy that you want to edit.

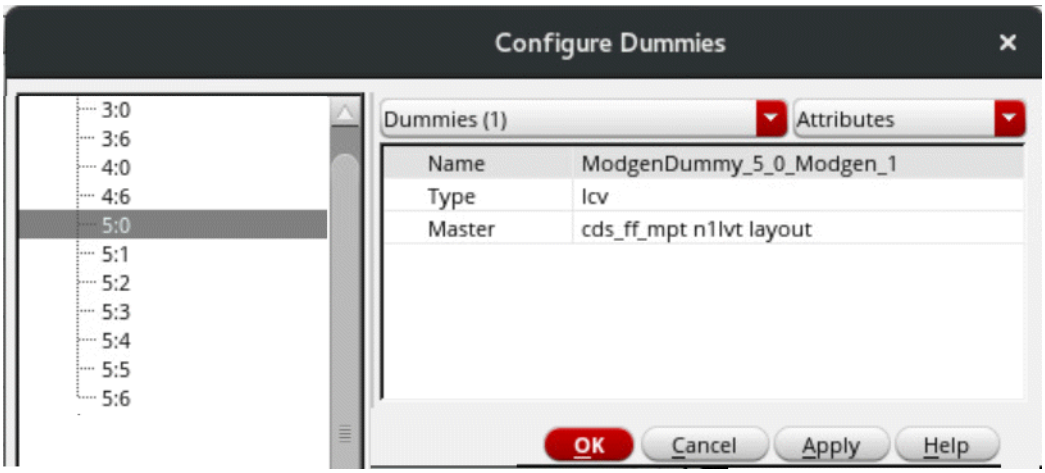
   You can either select individual dummies or multiple dummies in the left pane and edit their *Properties* in the right pane.

   Note: If the selected dummy is abutted, the Edit Dummy Form does not let you edit the terminals with modified connectivity due to abutment.

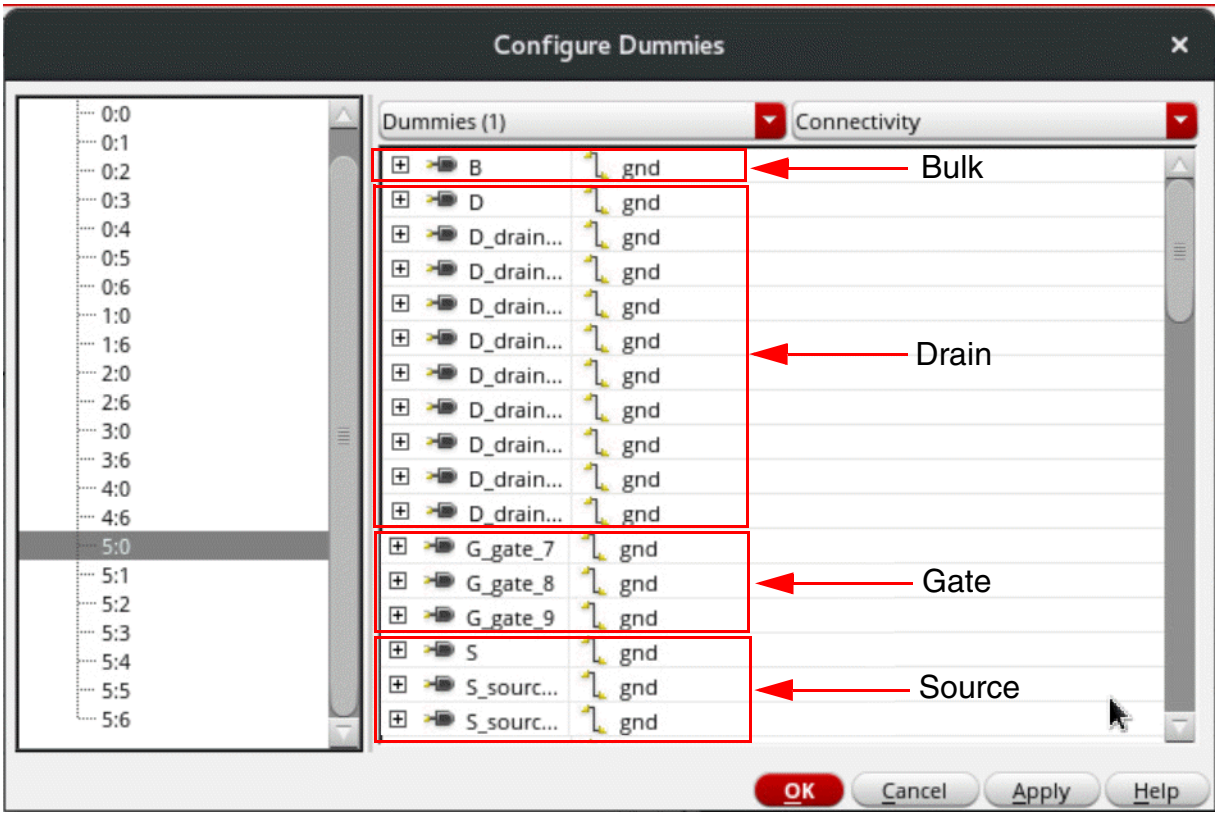
4. In the right pane, use the drop-down list at the top-left side to choose whether the *Attributes*, *CDF Parameters*, *Connectivity*, or *All* parameters are to be edited.



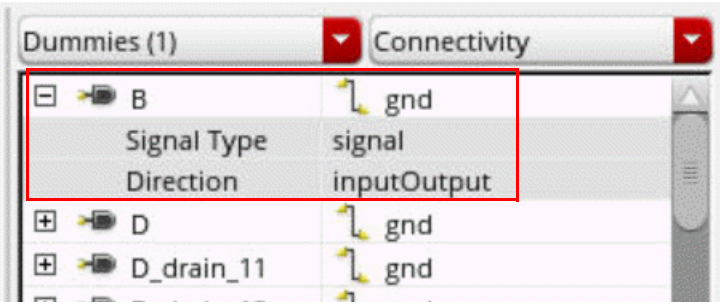
5. In *Attributes* mode, the dummy name, type, and master values are displayed. Edit the *Type* and *Master* fields as per your requirement.



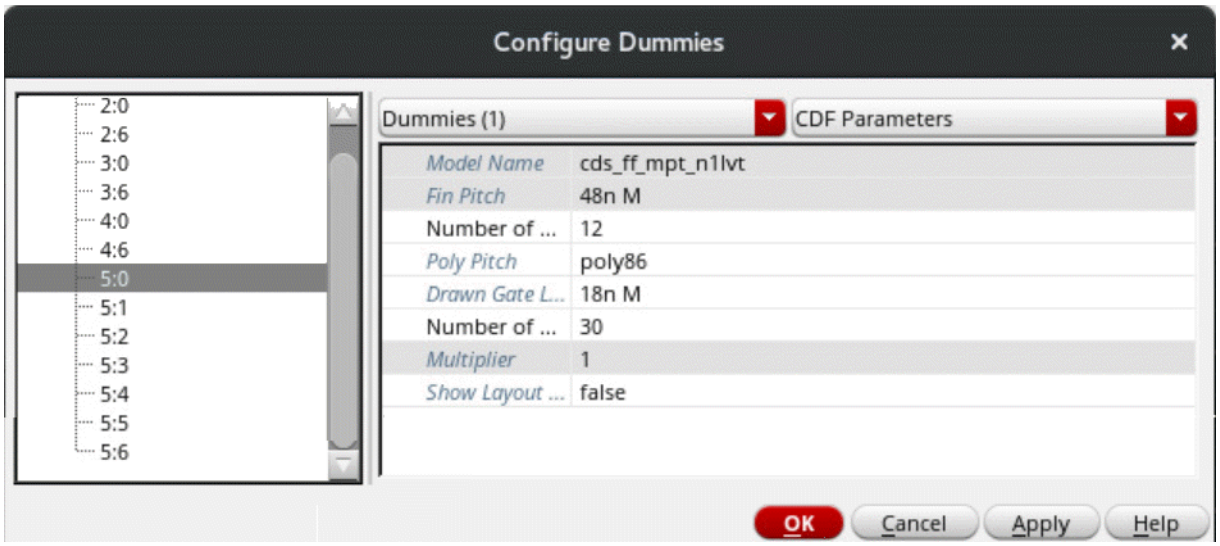
6. In *Connectivity* mode, a list of bulk, drain, gate, and source connections is displayed. Edit the net names associated with a terminal by typing the new net name or by choosing another net from the drop-down list.



7. Expand the net names to view the *Signal Type* and *Direction* of each net.



8. In *CDF Parameters* mode, edit the CDF parameters for the selected dummy.



The connectivity values are not editable for terminals or CDF parameters that have Modgen values that do not match the database value.

9. Click *OK*.

Dummy parameters are updated as per your specifications.

Related Topics

Configure Dummies Form

Adding Dummies around Modgen Instances using the Array Assistant

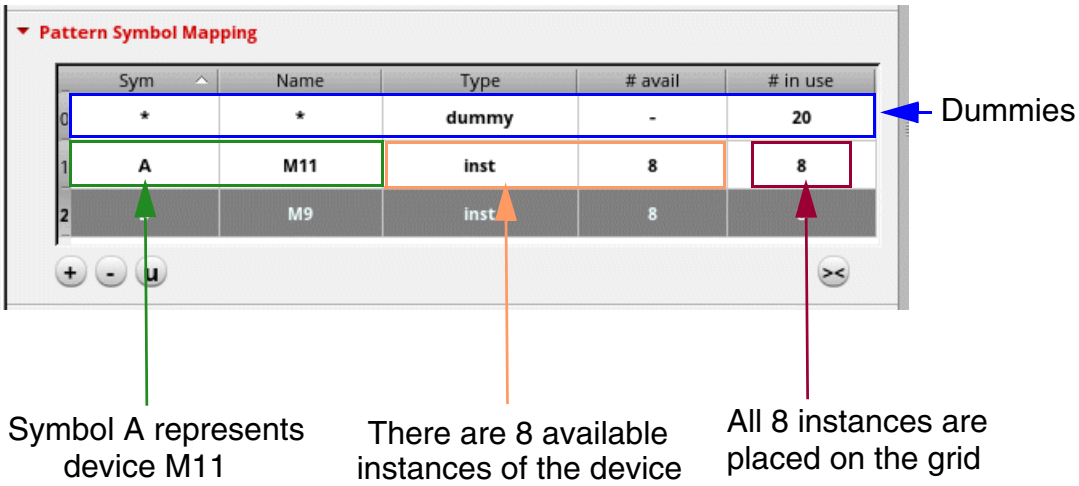
The Move Command in the Array Assistant

Array Assistant

Device Instance Management In the Array Assistant

Use the *Pattern Symbol Mapping* region of the Array Assistant to view and change the mapping of instances and dummies.

The following figure explains the columns in the *Pattern Symbol Mapping* table.

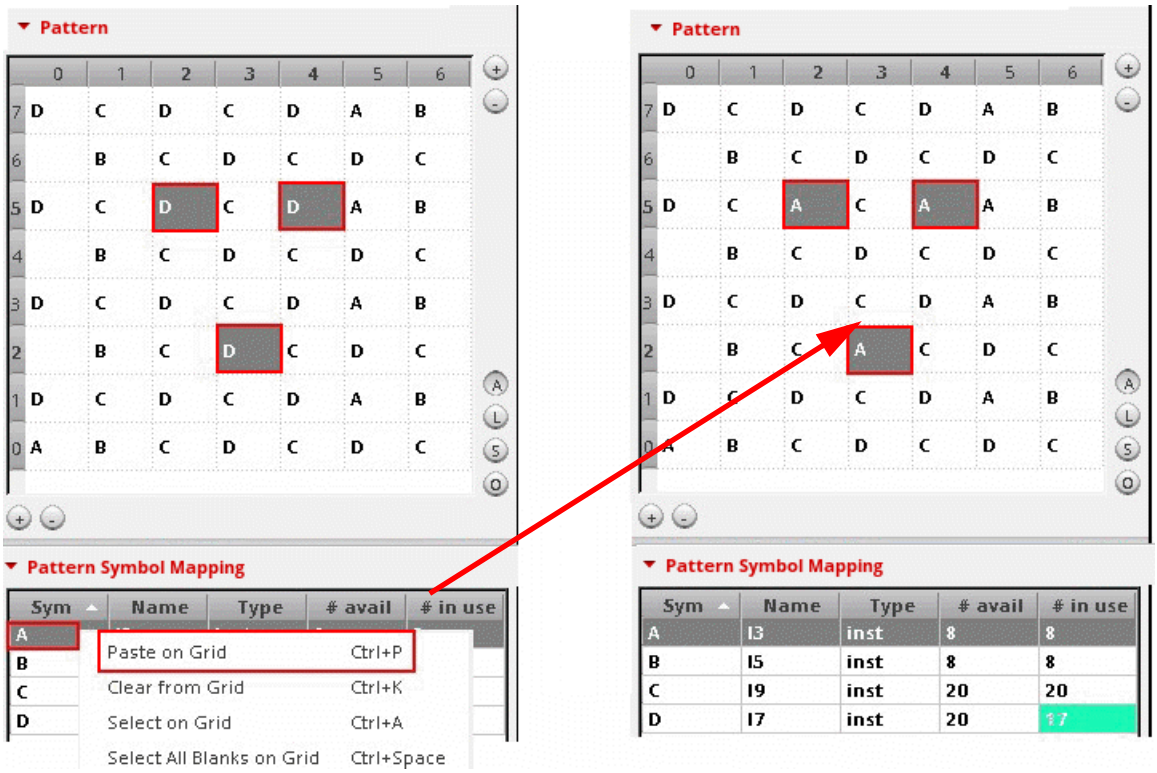


You can perform the following tasks in the *Pattern Symbol Mapping* table:

- **Select Instances of a Device:** When you select a device in the *Pattern Symbol Mapping* table, all instances of the device are highlighted in the layout canvas.
- **Sort Devices:** To sort the devices alphabetically, click the header of the column that you want to sort. Alternate clicks sort in ascending and descending order.
- **Remap Instances:** When you change the symbol of a device in the *Pattern Symbol Mapping* table, all instances of the device in the *Pattern* table are updated and remapped to the new symbol name.
- **Add and Remove Instances from Modgens:** Use the + and - buttons to add or remove selected instances from the active Modgen. You can add instances from the layout or schematic cellviews.
- **Update Modgen Instances:** Select the u button to update the connectivity and nets on the instances in the Modgen. The device connectivity, parameters, and multipliers are refreshed from the corresponding schematic cellview.
- **Reset Column Width:** If you have altered the size of any column, click the auto-fit button >< to automatically fit the columns.
- **Use the Shortcut Menu:** Select a device and use the shortcut menu to do the following:

- **Paste on Grid:** Select a few devices in the *Pattern* table before running this command. The selected devices are replaced by the device highlighted in the *Pattern Symbol Mapping* table.

  In the following example, the *Paste on Grid* command is used to replace all the selected instances of device D with device A.



- **Clear from Grid:** All instances of the selected device in the *Pattern Symbol Mapping* table are deleted from the *Pattern* table.

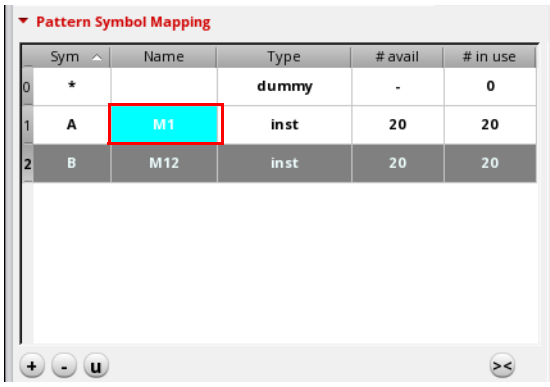
- **Select on Grid:** All instances of the selected device in the *Pattern Symbol Mapping* table are selected in the *Pattern* table.

- **Select All Blanks on Grid:** All blank instances are selected in the *Pattern* table. Following this command, you can select another device in the *Pattern Symbol Mapping* table and choose *Paste on Grid* from the shortcut menu.

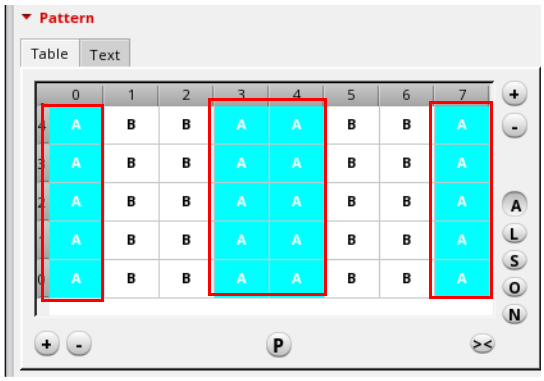
- **Assign Color:** Assigns a color to the selected device in the *Pattern Symbol Mapping* table. The color is applied to all instances of the device in the *Pattern*

table so that they are visually distinguishable. In the following image, all instances of device `A` are colored `cyan`.

Color assigned to a device in the Array Assistant



Color applied to all instances in the Array Assistant



You can select multiple entries in the *Pattern Symbol Mapping* table and apply the same color to all selected devices.

To remove coloring, select the required symbol in the *Pattern Symbol Mapping* table and choose *Assign Color—Clear Color* from the shortcut menu.

Colors assigned using this method could be overridden by Smart Display coloring, if enabled to work through the Modgen Editor.

Related Topics

Array Assistant

Modgen Placement Settings in the Array Assistant

Editing the Grid Pattern

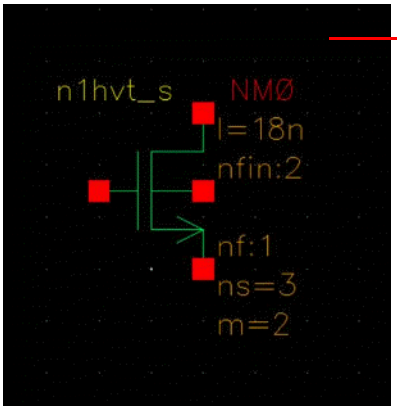
Methods to Add and Delete Rows and Columns in the Array Assistant

Display Names in Array Assistant

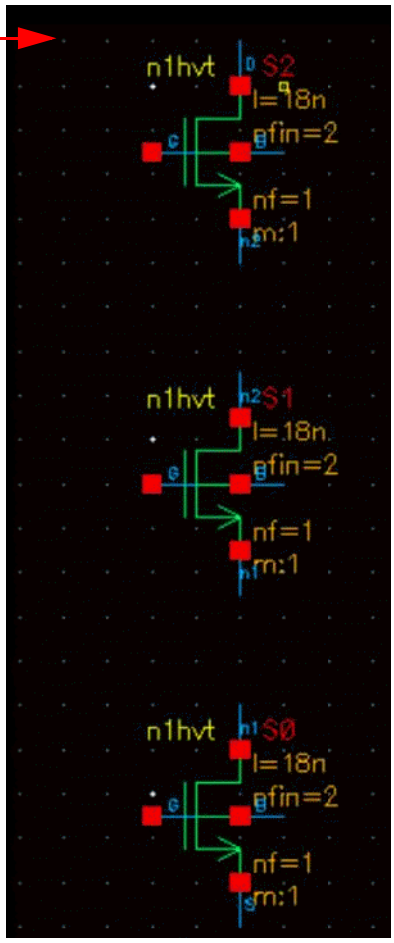
Support for Stacks in Modgens

(Virtuoso Advanced Node for Layout Only or Layout Standard) A stack of series-connected devices (stack) is a set of devices that are connected in sequence from the source to drain. The devices in a stack share a common gate connection, bulk connection (if the devices have a bulk terminal), and super-master. In addition, stacks have no external connections to any internal source or drain terminal. The following image depicts a stack in the schematic view.

A Stack Series Cell



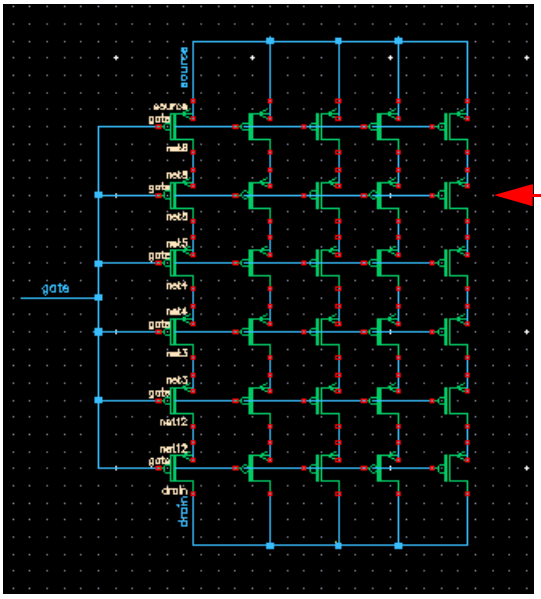
Stack of Cells Displayed when Descending the Stack



Stacking devices helps achieve circuit performance goals in advanced node PDKs, where the gate lengths for analog devices are more uniform and discrete. A properly arranged stack can be internally abutted to reduce the area of the layout.

When a new Modgen is created with the Array Assistant visible, all stacks within the Modgen are detected, rearranged according to connectivity (if necessary), and their members are abutted.

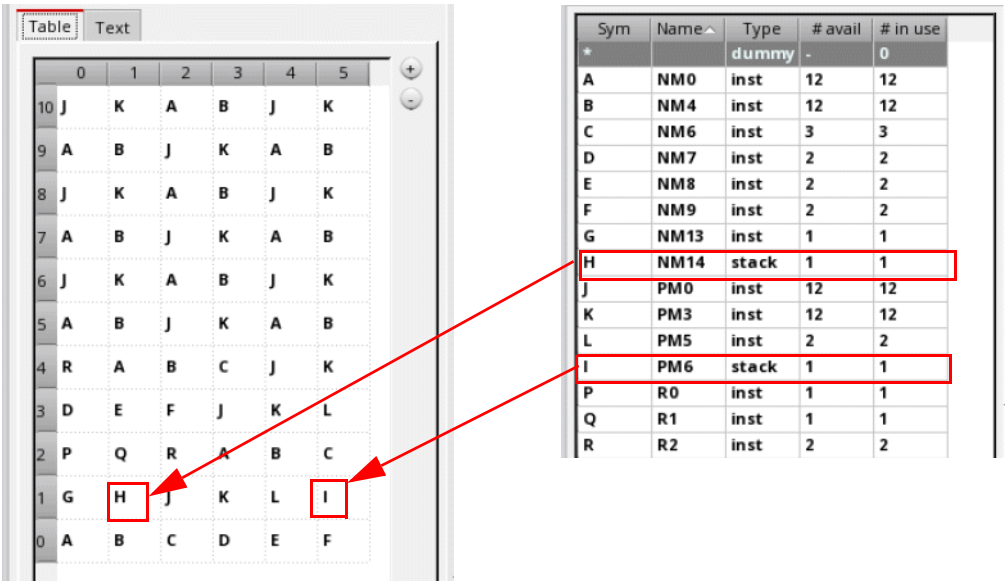
For existing Modgens, stacks are detected only if their devices are arranged and abutted properly.



A fully expanded and stacked m-factored device

Each abutted stack is represented as a single symbol, as if it were a single device, in the Array Assistant. Stacks sets that are connected in parallel also share a common symbol.

The tool tips for these symbols provide information about the constituent stacked devices.



Series-connected, unabutted devices with shared gate connectivity are displayed in the Array Assistant as individual symbols.

Dummies that are adjacent to abutted stacks are also stacked and abutted.

The Property Editor assistant displays the abstract and compressed grid dimension (row and column) information for Modgen arrays containing stack-type members. This behavior is similar to the Array Assistant. For Modgens with stack-type members, the Property Editor assistant might include a *Total* field that is not consistent with the row and column information.

Related Topics

Stack Attributes Controlled by the Array Assistants

Array Assistant

Automatic Generation of Modgens using the Array Assistant

Stack Attributes Controlled by the Array Assistants

(Virtuoso Advanced Node for Layout Only or Layout Standard) Each abutted stack of devices in a Modgen is represented as a single symbol, as if it were a single device, in the Array Assistant.

You can use these assistants to customize the following stack attributes:

- **Orientations of Stacks:** A stack may have valid orientations of `R0`, `MX`, `MY`, or `R180`. The orientation applies to the entire stack; the orientations of the stack members may (or may not) be identical to the orientation of the stack. This means that the absolute orientation of a stack is arbitrary. Therefore, the displayed stack orientation may not always be preserved when the stack is rotated. When a stack is rotated, the orientation is applied to each stacked device individually. `MY` and `R180` rotations also result in a left-to-right reversal of the ordering of the stack members.
- **Stack Names:** Stacks are named in one of the following ways:
  - If a naming pattern is recognized, then its base pattern is used to identify the stack.

    For example, a stack named `M0` is created in the following situations:
    - A stack that represents the layout devices `M0.0`, `M0.1`, `M0.2`, and `M0.3` that are bound to schematic device `M0` with an s-factor.
    - A stack that represents iterated layout devices `M0<0>`, `M0<1>`, `M0<2>`, and `M0<3>`.
  - If a naming pattern is not recognized, then the stack name is based on the name of the first stack member, alphabetically.
- **Dummies around Stacks:** Dummies can be added around stacks.
  - Dummies added above or below a stack result in a stack of dummies with the same number of members as the reference stack.
  - Dummies added to the left or right of a stack result in a single dummy being inserted.
  - Surround dummies inserts dummy stacks above and below the stack and a single dummy on each side.
- **Pattern Presets:** Built-in presets, other than the resistor presets, support the use of Modgen sandbox objects in both, the stacked and unstacked formats. Resistor presets support only the unstacked format.

Related Topics

Support for Stacks in Modgens

Array Assistant

Automatic Generation of Modgens using the Array Assistant

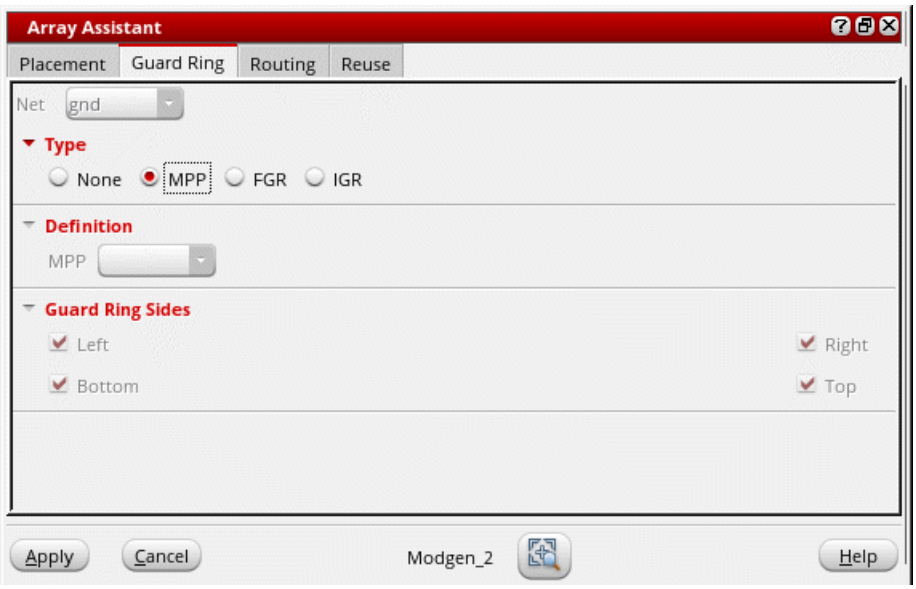
Creating Guard Rings Using the Array Assistant

(Layout EXL and Higher Tiers) Before creating guard rings, ensure that their definitions are added to the technology file.

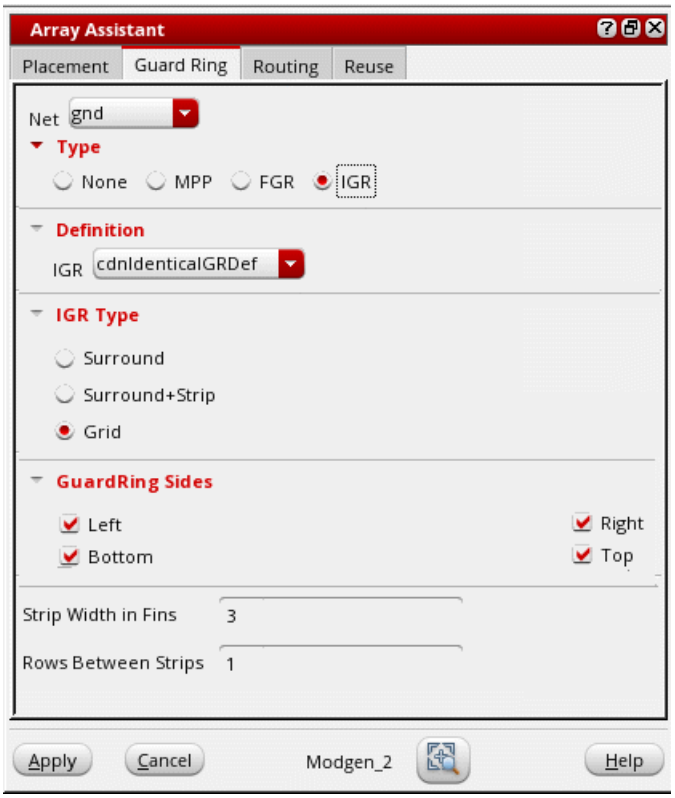
The options to create Identical Guard Rings (IGRs) are available only if the PDK that you are using is configured to support identical guard rings.

To create a guard ring:

1. Open the *Guard Ring* tab of the Array Assistant.
2. Select one of the following types of guard rings:
   - Multipart Path (*MPP*): The tool supports only surround MPP guard rings. Although the sides are listed, you cannot control the sides along which the guard ring is added.



- Fluid guard ring (*FGR*): The Create Guard Ring form is displayed. You can select a *Net* connection for the guard ring.



  Typically, in Layout XL, FGRs can be created in four different ways: by drawing a path, rectangle, or polygon, or by using the wrap mode. Each of these different modes represent a tab on the Create Guard Ring form. However, the Array Assistant only supports creation of FGRs in the *Wrap* mode. In this mode, an FGR is created around the objects you select. The fluid guard ring parameters are stored in the Modgen constraint.

  For information about the options on the *Wrap* tab of the Create Guard Ring form, see Wrap Mode.

- Identical guard ring (*IGR*): The IGR definition is provided by the PDK. See Creating an Identical Guard Ring Using the Array.

3. Specify the required guard ring settings.
4. Click *OK*.

A guard ring as per your definition is generated in the layout canvas.

Related Topics

Array Assistant

Creating an Identical Guard Ring Using the Array

Automatic Generation of Modgens using the Array Assistant

Modgen Placement Settings in the Array Assistant

Defining Modgen Topology Settings Using the Array Assistant

Reusing Modgen Templates Using the Array Assistant

Creating an Identical Guard Ring Using the Array

To create an Identical Guard Ring (IGR) using the Array Assistant:

1. Open the *Guard Ring* tab of the Array Assistant.
2. Select *IGR* from *Type*.
3. Select a *Net* connection for the guard ring.
4. Select an IGR definition from the *IGR* list. The IGR definition provided by the PDK defines all aspects of the guard ring, for example, the unit cell `lib:cell:view`, its parameters, and parameter callbacks, which helps the Modgen instantiate the guard ring correctly.
5. Select one of the following IGR types:
   - *Surround*: The IGR surrounds all the instances.
   - *Surround+Strip*: The IGR surrounds all the instances, and strips of guard ring instances are inserted between one or more Modgen rows.
   - *Grid*: The IGR surrounds every instance or group of abutted instances separately.
6. Select the sides around the Modgen on which you want to add guard rings: *Left*, *Right*, *Bottom*, *Top*. By default, all sides are selected.
7. Specify the strip width by specifying the number of fins in the *Strip Width in Fins* field.
8. Specify the number of rows to be inserted between strips of IGRs in the *Rows Between Strips* field.
9. Click *OK*.

An IGR is created as per your specifications.

Related Topics

Array Assistant

Automatic Generation of Modgens using the Array Assistant

Modgen Placement Settings in the Array Assistant

Defining Modgen Topology Settings Using the Array Assistant

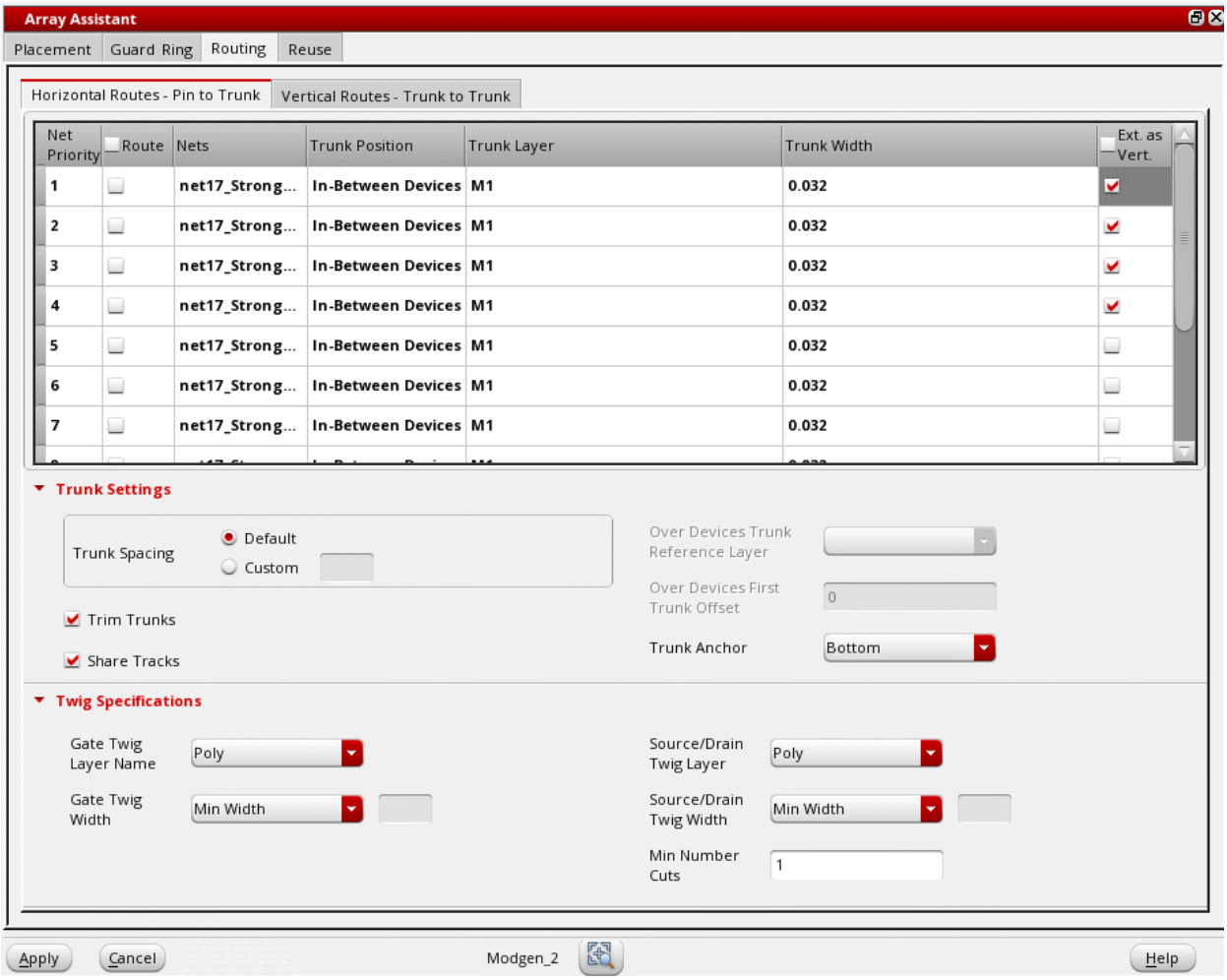
Reusing Modgen Templates Using the Array Assistant

Defining Modgen Topology Settings Using the Array Assistant

(Layout EXL and Higher Tiers) The *Routing* tab of the Array Assistant provides options to use the pin to trunk router to define Modgen topology patterns. This involves specifying routing preferences for horizontal nets and vertical nets. Modgen topology patterns help you to visualize, configure, and store routing information. Omit this task if you want to use any other router, such as the tree router, to route the Modgen devices.

To define the Modgen topology pattern and routing options:

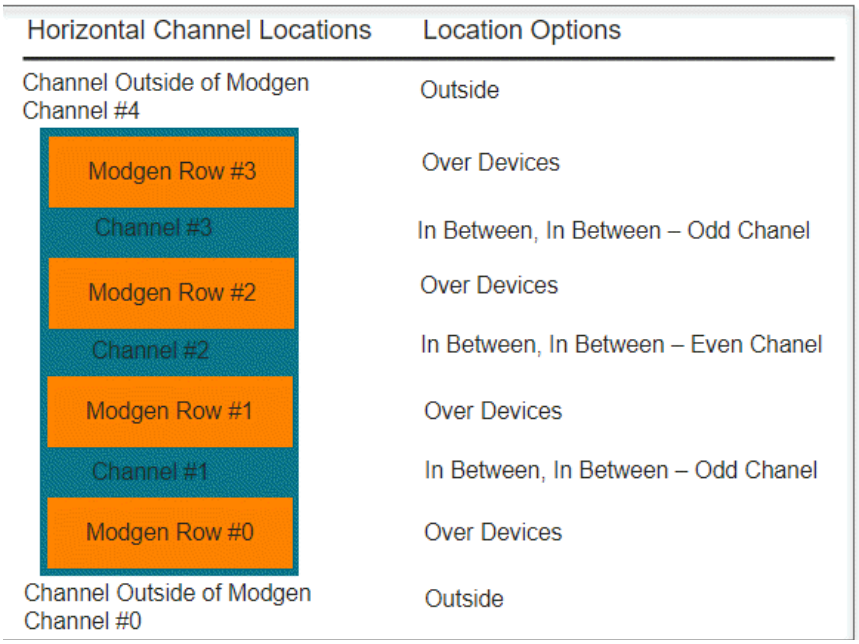
1. Open the *Routing* tab of the Array Assistant. The *Horizontal Routes - Pin to Trunk* tab is displayed by default.



2. Customize settings in the net table, which lists all the channel nets in the current Modgen.

- Customize the sequence of nets in the *Net Priority* column by dragging nets to the required position in the table.
- Select *Route* in individual rows to specify the nets to be used for routing. To select all nets, select *Route* in the column header.
- *Nets* lists the names of nets that are present in the current Modgen.
- Double-click *Trunk Position* and select the location for generating trunks.

  The following image shows the different channel locations:



- Double-click *Trunk Layer* for each net and select the required layer from the drop-down list.
- Specify the width of trunks in the *Trunk Width* field.
- Select *Ext. as Vert.* to specify whether the net is to be listed on the *Vertical Routes - Trunk to Trunk* tab.

3. Set the following options in the *Trunk Settings* section. These settings are applied only to the nets selected for routing.

a. Set *Trunk Spacing* to *Default* or select *Custom* to specify an absolute value.

b. Select *Trim Trunks* to trim the ends of horizontal and vertical trunks while routing.

c. Select *Share Tracks* to share the horizontal trunks that are on the same layer and have the same connectivity.

d. Select a layer from the *Over Devices Trunk Reference Layer* list. The twigs connected to the source and drain terminals are generated in this layer.

This option is available only for nets that have *Trunk Position* set to *Over Devices*.

e. Specify the trunk offset value in the *Over Devices First Trunk Offset* field.

This option is available only when *Trunk Position* of at least one net is set to *Over Devices*. When the field is left blank, the cell boundary of the Modgen is used as the reference layer.

f. Set *Trunk Anchor* to either *Top* (top-left vertex of the device row) or *Bottom* (bottom-left vertex of the device row).

4. Set the following options in the *Twig Specifications* section. These settings are applied only to the nets selected for routing.

a. Select a layer from the *Gate Twig Layer Name* list. The twigs that are connected to the gate terminal are generated on this layer.

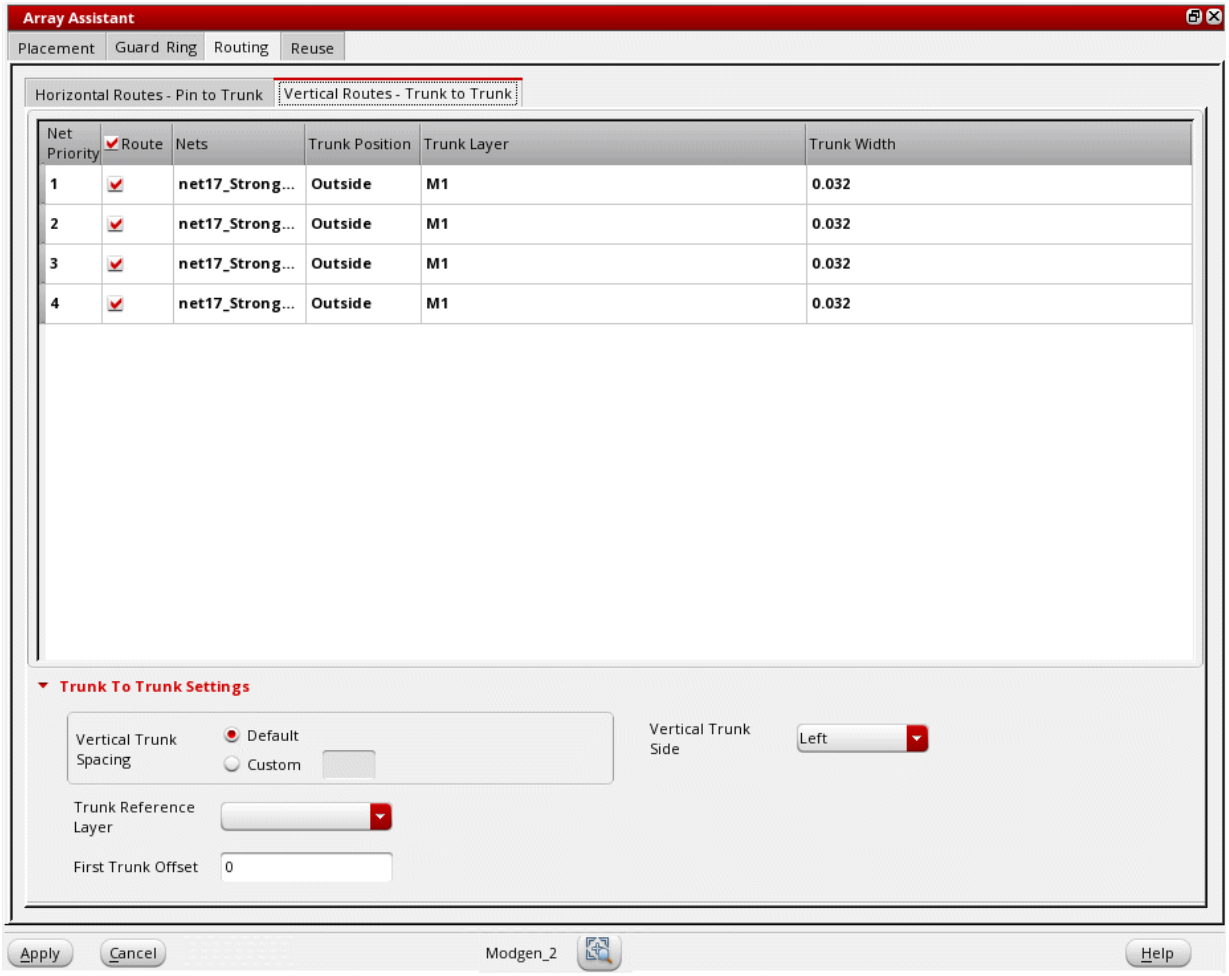
b. Specify the width of the gate twigs in the *Gate Twig Width* field.

c. Select a layer from the *Source/Drain Twig Layer* list. The twigs connected to the source and drain terminals are generated on this layer.

d. Specify the source and drain twig width values in the *Source/Drain Twig Width* field.

e. Specify the *Min Number Cuts* for the vias connecting the twigs to other objects.

5. Switch to the *Vertical Routes - Trunk to Trunk* tab to specify routing preferences for vertical nets.



The net table settings are similar to the *Horizontal Routes - Pin to Trunk* tab.

6. Specify the following options in the *Trunk To Trunk Settings* section:

   a. Set *Vertical Trunk Spacing* to *Default* or select *Custom* to specify an absolute value.

   b. Specify the trunk offset by selecting the reference layer from the *Trunk Reference Layer* list and specifying the device-to-trunk offset of the first device in the *First Track Offset* field.

   c. Select a *Vertical Trunk Side* to specify the side along which vertical trunks are to be generated.

7. Click *Apply* to apply the routing settings.

Related Topics

Array Assistant

Automatic Generation of Modgens using the Array Assistant

Modgen Placement Settings in the Array Assistant

Creating Guard Rings Using the Array Assistant

Reusing Modgen Templates Using the Array Assistant

Reusing Modgen Templates Using the Array Assistant

(Layout EXL and Higher Tiers) The *Reuse* tab of the Array Assistant provides options to load and save settings to Modgen template files.

Loading Settings from a Modgen Template File

To load placement settings from an existing Modgen template file:

1. Open the *Reuse* tab of the Array Assistant.

2. Select a *Device Group*.

   The default is the device group to which the selected instances belong. If the selected instances are not part of any device group, the *Device Group* is set to *Generic Group*.

3. Select a Modgen template file name from the *File* drop-down list.
4. Click *Load* to load the values stored in the file.

Settings from the selected Modgen template file are loaded into the form, which you can further customize.

Saving Settings to a Modgen Template File

You can store current placement settings from the form to a template file. After making the required updates to the form, in the *Template* section of the *Reuse* tab:

1. Specify a unique name in the *File* field.
2. Click *Save*.

The current placement settings are stored in the specified Modgen template file. Later, you can apply this template file to other device groups to generate similar Modgens.

Related Topics

Array Assistant

Automatic Generation of Modgens using the Array Assistant

Modgen Placement Settings in the Array Assistant

Creating Guard Rings Using the Array Assistant

Defining Modgen Topology Settings Using the Array Assistant

3

Modgen Tasks

In Virtuoso, you can use either the *Modgen Placement* toolbar in the Modgen Editor or the on-canvas Modgen commands to customize Modgens by performing the following tasks:

- Modgen Transparent Editing Mode
- Generating a Modgen Template File
- Reusing a Modgen Template File
- Adding and Deleting Dummies in the Modgen Editor
- Adding and Removing Modgen Body Contacts
- Grid Placement In Modgens
- Modgen Guard Rings
- Abutting Modgen Devices
- Specifying Modgen Device Alignment and Spacing
- Merging Layers and Wells

Related Topics

Modgen On-Canvas Commands

Modgen Reusable Templates

Modgen Dummies

Body Contacts in Modgens

Modgen Device Abutment

Prevent Unauthorized Edits to Modgens

You can prevent Modgen from unauthorized edits. If a user edits a Modgen by moving instances or body contacts or by stretching wires, the edit is ignored. When the Modgen is updated after such edits, the moved or stretched object are reset as per the constraint.

Instead of invoking the undo command, the Modgen layout is re-done according to the information in the constraint.

- If geometry is added or deleted from the Modgen, the Modgen constraint is disabled. This means that it is no longer a Modgen. However, it will remain a figGroup of type `none`.
- If the geometry is accidentally added or deleted, it can be removed or re-added by undoing the operation (or by pressing the `u` key). Invoking undo will remove (or re-add) the geometry and re-enable the Modgen constraint.
- If the geometry was added on purpose, you can exit the Modgen Editor and the result will be a standard figGroup.

  At this point, to turn the figGroup back into a Modgen, first edit the figGroup to remove (add) the added (deleted) geometry before re-enabling the Modgen constraint.

Related Topics

Open Modgens in the Modgen Editor

Modgen On-Canvas Commands

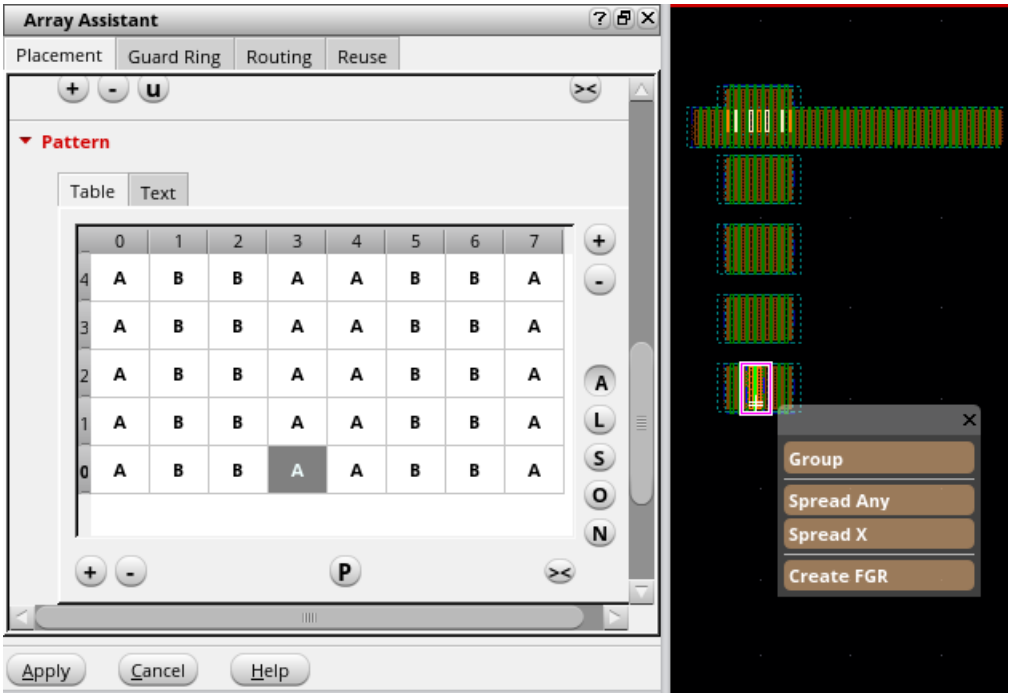
Modgen Transparent Editing Mode

A Modgen is an important structure in custom layouts. There are multiple ways to create and edit Modgens.

In Layout MXL, instead of invoking any GUI, you can use Modgen Transparent Editing mode to modify a Modgen directly from the top level. To invoke Modgen Transparent Editing mode, click the icon in the Options toolbar.

In this mode, double-clicking a Modgen does not invoke Modgen Editing mode. Modgen Transparent Editing mode allows all the functionalities of Modgen Editing mode, but with the top-level canvas object in context. The Virtuoso workspace menus and toolbars are also available in this mode.

You can use the Array Assistant in combination with Modgen Transparent Editing mode. In Modgen Transparent Editing mode, the tool supports cross-selection of instances between the layout canvas and the Array Assistant. Modifications done to array instances in the Array Assistant are immediately reflected in layout canvas. For example, if you apply a base pattern in the Array Assistant, the instances in the layout canvas are automatically re-arranged as per the specified base pattern.



In Modgen Transparent Editing mode, when the `leHiMove` enter function is used to move the selected array grid members outside the current array area, the array is regenerated and these grid members are removed from the Modgen array. Similarly, when the function is used

to move non-array grid members inside an array area and the array is regenerated, these members are added to the array. You can also move an instances directly from one Modgen array into another.

At any point, you can click the icon in the Options toolbar to turn off Modgen Transparent Editing mode.

Alternatively, use the transparentModgenArray environment variable to control the mode.

Related Topics

Modgen Placement Settings in the Array Assistant

Open Modgens in the Modgen Editor

Modgen On-Canvas Commands

Virtuoso provides the following Modgen on-canvas commands to perform certain tasks without opening the Modgen Editor. To access the Modgen on-canvas commands, either select the Modgen constraint in the layout canvas and choose *Modgen* from the shortcut menu or select *Place—Modgen*.

Command Name	Description
Create/Edit Modgen	If the selected device(s) are not part of an existing Modgen constraint, creates a new Modgen constraint. The Modgen Editor is not displayed. If the selected device(s) are part of an existing Modgen constraint, opens it in the Modgen Editor.
Create Reuse Template	Generates a reusable template for each device group available in the given Modgen.
Apply Reuse Template to Target	Applies existing Modgen reuse templates to the specified instances to generate matching Modgens.
Array Assistabt	Displays the Array Assistant.
Split Rows	Splits each Modgen row into two rows.
UnSplit Rows	Combines every two rows of Modgen into a single row.
Abut Instances	Abuts the Modgen devices.
UnAbut Instances	Unabuts the Modgen devices.
Dummies	Displays a submenu with the following commands that can be used to add or remove dummies: ■ Add Dummy Column Left ■ Add Dummy Column Right ■ Add Dummy Row Top ■ Add Dummy Row Bottom ■ Add Surround Dummies ■ Remove Surround Dummies ■ Delete All Dummies

Command Name	Description
Set Member Alignment/Spacing	Displays the Set Member Alignment and Spacing form to specify the alignment and spacing values for the Modgen devices.
Merge Layers	Displays the Select merge layers form, in which you can specify the layers that need to be merged.
Guard Ring	Displays a sub-menu with the following commands that can be used to create different types of guard rings: ■ Add Fluid Guard Ring ■ Add MPP Guard Ring ■ Add Identical Guard Ring ■ Remove Guard Ring

Related Topics

Creating a Modgen

The Move Command in the Array Assistant

Modgen Device Abutment

Modgen Dummies

Specifying Modgen Device Alignment and Spacing

Merging Layers

Modgen Guard Rings

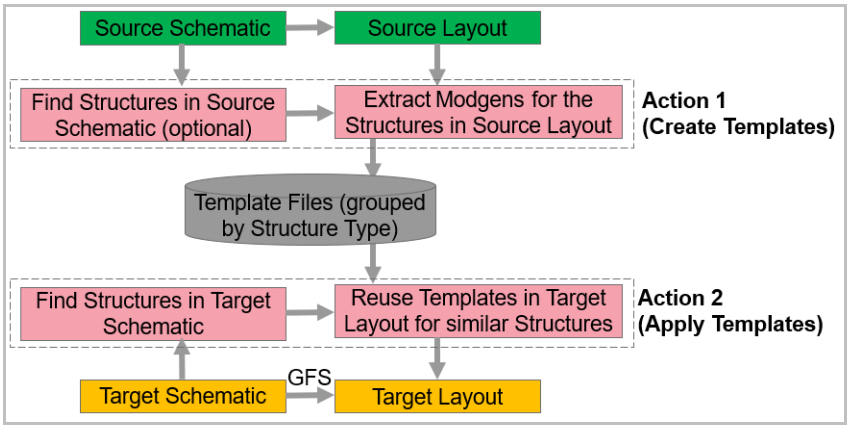
Modgen Reusable Templates

Virtuoso supports a template-driven Modgen reuse solution to help improve layout productivity. A Modgen template comprises a set of Modgen parameters, such as the interdigitation pattern, abutment, dummy definitions, and spacing values, that can be reused to create a gridded layout of matching structures.

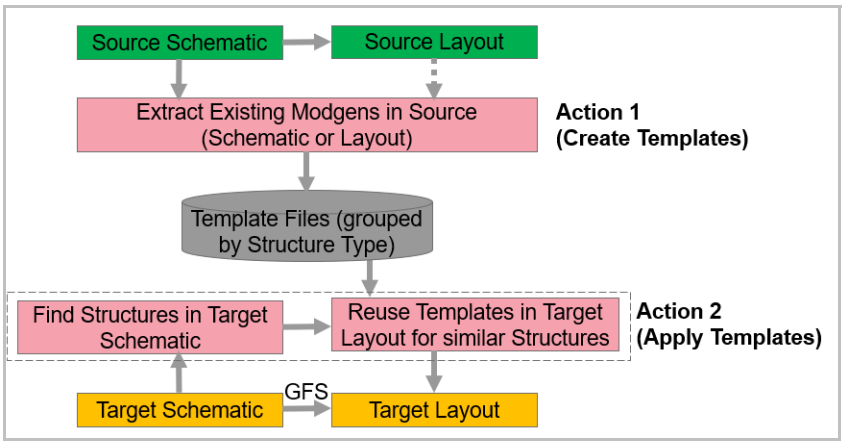
Note: Mosaics are not supported for extraction and reuse.

The following diagrams depict the template-driven Modgen reuse flow.

Flow 1 - There are no Modgen Structures in the Source



Flow 2 - There are Modgen Structures in the Source



1. The tool extracts Modgens from the source layout for important structures in the source schematic (Flow 1) and for existing Modgens, if available (Flow 2).
2. The information is saved in the Modgen template files.

3. The template files are then grouped by their structure type. The Modgen templates are reused in the target layout to generate Modgens that include similar structures in the target schematic.

Related Topics

gpeExtractTemplateFromMG (Corresponding SKILL Function)

Generating a Modgen Template File

Parameters of a Modgen Reuse Template File

Reusing a Modgen Template File

Generating a Modgen Template File

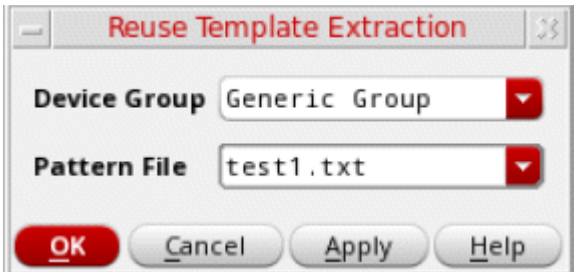
To generate a Modgen template file:

1. Select the required Modgen constraint either in the layout canvas or in the Constraint Manager assistant.

   If no Modgen constraint is selected, templates are extracted from all the Modgens in the active layout canvas.

2. Open the Reuse Template Exaction form using one of the following methods:

   - Choose *Place—Modgen—Extract Template*.



3. Specify a *Device Group*. Template files are created under matching or specified group names. The default is *Generic Group*.
4. Specify a *Pattern File* name with the `.txt` extension. This is the Modgen template file.
5. Click *Apply* to save the template file.
6. Click *OK*.

Constraint parameters from the source Modgen are stored in the template file.

Related Topics

gpeExtractTemplateFromMG (Corresponding SKILL Function)

Modgen Reusable Templates

Parameters of a Modgen Reuse Template File

Reusing a Modgen Template File

Parameters of a Modgen Reuse Template File

A Modgen template comprises a set of Modgen parameters that can be used to create a grid-based layout of matching structures. The following image shows a sample Modgen reuse template file:

```
rank=0
genericPattern
mapping="M9 A\nM3 B\nM4 C\nM10 D\nM7 E\nM5 F\nM6 G\nM8 H\n"
baseOrient="MY MY R0 R0"
orient="MY MY R0 R0\nMY MY R0 R0\n"
zigZag=nil
basePattern="E F G H\nA B C D\n"
pattern="E F G H\nA B C D\n"
interdigitateBy=0
rows=2
verticalSpacingLayer='("" "")
verticalSpacingDistance=2.0
horizontalSpacingLayer='("" "")
horizontalSpacingDistance=2.0
abut=nil
mergeLayer="default"
dummyLengthValue="18.0n"
dummyLength="Specify"
dummyNumFingersValue=0
dummyNumFingers="Same as Neighbor"
dummyNet="gnd"
dummyBracket=nil
dummyEdgeBottom=t
dummyEdgeTop=t
dummyEdgeRight=t
dummyEdgeLeft=t
dummyView="layout"
dummyCell="n1lvt"
dummyLib="cds_ff_mpt"
dummyTransitionColumns=2
routeEnable=t
routeOverDevice=nil
horizontalChannelNets=list("7" "8" "9" "10")
horizontalOutsideNets=list("7" "8" "9" "10")
horizontalChannelNetsDirection="bottom"
horizontalOutsideNetsDirection="bottom"
trimTrunks=nil
shareHorizontalTracks=t
horizontalTrunkLayerName="Metal2"
verticalTrunkLayerName="Metal3"
twigGLayerName="Poly"
twigSDLayerName="Metal1"
twigGLayerWidth=nil
twigSDLayerWidth=nil
horizontalMinNumCuts=1
verticalMinNumCuts=2
enableVerticalTrunks= t
verticalTrunkSide="auto"
router="none"
```

Pattern Parameters

Dummy Parameters

Router Parameters

The following table describes the parameters of a Modgen template file:

`mapping`	Specifies the mapping between devices and symbols. You can also define mapping for dummies. Value: Pairs of strings *`t_source t_symbol`* Example: `"Dummy * S0 B S1 C\nS2 A"`
`baseOrient`	Specifies the interdigitation pattern of Modgen instances in terms of their orientations. Value: A list of one or more space-separated symbols from the set {`"R0", "R90", "R180", "R270", "MX", "MY", "MXR90", "MyR90", "-"`}.) Example: `"R270 R90\nMX MY"`
`orient`	Specifies orientations of the Modgen instances. Value: A list of one or more space-separated symbols from the set {`"R0", "R90", "R180", "R270", "MX", "MY", "MXR90", "MyR90", "-"`}) Example: `"R0 R0 - -  R0 R0"`
`basePattern`	Specifies the interdigitation pattern of Modgen instances in terms of their symbols. Value: String of one or more space-separated symbols *`t_pattern`* Example: `"B A * A B"`
`pattern`	Specifies the pattern to place instances in a Modgen. Value: String of one or more space-separated symbols *`t_pattern`* Example: `"- B A *\nA B -"`
`zigZag`	Specifies whether a zigzag pattern is to be followed in the Modgen. Value: Boolean Example: `t`

`interdigitateBy`	Specifies the integer value based on which instances are interdigitated. Value: integers zero or greater Example: `0`
`rows`	Specifies the number of rows in the Modgen array or sandbox. Value: A positive integer *`x_rows`* Example: `2`
`verticalSpacingLayer`	Specifies the reference layer based on which the vertical spacing of devices must be calculated. Value: A list of two strings *`t_layer t_purpose`*, the string `all`, or an empty string. Example: `M3 drawing`
`verticalSpacingDistance`	Specifies the vertical spacing of devices in the Modgen. Value: A number *`x_verticalSpacingDistance`* (in user units)or `nil` Example: `3`
`horizontalSpacingLayer`	Specifies the reference layer based on which the horizontal spacing of devices must be calculated. Value: List of two strings in the format *`t_layer t_purpose`*, the string `all`, or an empty string. Example: `M3 drawing`
`horiztontalSpacingDistance`	Specifies the horizontal spacing of devices in the Modgen. Value: Either a number (in user units) *`x_horiztontalSpacingDistance`* or `nil` Example: `3`

`abut`	Specifies whether the Modgen instances are to be abutted. The tool attempts to abut all array members. Value: Boolean Default Value: `nil` Example: `t`
`mergeLayer`	Specifies the layers to be merged. Value: String of colon-separated layer names, the string `well`, or the string `default` Example: `Oxide:Metal`
`Dummy Type`	
`dummyOverrideParams`	Specifies dummy parameters to overrride existing values. Value: A list of two values in the format '(*`paramName paramValue`*)) Example: (`dummyParam1 5`)
`dummyWidthValue`	Specifies the dummy width value in terms of the number of fins. Value: Integer *`n_dummyWidthValue`* Example: `4`
`dummyWidth`	Specifies the mode for specifying dummy width value. Value: String. The valid values are: ■ `CDF Default` – The default number of fins specified in the CDF ■ `Same As Neighbor` – The number of fins of the neighboring device ■ `Specify` – You can specify the number of fins Example: `Same as Neighbor`
`dummyLengthValue`	Specifies the dummy length value in terms of the number of fins. Value: *`n_dummyLengthValue`* Example: `4`

`dummyLength`	Specifies the mode for specifying the length of dummy fingers. Value: String. The valid values are: ■ `CDF Default` – The default finger length specified in the CDF ■ `Same As Neighbor` – The length of fingers of the neighboring device ■ `Specify` – You can specify the length of fingers Example: `Specify`
`dummyNumFingersValue`	Specifies the number of dummy fingers to be created. Value: *`n_dummyNumFingersValue`* Example: `2`
`dummyNumFingers`	Specifies the mode for specifying the number of dummy fingers. Value: String. The valid values are: ■ `CDF Default` – The default number of fingers specified in the CDF ■ `Same As Neighbor` – The number of fingers of the neighboring device ■ `Specify` – You can specify the number of fingers Example: `Specify`
`dummyNet`	Specifies the net to which all dummy terminals must connect. Value: A string representing a valid net name. Example: `net1`
`dummyBracket`	Checks for the presence of surround dummies. Value: Boolean. The default value is `nil`. Example: `t`

`dummyEdgeBottom`	Checks for the presence dummies along the bottom edge of the Modgen constraint. Value: Boolean. The default value is `nil`. Example: `t`
`dummyEdgeTop`	Checks for the presence dummies along the top edge of the Modgen constraint. Value: Boolean. The default value is `nil`. Example: `t`
`dummyEdgeLeft`	Checks for the presence dummies along the left edge of the Modgen constraint. Value: Boolean. The default value is `nil`. Example: `t`
`dummyEdgeRight`	Checks for the presence dummies along the right edge of the Modgen constraint. Value: Boolean. The default value is `nil`. Example: `t`
`dummyTransitionColumns`	Specifies the number of transition dummy columns to be inserted. Value: *`n_dummyTransitionColumns`* Example: `2`
`dummyLib`	Specifies the default library to use when creating custom dummy devices. Value: String Example: `MyLib`
`dummyCell`	Specifies the default cell to use when creating custom dummy devices. Value: String Example: `MyCell`
`dummyView`	Specifies the default view to use when creating custom dummy devices. Value: String Example: `MyView`

Guard Ring Type

`guardRingType`

Determines the presence of the specified type of guard ring in the current Modgen array or sandbox.

Value: String. The valid values are:

- `None` (default): Does not look for guard rings.
- `FGR`: Identifies fluid guard rings.
- `MPP`: Identifies multipath part guard rings.
- `IGR`: Identifies identical guard rings.

Example: `FGR`

`guardRingSubType`

Determines the geometry configuration of the MPP or IGR guard ring in the current Modgen array or sandbox.

Value: String. the valid values are:

- `ring` (default): A single guard ring around the entire Modgen. This type is valid for MPPs guard rings and IGRs.
- `pane`: A guard ring around all devices as well as the entire Modgen. This type is valid for MPPs guard rings and IGRs.
- `stripe`: Strips of guard ring between one or more Modgen rows. This type is valid only for IGRs.

Example: `pane`

`guardRingNet`

Determines the net to which the guard ring geometry terminals are connected in the current Modgen array or sandbox.

Value: String (net name)

Example: `net1`

`guardRingShape`

Determines the surround style of the guard ring in the current Modgen array or sandbox.

Value: String. The valid values are `rectangular` and `rectilinear`.

Example: `rectilinear`

`guardRingDefName`	Determines an external guard ring definition in the Modgen array or sandbox. Value: String Example: `myGuardRing1`
`guardRingSpacing`	Specifies the spacing between the sides of a guard ring and the edges of its enclosed Modgen device. The value must be either equal to or greater than the `minDRC` value. This value applies only to MPP guard rings. Value: Integer (in user units) Example: `3`
`guardRingLeftSpacing`	Specifies the spacing between the left edge of a guard ring and the left edge of the enclosed Modgen device. The value must be either equal to or greater than the `minDRC` value. This value applies only to MPP guard rings. Value: Integer (in user units) Example: `3`
`guardRingRightSpacing`	Specifies the spacing between the right edge of a guard ring and the right edge of the enclosed Modgen device. The value must be either equal to or greater than the `minDRC` value. This value applies only to MPP guard rings. Value: Integer (in user units) Example: `3`
`guardRingTopSpacing`	Specifies the spacing between the top edge of a guard ring and the top edge of the enclosed Modgen device. The value must be either equal to or greater than the `minDRC` value. This value applies only to MPP guard rings. Value: Integer (in user units) Example: `3`

`guardRingBottomSpacing`	Specifies the spacing between the bottom edge of a guard ring and the bottom edge of the enclosed Modgen device. The value must be either equal to or greater than the `minDRC` value. This value applies only to MPP guard rings. Value: Integer (in user units) Example: `3`
`guardRingSides`	Specifies the sides of the Modgen array or sandbox to be enclosed by a guard ring. This value is applicable to MPP guard rings and IGRs. Value: A comma-separated string. The valid values are `top`, `bottom`, `left`, and `right`. Example: `top, bottom`
`guardRingHorizontalStripeWidth`	Specifies the vertical width of the horizontal guard ring strips in terms of either the number of fins or an absolute width value. This value applies only to IGRs. Value: Integer (in user units) Example: `2`
`guardRingRowsBetweenStripes`	Specifies the number of array rows between each guard ring stripe row in the array or Modgen sandbox. This value applies only to IGRs. Value: Integer (in user units) Example: `2`
`guardRingBreak`	Specifies whether the guard ring metal geometry <fill in here> in the Modgen array sandbox. This value applies only to IGRs. Value: Integer (in user units) Example: `2`

`guardRingAddCorners`

Specifies whether the corner guard ring devices must be added to the array or sandbox.

This value applies only to IGRs.

Value: Boolean. The default value is `nil`.

Example: `t`

Routing Type

`router`

Specifies whether the router must be invoked for the Modgen array or sandbox.

Value: String. The valid values are:

- `none` (default): Does not run any router.
- `pinToTrunk`: Runs the pin to trunk router

Example: `pinToTrunk`

`horizontalInteriorChannelNets`

`rank = n_TemplateRank`

Description: Specifies the rank of the template.

Valid Values: non-negative integer

The template with the lowest rank is considered the default value. This template is applied by the *Apply Default Reuse Template to Target* command.

`routeEnable = t | nil`

Description: Specifies whether the pin-to-trunk router is to be used to route the Modgen devices.

Valid Values: `t` and `nil`

When set to `nil`, all route-related parameters are ignored and a topology pattern is not created.

When `routeEnable` is set to `t` and `router` is set to `nil`, a topology pattern is created but the pin-to-trunk router is not called.

`routeOverDevice = t | nil`

Description: Allows horizontal routes to be generated over devices.

Valid Values: `t` and `nil`

`horizontalChannelNets = l_netNames | nil`

Description: (Optional) Lists channel nets for the horizontal trunks. If not specified (which is recommended), channel nets in the new Modgen are created based on the net connections.

Note: Channel nets are the nets between Modgen rows.

Valid Values: *`l_netNames`* and `nil`

Default Value: `nil`, which indicates no channel nets are to be created.

Example:

```
list("net1" "net2" "net3")
```

`horizontalOutsideNets = l_netNames | nil`

Description: Lists the nets that are above or below the Modgen. When this parameter is not specified, outside nets are created based on connections.

Valid Values: *l_netNames*, nil

Default Value: nil, which indicates that no trunks are to be created above or below the Modgen.

Example:
```
horizontalOutsideNets=list("net1
net2 net3")
```

horizontalTrunkWidths = *l_netWidths* | nil

Description: Specifies a space-separated list of horizontal trunk widths.

Valid Values: *l_netWidths*, nil

If a single value is specified, it is applied to all channel nets.

Default Value: nil, which indicates that the default values from either the technology file or a predefined WSP are used.

Example:

```
list(list("D1" 0.22) list("D2" 0.22))
```

horizontalNetOrder = *l_netOrder* | nil

Description: Specifies the net order of channel nets.

Valid Values: *l_netOrder* and nil

Default Value: nil

horizontalNetOrder values depends on the value of horizontalChannelNets.

trimTrunks = t | nil

Description: Specifies whether the ends of the horizontal trunks are to be trimmed while routing.

Valid Values: t and nil

`shareHorizontalTracks = t | nil`

Description: Specifies whether two nets can share the same horizontal trunk.

Valid Values: `t` and `nil`

`horizontalTrunkLayerName = t_layerName`

Description: Specifies the layer on which horizontal trunks are to be generated. This is a mandatory parameter.

Valid Values: *`t_layerName`*

`twigGLayerName = t_layerName`

Description: Specifies the layer in which the twigs that are connected to the gate terminal are to be generated. This is a mandatory parameter.

Valid Values: *`t_layerName`*

`twigGLayerWidth = f_twigGLayerWidth`

Description: Specifies the width of the gate net twigs.

Valid Values: *`f_twigGLayerWidth`*

`twigSDLayerName = t_layerName`

Description: Specifies the layer in which the twigs connected to the source and drain terminals are to be generated.

Valid Values: *`t_layerName`*

`twigSDLayerWidth = t_twigSDLayerWidth`

Description: Specifies the width of the source and drain gate twigs. This is a mandatory parameter.

Valid Values: *`t_twigSDLayerWidth`*

`horizontalMinNumCuts = nil | n_numCuts`

Description: Specifies the minimum number of cuts for the vias connecting the twigs to other objects.

Valid Values: `nil` and *`n_numCuts`* }

Default Value: `1`

`anchorReference` = `nil` | *`t_refLayer`*

Description: Specifies the reference layer or the anchor from which the trunk chain must start.

Valid Values: `nil` and *`t_refLayer`*

`firstHorizontalChannelTrackOffset` = `nil` | *`f_offset`*

Description: Specifies the offset of the first horizontal channel track from the specified `anchorReference` value.

Valid Values: `nil` and *`f_offset`*

`firstHorizontalOutsideTrackOffset` = `nil` | *`f_offset`*

Description: Specifies the offset of the first horizontal track outside the channel from the specified `anchorReference` value.

Valid Values: `nil` and *`f_offset`*

`firstHorizontalDeviceTrackOffset` = `nil` | *`f_offset`*

Description: Specifies the offset of the first horizontal device track of the `overDevice` trunk chain.

Valid Values: `nil` and *`f_offset`*

`enableVerticalTrunks` = `t` | `nil`

Description: Specifies whether vertical trunks can be created. The default value is `nil`.

Valid Values: `t` and `nil`

`verticalTrunkLayerName` = *`t_layerName`*

Description: Specifies the layer on which vertical trunks are to be generated. The option can be set only if `enableVerticalTrunks` is set to `t`.

Valid Values: *`t_layerName`*

`verticalTrunkWidths` = *`l_netWidths`* | `nil`

Description: Specifies a space-separated list of vertical trunk widths. If a single value is specified, it is applied to all channel nets. The option can be set only if `enableVerticalTrunks` is set to `t`.

Valid Values: *`l_netWidths`* and `nil`

Default Value: `nil`, which indicates that the default values from either the technology file or a predefined WSP are to be used.

Example:

```
list(list("D1" 0.22) list("D2" 0.22))
```

`verticalNetOrder = l_netOrder | nil`

Description: Specifies the net order of vertical channel nets. The default value is `nil` because the default `horizontalChannelNets` is `nil`. The option can be set only if `enableVerticalTrunks` is set to `t`.

Valid Values: *`l_netOrder`* and `nil`

`verticalMinNumCuts = nil | n_numCuts`

Description: Specifies the minimum number of cuts for the vias connecting the twigs to other objects. The default value is `1`. The option can be set only if `enableVerticalTrunks` is set to `t`.

Valid Values: `nil` and *`n_numCuts`*

`firstVerticalTrackOffset = nil | f_offset`

Description: Specifies the offset of the first vertical track from the specified `anchorReference` value.

Valid Values: `nil` and *`f_offset`*

`verticalTrunkSide ="both" | "auto" | "left" | "right"`

Description: Specifies the side along which vertical trunks are to be generated.

The option can be set only if `enableVerticalTrunks` is set to `t`.

Valid Values: `left`, `right`, `both`, and `auto`.

`router {"none" | "pinToTrunk" }`

Description: Specifies whether the pin-to-trunk router is to be used for routing.

`horizontalChannelNetsDirection ="top" | "bottom" | "both"`

Description: Specifies the direction of the horizontal channel nets, which is the direction from the trunk to the instance terminal.

Valid Values: `top, bottom, both`

Default Value: `both`

`horizontalOutsideNetsDirection ="top" | "bottom" | "both"`

Description: Specifies the direction of the horizontal nets that are outside channels.

Valid Values: `top, bottom, both`

Default Value: `both`

Related Topics

gpeExtractTemplateFromMG (Corresponding SKILL Function)

Generating a Modgen Template File

Parameters of a Modgen Reuse Template File

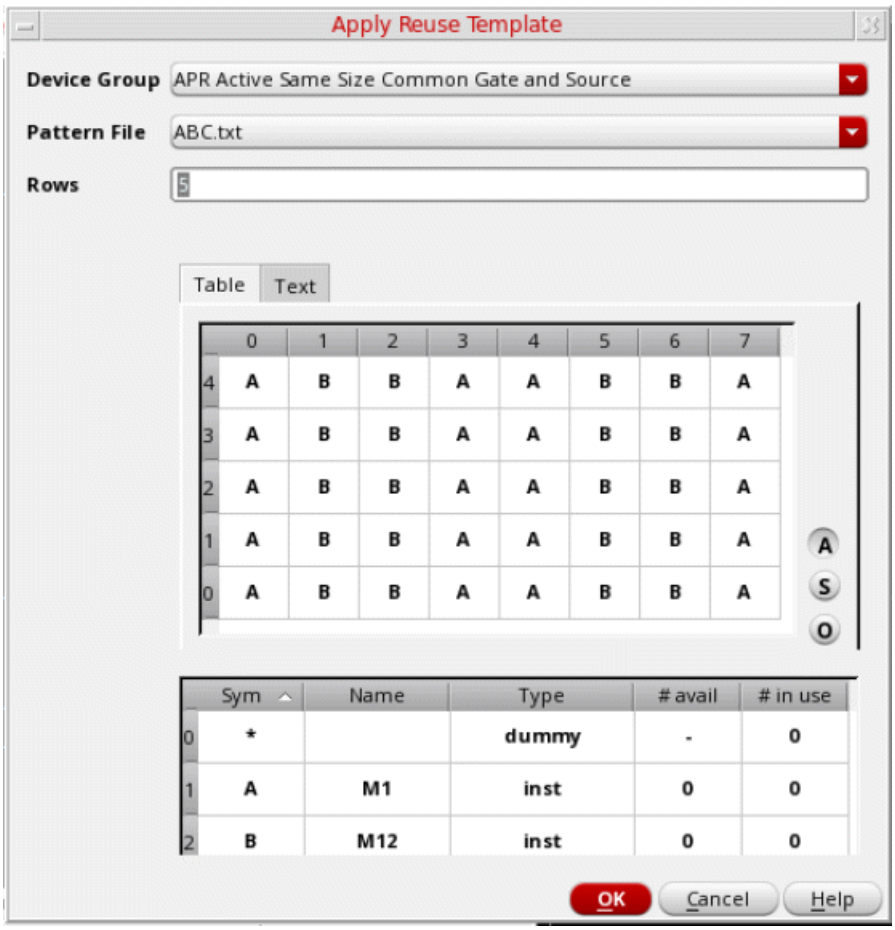
Reusing a Modgen Template File

Reusing a Modgen Template File

A Modgen template file can be reused to generate a new Modgen constraint.

To create a Modgen using a template file:

1. Open the schematic design in which a Modgen is to be generated.
2. Select the required devices. In the schematic view, select devices from the Circuit Prospector assistant. In the layout view, you can select the target instances or figGroups in many ways, for example, from the Navigator assistant or directly on the canvas.
3. Open the Apply Reuse Template form using one of the following methods:
   - Choose *Place—Modgen—Apply Reuse Template to Target*.
   - Use the `gpeLoadReuseTemplate` SKILL function.



4. Select the *Device Group* that contains the required Modgen reuse template.

5. Select the *Pattern File* corresponding to the Modgen template to be applied to the new Modgen. Parameters from the pattern file are loaded in the form.
6. Specify the number of *Rows* to be generated.
7. Update the pattern as per your requirements.
8. Click *OK* to generate the Modgen.

The new Modgen is listed in the Constraints Manager assistant.

Related Topics

Apply Reuse Template Form

gpeLoadReuseTemplate (SKILL function)

Generating a Modgen Template File

Parameters of a Modgen Reuse Template File

Parameters of a Modgen Reuse Template File

Modgen Dummies

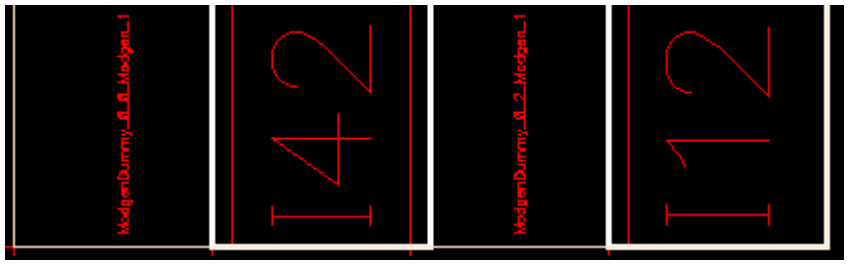
Dummy devices are created to counter electrical effects that are observed at small geometries. You can create dummies around devices in Modgens.

A device instance is considered a dummy if one or more of the following conditions are met:

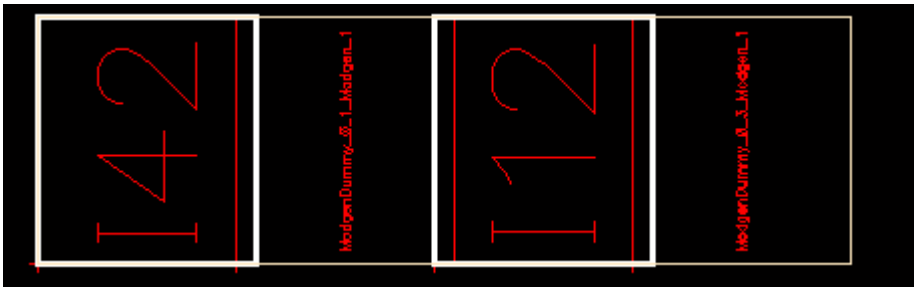
- The instance has one of the following properties set to `t`:
  - `lxDummy`
  - `ignore`
  - `lvsIgnore`
- The instance is not bound to the schematic. This behavior is controlled by a the `modgenCreateUnboundAsDummies` environment variable. The default value is `t`. In this state, the instance is considered a dummy. When set to `nil`, the instance is not considered a dummy.
- The instance gate net is the same as that assigned to the environment variable `modgenDummyNet`.

The *Add Dummy* button on the *Modgen Placement* toolbar provides the following options to add dummies at different locations:

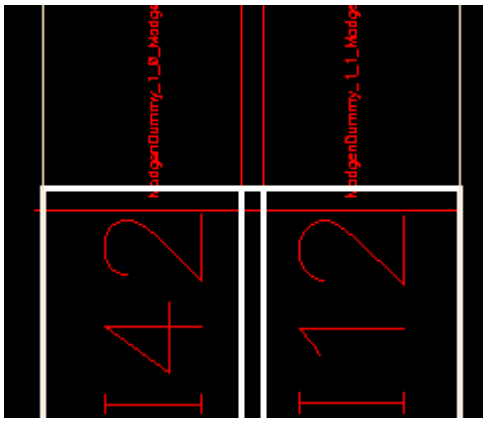
- **Add Dummy Left:** Adds dummy devices to the left of the selected Modgen instances



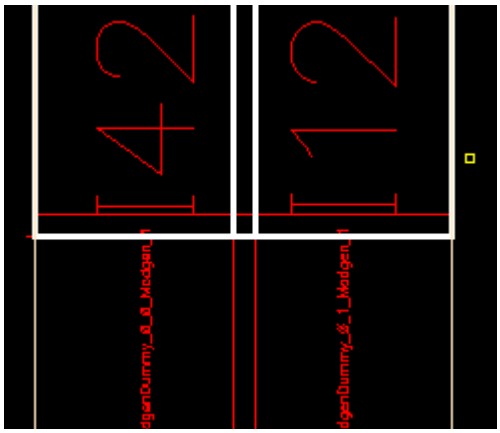
- **Add Dummy Right:** Adds dummy devices to the right of the selected Modgen instances



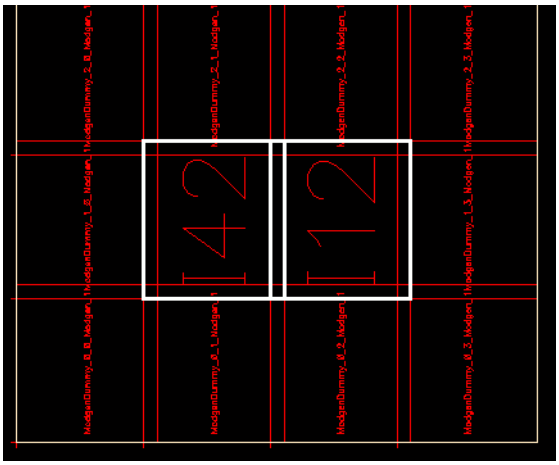
- **Add Dummy Top:** Adds dummy devices above the selected Modgen instances



- **Add Dummy Bottom:** Adds dummy devices below the selected Modgen instances



- **Surround Dummies:** A ring of dummies is added around the selected Modgen instances.



You can use the Modgen Editor to add and delete dummies and dummy device rows and columns, add dummy devices to the array, and backannotate dummy devices.

Related Topics

Adding and Deleting Dummies in the Modgen Editor

Adding Surround Dummies

Adding Dummy Device Rows or Columns

Neighbor Dummy Type for Modgens

Adding and Deleting Dummies in the Modgen Editor

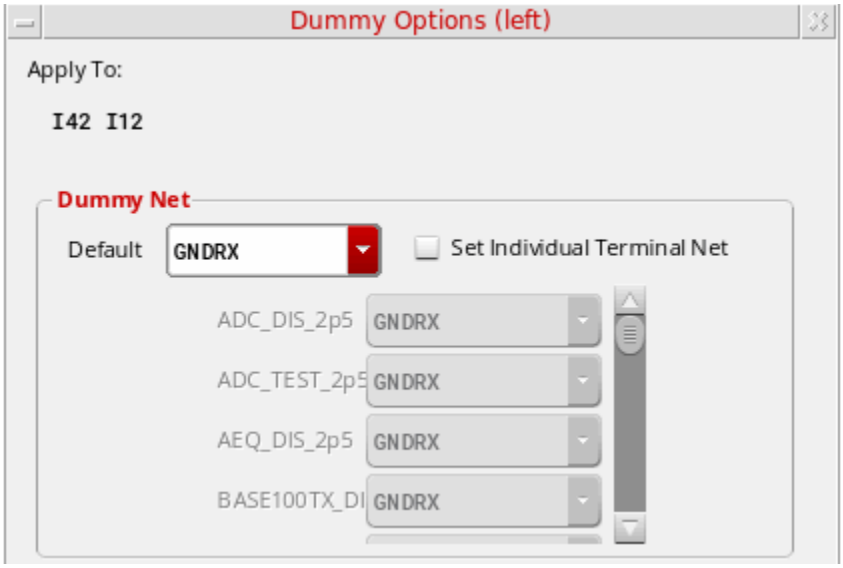
To create dummy devices, with the Modgen Editor open:

1. Select one or more Modgen instances.
2. Click the arrow next to the *Add Dummy* button on the toolbar.

   Alternatively, if the Modgen editor is not open, use the Modgen on-canvas command *Place—Modgen* and select the required location option.
3. Choose a dummy location.

   If no Modgen instance is selected, depending on the specified location option, either a dummy row or a dummy column is added to the array of instances. For example, selecting *Add Dummy Left* adds a column to the left of the array, and selecting *Add Dummy Bottom* adds a row of dummies at the bottom.

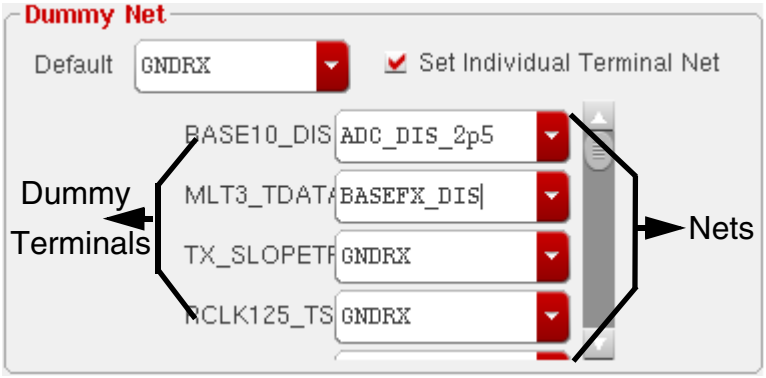
   The Dummy Options form appears.



4. *Apply To* lists the selected Modgen instances. This is not editable.
5. In the *Dummy Net* section, specify the nets to which you want the dummy terminals attached.
   - To attach all selected dummy terminals to the same net, select a net name from the *Default* list.
   - To choose a net that is present in the cellView but not in the current Modgen, type the net name in the *Default* combo box.

6. To connect individual dummy terminals to different nets, select *Set Individual Terminal Net*.

   A list of dummy terminals that are available in the current design is displayed. Use the list box beside each dummy terminal name to specify the net to which it needs to be connected.

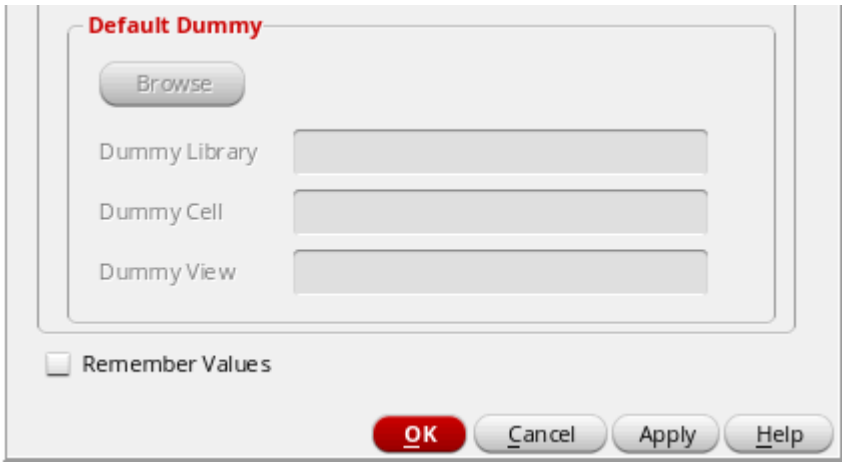


7. In the *Dummy Type* section, select a dummy type from the *Type* list.

8. Select *Specify Parameters* to edit the default dummy parameters.
9. In the *Dummy Parameters* section, the required value in the *Number of Fingers* field.
10. From the *Length* field, choose the required dummy length setting.

    Scale factors can be used to specify the length; for example .1u.
11. From the *Number of Fins* field, choose the required value.

12. If you chose the Dummy *Type* as *default*, click *Browse* under *Default Dummy* to browse for the library, cell, and view you want for the dummy devices.



13. Select *Remember Values* to save the values of all dummy devices.

These values are saved on a per-user basis. So the Module Generator will always load these values until you overwrite them with new saved values.

14. Click *OK* or *Apply*.

(Virtuoso Advanced Node for Layout Only) For FinFET devices to support operations such as dummy creation, their component class must be set to NFIN or PFIN. To define these component types and assign devices to them, use either the Configure Physical Hierarchy command or the library and attributes mapping (LAM) file.

In addition, the *Number of Fins* parameter must be included in the value of the `transistorWidthParamNames` environment variable:

```
envSetVal("layoutXL" "transistorWidthParamNames" 'string "nfin nFin
w wr")
```

Related Topics

Dummy Options Form

transistorWidthParamNames

Modgen Dummies

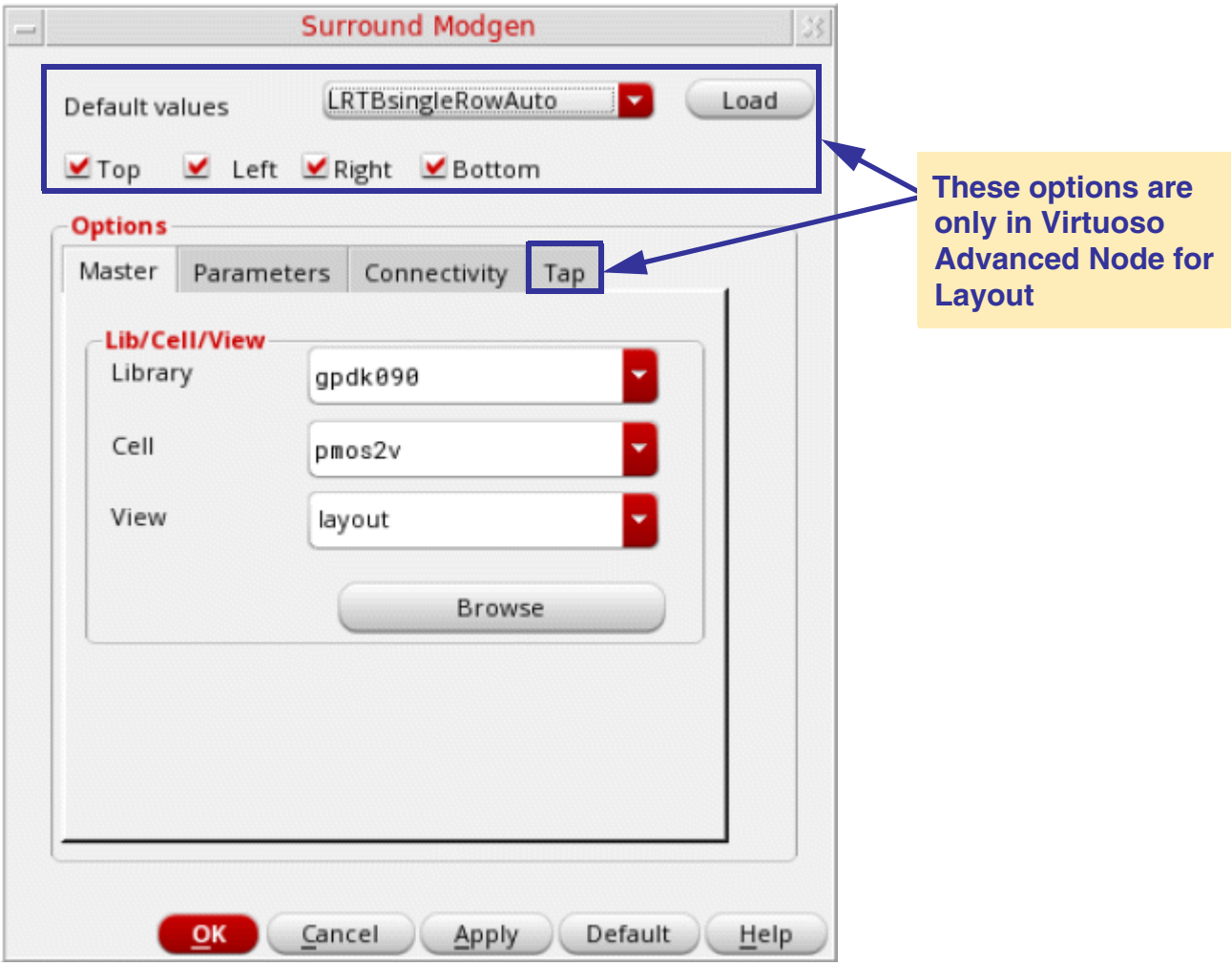
Adding Dummy Device Rows or Columns

Methods to Delete Dummy Devices, Dummy Rows, and Dummy Columns

Adding Surround Dummies

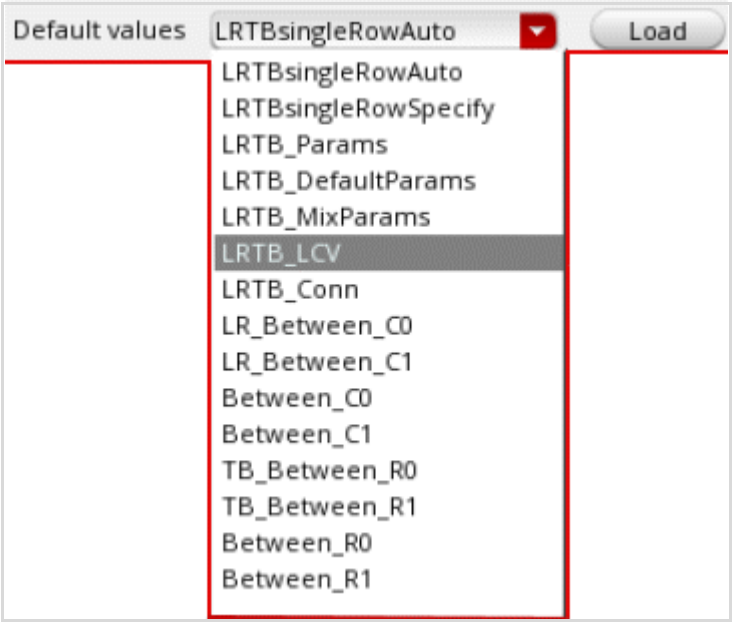
To add surround dummies around the selected Modgen instances, select *Surround Dummies* from the *Add Dummies* drop-down list in the Modgen editor.

The Surround Modgen form is displayed.



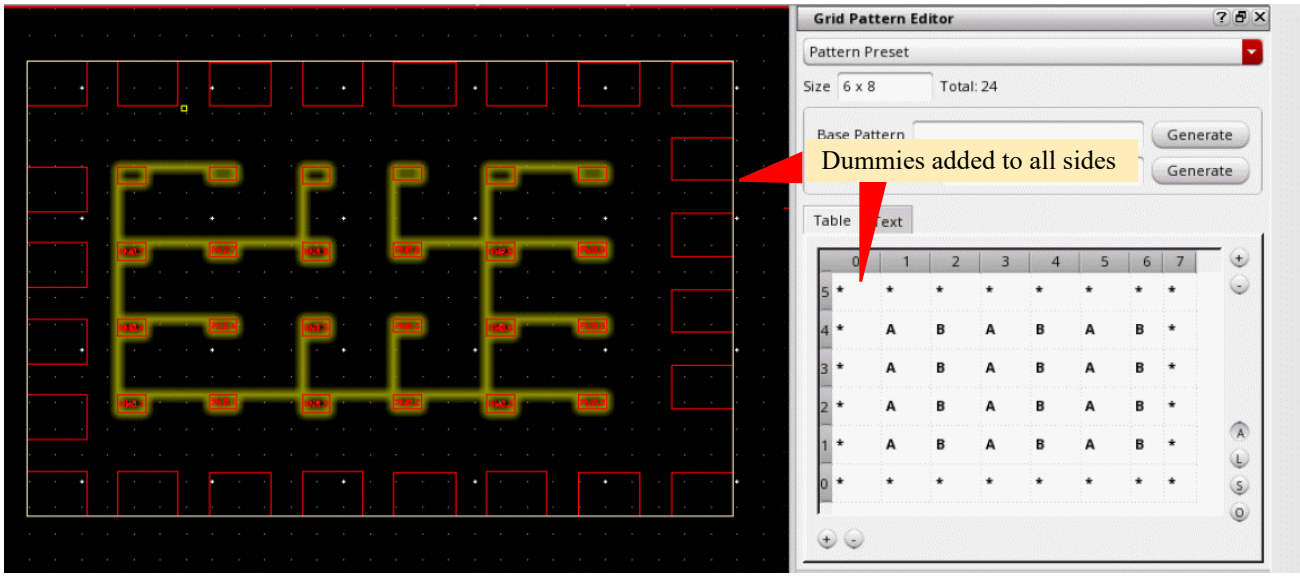
To insert surround dummies:

1. (Virtuoso Advanced Node for Layout Only) Select a pre-registered SKILL callback function from the *Default values* list that contains the values to be loaded.



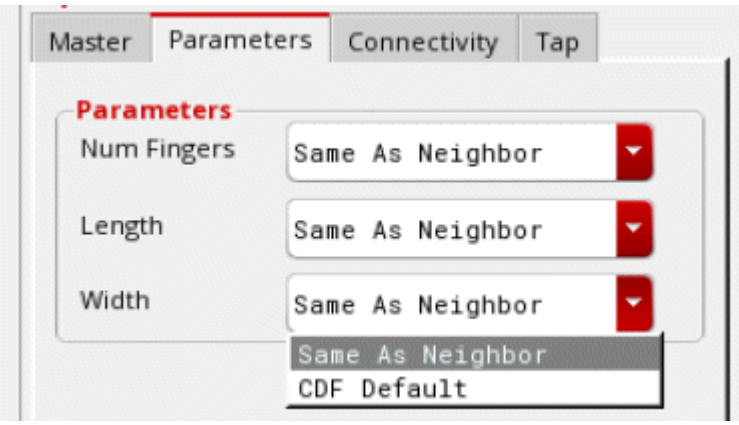
2. Click *Load* to load values from the selected pre-registered SKILL callback function.

3. (Virtuoso Advanced Node for Layout Only) Select one or more sides (*Top*, *Left*, *Right*, and *Bottom*) to add surround dummies. In the following example, dummies are added to all four sides of the Modgen constraint.

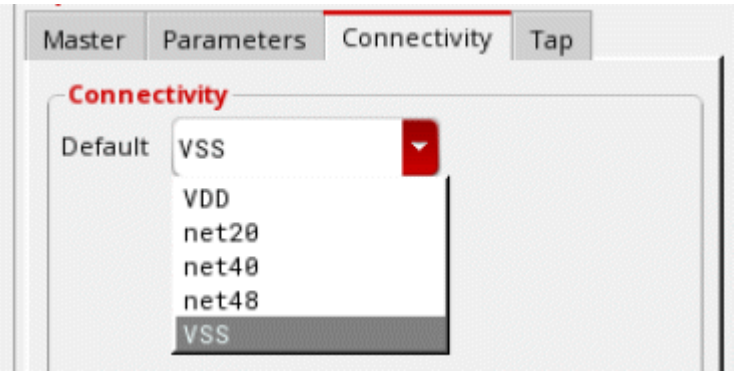


4. On the *Master* tab (default), specify the *Library*, *Cell*, and *View* to be used to create custom dummy devices.

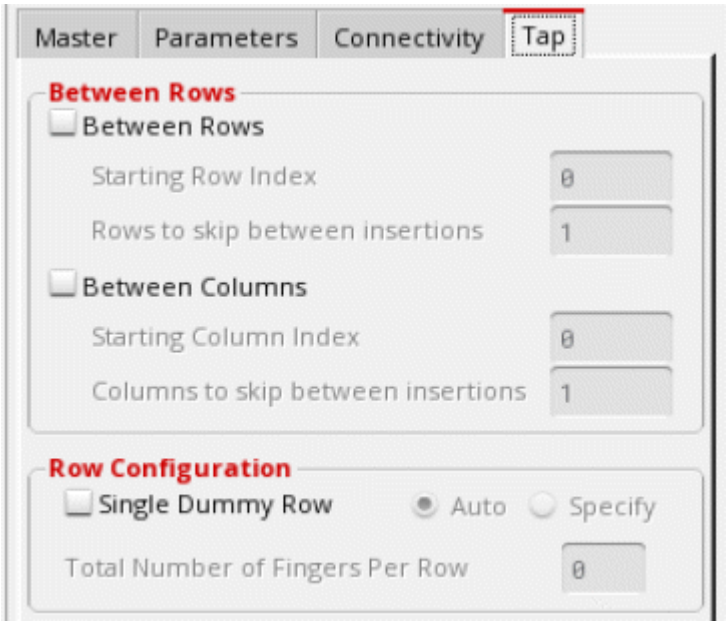
5. On the *Parameters* tab, specify the following dummy parameters.



6. On the *Connectivity* tab, choose the net to which all dummy terminals must connect. If left blank (no net is selected), no terminals are created.



7. (Virtuoso Advanced Node for Layout Only) On the *Tap* tab, select *Between Rows* to insert dummies between rows.



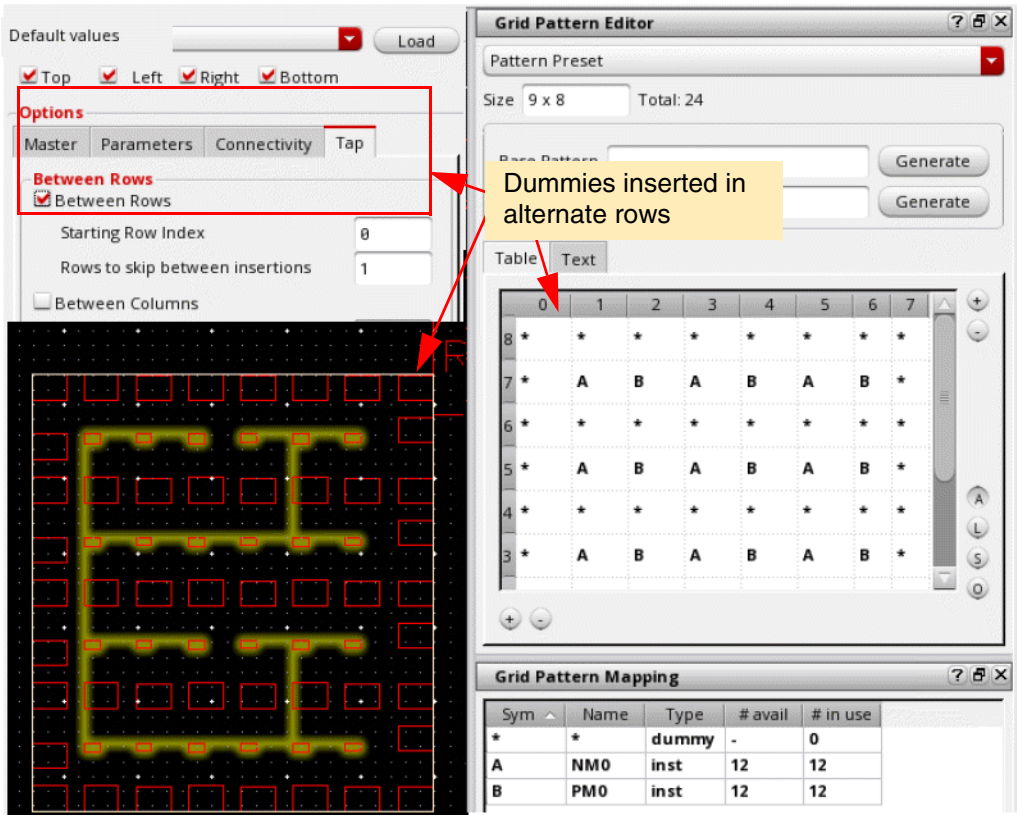
   a. Specify the first reference row for inserting dummy row in *Starting Row Index*.

   b. Specify the gap between dummy rows in *Rows to skip between insertions*.

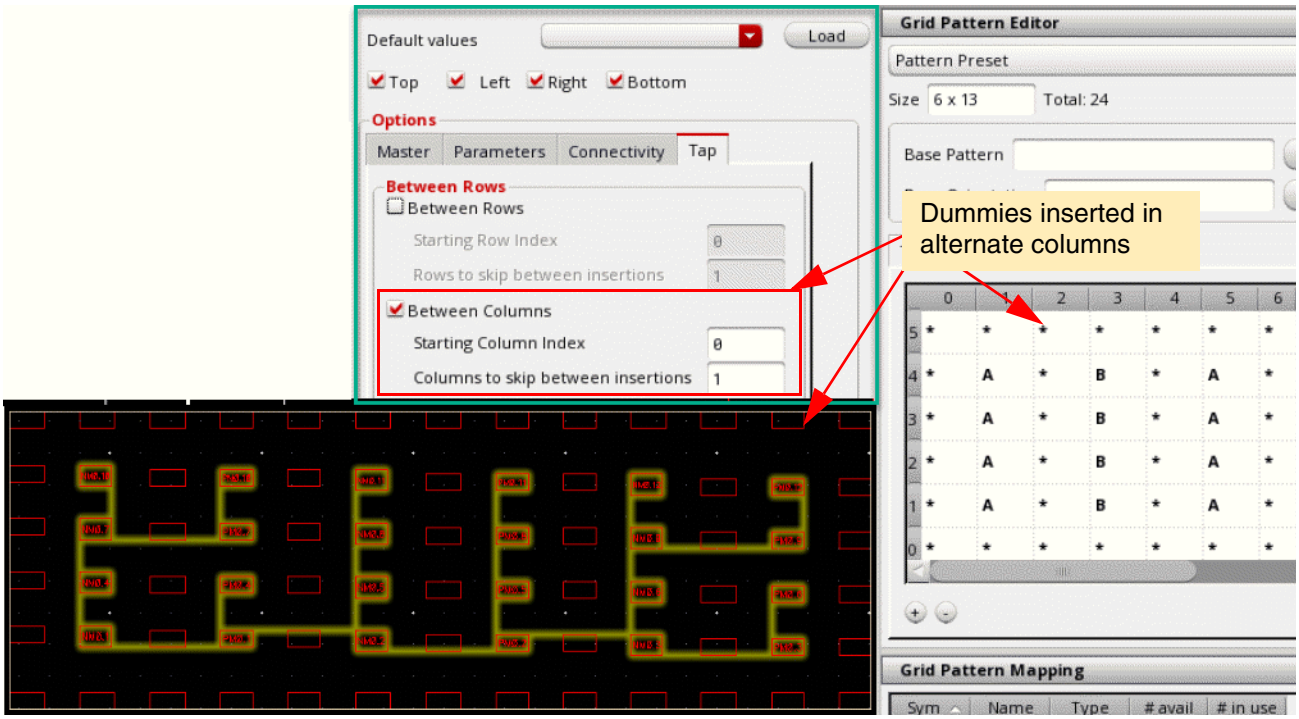
   c. Select *Between Columns* to insert dummies between columns.

   d. Specify the first reference column for inserting dummy row in *Starting Column Index*.

e. Specify the gap between dummy columns in *Columns to skip between insertions*.



8. Select *Single Dummy Row* to insert a single row of dummies instead of individual dummies for each device. You can specify the number of fingers to be inserted in each dummy row.



9. Click *OK*.

Surround dummies are inserted as per your specifications.

Related Topics

Surround Modgen Form

Adding and Deleting Dummies in the Modgen Editor

Adding Dummy Device Rows or Columns

Methods to Delete Dummy Devices, Dummy Rows, and Dummy Columns

Neighbor Dummy Type for Modgens

If the `modgenMakeMinDummies` environment variable is set to `nil`, then the dummy is created using the lib/cell/view of the neighboring device.

If the `modgenMakeMinDummies` environment variable is set to `t`, the following happens when a neighbor dummy is added to a MOSFET:

- If the dummy's location is to the right or left of an instance, a single finger, minimum length device is created to serve as the dummy.
- If the dummy's location is on the top or bottom of an instance, a minimum width device with the same number of fingers as the neighbor is created to serve as the dummy.

When setting the parameters to create minimum length and minimum width dummies, the CDF callbacks will be invoked.

If added to a resistor:

- If the dummy's location is on the right or left, a single segment, minimum length device is created to serve as the dummy.
- If the dummy's location is on the top or bottom, a minimum width device with the same number of segments is created to serve as the dummy.

In this case, the parameters for fingers (`lxFingeringNames`), width (`transistorWidthParamNames`), and s-factor (`sfactorNames`) are used to create the device.

The `lxFingeringNames` environment variable specifies parameter names that define Pcell gate fingering in VLS -XL. In addition, this environment variable is used by the Modgen placer when adding dummy devices to MOSFETs with the `modgenMakeMinDummies` environment variable set to t.

Related Topics

modgenMakeMinDummies

lxFingeringNames

transistorWidthParamNames

sfactorNames

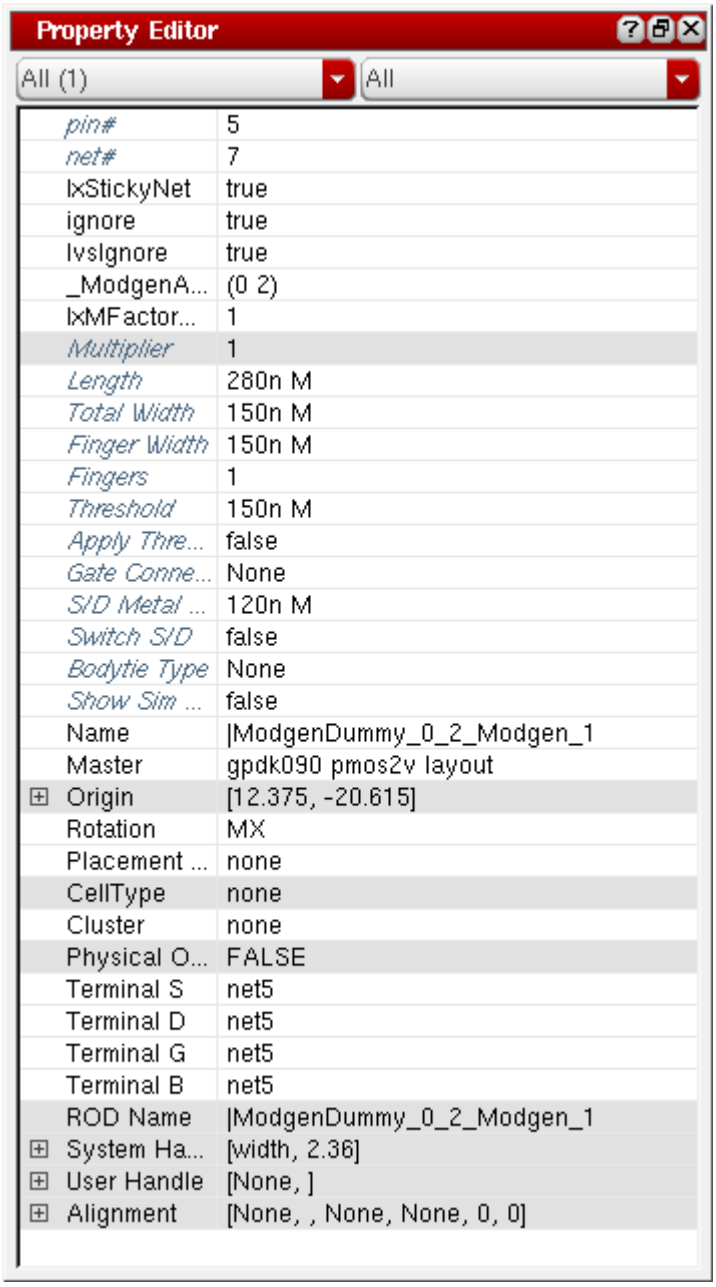
Modgen Dummies

Adding and Deleting Dummies in the Modgen Editor

Adding Surround Dummies

Modgen Dummy Properties

You can view and change parameters of Modgen dummy instances using the Property Editor assistant in the Modgen editor window. To open the Property Editor assistant, right-click the menu area and select the *Property Editor* option from the shortcut menu.

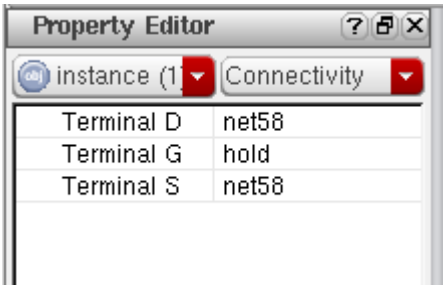


The updated parameter values are stored in the `dummyParams` member parameter in the Modgen constraint.

Use one of the following methods to reset the connectivity of dummies.

- **Using the Property Editor Assistant:** Use this method to individually reset connectivity of terminals within a dummy. Each terminal can be connected to a different net.

  To reset connectivity using this method, open the Property Editor assistant by right-clicking the menu area and selecting the *Property Editor* option from the shortcut menu. Filter the dummies by their connectivity and update the net name.



Related Topics

Modgen Dummies

Neighbor Dummy Type for Modgens

Backannotation of Dummy Devices

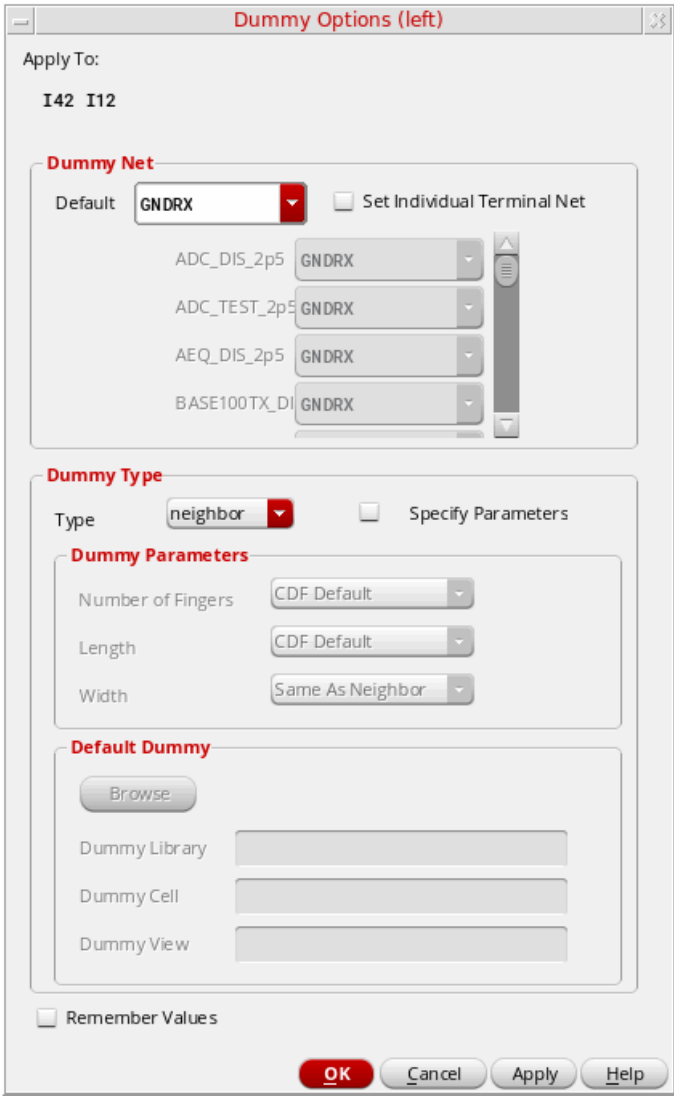
Adding Dummy Device Rows or Columns

Modgens provides the ability to add an entire row or column of dummy devices by selecting only one reference device. For instance, you can select any device in the left column and select *Add Dummy Row/Column* > *Left* to add an entire column of dummies to the left of the Modgen. You can also choose to add dummy devices to the entire array.

To add dummy devices around an array:

1. Select a device in an outer row or column of the Modgen array.
2. Click the arrow next to the *Add Dummy* button on the toolbar.
3. Select a dummy location.

The Dummy Options form is displayed.



You can either set up the dummy options or accept the default options.

4. Select *Specify Parameters* in the *Dummy Parameters* section to specify the number of fingers, length, and width for all dummy devices.

   The *Specify Parameters* check box is available only when the Dummy Options form is invoked by selecting *Add Dummy Row/Column* button.

   If *Specify Parameters* is not selected, then default values for these fields is determined by the modgenMakeMinDummies environment variable.

5. Modify dummy device options as required.

6. Click *OK* or *Apply*.

The Module Generator places the dummy rows or columns as specified.

You can also choose to add dummy devices to an entire array. To do this:

1. Select any device in an outer row or column of the array.
2. On the *Modgen Placement* toolbar, click the arrow next to the *Add Dummy Row/ Column* button and choose the location.

Related Topics

Dummy Options Form

Neighbor Dummy Type for Modgens

Modgen Dummies

Adding and Deleting Dummies in the Modgen Editor

Adding Surround Dummies

Methods to Delete Dummy Devices, Dummy Rows, and Dummy Columns

Backannotation of Dummy Devices

Modgen dummy devices in the layout can be backannotated to their corresponding schematic to keep the two views synchronized. If the dummy layout instance is already bound to a symbol in the schematic, no back annotation is performed for that instance.

You can backannotate dummies even when the layout view is open in the read-only mode.

At any point, you can run *Check Against Source* to check for any instance mismatches with the schematic.

Related Topics

BackAnnotating Dummy Instances (In Layout XL)

Modgen Dummies

Adding and Deleting Dummies in the Modgen Editor

Adding Surround Dummies

Modgen Dummy Properties

Adding Dummy Device Rows or Columns

Neighbor Dummy Type for Modgens

Methods to Delete Dummy Devices, Dummy Rows, and Dummy Columns

Methods to Delete Dummy Devices, Dummy Rows, and Dummy Columns

To delete a dummy devices, do one of the following:

- Select the dummy device and click the *Delete* button on the toolbar to delete the dummy device.
- Select the dummy device and press the *Delete* key delete the dummy device.
- Select *Place—Modgen—Remove Modgen Surround Dummies* to remove surround dummies in the layout canvas.

  Note: All empty rows and columns in the Modgen are deleted.
- To delete dummy rows or columns, in the *Modgen Placement* toolbar, click the arrow next to the *Add Dummy Row/Column* button and click *Delete All Dummy Rows/ Columns*. Alternatively, use the `mgDeleteAllDummyRowColumnCB` command.
- To delete all dummies in the design, in the *Modgen Placement* toolbar, click the arrow next to the *Add Dummy Row/Column* button and click *Delete All Dummies*.

  Note: Rows and columns with empty cells are deleted.

Related Topics

Modgen Dummies

Adding and Deleting Dummies in the Modgen Editor

Adding Surround Dummies

Modgen Dummy Properties

Adding Dummy Device Rows or Columns

Neighbor Dummy Type for Modgens

Backannotation of Dummy Devices

Body Contacts in Modgens

Adding body contacts is a two-step process in Modgens.

1. Define the body contact properties in the Body Contact Options form.

   You can specify the type, net, and separation distance for body contacts. The body contact created by the Modgen tool matches the height of the member instance, and therefore you need to specify which reference layer the body contact matches.

   If you do not specify a reference layer, the Modgen tool will match the body contact to the height of the bounding box of the member instance. The Modgen tool also uses the reference layer to calculate the separation distance.

2. Add body contacts to the Modgen array in the layout.

   You can either add body contacts in the layout on a row or column basis or add all body contacts using the *Modgen Placement* toolbar.

Related Topics

Body Contact Options Form

Defining Modgen Body Contact Properties

Adding and Removing Modgen Body Contacts

Grid Placement In Modgens

Placement of Modgen Body Contacts on Grids

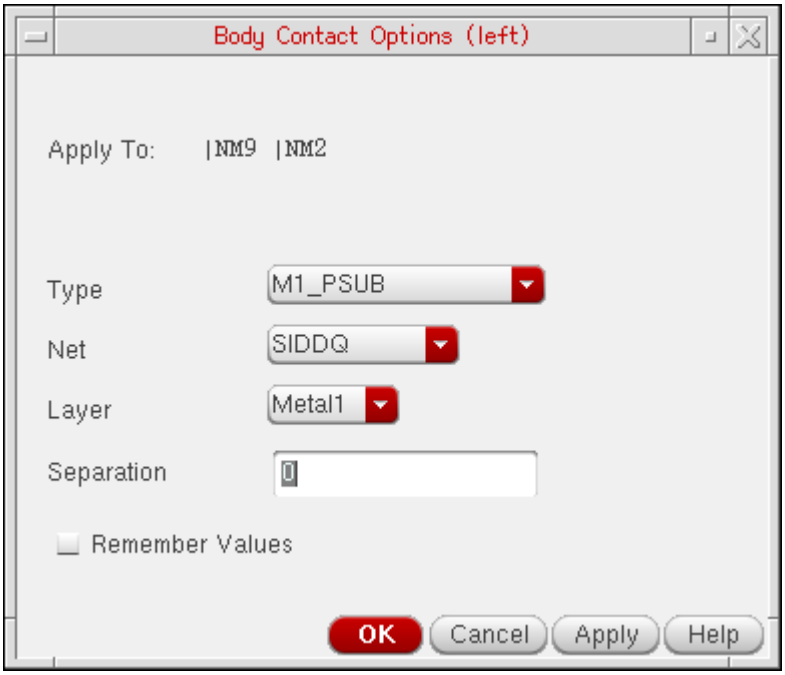
Defining Modgen Body Contact Properties

Before adding body contacts to Modgens, you must define the body contact properties.

To define body contact properties:

1. In the *Modgen Placement* toolbar, click the *Add Body Contact*  button.

The Body Contact Options form appears.



2. From *Type*, choose the type of body contact you want to add.

   This field is populated from the technology file.

3. Select the net to which you want to attach the body contacts.
4. From the *Layer* list, select the reference layer for the body contacts.
5. In *Separation*, enter the distance in microns that you want between body contacts and devices. This value is added to minimum DRC to space the body contact.
6. Select *Remember Values* to save values for all dummy devices.
7. Click *OK* or *Apply*.

Body contact properties are set as per your specifications.

You can edit body contact parameters using the Property Editor assistant in the Modgen editor window.

To open the Property Editor assistant, right-click the menu area and select the Property Editor option from the shortcut menu.

Member (2)	Member Parameters
Body Contacts	left
Body Contact Type	ruleVia
Body Contact Separation	0
Body Contact Reference Layer	Nimp
Body Contact Net	1
Dummy Net	0
Dummy Parameter Source	neighbor
Horizontal Custom Spacing	0
Horizontal Custom Spacing Re...	
Vertical Custom Spacing	0
Vertical Custom Spacing Refer...	
Row	0
Column	1
Abutment	0
Orientation	R0
Vertical Alignment	top
Horizontal Alignment	left

The modified parameter values are applied to the selected body contact.

Related Topics

Body Contact Options Form

Body Contacts in Modgens

Adding and Removing Modgen Body Contacts

Grid Placement In Modgens

Placement of Modgen Body Contacts on Grids

Adding and Removing Modgen Body Contacts

Before adding body contacts to Modgens, ensure that their properties are defined.

There are two methods to add body contacts. You can add them in the layout on a row or column basis, or you can add all body contacts using the *Modgen Placement* toolbar.

Adding Body Contact Rows and Columns

You can add body contacts on a row or column basis in the layout.

1. Select a device in a row or column of the array.

   The body contacts you can add are dependent on which device you select.

2. Do one of the following:

   - Right-click and choose *Add Body Contact – <Location>*.
   - In the *Modgen Placement* toolbar, click the arrow next to the *Add Body Contact* icon and choose a location option.

Adding Body Contacts to a Selected Device

You can also add body contacts to the selected devices.

1. Select one or more devices.
2. Do one of the following:

   - Right-click and choose *Add Body Contact – `<Location>`*.
   - In the *Modgen Placement* toolbar, click the arrow next to the *Add Body Contact* button and choose a location option.

Removing Body Contacts

To remove body contacts from a Modgen, do one of the following:

- Select the body contact and click the *Delete* button on the toolbar.
- Select the body contact and press the *Delete* key.

Related Topics

Body Contact Options Form

Body Contacts in Modgens

Defining Modgen Body Contact Properties

Placement of Modgen Body Contacts on Grids

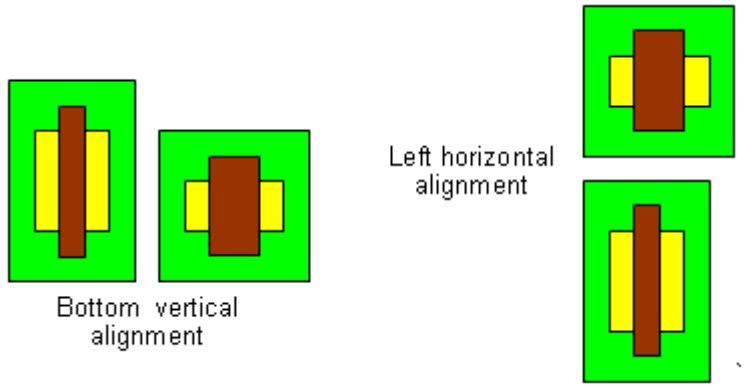
Grid Placement In Modgens

Grid Placer is a back end placement engine for Modgen to generate DRC-correct array placement. Grid Placer places a set of instances according to their grid indexes into an array-like placement compacted to the minimum DRC and constraint-correct spacing. It also lets you specify the array size in terms of the number of rows and columns.

Grid Placer provides fast and flexible placement and supports the constraints between member instances, well merging, and complex guard ring creation.

The Grid Placer algorithm scales better for large Modgens than the current cell by cell placement implementation.

In Grid Placer, the Modgen's horizontal and vertical alignment member parameters are relative to a member's left or bottom neighbor. Therefore, setting a member's alignment to the left aligns its left edge with the left edge of the member below it. If the member is in the bottom row, vertical alignment had no effect, as shown in the following image:



Typically, Modgens are snapped to the manufacturing grid resolution as specified in the technology file when *Generate From Source* or *Generate Selected From Source* is used. You can also snap Modgen origins to the snap spacing specified by the modgenUseSnapSpacing environment variable in the `.cdsinit` file.

Note: Modgen origin is the origin of the figGroup, which is the lower-left corner of the `figGroup` bbox. Snapping is with respect to the Modgen origin.

Placement of Modgen Body Contacts on Grids

Grid Placer supports placement of Modgen body contacts on grids. Grid Placer does not create the body contact geometry, but only places the body contacts.

You can define the relative position, such as left, right, top, or bottom and spacing between the body contacts and member instance of the grid. It also incorporates the DRC between the body contact and neighbor grid geometry as a hard constraint.

The Grid Placer also supports automatic minimum DRC spacing based placement for a body contact. This minimum DRC spacing would be obtained from the process rules on the database object.

If there is a conflict between the DRC rules of neighbor member instances, the Grid Placer uses the maximum value as the DRC rule between the two instances.

Calculating DRC Rules

You can use the Process Rule Editor to query minimum spacing rules for a single or different layer to calculate the DRC rules between the neighboring grid objects.

For same layer, use the `minSpacing` constraint definition, while for different layers use `minClearance`.

While setting DRC rules between two devices, you must consider the maximum of the `minSpacing` or `minClearance` values obtained from the Process Rule Editor on each of the two devices.

Corner Case Conditions for Modgen Grid Placer

User-defined Modgen parameters can be categorized into parameters that define the overall Module and parameters that defines each grid.

The module-level input parameters include the number of rows or columns, guard rings, row routing spacing estimation, and constraints between grid objects.

The grid-level input parameters define each grid object and its neighbors. This includes, row/column index, alignment, custom spacing, abutment, body contact, well layer, and DRC rules.

A combination of both these input parameters can create conflicts.

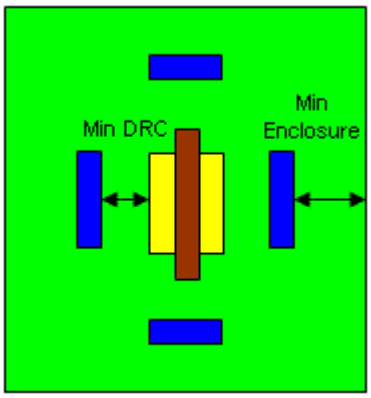
Note: In Grid Placer, if a member which is abutted to its neighbor also has custom spacing, then abutment will take precedence and the custom spacing will be ignored.

Body Contact v/s Well Merge

Let’s consider a scenario of a possible conflict between body contact and well merge.

Scenario 1:

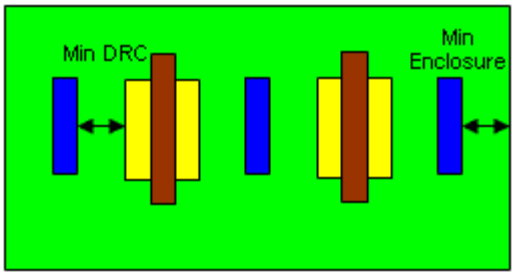
If the well-merge option is on, and the body contact has either the same well layer as the instance or no well layer in its geometry. Then, you enclose the body contact geometry with the well layer of the instance as shown in the image below.



Min-enclosure process rules need to be maintained between the well layer and the body contact geometry. However, if the body contact has a different well layer than the instance, you do not enclose the body contact with the well geometry of the instance.

Scenario 2:

If the well-merge option is on, and two neighboring instances in the grid sharing a common body contact also share the same well layer, then the well layers of both instances are merged maintaining minimum enclosure and minimum DRC rules as shown in the image below.



However, if the shared body contact happens to have a different well layer as the neighboring instances, well-merging would be disabled in this special case.

Note: If instance members in a Modgen have the same well layer but different bulk connectivity, then well merging is not allowed for these instances. Well merging is only applicable for instances that have the same well layer and same bulk connectivity.

Related Topics

Creating a Modgen

Specifying Modgen Parameters

Modgen Guard Rings

Guard rings are used to enclose one or more objects such as devices or device chains. You can use the Modgen tool to create multipart path (MPP) guard rings, fluid guard rings, and identical guard rings. Before creating guard rings, ensure that they are defined in the technology file.

Tip

You can define guard rings directly in the layout canvas without opening Modgen Editor. Select *Place—Modgen—Add Modgen Guard Ring* and choose the required guard ring type.

Related Topics

Creating a Multipath Part Guard Ring

Fluid Guard Rings Around Modgens

Creating Identical Guard Rings around Modgens (Virtuoso Advanced Node for Layout Only)

Creating a Multipath Part Guard Ring

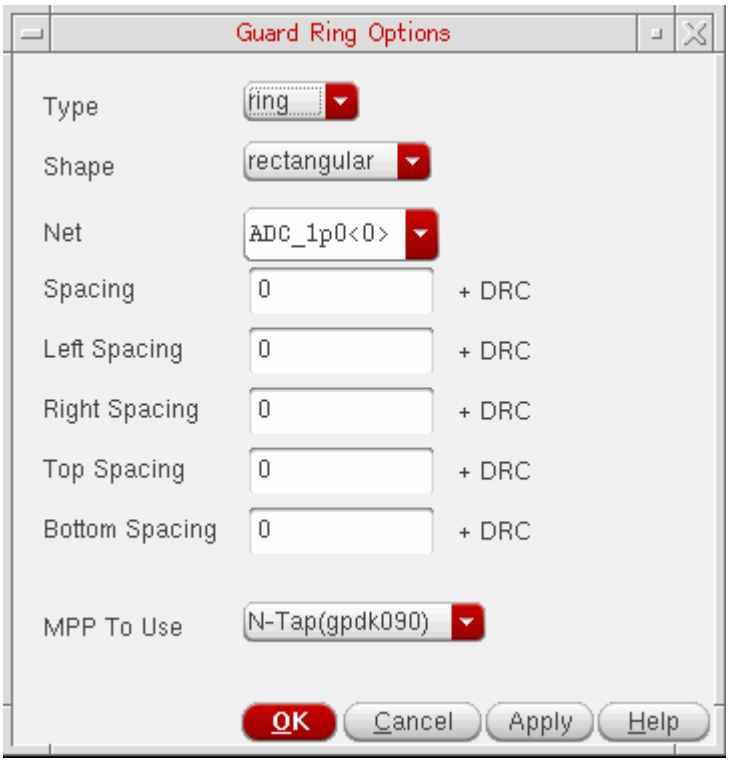
You can use the Modgen tool to generate guard rings based on multipart paths (MPPs) defined in the technology file.

You can specify one common spacing value to define the separation between all four sides of the guard ring and the devices, or a different value for each side. This spacing value is in addition to the minimum DRC distance. So, when creating a guard ring, the Modgen tool first finds the minimum DRC location at which to place the guard ring, examining each layer, then adds the specific spacing values to that distance in order to calculate the final location.

To create a MPP guard ring:

1. On the *Modgen Placement* toolbar, click the *Guard Ring* icon
2. Select *Create MPP Guard Ring*.

The Guard Ring Options form is displayed.



3. From *Type*, select a guard ring type.
4. From *Shape*, select the shape of guard ring.
5. From *Net*, select the net you want the guard ring attached to.
6. Specify the *Spacing* between the sides of the guard ring and the devices by doing one of the following:

   To specify the same *Spacing* value for all sides of the guard ring:

   - Enter the distance in microns in the *Spacing* field. By default, the value is `0`.

   To specify different spacing values for each side of the guard ring:

   a. Enter a value for the left side in the *Left Spacing* field.
   b. Enter a value for the right side in the *Right Spacing* field.
   c. Enter a value for the top side in the *Top Spacing* field.
   d. Enter a value for the bottom in the *Bottom Spacing* field.

   The spacing value is in addition to the DRC spacing rule.

7. From the *MPP To Use* cyclic field, choose an MPP for the guard ring.
8. Click *OK* or *Apply*.

The Module Generator creates a guard ring around the array using the parameters that you specified.

Note: Modgen pane guard ring stripes honor existing abutment between member instances. Therefore, guard ring stripes are inserted only between unabutted instances. If you unabut existing instances, the pane is regenerated, and the stripes are inserted between all instances.

Important

Cluster spacing to guard ring is the sum of spacing and DRC between instance Bbox and edge of guard ring edge. Modgen spacing to guard ring is the sum of spacing and DRC between outermost layer instance data to guard ring edge.

Related Topics

Modgen Guard Rings

Creating a Multipath Part Guard Ring

Fluid Guard Rings Around Modgens

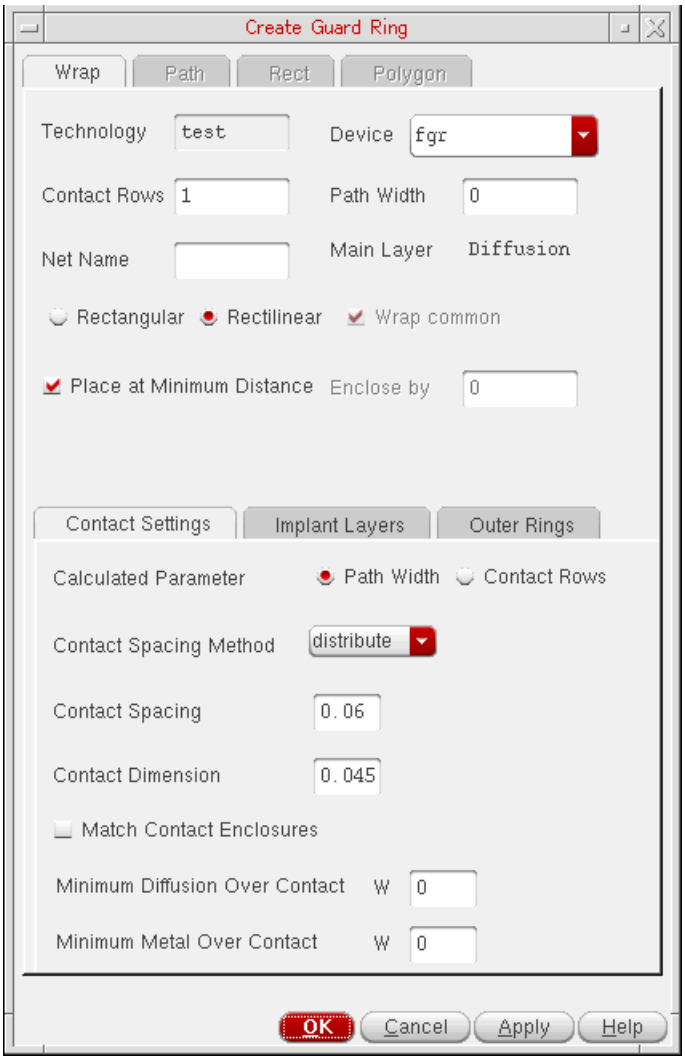
Creating Identical Guard Rings around Modgens (Virtuoso Advanced Node for Layout Only)

Fluid Guard Rings Around Modgens

In addition to multipart path (MPP) guard rings, you can use the Modgen editor to create fluid guard rings (FGRs) around objects. FGRs are a type of fluid Pcells that can be used to enclose one or more objects such as devices or device chains. Before creating FGRs, ensure that the FGR is installed as a device class in the technology file.

To create an FGR, click the *Guard Ring* icon on the *Modgen Placement* toolbar and choose *Create Fluid Guard Ring*.

The Create Guard Ring form is displayed.



FGRs can be created by drawing a path, a rectangle or polygon, or by using the wrap mode.

Each of these modes represent a tab on the Create Guard Ring form. However, the Modgen Editor only supports creation of FGRs in *Wrap* mode. In this mode, an FGR is created around the objects you select. The FGR parameters are stored in the Modgen constraint.

When you create an FGR for a FinFET device, the FGR instance automatically snaps to the underlying fin grids if the *Snap Pattern Snapping* check box is selected in the Layout Editor Options form.

Hand Editing Fluid Guard Rings

When a Modgen is edited, the FGRs in the Modgen are regenerated automatically. As a result, all the manual edits made to the FGRs are lost. To avoid this, switch to hand edit mode for the FGRs. In this mode, the FGRs are not regenerated each time the Modgen is updated. Therefore, all customizations remain.

To switch on hand edit mode, either select *Enable Fluid Guard Ring Hand Edit Mode* from the *Guard Ring* menu, or use the `mgFGREnterHandEdit` SKILL function.

To switch off hand edit mode, either select *Disable Fluid Guard Ring Hand Edit Mode* from the *Guard Ring* menu, or use the `mgFGRExitHandEdit` SKILL function.

Note: FGRs can be edited manually both in and out of hand edit mode.

Related Topics

Installing Fluid Guard Rings

Wrap Mode

mgFGREnterHandEdit

mgFGRExitHandEdit

Modgen Guard Rings

Layout Editor Options

FinFET Support in Layout L

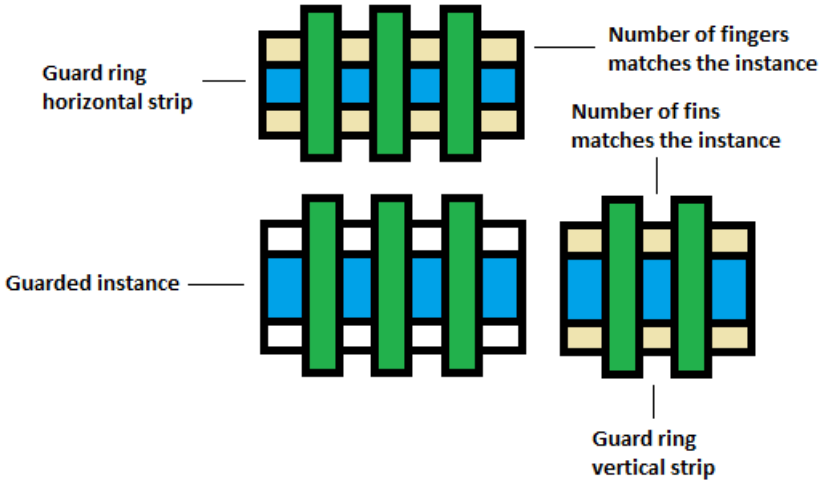
Creating Identical Guard Rings around Modgens (Virtuoso Advanced Node for Layout Only)

In addition to MPP and fluid guard rings, you can create identical guard rings in Modgens. Identical guard rings are composed of unit cells that match the guarded instances in terms of their number of fingers, number of fins, finger alignment, and other parameters.

Important

The *Create Identical Guard Ring* command is available only if the PDK is configured to support identical guard rings. For more information on this capability, contact your Cadence Customer Support representative.

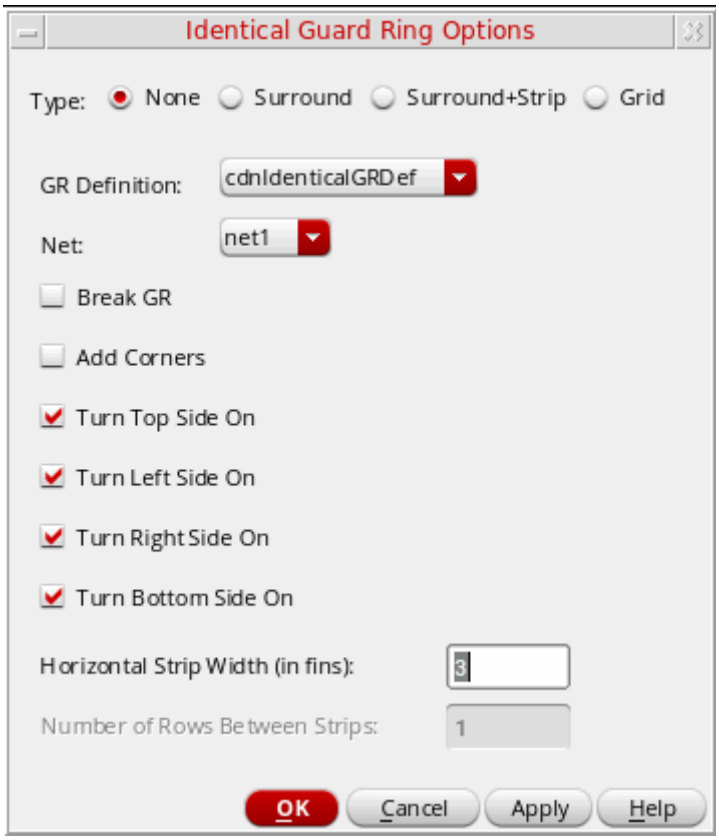
The following diagram depicts identical guard rings:



To create identical guard rings:

1. Open a Modgen in the Modgen editor.
2. Click the Guard Ring icon on the *Modgen Placement* toolbar.
3. Select *Create Identical Guard Ring*.

The Identical Guard Ring Option form appears.



4. Select a guard ring *Type* to define the way in which the identical guard ring surrounds the adjoining Modgen instances.
5. Select a *GR Definition* from the list. Values are listed from the current PDK.
6. Select a *Net* to which the guard ring must be connected.
7. Select *Break GR* to provide spaces for connecting guarded instances via the `Metal1` layer routes.
8. Select *Add Corners* to create guard rings at all available corners around the selected instances.
9. Select the required side switches—*Turn Top Side On*, *Turn Left Side On*, *Turn Right Side On*, and *Turn Bottom Side On*. These side switches turn on or off the guard ring for the specified sides.
10. In *Horizontal Strip Width (in fins)*, specify the vertical width of horizontal guard ring strips.

11. In *Number of Rows Between Strips,* specify the number of instance rows that are required between strips of guard rings, counting from the bottom row.
12. This option is available only when *Type* is set to either *Surround+Strip* or *Grid*.
13. Click *OK*.

An identical guard ring is created as per your specifications.

Related Topics

Identical Guard Ring Options Form

Modgen Guard Rings

Modifying a Modgen Guard Ring

To modify an existing guard ring:

1. Do one of the following:
   - In the toolbar, click the *Guard Ring* icon.
   - Right-click outside the array and choose *Edit Guard Ring*.

   The Guard Ring Options form appears.
2. Modify the guard ring options as desired.
3. Click *OK* to apply changes.

Related Topics

Modgen Guard Rings

Creating a Multipath Part Guard Ring

Fluid Guard Rings Around Modgens

Creating Identical Guard Rings around Modgens (Virtuoso Advanced Node for Layout Only)

Removing a Modgen Guard Ring

Removing a Modgen Guard Ring

Use one of the following methods to remove an existing guard ring:

1. On the *Modgen Placement* toolbar, click the *Guard Ring* icon.
2. Choose *Remove Guard Ring*.

Alternatively:

1. Do one of the following:
   - In the toolbar, click the *Guard Ring* icon.
   - Right-click outside the array and choose *Edit Guard Ring*.

   The Guard Ring Options form appears.
2. From the *Type* cyclic field, choose *none* as the type.
3. Click *OK* when finished.

Video

For more information, see the Removing a Guard Ring from a Modgen video.

Related Topics

Modgen Guard Rings

Creating a Multipath Part Guard Ring

Fluid Guard Rings Around Modgens

Creating Identical Guard Rings around Modgens (Virtuoso Advanced Node for Layout Only)

Modgen Device Abutment

Modgen supports abutment of MOSFET member instances. When you abut a device, Modgen remains aware of its connectivity and might flip the device to correctly abut it.

You can also abut dummy devices to other devices and dummies. Internally, the dummy device net might be changed from the default specified in the Dummy Options form, but if the dummy device is removed from abutment, the net reverts to the default. When two instances are abutted, body contacts between them are deleted.

Modgen abutment uses the Layout XL chaining engine. Therefore, the abutment results are subject to the settings of the Layout XL chaining environment variables. Notable examples are `chainMirror` and `chainMirrorEquivOrients`. These environment variables are enabled by default, and may therefore result in unintended changes to the orientation of instances.

Prerequisites for Modgen Device Abutment

The devices to be abutted must be registered as component types. It addition to abutment, this is a key requirement for operations such as chaining and folding devices, identifying pseudo-parallel nets, and identifying devices and structures using the Circuit Prospector assistant (this is a schematic XL feature) in Schematic view.

If the devices are not registered in the PDK, use one of the following methods to register devices as component types:

- Use the `ciRegisterDevice` function.
- Use the cph.lam file to assign the devices to component types NMOS, PMOS, NFIN, or PFIN.
- Use the *Component Type* mode of the Configure Physical Hierarchy window to assign the devices to component types NMOS, PMOS, NFIN, or PFIN.

  Note: The CPH settings must be done only once on a design before invoking the Modgen Editor.

Related Topics

Library and Attributes Mapping File Syntax

Defining a Design-Level Component Type

Abutting Modgen Devices

To abut Modgen devices:

1. Select two or more devices that you want to abut.

   Abutment in Modgen is a row-based operation. When you select multiple instances for abutment, the instances in each row are abutted separately.

2. Do one of the following:

   - In the *Modgen Placement* toolbar, click the *Abut* icon.
   - Right-click the array and choose *Abut*.
   - Select *Place—Modgen—Abut Modgen Instances*.

The Modgen devices are abutted.

To abut all devices, do one of the following:

- In the *Modgen Placement* toolbar, click the *Abut All* icon.
- Right-click the array and choose *Abut All*.
- Set the `modgenPatternFormAbutAll` environment variable to `t`.

By default, during abutment, the selected instances are mirrored, if needed. However, you can control whether to use mirroring instances or permutation of pins during the abutment process. To switch off mirroring and turn on pin permutation, set the `chainPermutePins` environment variable to `t`.

Note: For a set of selected instances, abutment is performed only between the instances that can be abutted. For example, if there are body contacts between two neighboring instances that are selected, then those instances cannot abut.

Synchronized Abutment of Instances

To run the same abutment for all rows in the Modgen, before running the abutment command, turn on synchronized abutment by clicking the *Synch Abut* button on the *Modgen Placement* toolbar.

Instances in all rows of the Modgen are abutted in a synchronized operation along the column.

Turn off Synch Abut (default) to abut each row individually, without considering the other rows.

Abutment of Dummy Shapes in Pcells (Virtuoso Advanced Node for Layout Only)

The Modgen editor supports the new Pcell abutment capability supported by Virtuoso Layout Suite XL. At advanced nodes, the Modgen editor supports the abutment of shapes in Pcell submasters that are not attached to pins. In addition to the usual abutment properties, these shapes also require an abutment name to be set in the Pcell SKILL code.

Related Topics

chainPermutePins

modgenPatternFormAbutAll

Modgen Device Abutment

Removing All Abutment

Advanced Node Abutment of Dummy Shapes in Pcells (Virtuoso Advanced Node for Layout Only).

Multiple Abutments in Modgen Constraints

Each Modgen constraint can be associated with multiple abutment scenarios. The Abutment parameter can take multiple values (0, 1, 2, 3, and so on), where each value is assigned to a different abutment scenario. The default value 0 means no abutment.

Use the following SKILL functions to register, retrieve, and unregister abutment scenarios:

- `mgRegUserProc`: Registers abutment scenarios in the Modgen code to enable callbacks.
- `mgGetRegUserProc`: Displays information about registered abutment scenarios.
- `mgUnRegUserProc`: Unregisters the specified abutment scenarios.

Related Topics

mgRegUserProc

mgGetRegUserProc

mgUnRegUserProc

Modgen Device Abutment

Abutting Modgen Devices

Removing Abutment from Devices

Removing Abutment from Devices

If you do not want devices in a module generator array abutted:

1. Select the devices you do not want abutted.
2. Do one of the following:
   - In the *Modgen Placement* toolbar, click the *Unabut* icon.
   - Right-click the array and choose *Unabut*.
   - Select *Place—Modgen—UnAbut Modgen Instances*.

Removing All Abutment

To remove abutment from all devices.

➤ Right-click the array and choose *Unabut All*.

Related Topics

Modgen Device Abutment

Abutting Modgen Devices

Multiple Abutments in Modgen Constraints

Specifying Modgen Device Alignment and Spacing

The Set Member Alignment and Spacing form lets you specify alignment and spacing settings for Modgen members.

A Modgen can contain instances with different Pcell masters. The instances can have different dimensions. To arrange instances with different heights within a module, you can use the Set Member Alignment and Spacing form to specify a vertical bounding box alignment (top, center, or bottom).

When devices are aligned at the top, the top edges of the instances are aligned at the same Y-coordinate, while bottom edges are aligned at the same Y-coordinate for bottom alignment. For center alignment, the center of shorter instances are aligned with the center of the largest instance.

The default vertical alignment is bottom. You can modify this alignment using either the align commands in the shortcut menu or the Set Member Alignment and Spacing form.

When you launch the Set Member Alignment and Spacing form, the alignment, reference layer, and horizontal and vertical spacing values of the selected Modgen devices are loaded in the form. Therefore, the values reflect the current design selection.

You can click *Load Values* in the form to load the default alignment, reference layer, horizontal and vertical spacing values from the corresponding environment variables.

Aligning Modgen Devices Using the Shortcut Menu

To modify device alignment using the shortcut menu:

1. Open a Modgen in the Modgen Editor.
2. Select the devices you want to align.
3. Right-click and choose one of the following options:
    - *Align – Top*
    - *Align – Bottom*
    - *Align – Center (V)*
    - *Align – Center (H)*
    - *Align – Left*
    - *Align – Right*

- In the toolbar, click the arrow next to the *Alignment* icon and choose one of the following options:
  - *Align Left*
  - *Align Center (H)*
  - *Align Right*
  - *Align Top*
  - *Align Center (V)*
  - *Align Bottom*

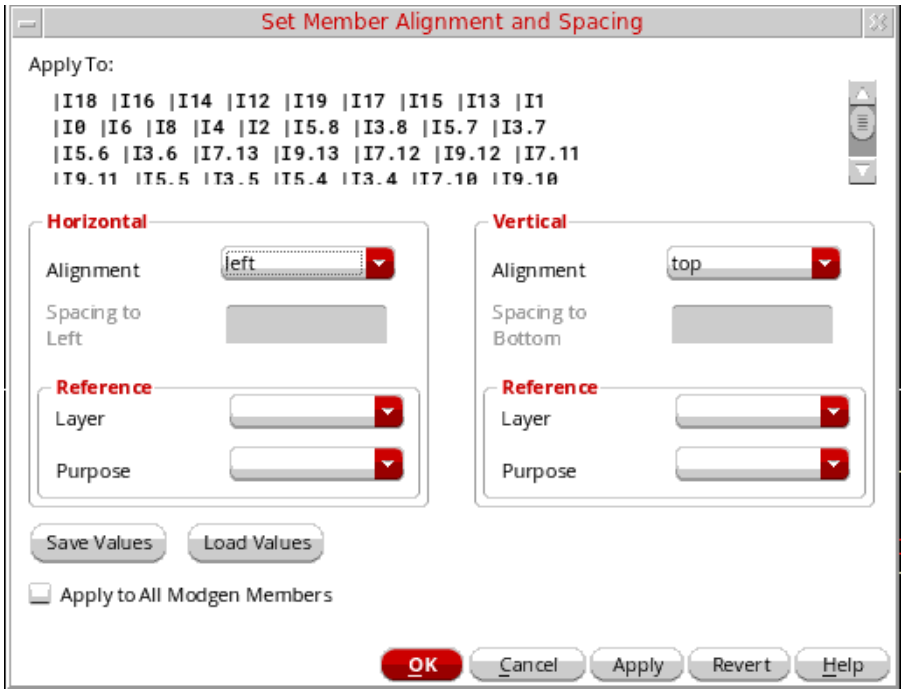
Aligning Modgen Devices Using the Alignment and Spacing Form

To align devices using the Set Member Alignment and Spacing form:

1. Open the Modgen in the Modgen Editor.
2. (Optional) Select the devices you want aligned.

   If you do not select any device, the settings are applied to all devices in the current Modgen.
3. Do one of the following:
   - Right-click and choose *Set Member Alignment/Spacing*.
   - Click the Member Spacing/Alignment icon on the *Modgen Placement* toolbar.

The Set Member Alignment and Spacing form appears.



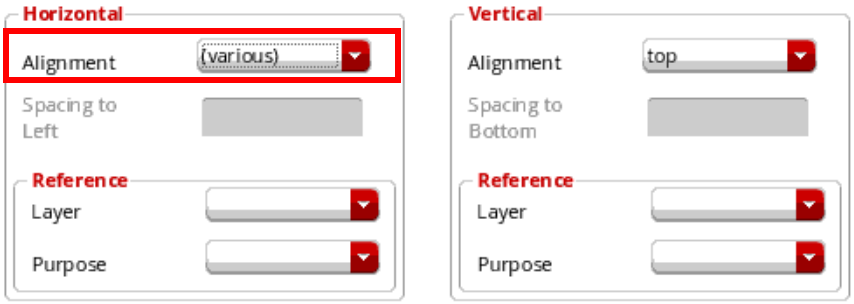
You can also specify Modgen alignment and spacing parameters directly in the layout canvas. To do this, select the required devices and choose *Place—Modgen—Set Member Alignment and Spacing* to display the Set Member Alignment and Spacing form.

4. Select an *Alignment* value in the *Horizontal* section to specify the alignment of instances along the horizontal axis, which is the x-axis.

The available options are *left*, *right*, *center*, and *custom*, and *customRight*.

In the above example, all selected instances have the same vertical and horizontal alignments, *left* and *top* respectively.

In the following example, the selected instances have the same vertical alignments, but their horizontal alignments are different. So, the horizontal *Alignment* is set to *(various)*.



Note: When you change the *Alignment* from *various* to any other value, the *various* option disappears from the *Alignment* drop-down list.

5. Select an *Alignment* value in the *Horizontal* section to specify the alignment of instances along the horizontal axis, which is the x-axis.

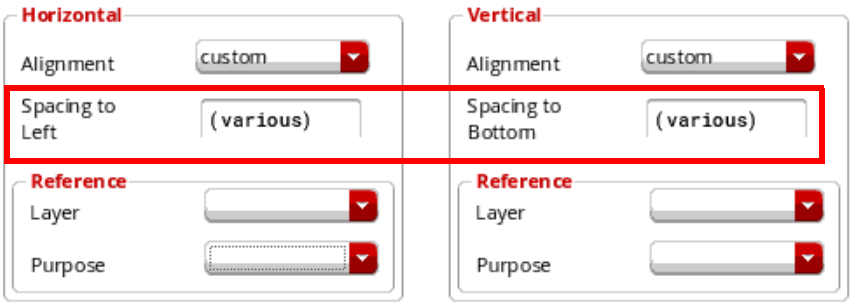
   The available options are *top*, *bottom*, *center*, *custom*, and *customTop*.

   The spacing field is available only when the *Alignment* is set to *custom* or *customRight* or *customTop* in either one or both of the *Horizontal* and *Vertical* sections.

6. Specify the horizontal spacing values in the *Spacing to Left* field.

7. Specify the vertical spacing in the and *Spacing to Bottom* field.

   By default, if the spacing values are different for the selected instances, then *(various)* is displayed.



8. In the *Reference* section, select a reference *Layer*.

9. Select a reference *Purpose*.

   Similar to *Alignment* and *Spacing* fields, if the layers and purposes are different for the selected instances, then *(various)* is displayed. You can reset these values.

Note: The reference layer and purpose fields are blank by default. In this state, the values are determined by the union of all non-text layers that contain the Modgen shapes and that honor the `minSpacing` DRC rules.

10. Click *Save Values* to overwrite environment variables with the values specified in the Set Member Alignment and Spacing form.
11. Select *Apply to All Modgen Members* to apply the custom spacing to all the devices in the module.

    If you do not select this option, custom spacing is applied only to the selected devices.
12. Click *OK*.

The alignment and spacing of the Modgen members is updated based on the values you specified.

In the presence of WSPs in a design, the WSP grid is ignored when placing individual Modgen members in rows. Instead, the settings in the Set Member Alignment and Spacing form are honored. However, the Modgen figGroup honors WSP settings and is snapped to the nearest WSP grid.

Removing a Custom Spacing Distance

Custom spacing is interpreted internally as a custom alignment. To completely remove a custom spacing distance, you must choose an alignment for the devices.

Related Topics

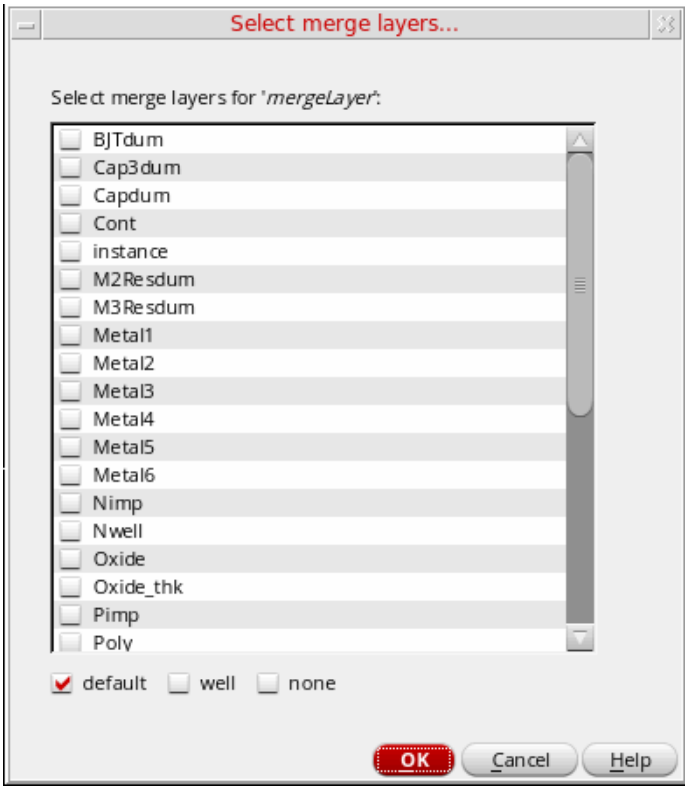
Set Member Alignment and Spacing Form

Specifying Modgen Device Alignment and Spacing

Merging Layers

Use the Select merge layers... form to specify the layers that need to be merged.

1. To invoke the Select merge layers... form, click the Merge Layers icon on the *Modgen Placement* toolbar. Alternatively, you can right-click anywhere in the Modgen Editor window and select *Merge Layers*. The Select merge layers... form is displayed.



   Note: The Select merge layers... form can also be invoked from Constraint Manager with the `mergeLayer` parameter in the Modgen `ci` constraint.

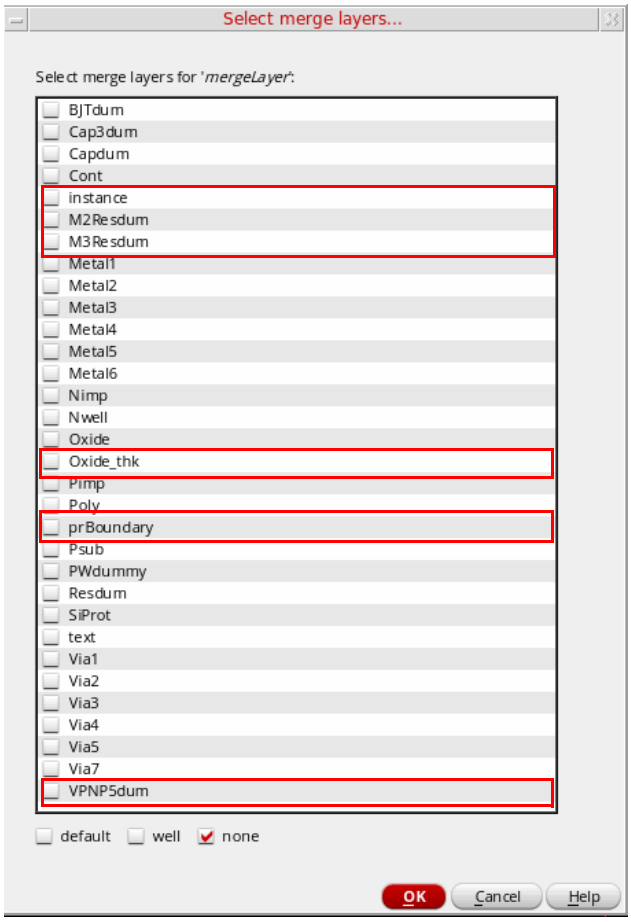
2. Select the check boxes adjacent to the layers that you want to merge.

   Alternatively, select one of the Layers Preset: *default*, *well*, or *none* to select all layers of the selected type.

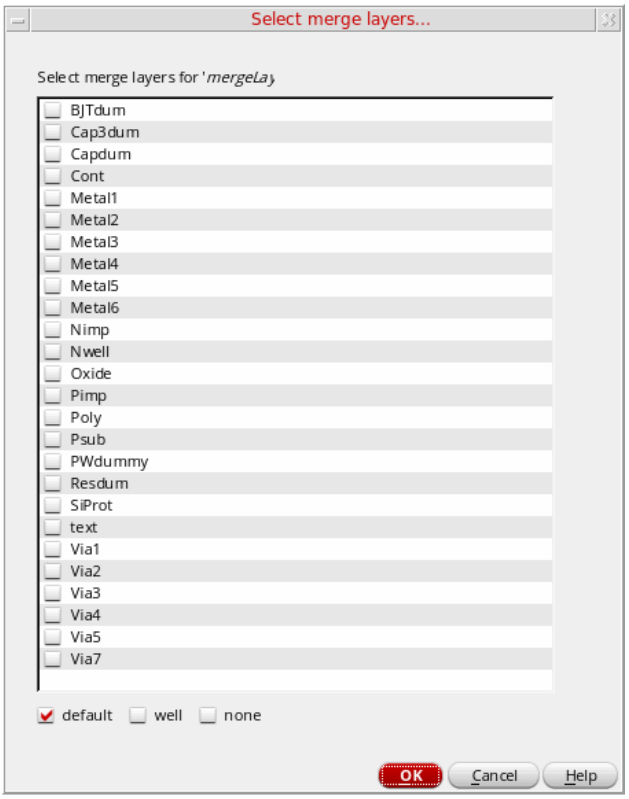
3. Click *OK*.

Selecting the *Use Layer Palette LPPs Only* check box in the Layout Editor Options form synchronizes the layers displayed in the list with the layers listed in the Palette assistant of Layout XL. To access the Layout Editor Options form, choose *Options - Editor...* from the Layout XL window.

When you enable layer synchronization and change the filters in the Palette assistant using the *Used*, *Valid*, and *Routing* check boxes, layers of specific categories are displayed in the merge layers list box. This change is visible when you update the filters in the Palette assistant and then launch the Select merge layers form.



Before the Layer Palette LPP sync up
(the highlighted entries are not visible in the image on the right)



After the Layer Palette LPP sync up

Related Topics

Select Merge Layers Form

modgenMergeLayers

Modgen Merge Wells

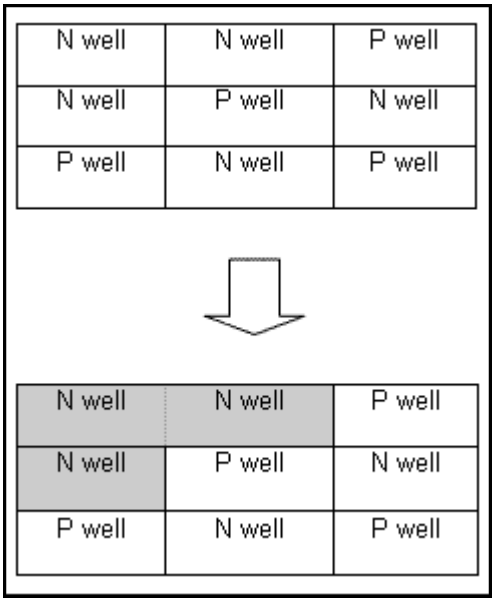
Modgen Merge Wells

Merging of wells in Modgens is controlled by the `mergeLayer` parameter in the Modgen `ci` constraint. The default value for the `mergeLayer` parameter in Modgen is `default`. For backward compatibility with the 614 Modgen constraints, the `modgenMergeWells` environment variable can be set in the `.cdsenv` file.

If this environment variable is set to `t` and the `mergeLayer` parameter is set to `default`, the wells of the Modgen instances are merged. However, if the environment variable is not set, and the `mergeLayer` parameter is set to `default`, the wells in the Modgen layout are not merged.

Other possible values of the `mergeLayer` parameter are: `none`, `well`, and any other custom layer, which can be selected from the Constraint Manager GUI. If `mergeLayer` is set to `none`, there is no well merging in the Modgen. If `mergeLayer` is set to `well`, the wells of the Modgen devices are merged and any DRCs between well layers and other layers of the devices are ignored. If `mergeLayer` is set to a custom layer, such as `Oxide_thickness`, the shapes on that layer are merged and DRCs between that layer and other layers of the devices are ignored.

In addition, you can merge multiple layers, such as `Nwell` and `Oxide_thk`. Before merging layers, only the top-level net connectivity between the layers is checked. So, layers across devices that have different master Pcells can also be merged. An example for well merging is shown below:



To specify whether the analog placer should have control over the well and body contact geometry, set the `aapDefaultWellsUnderPlacerControl` environment variable to `t`.

The merge layer functionality differs when there is a pane guard ring inside the Modgen structure. A layer across a pane guard ring is merged only if one of the following situations is applicable.

- The layer is defined as a recognition layer in the technology file.
- The layer is a well layer with the same net connectivity across all devices and the pane guard ring.

Related Topics

modgenMergeWells

Modgen Guard Rings

Merging Layers

4

Placing Modgen Members Interactively

Modgen supports the following interactive placements of its members in the Modgen Editor window.

- Moving Modgen Instances Interactively
- Adding Empty Rows and Columns to Modgens
- Rotating and Flipping Modgen Instances
- Swapping Instances

Related Topics

Modgen Tasks

Moving Modgen Instances Interactively

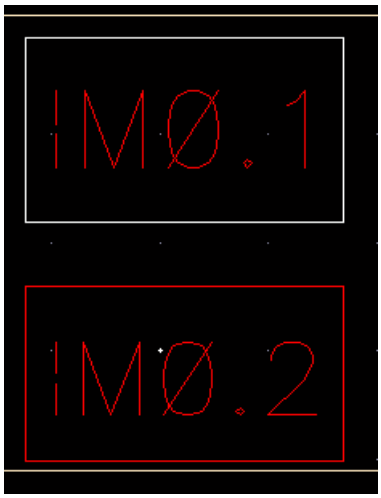
Modgen members can be moved interactively to other empty or non-empty cells on the same or a different row/column. If an instance is moved within the same row/column, the devices in that row/column are re-ordered. If an instance is moved to a different row/column, the positions of the moved instance and the device on which it is dropped are swapped.

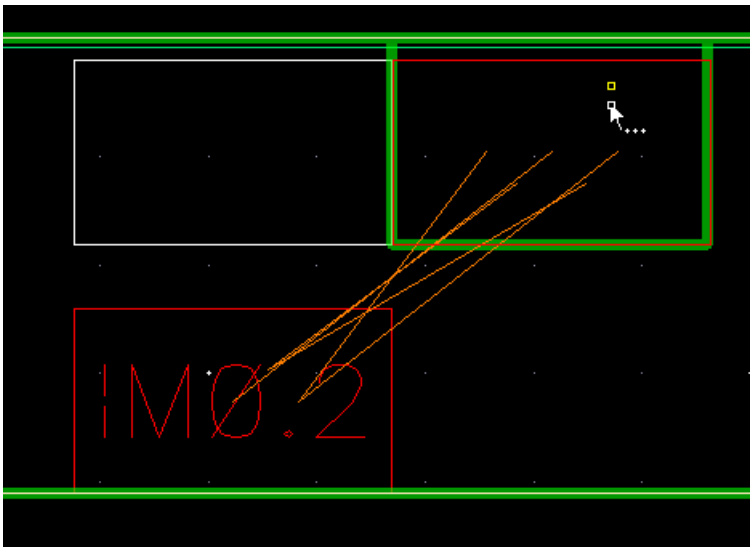
To move instances to an empty or non-empty cell, you can use one of the following methods:

- By dragging the instance to the target location

a. Click the instance you want to move.

b. While holding down the mouse button, drag the instance to the required empty cell.





c. Release the mouse button to place the instance.

- Using the shortcut menu.
  - **a.** Right-click the instance and select *Move*.
  - **b.** Click and release the left mouse button on the instance.
  - **c.** Drag the instance to the desired cell.
  - **d.** Click to place the instance.
- Using the Move icon.
  - **a.** Select the instance.
  - **b.** On the *Modgen Placement* toolbar, click *Move* .
  - **c.** Click and release the left mouse button on the instance.
  - **d.** Drag it to the desired cell
  - **e.** Click to place the instance.

Related Topics

Placing Modgen Members Interactively

Rotating and Flipping Modgen Instances

Swapping Instances

Customizing Rows and Columns

You can use the interactive Modgen commands to customize rows and columns by:

- Adding empty rows and columns
- Highlighting and deleting empty rows and columns
- Selecting rows and columns

Related Topics

Placing Modgen Members Interactively

Adding Empty Rows and Columns to Modgens

Highlighting and Deleting Empty Modgen Rows and Columns

Selecting Rows and Columns in Modgens

Adding Empty Rows and Columns to Modgens

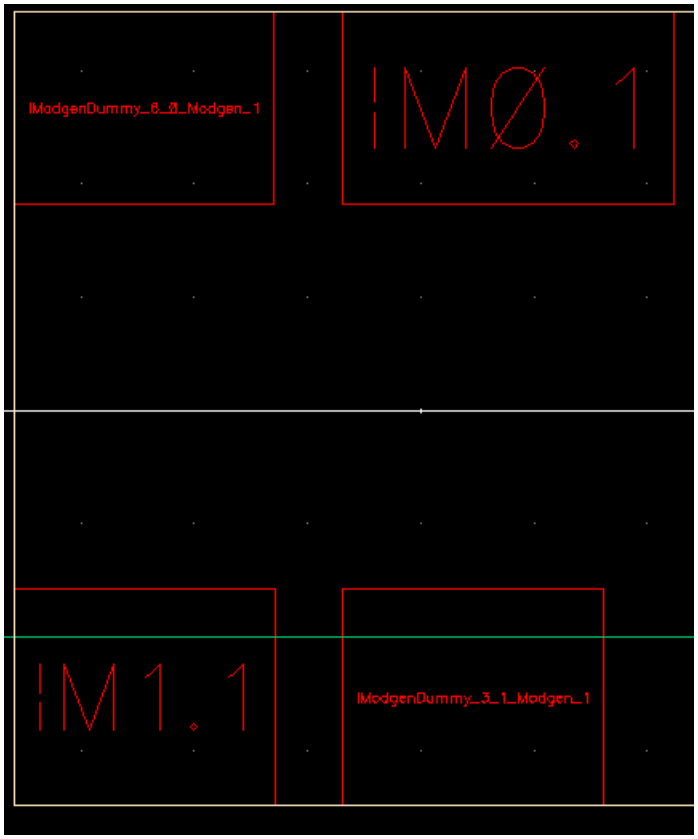
To add an empty row, select an instance in the row above or below which you want to add the row.

Do one of the following:

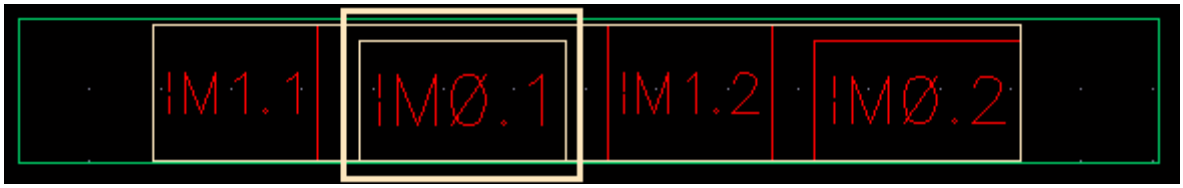
- On the *Modgen Placement* toolbar, click the arrow next to the *Add Empty Row/ Column* button and select *Add Empty Row Top* or *Add Empty Row Bottom*.

- Right-click the instance, navigate to *Add Empty Row/Column*, and select *Top* or *Bottom*.

A new empty row is added.



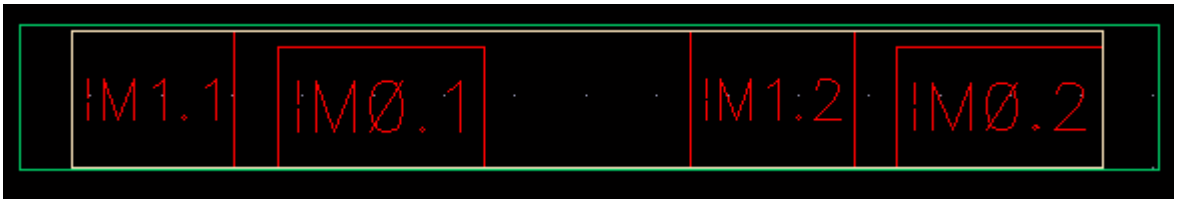
To add an empty column, select the instance to the right or left of which you want to add the column.



Do one of the following:

- On the *Modgen Placement* toolbar, click the down arrow next to the *Add Empty Row/Column* button and select *Add Empty Row Left* or *Add Empty Row Right*.
- Right-click the instance, point to *Add Empty Row/Column*, and select *Left* or *Right*.

An empty column is added.



Instead of using the GUI, use the `mgAddEmptyRCCB` API to add empty rows and columns.

Related Topics

Placing Modgen Members Interactively

Adding Empty Rows and Columns to Modgens

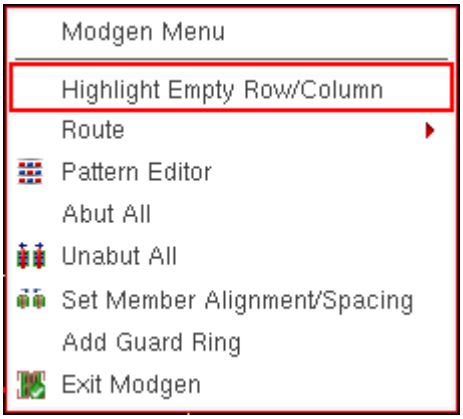
Highlighting and Deleting Empty Modgen Rows and Columns

Selecting Rows and Columns in Modgens

Highlighting and Deleting Empty Modgen Rows and Columns

An empty row or column is highlighted in the Modgen editor window. The highlighting is on by default. When highlighting is on, the shortcut menu displays the *Remove Empty Row/column Highlight* option to turn the highlighting off as shown in the figure below.

When highlighting is off, the shortcut menu displays the *Highlight Empty Row/Column* option to turn the highlighting on, as shown in the figure below.



These options are displayed in the shortcut menu when:

- One or more instances are selected.
- No instance is selected.

However, the placement of this option in the shortcut menu is different in both these cases.

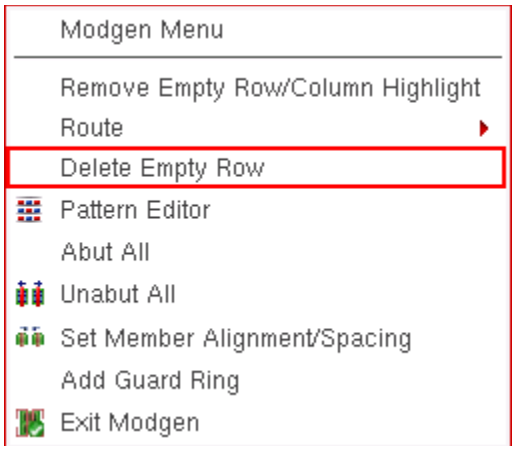
When you perform an *undo* operation in the Modgen editor window, the highlighting, if on, gets automatically turned off.

Alternative SKILL Function: `mgHilightEmptyRowColumnCB`

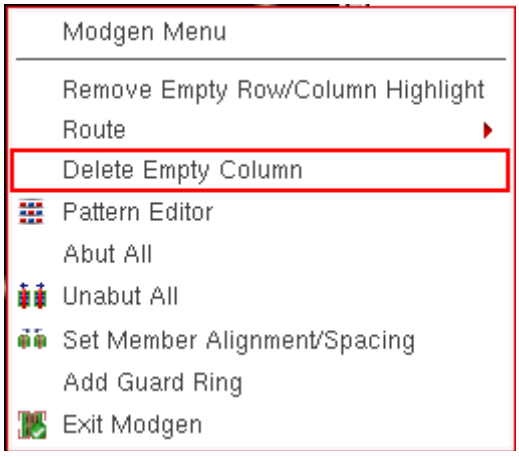
The Zoom to Fit option also includes empty rows and columns in the Modgen editor window even if they are at the outer edges of the window.

Deleting Empty Rows and Columns

As a part of interactive Modgen editing, you can delete empty rows or columns from the Modgen editor window. To delete empty rows, you need to right-click the empty row and select the *Delete Empty Row* option from the shortcut menu, as shown in the figure below.



To delete empty columns, you need to right-click the empty columns and select the *Delete Empty Columns* option.



If you right-click at the location where an empty row and an empty column are overlapping, then both these options are displayed in the shortcut menu. You can select any of these options to either delete the row or column.

Note: If you right-click at the location where there is no empty row or columns, then the *Delete Empty Rows* or *Delete Empty Columns* option is not displayed in the shortcut menu.

These options are displayed in the shortcut menu when:

- One or more instances are selected.
- No instance is selected.

Alternative SKILL Functions: `mgDeleteEmptyRowColumnCB`, `mgDeleteAllEmptyRowColumnCB`.

Related Topics

Placing Modgen Members Interactively

Adding Empty Rows and Columns to Modgens

Adding Empty Rows and Columns to Modgens

Selecting Rows and Columns in Modgens

Selecting Rows and Columns in Modgens

To move or swap the entire row or column, you may need to select the entire row or column.

To select the entire row:

1. Select the instance in a row.
2. Right-click the instance, point to *Select Row/Column*, and select *Select Entire Row*.
3. The entire row will be selected.

To select the entire column:

1. Select the instance in a column.
2. Right-click the instance, point to *Select Row/Column*, and select *Select Entire Column*.
3. The entire column will be selected.

Alternative SKILL Function: `mgSelectRowColCB`

Related Topics

Placing Modgen Members Interactively

Customizing Rows and Columns

Rotating and Flipping Modgen Instances

You rotate or flip instances for the following reasons:

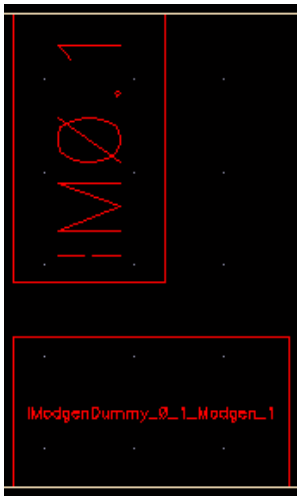
- To optimize placement
- To optimize routing; for example, to optimally connect resistors in a series
- To allow for matched routing

To rotate instances:

1. Select the instance that you want to rotate.

2. Do one of the following:
   - On the *Modgen Placement* toolbar, click the *Rotate Left* or *Rotate Right* button.
   - Right-click the instance, point to *Rotate/Flip*, and select *Rotate Left* or *Rotate Right*.

The selected instance is rotated.



Related Topics

mgRotateLeftCB

mgRotateRightCB

Placing Modgen Members Interactively

Moving Modgen Instances Interactively

Adding Empty Rows and Columns to Modgens

Flipping Instances

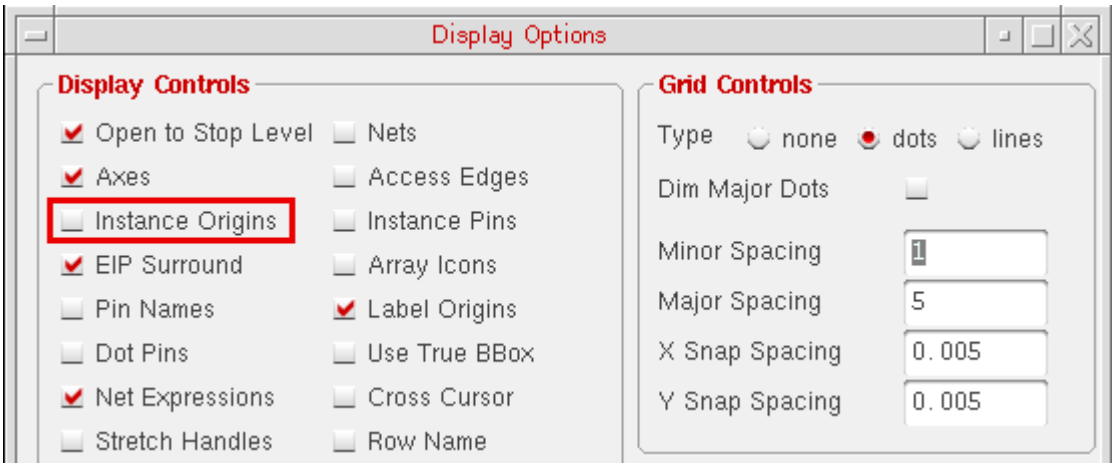
Swapping Instances

Flipping Instances

You can flip Modgen instances for compact placement and to allow abutment with adjoining instances.

Before you flip a Modgen instance:

1. In the layout window, choose *Options – Display* to view the Display Options form.



2. In the *Display Controls* group box, select the *Instance Origins* check box.
3. Click the *OK* button to save the settings.

A plus sign appears on the instance, which indicates whether the instance was flipped.

To flip a Modgen instance:

1. Select the instance in the design display area.

2. Do one of the following:
   - On the *Modgen Placement* toolbar, click the *Flip Horizontal* or *Flip Vertical* button.

- Right-click the instance, point to *Rotate/Flip*, and select *Flip Horizontal* or *Flip Vertical*.

  Alternative SKILL Functions: `mgFlipHorizontalCB`, `mgFlipVerticalCB`

The selected instances is flipped. In the figure below, notice that the plus sign has moved upward.

Related Topics

Placing Modgen Members Interactively

Moving Modgen Instances Interactively

Adding Empty Rows and Columns to Modgens

Swapping Instances

Swapping Instances

By using the swap function, you can interchange the position of two instances, rows, or columns. This helps optimize routing. For example, by swapping instances, you can optimally connect resistors in a series.

To swap instances:

1. Select the two required instances in the design display area.

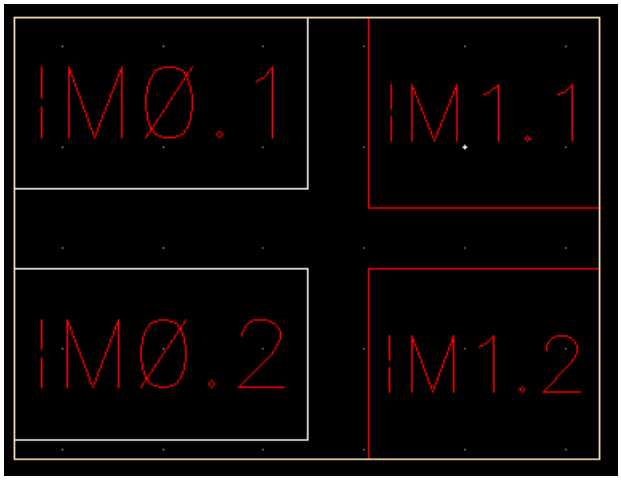
2. Do one of the following:
   - On the *Modgen Placement* toolbar, click the arrow next to the *Swap* button, and select *Swap Instances*.
   - Right-click the selected instances, point to *Swap*, and select *Swap Instances*.

     Alternative SKILL Function: `mgSwapCB`

The selected instances are swapped. In the figure below, notice that `M1.2` and `M0.2` are swapped.

To swap an entire row:

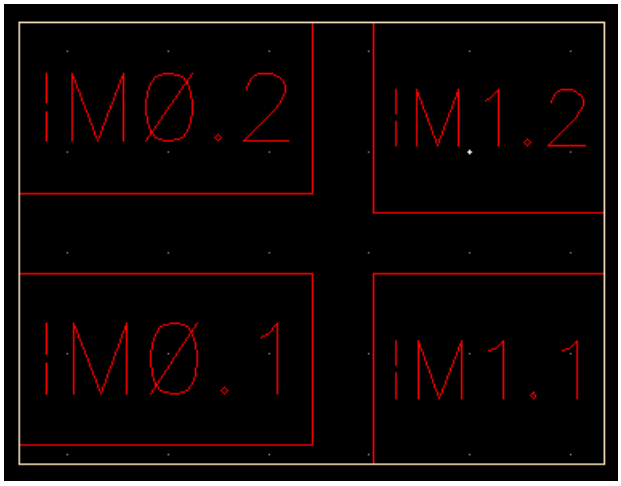
1. Select an instance each in the two rows that you want to swap.



2. Do one of the following:

   - On the *Modgen Placement* toolbar, click the arrow next to the *Swap* button, and select *Swap Rows*.
   - Right-click a selected instance, point to *Swap*, and select *Swap Rows*.

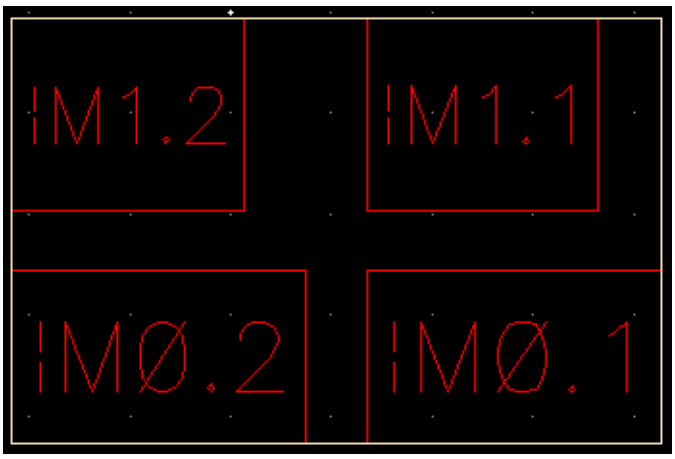
3. The selected rows are swapped. In the figure below, notice that the row (`M0.2 | M1.2`) is swapped with the row (`M0.1 | M1.1`).



To swap entire columns:

1. Select an instance each in the two columns that you want to swap.

2. Do one of the following:
   - On the *Modgen Placement* toolbar, click the arrow next to the *Swap* button, and select *Swap Columns*.
   - Right-click a selected instance, point to *Swap*, and select *Swap Columns*.
3. The selected columns are swapped. In the figure below, notice that the column `(Mi.2 | M0.2)` is swapped with the column `(M1.1 | M0.1)`.



Related Topics

Placing Modgen Members Interactively

Moving Modgen Instances Interactively

5

Modgen Forms

This topic lists the forms that are used to customize Modgens.

Apply Reuse Template Form

(Layout EXL and Higher Tiers) Use the Apply Reuse Template form to generate a new Modgen constraint using a Modgen template file.

Field	Description
Device Group	Specifies the device group under which the required Modgen pattern file is located.
Pattern File	Specifies the name of the Modgen reuse template file.
Rows	Specifies the number of rows to be included in the new Modgen. The default value is 1.

Related Topics

Reusing a Modgen Template File

Body Contact Options Form

Use the Body Contact Options form to specify the parameters for body contacts created by Modgens.

Field	Description
Apply To	Lists the instances for which body contacts need to be created.
Type	Provides a list of body contacts defined in the technology file. Select the required type of body contact.
Net	Lists the nets that are connected to objects in the active Modgen. All body contacts are attached to the selected net. *Net* combo box lists the nets that are connected to objects in the active Modgen. To choose a net that is present in the cellview, but not in the current Modgen, type the net name in the *Net* box. The entry is validated against all existing nets in the current cellView. If a matching net is not available in the current cellView, then a warning message is issued and the net is created.
Layer	Lets you select the reference layer for the body contacts. Environment Variable: modgenDefVCSRefPurpose
Separation	Specifies the distance (in microns) between the body contacts and devices. This value is added to minimum DRC to space the body contact. You can also specify a negative value in this field. When you specify a negative value, the body contact is placed inside of the value that Modgen is calculating as a legal DRC distance.
Remember Values	Saves the specified values for all dummy devices. Environment Variable: modgenRememberBodyContactVals

Related Topics

Defining Modgen Body Contact Properties

Grid Placement In Modgens

Placement of Modgen Body Contacts on Grids

Configure Dummies Form

Use the Configure Dummies form to edit dummy attributes.

Field	Description
Left pane	Lists the dummies that are selected in the layout canvas.
Right pane	Displays the attributes, connectivity information, and CDF parameters of the selected dummies.
List at the top-left corner	Specifies the following dummy properties: ■ *Attributes* lets you edit the *Name*, *Type*, and *Master* of dummies. ■ *CDF Parameters* displays a list of CDF parameters for the dummies, which you can edit. ■ *Connectivity* lists connectivity information for the Modgen dummy devices, which you can edit.
Parameters	Specifies the following dummy parameters.
Name	Specifies the name of the dummy. If multiple dummies are selected, then (various) is displayed.
Type	Specifies the type of dummy. Valid values are: ■ *copy*: Uses the dummy parameters and default values of the source instances. ■ *lcv*: You can specify the Master lib:cell:view of the dummies. ■ *neighbor*: Considers the lib:cell:view of the neighboring device as the master.

Field	Description
Master	Defines the master lib:cell:view when the Type is set to `lcv`. ■ *Fingers*: Number of fingers. Valid values are default from the `Pcell`, `match_src` from the reference dummy, and the numeric value found from `match_src`. ■ *l*: Length of the dummies. Valid values are default from the `Pcell`, `match_src` from the reference dummy, and the numeric value found from `match_src`. ■ *w*: Width of the dummies. Valid values are default from the `Pcell`, `match_src` from the reference dummy, and the numeric value found from `match_src`. For all above options, when multiple dummies of different values are selected, then value (various) is displayed.

Related Topics

Editing Modgen Dummy Properties in Array Assistant

Adding Dummies around Modgen Instances using the Array Assistant

Dummy Options Form

Use the Dummy Options form to specify the net to which dummy devices need to be attached and the type of dummy devices.

Field	Description
Apply To	Lists the selected Modgen instances. This is not editable.
Dummy Net	Specifies the net to which dummies are to be connected.
Default	Specifies the default net to which all dummies are to be connected. You can select a different net from the drop-down list. To select a net that is present in the cellview but not in the current Modgen, type the net name in the *Default* combo box. The entry is validated against all existing nets in the current cellView. If a matching net is not available, then a warning message is issued and the net is created. Environment Variable: modgenDummyNet
Set Individual Terminals Net	Sets individual nets for dummy terminals, as opposed to the same net for all four terminals. When the option is selected, the *Default* combo box is still available. If you choose a different net in the *Default* combo box, the selected net is applied only to the dummy terminals for which you have not specified individual terminal nets.
List of terminals	Lists the nets that are connected to objects in the active Modgen.
Dummy Type	

Field	Description
Type	Specifies the default type for dummy devices. The available options are: ■ *neighbor*: The master `lib:cell:view` of the dummy is the same as the neighboring device. The dummy device type depends on the location of the dummy and the setting of the modgenMakeMinDummies environment variable. ■ *default*: You can specify a different master `lib:cell:view` for the dummy devices or the modgenDummyLib, modgenDummyCell, and modgenDummyView environment variables to specify the default dummy device type. ■ *copy*: Creates identical dummies of the selected instances. In this mode, the dummy parameters and default values of the source instances are used. Different values cannot be specified. Environment Variable: modgenPhysConfigs
Specify Parameters	Activates the options in the *Dummy Parameters* section.
Dummy Parameters	Specifies the default dummy parameters. The default values for these fields are determined by the modgenMakeMinDummies environment variable.
Number of Fingers	Specifies the number of dummy fingers. The available options are: ■ *CDF Default*: The default number of fingers, as specified in the CDF, is created. ■ *Same As Neighbor*: The same number of fingers as the neighboring device is created. ■ *Specify*: You can specify the number of fingers to be created in the box beside the *Number of Fingers* cyclic field. Environment variables: modgenDummyNumFingersOptions, modgenDummyNumFingersValue

Field	Description
Length	Specifies the length of the dummies. The available options are: ■ *CDF Default* – The default finger length, as specified in the CDF, is considered. ■ *Same As Neighbor* – The length of fingers of the neighboring device is considered. ■ *Specify* – You can specify the length of fingers in the box beside the *Length* cyclic field. Scale factors can be used to specify the length; for example .1u. Environment variables: modgenDummyLengthValue

Field	Description
Width *Number of Fins* (Virtuoso Advanced Node for Layout Only)	Specifies the width of the dummies. Select one of the following: ■ *CDF Default* – The default number of fins, as specified in the CDF is considered ■ *Same As Neighbor* – The width of fingers of the neighboring device is considered ■ *Specify* – You can specify the width of fingers in the box beside the *Width* cyclic field Scale factors can be used to specify the width; for example 200n. When *Width* represents the total width for the device (number of fingers * finger width), set the modgenWidthParamProportionalToFingers environment variable to `t` to indicate that the finger width of Modgen dummy instances must be proportional to the number of fingers. For FinFET devices, the default value is `no`; for non-FinFET devices, the default value is `yes`. In Advanced Node for Layout, the *Width* field is replaced by the *Number of Fins* field, which specifies the number of fins in the dummies. The available options are: ■ *CDF Default* – The default number of fins, as specified in the CDF is considered ■ *Same As Neighbor* – The number of fins of the neighboring device is considered ■ *Specify* – You can specify the number of fins in the box beside the *Number of Fins* cyclic field Environment variables: modgenDummyWidthOptions, modgenDummyWidthOptions, modgenDummyWidthValue
Default Dummy	Specifies the default dummy to be used when *Dummy Type* is set to *Default*.
Browse	Lets you select the required library, cell, and view for the dummy devices.
Dummy Library	Specifies the library to which the default dummy device belongs. Environment variable: modgenDummyLib

Field	Description
Dummy Cell	Specifies the cell to which the default dummy device belongs. Environment variable: modgenDummyCell
Dummy View	Specifies the view to which the default dummy device belongs. Environment variable: modgenDummyView
Remember Values	Save the form values for all dummy devices and loads them until the values are overwritten with new values. Environment Variable: modgenRememberDummyVals

Related Topics

Adding Dummy Device Rows or Columns

Guard Ring Options Form

Use the Guard Ring Options form to create multipath Part (MPP) guard rings.

Field	Description
Type	Specifies the type of guard ring to be created. ■ *none*: Indicates no guard ring and can be used to delete an existing one. ■ *ring*: Indicates a single guard ring around the entire Modgen. ■ *pane*: Indicates the new window pane configuration that includes a guard ring around all devices as well as the entire Modgen.
Shape	Specifies the shape of the guard ring.
Net	Specifies the net to which the guard ring needs to be attached.
Spacing	Specifies the spacing between the guard ring and other devices. You can specify a negative spacing value. In this case, the guard ring is placed inside the value that Modgen calculates as a legal DRC distance.
Left Spacing	Specifies the spacing of the guard ring from the left edge.
Right Spacing	Specifies the spacing of the guard ring from the right edge.
Top Spacing	Specifies the spacing of the guard ring from the top edge.
Bottom Spacing	Specifies the spacing of the guard ring from the bottom edge. If both *Spacing* and <Side> *Spacing* values are specified, the <Side> *Spacing* values are applied and the *Spacing* value is ignored.
MPP to Use	Specifies the MPP to be used for the guard ring.

Related Topics

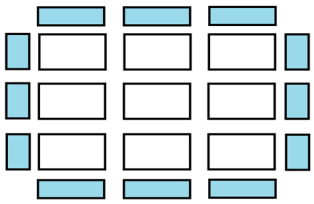
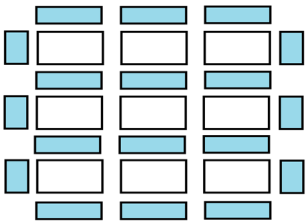
Modgen Guard Rings

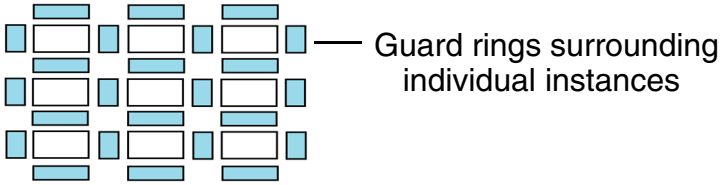
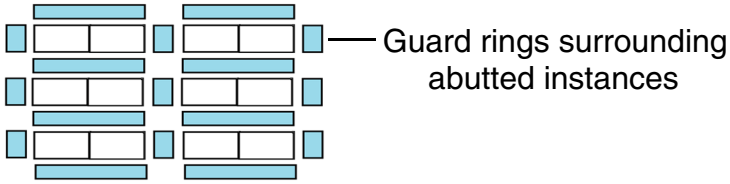
Creating a Multipath Part Guard Ring

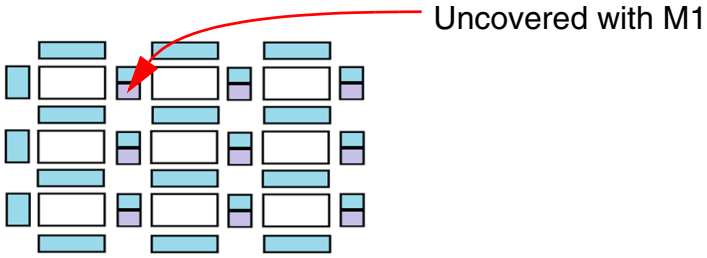
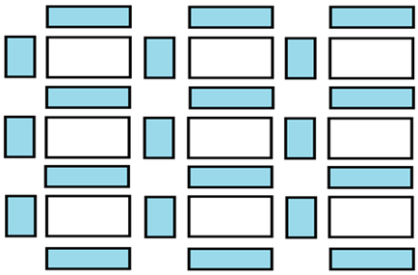
Identical Guard Ring Options Form

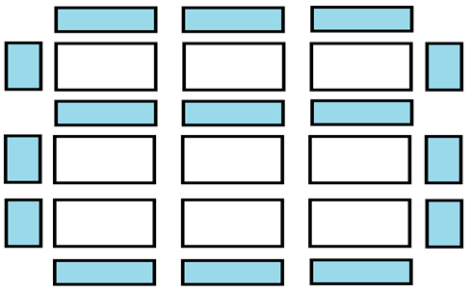
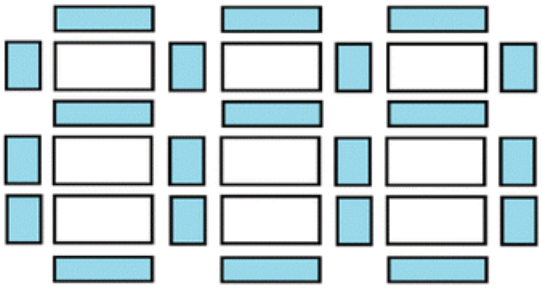
(Advanced Node for Layout) Use the Identical Guard Ring Options form to define identical guard rings for a Modgen.

The *Create Identical Guard Ring* command is available only if the PDK is configured to support identical guard rings. For more information on this capability, contact your Cadence Customer Support representative.

Field	Description
Type	Defines the way in which the identical guard ring surrounds the adjoining Modgen instances. The available options are: ■ *None*: Deletes the identical guard ring. ■ *Surround*: The identical guard ring surrounds all the instances.  ■ **Surround+Strip**: The guard ring surrounds all the instances, and strips of guard ring instances are inserted between one or more Modgen rows. 

Field	Description
	■ **Grid:** The guard ring surrounds every instance or group of abutted instances separately.   ■ *Horizontal Strip Width (in fins)* value specifies the vertical width (as a number of fins) of the horizontal guard ring strips, represented by the horizontal blue rectangles in the diagram above. 
GR Definition	Lists the available guard ring definitions provided by the PDK, which define various aspects of the guard ring. They provide information such as the unit cell lib:cell:view, its parameters, and parameter callbacks, which helps Modgens instantiate the guard ring correctly.
Net	Specifies the net to which the guard ring needs to be connected.

Field	Description
Break GR	Provides spaces for connecting guarded instances via the `Metal1` layer routes. This option affects the `Metal1` layer of all vertical guard ring strips except those on the left side of the guard ring. The effect is determined by the *GR Definition*, but usually the `Metal1` layer is trimmed from the bottom as depicted in the following diagram. 
Add Corners	Creates guard rings at all available corners around the selected instances.
Turn Top Side On, Turn Left Side On, Turn Right Side On, Turn Bottom Side On	Specify the side switches that can be used turn on or off the guard ring for the specified sides. For example, the following diagram depicts a *Grid* guard ring in which the right side is turned off. 
Horizontal Strip Width (in fins)	Specifies the vertical width (as a number of fins) of the horizontal guard ring strips.

Field	Description
Number of Rows Between Strips	Specifies the number of instance rows that are required between strips of guard rings, counting from the bottom row. **Example 1:** *Type* is set to *Surround+Strip*; *Number of Rows Between Strips* is set to `2`  **Example 2:** *Type* is set to *Grid*; *Number of Rows Between Strips* value is set to `2`. 

Related Topics

Creating Identical Guard Rings around Modgens (Virtuoso Advanced Node for Layout Only)

Reuse Template Exaction Form

(Layout EXL and Higher Tiers) Use the Reuse Template Extraction form to extract reusable templates from the given source layout or Modgen.

Field	Description
Device Group	Specifies the group under which the pattern file must be created.
Pattern File	Specifies the name of the Modgen reuse template file (with the .txt file extension).

Related Topics

Generating a Modgen Template File

Select Merge Layers Form

Use the Select merge layers form to specify the layers that need to be merged.

Field	Description
Select merge layers for 'mergeLayer'	Displays a list of all used layers in the design.
default	Selects all default layers in the Select merge layers for `mergeLayer` list.
well	Selectsall well layers in the Select merge layers for `mergeLayer` list.
none	Deselecta all layers in the Select merge layers for `mergeLayer` list.

Related Topics

Merging Layers

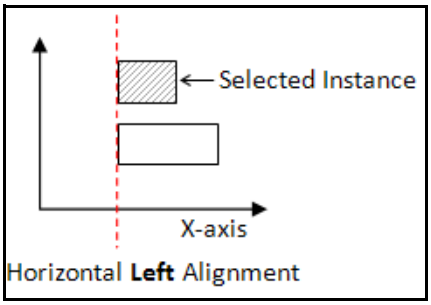
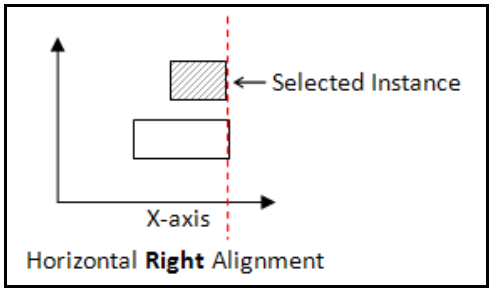
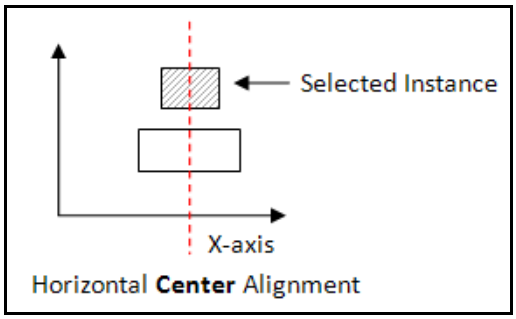
modgenMergeWells (environment variable)

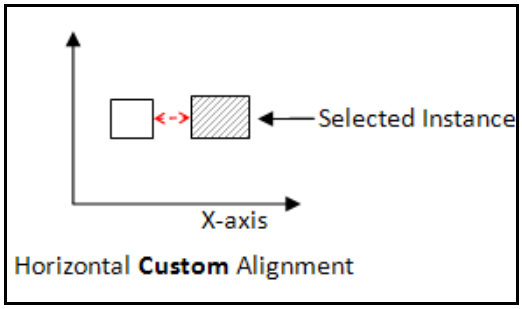
modgenMergeLayers (environment variable)

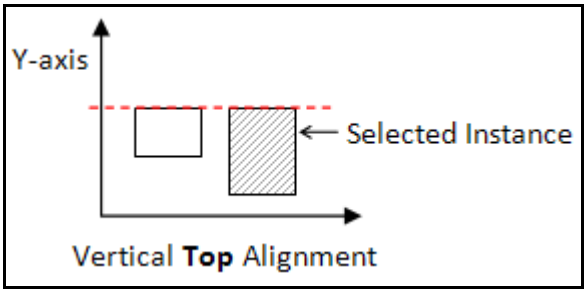
Set Member Alignment and Spacing Form

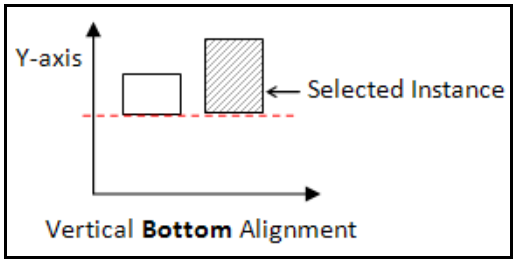
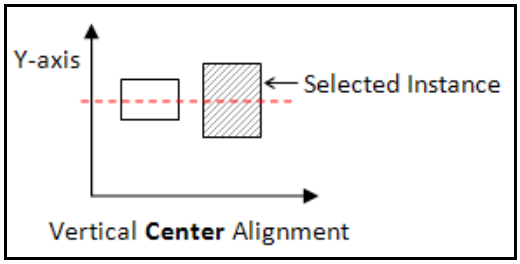
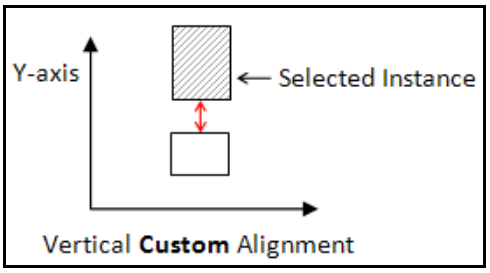
Use the Set Member Alignment and Spacing form to specify alignment and spacing values for instances in Modgens.

Field	Description
Apply To	Lists the Modgen members to which the alignment and spacing settings need to be applied.
Horizontal	Specifies the alignment of instances along the horizontal axis, which is the x-axis.

Field	Description
Alignment	Specifies the horizontal alignments of the selected instances. The available options are: ■ **left:** The left edge of the selected instance is aligned with the left edge of the instance below it.  ■ **right:** The right edge of the selected instance is aligned with the right edge of the instance below it.  ■ **center:** The center of the selected instance is aligned with the center of the instance below it. 

Field	Description
	■ **custom:** The left edge of the selected instance is separated by the specified horizontal distance from the right edge of the neighboring instance.  ■ **customRight:** The right edge of the selected instance is separated by the specified horizontal distance from the left edge of the neighboring instance. Environment variables: modgenDefHoriAlignment
Spacing to Left	Specifies the spacing of selected instances from the left edge. Environment variables: modgenDefHoriSpacing
Layer	Specifies the reference layer based on which: ■ The horizontal spacing of devices must be calculated. ■ Instances must be aligned horizontally. Environment variable: modgenDefHCSRefLayer
Purpose	Specifies the reference purpose for calculating custom spacing value and for specifying the alignment of instances. Environment variable: modgenDefHCSRefPurpose
Vertical	Specifies the alignment of instances along the vertical axis, which is the y-axis.

Field	Description
Alignment	Specifies the vertical alignments of the selected instances. The available options are: ■ **top:** The top edge of the selected instance is aligned with the top edge of the instance to the left. 

Field	Description
	■ **bottom:** The bottom edge of the selected instance is aligned with the bottom edge of the instance to the left.  ■ **center:** The center of the selected instance is aligned with the center of the instance to the left.  ■ **custom:** The bottom edge of the selected instance is separated by the specified vertical distance from the top edge of the instance below it.  ■ **customTop:** The top edge of the selected instance is separated by the specified vertical distance from the bottom edge of the instance above it. Multiple instances can be selected and their horizontal and vertical alignments can be set. For example, if multiple instances in a row are selected and their vertical alignment is set to top, all instances are top aligned. Environment variable: modgenDefVertAlignment

Field	Description
Spacing to Bottom	Specifies the spacing of selected instances from the bottom edge. Environment variables: modgenDefVertSpacing
Layer	Specifies the reference layer based on which: ■ The vertical spacing of devices must be calculated. ■ Instances must be aligned vertically. Environment variable: modgenDefVCSRefLayer
Purpose	Specifies the reference purpose for calculating custom spacing value and for specifying the alignment of instances. Environment variable: modgenDefVCSRefPurpose
Save Values	Overwrites environment variables with the values specified in the Set Member Alignment and Spacing form.
Load Values	Resets all values in the form with values from their corresponding environment variables stored in the .cdsenv file. These values were saved earlier by clicking *Save Values*.
Apply to All Modgen Members	Applies the custom spacing to all the devices in the module.

Related Topics

Specifying Modgen Device Alignment and Spacing

Surround Modgen Form

Use the Surround Modgen form to add surround dummies around Modgen instances. The form includes a few common fields and a few tabs. The following table describes the common fields in the Surround Modgen form, which are available only in the Virtuoso Advanced Node for Layout branch.

Field	Description (Virtuoso Advanced Node for Layout)
Default values	Provides a list of presets that have been registered using a SKILL callback function. This is helpful when you want to apply the same style across designs. You can load the values and edit them as per your requirement.
Load	Loads values from the selected preset.
Top	Adds dummies to the top of the selected Modgen instances.
Left	Adds dummies to the left of the selected Modgen instances.
Right	Adds dummies to the right of the selected Modgen instances.
Bottom	Adds dummies to the bottom of the selected Modgen instances.

In addition to the above fields, the Surround Modgen form contains the following tabs.

Tab	Description
Master	Specifies the cellview to be used to create custom dummy devices.
Parameter	Specifies the dummy parameters.
Connectivity	Specifies the connectivity settings for the dummies.
Tap	(Virtuoso Advanced Node for Layout) Specifies settings for inserting tap cells.

Master

The following table describes the fields available on the *Master* tab of the Surround Modgen form.

Field	Description
Library	Specifies the library that contains the required dummy instances.

Field	Description
Cell	Specifies the cell that contains the required dummy instances.
View	Specifies the view that contains the required dummy instances. *View* lists only the `maskLayout` type views.
Browse	Opens the Library Manager from which you can select the required cellview.

Parameter

The following table describes the fields available on the *Parameter* tab of the Surround Modgen form.

Field	Description
Num Fingers	Specifies the number of fingers in each dummy. You can either type the required value in the field or choose from the following values: ■ *Same As Neighbor*: The same number of fingers as the neighboring device are created. ■ *CDF Default*: The default number of fingers, as specified in the CDF, are created.
Length	Specifies the length of dummy fingers. You can either type the required value in the field or choose *Same as Neighbor* or *CDF Default*.
Width in IC6.1.8;	Specifies the width of dummy fingers. You can either type the required value in the field or choose *Same as Neighbor* or *CDF Default*.
Number of Fins in Virtuoso Advanced Node for Layout	In Virtuoso Advanced Node for Layout, the *Width* cyclic field is replaced by the *Number of Fins* cyclic field. Valid values are the same as *Width*.

Connectivity

The following table describes the fields available on the *Connectivity* tab of the Surround Modgen form.

Field	Description
Default	Specifies the net to which all dummy terminals must be connected. If left blank (no net is selected), no terminals are created.

Tap

(Virtuoso Advanced Node for Layout) The following table describes the fields available on the *Tap* tab of the Surround Modgen form.

Field	Description
Between Rows	Specifies whether dummies are to be inserted between rows.
Starting Row Index	Specifies the first reference row for inserting dummies.
Rows to skip between insertions	Specifies the gap (number of rows) after which dummies must be inserted.
Between Columns	Specifies whether dummies are to be inserted between columns.
Starting Column Index	Specifies the first reference column for inserting dummies.
Columns to skip between insertions	Specifies the gap (number of columns) after which dummies must be inserted.
Row Configuration	Lets you specify dummy row settings.
Single Dummy Row	Inserts a single dummy row, instead of individual dummies, in each direction. The available options are: ■ *Auto* automatically calculates the number of fingers to be included in a dummy row. ■ *Specify* lets you specify the *Total Number of Fingers Per Row*.

Field	Description
Total Number of Fingers Per Row	Specifies the number of fingers to be included in a row of dummies.

Related Topics

Adding Surround Dummies

Array Assistant

Use the Array Assistant to quickly create and edit Modgens without invoking the Modgen Editor. The Array Assistant is accessible from both the Constraint Manager assistant and the Auto P&R assistant.

Tab	Description
Placement	Specifies the Modgen pattern settings.
Guard Ring	Creates guard rings around Modgens.
Routing	Defines Modgen topology patterns and routing preferences.
Reuse	Loads and saves Modgen settings to template files.

The following buttons are common to all tabs:

Button	Description
Apply	Applies the form settings to the Modgen.
Cancel	Discards settings and closes the Array Assistant.
`Modgen_Name`	Name of the Modgen for which settings are displayed in the Array Assistant.
Help	Opens help documentation for Array Assistant.

Placement

The following table describes the fields available on the *Placement* tab of the Array Assistant.

Field	Description
Pattern Preset	Specifies the pattern in which the devices are to be placed within a Modgen. You can select a pattern preset from the drop-down list, specify the base symbol pattern, or specify the base orientation pattern.

Field	Description
Presets	Lists the following pattern presets ■ *Current* uses the current Modgen pattern. ■ *Clustered* displays a pattern based on the bottom-up approach, with row-based split. ■ *Interdigitated* applies an interdigitation of 1. ■ *Compact* adjusts the pattern to achieve the maximum abutment of devices. ■ *Custom* lets you specify a base pattern for the Modgen in the adjoining text box. ■ *Resistor Topology* provides a list of predefined array topologies.
Input Base Symbol Pattern	Specifies the base symbol pattern. In Layout EXL and higher tiers, set *Pattern Preset* to *Custom* before specifying the base pattern.
Input Base Orientation Pattern	Specifies the base orientation pattern. In Layout EXL and higher tiers, set *Pattern Preset* to *Custom* before specifying the base orientation.
Aspect Ratio	Specifies the aspect ratio of the Modgen.
Active Rows, Cols	Specifies the number of rows and columns in the Modgen. You can specify the number of active rows in the Modgen. The number of columns is automatically calculated and displayed.
Total available	Specifies the total number of non-dummy array members available in the Modgen.
Best Fit	This option is available only when *Pattern* is set to *Custom* and a value is entered in the custom pattern text box. In this mode, an optimal placement of the Modgen devices is achieved according to the specified pattern.

Field	Description
Allow Placer to change Aspect Ratio	Lets the Virtuoso device-level automatic placer adjust the aspect ratio of the Modgen to achieve optimized placement for the given floorplan. The rows and columns of the Modgen are modified, while retaining the base pattern. By default, this option is selected, implying that the placer can reshape the Modgen. When *Pattern Preset* is set to *Custom*, the aspect ratio of the design is locked, and therefore, *Allow Placer to change Aspect Ratio* the option is automatically turned off.
Dummy Control	Specifies dummy parameters.
Dummy Net	Specifies the net to which the dummies are connected.
Dummy Type	Specifies the type of cells to be used as dummies. The available options are: ■ *Default*: Uses the default dummy defined in the Modgen Dummy Options form. ■ *Analog Dummy*: Inserts dummies of type analog cell. ■ *Stacked Dummy*: Insert dummies of type stack cell. This option is available only in certain advanced node flows. Environment variables: modgenDummyTypeAnalogLCV, modgenDummyTypeStackLCV, modgenDummyTypeStackNum

Field	Description
Fill Gaps with Dummies	Fills gaps between devices in the Modgen with dummy instances and adds dummy columns at locations where there are abutment breaks. Gaps caused by unabuttable devices are filled with the least number of cells. Dummies are filled in the gaps inside the Modgen, and not on the edges. This is done to ensure that the Modgen can be completely abutted and is rectangular. After running this command, making any further modifications to the grid might result in unabutted instances. Re-run the command to ensure that all devices are abutted. The *Fill Gaps with Dummies* command is command is run when you click the button, but the changes are not applied to the canvas until you click *Apply* in Array Assistant. **Note:** In Layout EXL and higher tiers, the *Fill Modgen Dummies* option is available on the *Placer* tab of the Auto P&R assistant that provides a similar functionality. This is a stand-alone command that immediately modifies either the selected Modgen figGroups or all Modgen figGroups.
Top/Bot	Specifies the number of dummy rows to be inserted at the top and bottom of the Modgen.
Left/Right	Specifies the number of dummy rows to be inserted at the left and right of the Modgen.
Transition	Inserts a dummy column after the specified number of active device columns in the Modgen. Transition dummies are columns of dummy devices inserted after the specified number of active device columns in the Modgen. For example, if *Transition* is set to `1`, every alternate column is a dummy column.
Match	Creates the same number of fins or fingers as the neighboring device.
Specify	Specifies the required number of fins or fingers.
Pattern Symbol Mapping	Lets you view and change the mapping of devices in the *Pattern* box.

Field	Description
Add Selected Schematic Instances	Adds the selected instances to the active Modgen. You can add instances from the layout or schematic cellviews.
Remove Selected Instances	Removes the selected instances from the active Modgen.
Update Sandbox Instances from Schematic	Updates the device connectivity, parameters, and multipliers from the corresponding schematic cellview.
Auto-fit Columns	Automatically fit the columns in the *Pattern Symbol Mapping* table.
Pattern	Displays the Modgen pattern derived based on the specified preset, active rows, and dummies. The pattern is derived based on the specified preset, active rows, and dummies. Use the options in the previous sections to customize the pattern as per your requirements. You can also directly edit the pattern in the *Pattern* box. For example, move instances or columns, copy and paste instances, swap instances or rows, clear cells, toggle between displays, and add dummies.
Add Row/Column	Adds a new row or column to the grid. Grid members are adjusted to fit the revised array dimensions.
Remove Row/ Column	Deletes a row or column from the grid. Grid members are adjusted to fit the revised array dimensions.
Show Symbols	Shows symbol names in the *Pattern* table.
Show Layout Names	Shows layout names in the *Pattern* table.
Show Schematic Names	Shows schematic names in the *Pattern* table.
Show Orientations	Shows device orientations in the *Pattern* table.
Show Gate Net Names	Shows gate names in the *Pattern* table.

Field	Description
Bring Power/ Ground-Connected Terminals to Row Edges	Bring all power and ground-connected terminals to their nearest row edges. The orientations of the related instances are updated based on their locations and the pattern preset is set to *Custom*. This command does not run on dummies added to the grid.
Table tab	Displays the grid pattern in a tabular format.
Text tab	Displays a textual pattern for the grid.
Save/Load CSV	Lets you save the current array settings to a comma-separated values (CSV) file or load array settings from a CSV file.
File Name	Path to the CSV file in which array settings are to be saved or from which array settings are to be loaded.
Pattern View	Specifies the format in which the pattern is to be generated in the CSV file. The valid values are: *Symbol:* The symbols to which devices are mapped to in the *Pattern Symbol Mapping* section. *Schematic Name:* The schematic counterparts of the devices.
Save	Saves the array settings to the specified CSV file.
Load	Loads array settings from the specified CSV file.
Spacing	Specifies the spacing between devices in a Modgen. An unselected state for the spacing, alignment, and abut fields implies that if the current Array Assistant settings for these properties are customized (their state is unreachable by the spacing, alignment, or abut Array Assistant fields), the custom settings remain untouched during an Array Assistant *Apply* operation. To customize Array Assistant spacing, alignment, and abut settings without using these fields, you can load an existing customized Modgen array into the Array Assistant. Unselecting the spacing, alignment, and abut fields from a selected state always resets these values to their defaults, align left for *Horizontal*, align bottom for *Vertical*, and unabut all.
Horizontal	Specifies the spacing between devices in rows.
Vertical	Specifies the spacing between devices in columns.

Field	Description
Abut	Abuts all devices in the Modgen. *Abut* is selected by default for new Modgens. If a Modgen reuse template file is loaded on the *Reuse* tab, *Abut* is updated based on the template.
Sync Row Abutment	Abuts all rows in the Modgen in a synchronized operation along a column. When this option is not selected, each row is abutted individually, without considering the other rows. *Sync Row Abutment* is selected by default for new Modgens.

Guard Ring

(Layout EXL and Higher Tiers) The following table describes the fields available on the *Guard Ring* tab of the Array Assistant.

Field	Description
Net	Specifies the net to which the guard ring has to be connected. This option is available only when *Type* is set to **MPP**, **FGR**, or **IGR**.
Type	Specifies the type of guard ring to create. The available options are: ■ *MPP*: Creates a multipath part (MPP) guard ring. ■ *FGR*: Creates a fluid guard ring (FGR). ■ *IGR*: Creates an identical guard ring (IGR).
Definition	Specifies the guard definition to be used. Select from the list of available *MPP*, *FGR*, or *IGR* definitions depending on the selected *Type*.
Guard Ring Sides	Specifies the sides on which guard rings are to be inserted. For MPP guard rings and FGRs, the available options are *Left* and *Bottom*. For IGRs, the available options are *Left*, *Right*, *Top*, and *Bottom*.

Field	Description
IGR Type (IGR Only)	Specifies the type of IGR.
	■ *Surround* creates IGRs that surround all instances. ■ *Surround+Strip* creates IGRs that surround all the instances, and inserts strips of guard ring instances between one or more Modgen rows. ■ *Grid* creates IGRs that surround every instance or group of abutted instances separately.
Strip Width in Fins (IGR Only)	Specifies the strip width.
Rows Between Strips (IGR Only)	Specifies the number of rows to be inserted between strips of IGRs.

Routing

(Layout EXL and Higher Tiers) Use the *Routing* tab of the Array Assistant to specify options to use the pin to trunk router to define Modgen topology patterns.

Tab	Description
Horizontal Routes - Pin to Trunk	Specifies routing preferences for horizontal nets.
Vertical Routes - Pin to Trunk	Specifies routing preferences for vertical nets.

Horizontal Routes - Pin to Trunk

The following table describes the fields available on the *Horizontal Routes - Pin to Trunk* tab on the *Routing* tab of the Array Assistant.

Field	Description
Net Priority	Specifies the order in which nets are to be routed.
Route	Specifies the nets to be used for routing.

Field	Description
Nets	Lists the names of nets that are present in the current Modgen.
Trunk Position	Specifies where the trunks are to be generated. The available options are: ■ *In-Between Devices*: Generates trunks in between devices in both, odd and even channels. ■ *In-Between Odd*: Generates trunks only in the odd-numbered channels in between devices. ■ *In-Between Even*: Generates trunks only in the even-numbered channels in between devices. ■ *Outside*: Generates trunks in the area outside the device row. ■ *Over Devices*: Draws horizontal routes over devices.
Trunk Layer	Specifies the trunk layers.
Trunk Width	Specifies the trunk width. By default, values from either the technology file or a predefined WSP are used.
Ext. as Vert.	Specifies whether the net is to be listed on the *Vertical Routes - Trunk to Trunk* tab. By default, the table is blank. The nets selected here are listed in the table on the *Vertical Routes - Trunk to Trunk* tab.
Trunk Settings	Specifies the trunk settings to be applied only to the nets selected for routing.
Trunk Spacing	Sets the spacing between trunks to one of the following values: ■ *Default*: Uses spacing value from either the technology file or WSP, if defined. ■ *Custom*: Lets you specify an absolute value.
Trim Trunks	Trims the ends of horizontal and vertical trunks while routing.
Share Tracks	Shares horizontal trunks that are on the same layer and have the same connectivity.
Over Devices Trunk Reference Layer	Specifies the layer in which the twigs connected to the source and drain terminals must be generated.
Over Devices First Trunk Offset	Specifies the trunk offset value. The default value is `0`.

Field	Description
Trunk Anchor	Specifies the trunk anchor point of the trunks in a device row. ■ *Top* (default): The trunks are anchored to the top-left vertex of the device row. ■ *Bottom*: The trunks are anchored to the bottom-left vertex of the device row.
Twig Specifications	Specifies twig settings for the nets selected for routing.
Gate Twig Layer Name	Specifies the layer on which twigs that are connected to the gate terminal are generated.
Gate Twig Width	Specifies the width of the gate twigs
Source/Drain Twig Layer	Specifies the layer in which the twigs connected to the source and drain terminals are generated.
Source/Drain Twig Width	Specifies the source and drain twig widths.
Min Number Cuts	Specifies he minimum number of cuts for the vias connecting the twigs to other objects. The default value is `1`.

Vertical Routes - Pin to Trunk

The following table describes the fields available on the *Vertical Routes - Pin to Trunk* tab on the *Routing* tab of the Array Assistant.

The net table settings are similar to the *Horizontal Routes - Pin to Trunk* tab.

Field	Description
Trunk to Trunk Settings	Specifies the trunk to trunk settings for vertical trunks.
Vertical Trunk Spacing	Specifies the spacing between trunks. The available options are: ■ *Default*: Uses spacing value from either the technology file or WSP, if defined. ■ *Custom*: Lets you specify an absolute value.
Trunk Reference Layer	Specifies the reference layer for calculating the trunk offset.

Field	Description
First Track Offset	Specifies the offset of the first track.
Vertical Trunk Side	Specifies the side along which vertical trunks are to be generated. The available options are: *left*, *right*, *both*, and *auto* (default).

Reuse

(Layout EXL and Higher Tiers) Use the *Reuse* tab of the Array Assistant form to load and save settings to Modgen template files.

Field	Description
Template	Provides options to load and save settings to Modgen template files.
Device Group	Specifies the devices to which the reuse template is to be applied.
File	Specifies the Modgen template file to be used.
Load	Loads setting from the selected Modgen template file.
Save	Saves the current placement setting in the specified Modgen template file.

Related Topics

Reusing Modgen Templates Using the Array Assistant

Modgen Placement Settings in the Array Assistant

Creating Guard Rings Using the Array Assistant

Defining Modgen Topology Settings Using the Array Assistant

6

Modgen Environment Variables

The following is a list of environment variables that can be used in the Virtuoso Module Generator (Modgen) environment.

- modgenAllowPinPermutation
- modgenBodyContactNetToUse
- modgenBodyContactReferenceLayer
- modgenBodyContactSep
- modgenBodyContactType
- modgenCreatePreserveLayout
- modgenCreatePreserveSpacing
- modgenCreateReferenceLPP
- modgenCreateUseDefAlignSpacing
- modgenCreateUnaboundAsDummies
- modgenDefHCSRefLayer
- modgenDefHCSRefPurpose
- modgenDefHoriAlignment
- modgenDefHoriSpacing
- modgenDefVCSRefLayer
- modgenDefVCSRefPurpose
- modgenDefVertAlignment
- modgenDefVertSpacing
- modgenDummyCell

- modgenDummyLengthOptions
- modgenDummyLengthValue
- modgenDummyLib
- modgenDummyNet
- modgenDummyNumFingersOptions
- modgenDummyNumFingersValue
- modgenDummySpecifyParams
- modgenDummyType
- modgenDummyView
- modgenDummyWidthOptions
- modgenDummyWidthValue
- modgenGuardRingSep
- modgenInterdigitationFactor
- modgenMakeMinDummies
- modgenMergeLayers
- modgenMergeWells
- modgenMPPGuardRingToUseCB
- modgenPassiveCreateOnConnectivity
- modgenPatternFormAbutAll
- modgenPhysConfigs
- modgenPlacementConstraintGroup
- modgenPreviewIgnoreXLConnVios
- modgenReferencePoint
- modgenRememberBodyContactVals
- modgenRememberDummyVals

Modgen Topology and Routing Environment Variables

- modgenPToTChannelWidth
- modgenPToTGenTrunkTwigs
- modgenPToTOnCreateFigMode
- modgenPToTPinCoverTapLowerViaPercent
- modgenPToTPinCoverTapLowerViaPercentEnable
- modgenPToTSpecifyChannelWidth
- modgenPToTPinCoverTapViaPercentEnable
- modgenPToTPinCoverViaModePercentTrunkOver
- modgenPToTPinCoverViaModePercentTrunkOverEnable
- modgenPToTPinCoverViaModeTrunkOver
- modgenPToTPinCoverViaModeTrunkOverEnable
- modgenPToTSpecifyChannelWidth
- modgenPToTSpecifyTrunk2DevSpacing
- modgenPToTTrunk2DevSpacing
- modgenPToTTrunkInsertMode
- modgenPToTTrunkLayer
- modgenPToTTrunkNets
- modgenPToTTrunkSpacing
- modgenPToTTrunkWidth
- modgenPToTTwigAbsoluteWidth
- modgenPToTTwigDirectionDown
- modgenPToTTwigDirectionLeft
- modgenPToTTwigDirectionOver
- modgenPToTTwigDirectionRight
- modgenPToTTwigDirectionUp
- modgenPToTTwigLayer

- modgenPToTTwigMinNumCuts
- modgenPToTTwigRelativeWidth
- modgenPToTTwigWidthType
- modgenPToTViaControlCutClass1
- modgenPToTViaControlCutClass2
- modgenPToTViaControlCutClassLayer
- modgenPToTViaControlCutClassName
- modgenPToTViaControlCutClassType
- modgenPToTViaControlCutClassEnable
- modgenPToTViaControlExtensionOrientEnable
- modgenPToTViaControlInline
- modgenPToTViaControlInlineEnable
- modgenPToTViaControlOffset
- modgenPToTViaControlOffsetEnable
- modgenPToTViaControlOrient
- modgenPToTViaControlOrientEnable
- modgenPToTViaWidthPercent
- modgenPToTViaWidthPercentEnable
- modgenSaveOnClosePreviewWindow
- modgenTransferDiffInstances
- modgenTransferIgnoreParamsList
- modgenUseSnapSpacing
- modgenUseIteratedAsMfactor
- modgenWidthParamProportionalToFingers
- modgenWindowConfigFile
- transparentModgenArray
- chainPermutePins

Modgen Patterns Environment Variables

- moveAsSwap
- patternDefaultDisplayMode
- patternPresetItemList
- useDummyOwner

Related Topics

Environment Variables

modgenAllowPinPermutation

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

Description

Enables automatic pin permutation within the Modgen figGroup. Pin permutation refers to the exchange of connectivity or net connections between the pins of a component.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenAllowPinPermutation")
envSetVal("layoutXL" "modgenAllowPinPermutation" 'boolean t)
```

Related Topics

Pin Permutation

modgenBodyContactNetToUse

```
layoutXL modgenBodyContactNetToUse string "net_name"
```

Description

Specifies the default net to use when creating body contacts.

The default value is "".

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenBodyContactNetToUse")
envSetVal("layoutXL" "modgenBodyContactNetToUse" 'string "myNet")
```

Related Topics

Body Contact Options Form

modgenBodyContactReferenceLayer

`layoutXL modgenBodyContactReferenceLayer string "layer_name"`

Description

Specifies the default layer for body contact creation.

The default value is "".

GUI Equivalent

Command	Modgen Editor – *Body Contacts* button on the *Modgen Placement* toolbar
Field	*Layer*

Examples

```
envGetVal("layoutXL" "modgenBodyContactReferenceLayer")
envSetVal("layoutXL" "modgenBodyContactReferenceLayer" 'string "myLayer")
```

Related Topics

Body Contact Options Form

modgenBodyContactSep

`layoutXL modgenBodyContactSep float` *`float_number`*

Description

Specifies the default separation distance for body contact creation.

The default is `0`.

GUI Equivalent

Command	Modgen Editor – *Body Contacts* button on the *Modgen Placement* toolbar
Field	*Separation*

Examples

```
envGetVal("layoutXL" "modgenBodyContactSep")
envSetVal("layoutXL" "modgenBodyContactSep" 'float 1.0)
```

Related Topics

Body Contact Options Form

modgenBodyContactType

`layoutXL modgenBodyContactType string "type"`

Description

Specifies the default type for body contact creation.

The default value is "".

GUI Equivalent

Command	Modgen Editor – *Body Contacts* button on the *Modgen Placement* toolbar
Field	*Type*

Examples

```
envGetVal("layoutXL" "modgenBodyContactType")
envSetVal("layoutXL" "modgenBodyContactType" 'string "myType")
```

Related Topics

Body Contact Options Form

modgenCreatePreserveLayout

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

Description

Specifies whether the layout instance member positions are to preserved when generating a Modgen. When the environment variable is set to `t`, the following environment variable settings are used when creating a Modgen:

```
layoutXL modgenDefHoriAlignment string "custom"
layoutXL modgenDefVertAlignment string "custom"
layoutXL modgenDefVertSpacing  float 0
layoutXL modgenDefHoriSpacing  float 0
layoutXL modgenDefVCSRefLayer  string "prBoundary"
layoutXL modgenDefHCSRefLayer  string "prBoundary"
```

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenCreatePreserveLayout")
envSetVal("layoutXL" "modgenCreatePreserveLayout" 'boolean t)
```

Related Topics

Creating a Modgen

Closing and Regenerating a Modgen

modgenCreatePreserveSpacing

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

Description

Specifies whether the relative spacing of the Modgen members must be respected when creating a Modgen from a layout selection set. The Modgen respects the existing spacing value by setting the custom spacing values between members. These spacing values may vary slightly from the spacings in the selection set because the Modgen aligns parallel member edges that are within a certain proximity and border member edges.

The default is `nil`, which means that a Modgen with minDRC spacing values is created.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenCreatePreserveSpacing")
envSetVal("layoutXL" "modgenCreatePreserveSpacing" 'boolean t)
```

Related Topics

Creating a Modgen

Closing and Regenerating a Modgen

modgenCreateReferenceLPP

```
layoutXL modgenCreateReferenceLPP string { "layer" | "layer_purpose" }
```

Description

This setting impacts how Modgens are created when `modgenCreatePreserveSpacing` is set to `t`. When a layer (or layer purpose pair) is specified, the layer's geometry is used to determine the Modgen's pattern.

The default value is "". In this state, the bounding box of the selected instances is used as reference to determine the Modgen's pattern.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenCreateReferenceLPP")
envSetVal("layoutXL" "modgenCreateReferenceLPP" 'string "Metal2")
envSetVal("layoutXL" "modgenCreateReferenceLPP" 'string "Poly drawing")
```

Related Topics

modgenCreatePreserveSpacing

Creating a Modgen

Closing and Regenerating a Modgen

modgenCreateUseDefAlignSpacing

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

Description

Applies the alignment and spacing settings to all Modgens in the current layout view. This environment variable is honored only when `modgenCreatePreserveSpacing` and `modgenCreatePreserveLayout` are set to `nil`.

The default value is `nil`, in which case the alignment and spacing values for each Modgen are to be specified individually.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenCreateUseDefAlignSpacing")
envSetVal("layoutXL" "modgenCreateUseDefAlignSpacing" 'boolean t)
```

Related Topics

modgenCreatePreserveSpacing

modgenCreatePreserveLayout

modgenCreateUnaboundAsDummies

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

Description

Specifies whether unbound layout dummies in the selected set are to be included when creating a Modgen.

The default value is `t`, where unbound layout dummies are included.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenCreateUnaboundAsDummies")
envSetVal("layoutXL" "modgenCreateUnaboundAsDummies" 'boolean nil)
```

Related Topics

Modgen Dummies

modgenDefHCSRefLayer

```
layoutXL modgenDefHCSRefLayer string { "layer_name" }
```

Description

Specifies the reference layer for horizontal spacing of devices. The spacing value is the distance between the bounding boxes of all shapes on this layer in the devices.

The default value is "".

GUI Equivalent

Command	Modgen Editor – *Member Alignment/Spacing* button on the *Modgen Placement* toolbar
Field	*Horizontal:Reference*

Examples

```
envGetVal("layoutXL" "modgenDefHCSRefLayer")
envSetVal("layoutXL" "modgenDefHCSRefLayer" 'string "Metal2")
envSetVal("layoutXL" "modgenDefHCSRefLayer" 'string "Poly")
```

Related Topics

Set Member Alignment and Spacing Form

Specifying Modgen Device Alignment and Spacing

modgenDefHCSRefPurpose

`layoutXL modgenDefHCSRefPurpose string "purpose_name"`

Description

Specifies the purpose to be set for horizontal spacing of devices. The spacing value is the distance between the bounding boxes of all shapes on this layer in the devices.

The default value is "".

GUI Equivalent

Command	Modgen Editor – *Member Alignment/Spacing* button on the *Modgen Placement* toolbar
Field	*Horizontal: Reference: Purpose*

Examples

```
envGetVal("layoutXL" "modgenDefHCSRefPurpose")
envSetVal("layoutXL" "modgenDefHCSRefPurpose" 'string "drawing")
envSetVal("layoutXL" "modgenDefHCSRefPurpose" 'string "label")
envSetVal("layoutXL" "modgenDefHCSRefPurpose" 'string "pin")
```

Related Topics

Set Member Alignment and Spacing Form

Specifying Modgen Device Alignment and Spacing

modgenDefHoriAlignment

```
layoutXL modgenDefHoriAlignment string { "left" | "right" | "center" | "custom" |
     "customRight" }
```

Description

Specifies the default value for the *Horizontal Alignment* cyclic field in the Set Member Alignment and Spacing form. If “custom” or “customRight” is specified, the value defined in the `modgenDefHoriSpacing` environment variable is used.

The default is `left`.

GUI Equivalent

Command	Modgen Editor – *Member Alignment/Spacing* button on the *Modgen Placement* toolbar
Field	*Horizontal: Alignment*

Examples

```
envGetVal("layoutXL" "modgenDefHoriAlignment")
envSetVal("layoutXL" "modgenDefHoriAlignment" 'string "right")
envSetVal("layoutXL" "modgenDefHoriAlignment" 'string "center")
```

Related Topics

modgenDefHoriSpacing

Set Member Alignment and Spacing Form

Specifying Modgen Device Alignment and Spacing

modgenDefHoriSpacing

`layoutXL modgenDefHoriSpacing float` *`float_number`*

Description

Specifies the default value for the Horizontal Spacing field in the Set Member Alignment and Spacing form. This value is used only if the `modgenDefHoriAlignment` variable is set to `custom` or `customRight`.

The default is `0.0`.

GUI Equivalent

Command	Modgen Editor – *Member Alignment/Spacing* button on the *Modgen Placement* toolbar
Field	*Horizontal: Spacing to Left*

Examples

```
envGetVal("layoutXL" "modgenDefHoriSpacing")
envSetVal("layoutXL" "modgenDefHoriSpacing" 'float 1.0)
```

Related Topics

modgenDefHoriAlignment

Set Member Alignment and Spacing Form

Specifying Modgen Device Alignment and Spacing

modgenDefVCSRefLayer

```
layoutXL modgenDefVCSRefLayer string { "layer_name" }
```

Description

Specifies the reference layer for vertical spacing of devices. The spacing value is the distance between the bounding boxes of all shapes on this layer in the devices.

The default value is "".

GUI Equivalent

Command	Modgen Editor – *Member Alignment/Spacing* button on the *Modgen Placement* toolbar
Field	*Vertical: Reference*

Examples

```
envGetVal("layoutXL" "modgenDefVCSRefLayer")
envSetVal("layoutXL" "modgenDefVCSRefLayer" 'string "Metal2")
envSetVal("layoutXL" "modgenDefVCSRefLayer" 'string "Poly")
```

Related Topics

Set Member Alignment and Spacing Form

Specifying Modgen Device Alignment and Spacing

modgenDefVCSRefPurpose

```
layoutXL modgenDefVCSRefPurpose string "purpose_name"
```

Description

Specifies the purpose to be set for vertical spacing of devices. The spacing value is the distance between the bounding boxes of all shapes on this layer in the devices.

The default value is "".

GUI Equivalent

Command	Modgen Editor – *Member Alignment/Spacing* button on the *Modgen Placement* toolbar
Field	*Vertical: Reference: Purpose*

Examples

```
envGetVal("layoutXL" "modgenDefVCSRefPurpose")
envSetVal("layoutXL" "modgenDefVCSRefPurpose" 'string "drawing")
envSetVal("layoutXL" "modgenDefVCSRefPurpose" 'string "label")
envSetVal("layoutXL" "modgenDefVCSRefPurpose" 'string "pin")
```

Related Topics

Set Member Alignment and Spacing Form

Specifying Modgen Device Alignment and Spacing

modgenDefVertAlignment

```
layoutXL modgenDefVertAlignment string { "top" | "bottom" | "center" | "custom" |
     "customTop" }
```

Description

Specifies the default value for the Vertical Alignment cyclic field in the Set Member Alignment and Spacing form. If “`custom`” or “`customTop`” is specified, the value defined in the `modgenDefVertSpacing` environment variable is used.

The default value is `top`.

GUI Equivalent

Command	Modgen Editor – *Member Alignment/Spacing* button on the *Modgen Placement* toolbar
Field	*Vertical: Alignment*

Examples

```
envGetVal("layoutXL" "modgenDefVertAlignment")
envSetVal("layoutXL" "modgenDefVertAlignment" 'string "bottom")
envSetVal("layoutXL" "modgenDefVertAlignment" 'string "center")
```

Related Topics

modgenDefVertSpacing

Set Member Alignment and Spacing Form

Specifying Modgen Device Alignment and Spacing

modgenDefVertSpacing

`layoutXL modgenDefVertSpacing float` *`float_number`*

Description

Specifies the default value for the Vertical Spacing field in the Set Member Alignment and Spacing form. This value is used only if the `modgenDefVertAlignment` variable is set to `custom` or `customTop`.

The default value is `0.0`.

GUI Equivalent

Command	Modgen Editor – *Member Alignment/Spacing* button on the *Modgen Placement* toolbar
Field	*Vertical: Spacing to Bottom*

Examples

```
envGetVal("layoutXL" "modgenDefVertSpacing")
envSetVal("layoutXL" "modgenDefVertSpacing" 'float 1.0)
```

Related Topics

modgenDefVertAlignment

Set Member Alignment and Spacing Form

Specifying Modgen Device Alignment and Spacing

modgenDummyCell

```
layoutXL modgenDummyCell string "cellName"
```

Description

Specifies the default cell to use when creating custom dummy devices.

The default value is "".

GUI Equivalent

Command	Modgen Editor – *Dummies* button on the *Modgen Placement* toolbar
Field	*Default Dummy: DummyCell*

Examples

```
envGetVal("layoutXL" "modgenDummyCell")
envSetVal("layoutXL" "modgenDummyCell" 'string "myCell")
```

Related Topics

Dummy Options Form

Adding and Deleting Dummies in the Modgen Editor

modgenDummyLengthOptions

```
layoutXL modgenDummyLengthOptions cyclic { "CDF Default" | "Same As Neighbor" |
     "Specify" }
```

Description

Specifies valid values for the dummy *Length* field.

- `CDF Default` – The default finger length, as specified in the CDF, is considered. This is the default value.
- `Same As Neighbor` – The length of fingers of the neighboring device is considered.
- `Specify` – You can specify the length of fingers.

The default value is "", in which case the value specified in the technology file is honored.

GUI Equivalent

Command	Modgen Editor – *Dummies* button on the *Modgen Placement* toolbar
Field	*Dummy Type: Dummy Parameters: Length*

Examples

```
envGetVal("layoutXL" "modgenDummyLengthOptions")
envSetVal("layoutXL" "modgenDummyLengthOptions" 'cyclic "Same as Neighbor")
envSetVal("layoutXL" "modgenDummyLengthOptions" 'cyclic "Specify")
```

Related Topics

Dummy Options Form

Adding and Deleting Dummies in the Modgen Editor

modgenDummyLengthValue

`layoutXL modgenDummyLengthValue string "Default_Length"`

Description

Specifies default value for the dummy *Length* field. This variable can be used only if the modgenDummyWidthOptions environment variable is set to `Specify`.

GUI Equivalent

Command	Modgen Editor – *Dummies icon on the Modgen Placement toolbar*
Field	*Length* (Dummy Options Form)

Examples

```
envGetVal("layoutXL" "modgenDummyLengthValue")
envSetVal("layoutXL" "modgenDummyLengthValue" 'string "2")
```

Related Topics

Dummy Options Form

modgenDummyWidthOptions

modgenDummyLib

```
layoutXL modgenDummyLib string "library_name"
```

Description

Specifies the default library to use when creating custom dummy devices.

The default value is "".

GUI Equivalent

Command	Modgen Editor – *Dummies* button on the Modgen Placement toolbar
Field	*Default Dummy: DummyLibrary*

Examples

```
envGetVal("layoutXL" "modgenDummyLib")
envSetVal("layoutXL" "modgenDummyLib" 'string "myLib")
```

Related Topics

Dummy Options Form

Adding and Deleting Dummies in the Modgen Editor

modgenDummyNet

```
layoutXL modgenDummyNet string "net_name"
```

Description

Specifies the default net to use when creating dummy devices.

The default value is "".

GUI Equivalent

Command	Modgen Editor – *Dummies* button on the Modgen Placement toolbar
Field	*Dummy Net: Default*

Examples

```
envGetVal("layoutXL" "modgenDummyNet")
envSetVal("layoutXL" "modgenDummyNet" 'string "myNet")
```

Related Topics

Dummy Options Form

Adding and Deleting Dummies in the Modgen Editor

modgenDummyNumFingersOptions

```
layoutXL modgenDummyNumFingersOptions cyclic { "CDF Default" | "Same As Neighbor"
     | "Specify" }
```

Description

Specifies valid values for the dummy *Number of Fingers* field.

- `CDF Default` – The default number of fingers, as specified in the CDF, is created. This is the default value.
- `Same As Neighbor` – The same number of fingers as the neighboring device is created.
- `Specify` – You can specify the number of fingers to be created in the box beside the *Number of Fingers* cyclic field.

The default value is "", in which case the value specified in the technology file is honored.

GUI Equivalent

Command	Modgen Editor – *Dummies* button on the Modgen Placement toolbar
Field	*Dummy Type: Dummy Parameters: Number of Fingers*

Examples

```
envGetVal("layoutXL" "modgenDummyNumFingersOptions")
envSetVal("layoutXL" "modgenDummyNumFingersOptions" 'cyclic "Same as Neighbor")
envSetVal("layoutXL" "modgenDummyNumFingersOptions" 'cyclic "Specify")
```

Related Topics

Dummy Options Form

Adding and Deleting Dummies in the Modgen Editor

modgenDummyNumFingersValue

`layoutXL modgenDummyNumFingersValue string "Number_of_Fingers"`

Description

Specifies valid values for the dummy *Number of Fingers* field. This `modgenDummyNumFingersOptions` variable can be used only if the environment variable is set to `Specify`.

The default value is "".

GUI Equivalent

Command	Modgen Editor – *Dummies* button on the Modgen Placement toolbar
Field	*Number of Fingers*

Examples

```
envGetVal("layoutXL" "modgenDummyNumFingersValue")
envSetVal("layoutXL" "modgenDummyNumFingersValue" 'string "2")
```

Related Topics

modgenDummyNumFingersOptions

Dummy Options Form

Adding and Deleting Dummies in the Modgen Editor

modgenDummySpecifyParams

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

Description

Specifies whether the number of fingers, length, and width values for dummies need to be specified. When set to `nil`, the default values are as specified in the `modgenMakeMinDummies` environment variable.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenDummySpecifyParams")
envSetVal("layoutXL" "modgenDummySpecifyParams" 'boolean t)
```

Related Topics

modgenMakeMinDummies

modgenDummyType

```
layoutXL modgenDummyType cyclic { "neighbor" | "default" | "copy" }
```

Description

Specifies the default type for dummy devices.

The default type is `neighbor`, where the type of device created is dependent on the location of the dummy and the setting of the `modgenMakeMinDummies` environment variable.

If the default type is `default`, then the type of device can be specified using `modgenDummyLib`, `modgenDummyCell`, and `modgenDummyView` variables.

If the default type is `copy`, then identical dummies of the selected instances are created. In this mode, the dummy parameters and default values of the source instances are used. Different values cannot be specified.

GUI Equivalent

Command	Modgen Editor – *Dummies* button on the Modgen Placement toolbar
Field	*Dummy Type: Type*

Examples

```
envGetVal("layoutXL" "modgenDummyType")
envSetVal("layoutXL" "modgenDummyType" 'cyclic "neighbor")
envSetVal("layoutXL" "modgenDummyType" 'cyclic "copy")
```

Related Topics

modgenMakeMinDummies

modgenDummyLib

modgenDummyCell

modgenDummyView

Dummy Options Form

Adding and Deleting Dummies in the Modgen Editor

modgenDummyView

```
layoutXL modgenDummyView string "viewName"
```

Description

Specifies the default cell view to use when creating custom dummy devices.

The default value is "".

GUI Equivalent

Command	Modgen Editor – *Dummies* button on the Modgen Placement toolbar
Field	*Default Dummy: DummyView*

Examples

```
envGetVal("layoutXL" "modgenDummyView")
envSetVal("layoutXL" "modgenDummyView" 'string "myView")
```

Related Topics

Dummy Options Form

Adding and Deleting Dummies in the Modgen Editor

modgenDummyWidthOptions

```
layoutXL modgenDummyWidthOptions cyclic { "CDF Default" | "Same As Neighbor" |
     "Specify" }
```

Description

Specifies the default value for the dummy *Number of Fins* field. Valid values are:

- `CDF Default` – The default finger width, as specified in the CDF, is considered.
- `Same As Neighbor` – The width of fingers of the neighboring device is considered.
- `Specify` – You can specify the width of fingers in the box beside the Width cyclic field.

The default value is "".

GUI Equivalent

Command	Modgen Editor – *Dummies* button on the Modgen Placement toolbar
Field	*Number of Fins*

Examples

```
envGetVal("layoutXL" "modgenDummyWidthOptions")
envSetVal("layoutXL" "modgenDummyWidthOptions" 'cyclic "Same as Neighbor")
envSetVal("layoutXL" "modgenDummyWidthOptions" 'cyclic "Specify")
```

Related Topics

Dummy Options Form

Adding and Deleting Dummies in the Modgen Editor

modgenDummyWidthValue

`layoutXL modgenDummyWidthValue string "Default_Width"`

Description

Specifies the default values for the dummy *Number of Fins* field. This variable can be used only if the `modgenDummyWidthOptions` environment variable is set to `Specify`.

The default value is "".

GUI Equivalent

Command	Modgen Editor – *Dummies* button on the Modgen Placement toolbar
Field	*Number of Fins*

Examples

```
envGetVal("layoutXL" "modgenDummyWidthValue")
envSetVal("layoutXL" "modgenDummyWidthValue" 'string "2")
```

Related Topics

modgenDummyWidthOptions

Dummy Options Form

Adding and Deleting Dummies in the Modgen Editor

modgenGuardRingSep

`layoutXL modgenGuardRingSep float` *`float_number`*

Description

Specifies the default separation distance for all sides of a guard ring.

The default value is `0.0`.

GUI Equivalent

Command	Modgen Editor – *Body Contacts* button on the Modgen Placement toolbar
Field	Separation

Examples

```
envGetVal("layoutXL" "modgenGuardRingSep")
envSetVal("layoutXL" "modgenGuardRingSep" 'float 1.0)
```

Related Topics

Body Contact Options Form

Body Contacts in Modgens

modgenInterdigitationFactor

`layoutXL modgenInterdigitationFactor int` *`integer_number`*

Description

Specifies the interdigitation pattern for the device, which is specific to the current module. In case no interdigitation in the Modgen is required, specify `0` as the interdigitation value.

The default value is `1`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenInterdigitationFactor")
envSetVal("layoutXL" "modgenInterdigitationFactor" 'int 0)
envSetVal("layoutXL" "modgenInterdigitationFactor" 'int 1)
```

Related Topics

modgenMakeMinDummies

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

Description

Determines the behavior of the Modgen when dummies are added to MOSFETs or resistors.

If true, the following happens when a neighbor dummy is added to a MOSFET:

- If the dummy's location is on the right or left, a single finger device is created to serve as the dummy.
- If the dummy's location is on the top or bottom, a minimum width device with the same number of fingers as the neighbor is created to serve as the dummy.

If true, the following happens when a neighbor dummy is added to a resistor:

- If the dummy's location is on the right or left, a single segment device is created to serve as the dummy.
- If the dummy's location is on the top or bottom, a minimum width device with the same number of segments as the neighbor is created to serve as the dummy.

In this case, the parameters for fingers (`lxFingeringNames`), width (`transistorWidthParamNames`), and s-factor (`sfactorNames`) are used to create the device.

If it is set to false, then the dummy is created as identical to the neighboring device (same number of fingers or segments).

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenMakeMinDummies")
envSetVal("layoutXL" "modgenMakeMinDummies" 'boolean nil)
```

Related Topics

Dummy Options Form

Adding and Deleting Dummies in the Modgen Editor

modgenMergeLayers

`layoutXL modgenMergeLayers string "layer_names"`

Description

Specifies the layers that need to be merged in the Modgens that have been created interactively. The string value is delimited by a colon (:), where the characters on both sides of the delimiter are layer names. Other valid values are `well` and `default`.

The default value is `default`.

GUI EquivalentExamples

Command	Modgen Editor – *Merge Layers* button on the Modgen Placement toolbar
Field	*Select merge layers for 'mergeLayer'*

```
envGetVal("layoutXL" "modgenMergeLayers")
envSetVal("layoutXL" "modgenMergeLayers" 'string "Nimp:Pimp")
```

Related Topics

Select Merge Layers Form

Merging Layers

modgenMergeWells

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

Description

Specifies whether shared well layer shapes can be created for member instances that have the same well layer.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenMergeWells")
envSetVal("layoutXL" "modgenMergeWells" 'boolean t)
```

Related Topics

Select Merge Layers Form

Merging Layers

modgenMPPGuardRingToUseCB

```
layoutXL modgenMPPGuardRingToUseCB string { "instance_names" }
```

Description

Sets the default MPP guard ring when the MPP Guard Ring Options form is launched.

The environment variable accepts a user-defined callback function with a list of instances in the Modgen, and returns the name of the valid MPP guard ring that surrounds the Modgen.

The default value is "".

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenMPPGuardRingToUseCB")
envSetVal("layoutXL" "modgenMPPGuardRingToUseCB" 'string "mgGRProc")
```

Related Topics

Guard Ring Options Form

Modgen Guard Rings

modgenPassiveCreateOnConnectivity

```
layoutXL modgenPassiveCreateOnConnectivity cyclic { "none" | "zigzag" }
```

Description

(Layout EXL and Higher Tiers) Specifies whether the Modgen placement is to be reordered to one of the array topologies.

By default, a suitable array topology is automatically selected. If set to `none` (default), the Modgen placement is based on the current instances in the canvas.

This option is available only in certain advanced node flows for Modgen arrays with all Modgen members registered as resistors, capacitors, or inductors.

The default value is "".

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPassiveCreateOnConnectivity")
envSetVal("layoutXL" "modgenPassiveCreateOnConnectivity" 'cyclic "zigzag")
```

Related Topics

Creating a Modgen

modgenPatternFormAbutAll

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

Description

Sets the default state of the *Abut All* parameter for Modgen patterns. The last used state of this option is saved as the default for subsequent uses.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPatternFormAbutAll")
envSetVal("layoutXL" "modgenPatternFormAbutAll" 'boolean t)
```

Related Topics

modgenPhysConfigs

```
layoutXL modgenPhysConfigs string { "physConfig" }
```

Description

Specifies the physConfig view name to be used by the Modgen when run from the schematic, if the physConfig view exists with this name.

The default value is "".

When none of the specified physConfigs exists, the Modgen creates a temporary default physConfig and uses it. You can specify multiple physConfig view names by separating them by a space.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPhysConfigs")
envSetVal("layoutXL" "modgenPhysConfigs" 'string "physConfig")
envSetVal("layoutXL" "modgenPhysConfigs" 'string "viewName1 viewName2")
envSetVal("layoutXL" "modgenPhysConfigs" 'string "")
```

Related Topics

Creating a Modgen

modgenPlacementConstraintGroup

```
layoutXL modgenPlacementConstraintGroup string { "Modgen_ConstraintGroup_Name"
     | t }
```

Description

Specifies the name of the tool constraint group to be used when creating, generating, or updating Modgens. The tool constraint group comprises a set of process rules that control creation of Modgens. If you change the `modgenPlacementConstraintGroup` setting before regenerating a Modgen, then the new Constraint Group is automatically used, potentially resulting in a different placement of the Modgen.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPlacementConstraintGroup")
envSetVal("layoutXL" "modgenPlacementConstraintGroup" 'string "My_ConstraintGp")
```

Related Topics

Creating a Modgen

modgenPreviewIgnoreXLConnVios

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

Description

Selectively reports or hides the number of open violations in the design in the Modgen previewer window.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPreviewIgnoreXLConnVios")
envSetVal("layoutXL" "modgenPreviewIgnoreXLConnVios" 'boolean t)
```

Related Topics

Open Modgens in the Modgen Editor

Closing and Regenerating a Modgen

modgenReferencePoint

```
layoutXL modgenReferencePoint cyclic { "center" | "interactive" | "lowerLeft" |
     "upperRight" | "upperLeft" | "lowerRight" }
```

Description

Specifies the reference point for aligning Modgens when they are either created or repositioned, for example when a Modgen is moved or when the number of rows in a Modgen is increased. Valid values are:

- `center` – Aligns the center of the new or edited Modgen with the center of the existing Modgen.
- `interactive` – Aligns the center of the new or edited Modgen with the center of the existing Modgen. However, the center excludes any user-moved instance.

  Example:

  Instances in an existing Modgen are positioned as following:

  ```
  M1

  M2   M3  M4
  ```

  Instance `M1` is moved to the right of the three instances. The revised alignment of the Modgen is follows:

  ```
  M2   M3   M4   M1
  ```

  When set to `center`, all instances, `M1 M2 M3 M4`, are considered for determining the center of the Modgen.

  When set to `interactive`, only instances, `M2 M3 M4`, are considered.
- `lowerLeft` – Aligns the lower left corner of the new or edited Modgen with that of the existing Modgen.
- `upperRight` – Aligns the upper right corner of the new or edited Modgen with that of the existing Modgen.
- `upperLeft` – Aligns the upper left corner of the new or edited Modgen with that of the existing Modgen.
- `lowerRight` – Aligns the lower right corner of the new or edited Modgen with that of the existing Modgen.

The default is `interactive`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenReferencePoint")
envSetVal("layoutXL" "modgenReferencePoint" 'cyclic "center")
envSetVal("layoutXL" "modgenReferencePoint" 'cyclic "lowerLeft")
```

Related Topics

Creating a Modgen

Modgen On-Canvas Commands

modgenRememberBodyContactVals

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

Description

Specifies whether the values from the body contacts form should be saved.

The default is `nil`.

GUI Equivalent

Command	Modgen Editor – *Body Contacts* button on the Modgen Placement toolbar
Field	*Remember Values*

Examples

```
envGetVal("layoutXL" "modgenRememberBodyContactVals")
envSetVal("layoutXL" "modgenRememberBodyContactVals" 'boolean t)
```

Related Topics

Body Contact Options Form

Body Contacts in Modgens

modgenRememberDummyVals

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

Description

Specifies whether the values from the Dummy Options form should be saved.

The default is `nil`.

GUI Equivalent

Command	Modgen Editor – *Dummies* button on the Modgen Placement toolbar
Field	*Remember Values*

Examples

```
envGetVal("layoutXL" "modgenRememberDummyVals")
envSetVal("layoutXL" "modgenRememberDummyVals" 'boolean t)
```

Related Topics

Dummy Options Form

Modgen Dummies

modgenPToTChannelWidth

`layoutXL modgenPToTChannelWidth float` *`float_number`*

Description

Specifies the width of channel nets in the current Modgen. This option can be set only if the `modgenPToTSpecifyChannelWidth` environment variable is set to `t`. The default value is `0.0`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTChannelWidth")
envSetVal("layoutXL" "modgenPToTChannelWidth" 'float 1.2)
```

Related Topics

modgenPToTSpecifyChannelWidth

modgenPToTGenTrunkTwigs

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

Description

Controls whether topological twigs, which connect trunks to channel objects (instances and body contacts), need to be generated when creating trunks.

The default is `t`, and so twigs will be generated. The options in the *Twig Options* section are enabled.

When set to `nil`, twigs will not be generated. The options in the *Twig Options* section are disabled.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTGenTrunkTwigs")
envSetVal("layoutXL" "modgenPToTGenTrunkTwigs" 'boolean nil)
```

Related Topics

modgenPToTSpecifyChannelWidth

modgenPToTOnCreateFigMode

```
layoutXL modgenPToTOnCreateFigMode cyclic { "ManualRoute" | "Ignore" }
```

Description

Specifies whether the manual editing mode should be turned on. When set to `ManualRoute`, the manual editing mode is turned on. Shapes and vias added in this mode are honored.

When set to `Ignore`, any new shape added is immediately deleted, and an appropriate message is displayed.

- `ManualRoute`: Manual editing mode is turned on.
- `Ignore`: Manual editing mode is turned off.

The default value is `ManualRoute`

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTOnCreateFigMode")
envSetVal("layoutXL" "modgenPToTOnCreateFigMode" 'cyclic "Ignore")
```

modgenPToTPinCoverTapLowerViaPercent

`layoutXL modgenPToTPinCoverTapLowerViaPercent int` *`integer_number`*

Description

Specifies the percentage of a pin shape to be covered by the lower layers of the tapping vias. This value applies to all the layers expect the top-most via. The percentage is measured in the direction perpendicular to the trunk.

The default value is `100`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTPinCoverTapLowerViaPercent")
envSetVal("layoutXL" "modgenPToTPinCoverTapLowerViaPercent" 'int 150)
envSetVal("layoutXL" "modgenPToTPinCoverTapLowerViaPercent" 'int 133)
```

modgenPToTPinCoverTapLowerViaPercentEnable

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

Description

Specifies whether the percentage of a pin shape, which is to be covered by the lower layers of the tapping vias, can be specified.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTPinCoverTapLowerViaPercentEnable")
envSetVal("layoutXL" "modgenPToTPinCoverTapLowerViaPercentEnable" 'boolean t)
```

modgenPToTPinCoverTapViaPercent

`layoutXL modgenPToTPinCoverTapViaPercent int` *`integer_number`*

Description

Specifies the percentage of a pin shape to be covered by the tapping via on all layers. The percentage is measured in the direction perpendicular to the trunk.

The default value is `100`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTPinCoverTapViaPercent")
envSetVal("layoutXL" "modgenPToTPinCoverTapViaPercent" 'int 150)
envSetVal("layoutXL" "modgenPToTPinCoverTapViaPercent" 'int 133)
```

modgenPToTPinCoverTapViaPercentEnable

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

Description

Specifies whether the percentage of the pin shape to be covered by the tapping via on all layers can be specified.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTPinCoverTapViaPercentEnable")
envSetVal("layoutXL" "modgenPToTPinCoverTapViaPercentEnable" 'boolean t)
```

modgenPToTPinCoverViaModePercentTrunkOver

`layoutXL modgenPToTPinCoverViaModePercentTrunkOver int` *`integer_number`*

Description

Controls the percentage of a trunk or a pin to be covered by the via stack.

The default value is `100`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTPinCoverViaModePercentTrunkOver")
envSetVal("layoutXL" "modgenPToTPinCoverViaModePercentTrunkOver" 'int 150)
envSetVal("layoutXL" "modgenPToTPinCoverViaModePercentTrunkOver" 'int 133)
```

modgenPToTPinCoverViaModePercentTrunkOverEnable

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

Description

Specifies whether the percentage of a trunk or a pin to be covered by the via stack can be specified.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTPinCoverViaModePercentTrunkOverEnable")
envSetVal("layoutXL" "modgenPToTPinCoverViaModePercentTrunkOverEnable" 'boolean
t)
```

modgenPToTPinCoverViaModeTrunkOver

```
layoutXL modgenPToTPinCoverViaModeTrunkOver cyclic { "matchTrunk" | "full" |
     "fullExceptTop" }
```

Description

Specifies the via mode to be applied when a trunk or a pin is covered by a via stack. The following options are available:

- `matchTrunk`: Covers the width of the trunk.
- `full`: Covers the entire pin shape.
- `fullExceptTop`: Covers the entire trunk on the top via layer and the entire pin on the lower via layers.

The default value is `matchTrunk`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTPinCoverViaModeTrunkOver")
envSetVal("layoutXL" "modgenPToTPinCoverViaModeTrunkOver" 'cyclic "full")
envSetVal("layoutXL" "modgenPToTPinCoverViaModeTrunkOver" 'cyclic
"fullExceptTop")
```

modgenPToTPinCoverViaModeTrunkOverEnable

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

Description

Specifies whether the via mode, which is to be applied when the trunk or a pin is covered by the via stack, can be specified.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTPinCoverViaModeTrunkOverEnable")
envSetVal("layoutXL" "modgenPToTPinCoverViaModeTrunkOverEnable" 'boolean t)
```

modgenPToTSpecifyChannelWidth

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

Description

Specifies whether the width of channel nets can be specified for the current Modgen.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTSpecifyChannelWidth")
envSetVal("layoutXL" "modgenPToTSpecifyChannelWidth" 'boolean t)
```

modgenPToTSpecifyTrunk2DevSpacing

```
layoutXL modgenPToTSpecifyTrunk2DevSpacing string { "Center Trunks in Channel" |
     nil }
```

Description

Specifies whether trunks must be centered in the channel such that their distances from devices is the same from both, the top and bottom edges.

The default value is `Center Trunks in Channel`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTSpecifyTrunk2DevSpacing")
envSetVal("layoutXL" "modgenPToTSpecifyTrunk2DevSpacing" 'string "nil")
```

modgenPToTTrimTrunks

```
layoutXL modgenPToTTrimTrunks cyclic { "None" | "Both" | "Left/Bottom" | "Right/
     Top" }
```

Description

Specifies the side from which trunks need to be trimmed while routing. The following options are available:

- `None`: Trunks are not trimmed.
- `Both`: Trunks are trimmed from both the ends.
- `Left/Bottom`: Trunks are trimmed along the left and bottom edges. Trunks are trimmed to the last twig connection.
- `Right/Top`: Trunks are trimmed along the right and top edges. Trunks are trimmed to the last twig connection.

The default value is `None`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTTrimTrunks")
envSetVal("layoutXL" "modgenPToTTrimTrunks" 'cyclic "Both")
envSetVal("layoutXL" "modgenPToTTrimTrunks" 'cyclic "Left/Bottom")
```

modgenPToTTrunk2DevSpacing

`layoutXL modgenPToTTrunk2DevSpacing float` *`float_number`*

Description

Specifies the trunk-to-device spacing value for the current Modgen.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTTrunk2DevSpacing")
envSetVal("layoutXL" "modgenPToTTrunk2DevSpacing" 'float 1.2)
```

modgenPToTTrunkInsertMode

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

Description

With this option set to `t`, whenever a trunk is deleted, the adjacent (existing) trunks are moved to ensure that the relative distances between them remain the same.

When the option is set to `nil` (default state), the adjacent (existing) trunks remain at their original positions, irrespective of whether trunks are deleted.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTTrunkInsertMode")
envSetVal("layoutXL" "modgenPToTTrunkInsertMode" 'boolean t)
```

modgenPToTTrunkLayer

`layoutXL modgenPToTTrunkLayer string "layer_name"`

Description

Specifies the name of the layer on which trunks need to be created for the current Modgen. The layer name must be enclosed in quotation marks; for example, `"metal1"`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTTrunkLayer")
envSetVal("layoutXL" "modgenPToTTrunkLayer" 'string "metal1")
```

modgenPToTTrunkNets

`layoutXL modgenPToTTrunkNets string "net_name"`

Description

Specifies the names of the nets to which trunks need to be added. The net name must be enclosed in quotation marks; for example, `"(\"GND\")"`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTTrunkNet")
envSetVal("layoutXL" "modgenPToTTrunkNet" 'string "(\"GND\")")
envSetVal("layoutXL" "modgenPToTTrunkNet" 'string "(\"GND\" \"INP\" \"OUT\")")
```

modgenPToTTrunkSpacing

`layoutXL modgenPToTTrunkSpacing float` *`float_number`*

Description

Specifies the trunk spacing for the current Modgen.

The default value is `0.0`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTTrunkSpacing")
envSetVal("layoutXL" "modgenPToTTrunkSpacing" 'float 1.2)
```

modgenPToTTrunkWidth

`layoutXL modgenPToTTrunkWidth float` *`float_number`*

Description

Specifies the width of trunks in the current Modgen.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTTrunkWidth")
envSetVal("layoutXL" "modgenPToTTrunkWidth" 'float 1.2)
```

modgenPToTTwigAbsoluteWidth

`layoutXL modgenPToTTwigAbsoluteWidth float` *`float_number`*

Description

Specifies the width of twigs in the current Modgen. This option can be used only if the modgenPToTTwigWidthType environment variable is set to `Absolute Width`. The default value is `0.0`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTTwigAbsoluteWidth")
envSetVal("layoutXL" "modgenPToTTwigAbsoluteWidth" 'float 1.2)
```

Related Topics

modgenPToTTwigWidthType

modgenPToTTwigDirectionDown

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

Description

Specifies the direction (down) along which twigs should be searched, starting from the selected source. It also indicates the direction in which twigs must be created.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTTwigDirectionDown")
envSetVal("layoutXL" "modgenPToTTwigDirectionDown" 'boolean t)
```

modgenPToTTwigDirectionLeft

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

Description

Specifies the direction (left) along which twigs should be searched, starting from the selected source. It also indicates the direction in which twigs must be created.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTTwigDirectionLeft")
envSetVal("layoutXL" "modgenPToTTwigDirectionLeft" 'boolean t)
```

modgenPToTTwigDirectionOver

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

Description

Specifies that for a trunk located over the device or pin, a twig must be created, which connects the trunk to an overlapping pin on the same net.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTTwigDirectionOver")
envSetVal("layoutXL" "modgenPToTTwigDirectionOver" 'boolean t)
```

modgenPToTTwigDirectionRight

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

Description

Specifies the direction (right) along which twigs should be searched, starting from the selected source. It also indicates the direction in which twigs must be created.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTTwigDirectionRight")
envSetVal("layoutXL" "modgenPToTTwigDirectionRight" 'boolean t)
```

modgenPToTTwigDirectionUp

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

Description

Specifies the direction (up) along which twigs should be searched, starting from the selected source. It also indicates the direction in which twigs must be created.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTTwigDirectionUp")
envSetVal("layoutXL" "modgenPToTTwigDirectionUp" 'boolean t)
```

modgenPToTTwigLayer

`layoutXL modgenPToTTwigLayer string "layer_name"`

Description

Specifies the name of the layer on which twigs need to be created for the current Modgen. The layer name must be enclosed in quotation marks; for example, `"metal1"`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTTwigLayer")
envSetVal("layoutXL" "modgenPToTTwigLayer" 'string "metal1")
```

modgenPToTTwigMinNumCuts

`layoutXL modgenPToTTwigMinNumCuts int` *`integer_number`*

Description

Specifies the minimum number of cuts for twigs in the current Modgen. Value must be a non-zero, positive integer that specifies the number of cuts for twigs. The default value is `1`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTTwigMinNumCuts")
envSetVal("layoutXL" "modgenPToTTwigMinNumCuts" 'int 2)
```

modgenPToTTwigRelativeWidth

`layoutXL modgenPToTTwigRelativeWidth int` *`integer_number`*

Description

Specifies the relative width of twigs in the current Modgen. This option can be used only if the `modgenPToTTwigWidthType` environment variable is set to `Relative Width (Pin %)`. Value must be a non-zero, positive integer that specifies the relative width of twigs.

Default value is `100`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTTwigRelativeWidth")
envSetVal("layoutXL" "modgenPToTTwigRelativeWidth" 'int 70)
```

Related Topics

modgenPToTTwigWidthType

modgenPToTTwigWidthType

```
layoutXL modgenPToTTwigWidthType cyclic { "MinWidth" | "Absolute Width" | "Relative
     Width (Pin %)" }
```

Description

Specifies the mode in which the twig width can be specified for the current Modgen. This environment variable can take three values.

- `MinWidth`: Minimum width of twigs as per the defaults is used if this option is specified.
- `Absolute Width`: Absolute width of twigs can be specified (using the `modgenPToTTwigAbsoluteWidth` environment variable), if this option is specified.
- `Relative Width (Pin %)`: Relative width of twigs can be specified (using the `modgenPToTTwigRelativeWidth` environment variable), if this option is specified.

The default value is `MinWidth`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTTwigWidthType")
envSetVal("layoutXL" "modgenPToTTwigWidthType" 'cyclic "Absolute Width")
envSetVal("layoutXL" "modgenPToTTwigWidthType" 'cyclic "Relative Width (Pin %)")
```

Related Topics

modgenPToTTwigAbsoluteWidth

modgenPToTTwigRelativeWidth

modgenPToTViaControlCutClass1

`layoutXL modgenPToTViaControlCutClass1 float` *`float_number`*

Description

Stores the last used cut class width value.

The default value is `0.0`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTViaControlCutClass1")
envSetVal("layoutXL" "modgenPToTViaControlCutClass1" 'float 1.2)
```

modgenPToTViaControlCutClass2

`layoutXL modgenPToTViaControlCutClass2 float` *`float_number`*

Description

Stores the last used cut class height value.

The default value is `0.0`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTViaControlCutClass2")
envSetVal("layoutXL" "modgenPToTViaControlCutClass2" 'float 1.2)
```

modgenPToTViaControlCutClassLayer

```
layoutXL modgenPToTViaControlCutClassLayer string "layer_name"
```

Description

Stores the layer name of the last used cut class.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTViaControlCutClassLayer")
envSetVal("layoutXL" "modgenPToTViaControlCutClassLayer" 'string "metal1")
```

modgenPToTViaControlCutClassName

`layoutXL modgenPToTViaControlCutClassName string "cutClass_name"`

Description

Stores the name of the last used cut class.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTViaControlCutClassName")
envSetVal("layoutXL" "modgenPToTViaControlCutClassName" 'string "CutClass1")
```

modgenPToTViaControlCutClassType

`layoutXL modgenPToTViaControlCutClassType string "cutClass_type"`

Description

Specifies how the last used cut class must be stored. Valid values are the following:

- `By Size`: Lets you specify the width and height of the associated cut classes.
- `By Name`: Lets you specify the name and layer of the associated cut classes.

The default value is `By Size`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTViaControlCutClassType")
envSetVal("layoutXL" "modgenPToTViaControlCutClassType" 'string "By Name")
```

modgenPToTViaControlCutClassEnable

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

Description

When set to `t`, enables the *Cut Class Width* and *Height* options on the *Via Controls* tab.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTViaControlCutClassEnable")
envSetVal("layoutXL" "modgenPToTViaControlCutClassEnable" 'boolean t)
```

modgenPToTViaControlExtensionOrient

```
layoutXL modgenPToTViaControlExtensionOrient cyclic { "None" | "Horizontal" |
     "Vertical" }
```

Description

Specifies the preference of the via extension or enclosure orientations.

The default value is `None`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTViaControlExtensionOrient")
envSetVal("layoutXL" "modgenPToTViaControlExtensionOrient" 'cyclic "Horizontal")
envSetVal("layoutXL" "modgenPToTViaControlExtensionOrient" 'cyclic "Vertical")
```

modgenPToTViaControlExtensionOrientEnable

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

Description

When set to `t`, enables the *Via Extension Orientation* option on the *Via Controls* tab.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTViaControlExtensionOrientEnable")
envSetVal("layoutXL" "modgenPToTViaControlExtensionOrientEnable" 'boolean t)
```

modgenPToTViaControlInline

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

Description

When selected, places a higher preference for vias that are fully enclosed or in line with the wire.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTViaControlInline")
envSetVal("layoutXL" "modgenPToTViaControlInline" 'boolean t)
```

modgenPToTViaControlInlineEnable

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

Description

When set to `t`, enables the *Via Inline* option on the *Via Controls* tab.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTViaControlInlineEnable")
envSetVal("layoutXL" "modgenPToTViaControlInlineEnable" 'boolean t)
```

modgenPToTViaControlOffset

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

Description

When set to `t`, honors the via offset values.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTViaControlOffset")
envSetVal("layoutXL" "modgenPToTViaControlOffset" 'boolean t)
```

modgenPToTViaControlOffsetEnable

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

Description

When set to `t`, enables the *Via Offset* option.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTViaControlOffsetEnable")
envSetVal("layoutXL" "modgenPToTViaControlOffsetEnable" 'boolean t)
```

modgenPToTViaControlOrient

```
layoutXL modgenPToTViaControlOrient cyclic { "None" | "Horizontal" | "Vertical" }
```

Description

Specifies the default via orientation. Default is `None`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTViaControlOrient")
envSetVal("layoutXL" "modgenPToTViaControlOrient" 'cyclic "Horizontal")
envSetVal("layoutXL" "modgenPToTViaControlOrient" 'cyclic "Vertical")
```

modgenPToTViaControlOrientEnable

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

Description

When set to `t`, enables the *Via Orientation* option.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTViaControlOrientEnable")
envSetVal("layoutXL" "modgenPToTViaControlOrientEnable" 'boolean t)
```

modgenPToTViaWidthPercent

`layoutXL modgenPToTViaWidthPercent int` *`integer_number`*

Description

Specifies the depth (in percentage) of via coverage when the via overlaps with a pin.

The default value is `100`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTViaWidthPercent")
envSetVal("layoutXL" "modgenPToTViaWidthPercent" 'int 120)
envSetVal("layoutXL" "modgenPToTViaWidthPercent" 'int 170)
```

modgenPToTViaWidthPercentEnable

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

Description

Specifies whether the depth (in percentage) of via coverage, when the via overlaps a pin, can be specified.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenPToTViaWidthPercentEnable")
envSetVal("layoutXL" "modgenPToTViaWidthPercentEnable" 'boolean t)
```

modgenSaveOnClosePreviewWindow

```
layoutXL modgenSaveOnClosePreviewWindow cyclic { "prompt" | "yes" | "no" }
```

Description

Specifies the behavior when the Modgen Previewer window is closed. When set to `prompt`, a pop-up message is displayed requesting for confirmation whether changes to the Modgen need to be saved. Choose *Yes* to save changes, and *No* to discard changes. Click *Cancel* to reject the closing of the window.

When set to `yes`, no pop-up is displayed. Instead, all modifications are saved and the Modgen Previewer window is closed.

When set to `no`, no pop-up is displayed. Instead, all modifications are discarded and the Modgen Previewer window is closed.

The default is `prompt`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenSaveOnClosePreviewWindow")
envSetVal("layoutXL" "modgenSaveOnClosePreviewWindow" 'cyclic "yes")
envSetVal("layoutXL" "modgenSaveOnClosePreviewWindow" 'cyclic "no")
```

Related Topics

Open Modgens in the Modgen Editor

Closing and Regenerating a Modgen

modgenTransferDiffInstances

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

Description

When set to `t`, verifies the layout and schematic parameters during Modgen transfer from the schematic to layout or vice versa.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenTransferDiffInstances")
envSetVal("layoutXL" "modgenTransferDiffInstances" 'boolean t)
```

Related Topics

Creating a Modgen

Closing and Regenerating a Modgen

modgenTransferIgnoreParamsList

`layoutXL modgenTransferIgnoreParamsList string "parameters_list"`

Description

Indicates the parameters to be ignored while verifying parameters between the layout and the schematic during transfer of the Modgen from schematic to layout or vice versa. This environment variable can be used only when `modgenTransferDiffInstances` is set to `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenTransferIgnoreParamsList")
envSetVal("layoutXL" "modgenTransferIgnoreParamsList" 'string "paramName")
```

Related Topics

modgenTransferDiffInstances

Creating a Modgen

Closing and Regenerating a Modgen

modgenUseSnapSpacing

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

Description

Allows to snap Modgen origin to snap spacing.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenUseSnapSpacing")
envSetVal("layoutXL" "modgenUseSnapSpacing" 'boolean t)
```

Related Topics

Creating a Modgen

Closing and Regenerating a Modgen

modgenUseIteratedAsMfactor

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

Description

Allows Modgen to consider iterated instances, with same or different connectivity, equivalent to an m-factor (multiplier). As a result, the pattern mapping shows the number of instances equivalent to the iterations for each different master.

The default value is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenUseIteratedAsMfactor")
envSetVal("layoutXL" "modgenUseIteratedAsMfactor" 'boolean nil)
```

Related Topics

Creating a Modgen

Closing and Regenerating a Modgen

modgenWidthParamProportionalToFingers

```
layoutXL modgenWidthParamProportionalToFingers cyclic { "default" | "yes" | "no" }
```

Description

Specifies whether the finger width of Modgen dummy instances must be proportional to the number of fingers.

The default value is `default`.

For FinFET devices, the `default` is `no`, and for non-FinFET devices, the `default` is `yes`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "modgenWidthParamProportionalToFingers")
envSetVal("layoutXL" "modgenWidthParamProportionalToFingers" 'cyclic "default")
envSetVal("layoutXL" "modgenWidthParamProportionalToFingers" 'cyclic "no")
```

Related Topics

Modgen Dummies

Dummy Options Form

modgenWindowConfigFile

`layoutXL modgenWindowConfigFile string "workspace_name"`

Description

Specifies the workspace that Modgen needs to load during startup.

GUI Equivalent

None

Example

```
envGetVal("layoutXL" "modgenWindowConfigFile")
envSetVal("layoutXL" "modgenWindowConfigFile" 'string "modgenUserWorkspace")
```

Related Topics

Open Modgens in the Modgen Editor

Closing and Regenerating a Modgen

transparentModgenArray

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

Description

Specifies whether Modgen Transparent Editing Mode must be turned on or off. In this mode, you can modify a Modgen directly from the top level, without invoking any GUI.

The default value is `nil`.

GUI Equivalent

Command	Options toolbar
Field	

Examples

```
envGetVal("layoutXL" "modgenUseIteratedAsMfactor")
envSetVal("layoutXL" "modgenUseIteratedAsMfactor" 'boolean t)
```

Related Topics

Modgen Transparent Editing Mode

chainPermutePins

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

Description

Specifies whether to use mirroring instances or permutation of pins during the abutment process.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "chainPermutePins")
envSetVal("layoutXL" "chainPermutePins" 'boolean nil)
```

Related Topics

Modgen Device Abutment

moveAsSwap

```
gpe moveAsSwap boolean { t | nil }
```

Description

Specifies the behavior when instances are moved in the Grid Pattern Editor. With this environment variable set to `t`, the instances in the source and target cells are swapped. When the environment variable is set to `nil`, the target instance first shifts horizontally and then vertically, until the source location is backfilled.

The default value is `t`.

GUI Equivalent

None

Examples

```
envGetVal("gpe" "moveAsSwap")
envSetVal("gpe" "moveAsSwap" 'boolean nil)
```

Related Topics

The Move Command in the Array Assistant

patternDefaultDisplayMode

```
gpe patternDefaultDisplayMode cyclic { "A" | "L" | "S" | "O" | "N" }
```

Description

Specifies the default pattern display mode in the Array Assistant. The available options are:

- `A`: Symbol names
- `L`: Layout names
- `S`: Schematic names
- `O`: Orientations
- `N`: Net names

The default value is `A`.

GUI Equivalent

None

Examples

```
envGetVal("gpe" "patternDefaultDisplayMode")
envSetVal("gpe" "patternDefaultDisplayMode" 'cyclic "L")
```

Related Topics

Modgen Placement Settings in the Array Assistant

patternPresetItemList

`gpe patternPresetItemList string l_presetList`

Description

Specifies the pattern presets to be listed in the *Pattern Preset* drop-down list in the Array Assistant. You can alter the entries and the sequence in which they are listed.

The default value is `default`, where the sequence is:

```
"Current|Clustered|Interdigitated|Compact|Custom|Zigzag|Shift|Common
Centroid|Resistor Topology"
```

Note: The `Resistor Topology` preset is conditionally visible in the tool. The visibility condition is applied only when `patternPresetItemList` is set to either `Resistor Topology` or `default`.

Note: If neither `Current` nor `Custom` presets are included in the environment variable string, they are added at the bottom of the list.

GUI Equivalent

Command	Array Assistant – *Placement*
Field	*Pattern Preset*

Examples

```
envGetVal("gpe" "patternPresetItemList")
envSetVal("gpe" "patternPresetItemList" 'string
"Compact|Interdigitated|Clustered|Common Centroid")
```

Related Topics

Array Assistant

Modgen Placement Settings in the Array Assistant

useDummyOwner

`layoutXL moveAsSwap boolean { t | nil }`

Description

Adds an `lxDummyOwner` property to the `library:cell:view` (LCV)-type dummies when an explicit dummy master is not specified at the top of the Modgen array dummy creation and edit flow. These dummies are maintained similar to non-LCV type array dummies.

For example, with the environment variable set, you can add a * character to an Array Assistant grid table widget. The neighbor device information is used to create an LCV-type dummy.

If an explicit dummy master is specified at all points in the flow, this environment variable does not impact the LCV-type dummies.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layoutXL" "useDummyOwner")
envSetVal("layoutXL" "useDummyOwner" 'boolean t)
```

Related Topics

The Move Command in the Array Assistant

7

Modgen Topology Constraints

The Modgen pin-to-trunk router recognizes topology objects and their constraints and creates geometries that match the topology pattern. You can add topology constraints to customize certain routing attributes. All topology constraints have a name, a layer, and a value. The layer can be set to `nil` if it is not applicable.

The Modgen topology constraints can be passed as arguments to certain Modgen place and route SKILL APIs such as `gpeCreateTrunkEntries`, `gpeCreateTwigEntries`, and `gpeCreateStrapEntries`. These constraints are honored while creating the respective topology objects.

In the following example, values for `twigConstraints` are first instantiated. These values are then passed as an argument to `gpeCreateTwigEntries`.

```
twigConstraints= list(
list("validLayers" nil list("Metal1" "Metal2"))
list("minWidth" list("Metal1") 0.14)
list("minWidth" list("Metal2") 0.12)
)
twigEntry = gpeCreateTwigEntries(
?instTermEntries list(iterm)
?constraints twigConstraints
)
```

Related Topics

.Defining Modgen Topology Settings Using the Array Assistant

minNumCut

Definition

Specifies the minimum number of cuts that a via object or a via instance, which is created between a pin and the topology connecting that pin, must contain. Use constraint trunkAccessingNumCuts for the vias that are created between trunk pathSegs and the topology connecting that trunk.

Wide wires are capable of carrying more current, and a sufficient number of via cuts is required to carry that additional current. Using multiple via cuts increases redundancy, and therefore reliability.

Values

- *`l_layerNames`*

  Specifies the layers to which the constraint must be applied.

  Type: List of strings (layer and purpose names) or integers (layer numbers)

- *`x_minNumCut`*

  The minimum number of cuts that a via object or a via instance must contain.

  Type: Integer

Parameters

None

Applies To

- Twig constraints
- Strap constraints

Example

```
'("minNumCut" '("Metal1" "Metal2" "Metal3") 4 )
```

Specifies that the vias on the specified layers must contain a minimum of four cuts.

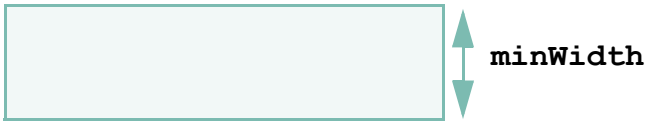
Related Topics

Defining Modgen Topology Settings Using the Array Assistant

minWidth

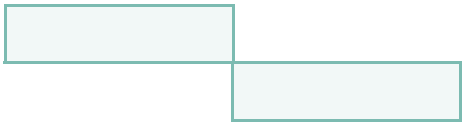
Definition

Specifies the minimum orthogonal width of shapes on the specified layers.

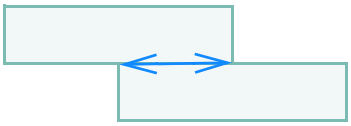


The minimum width constraint does not apply in the following situations:

- Two shapes touch at a point, as shown in the following figure:



- Two shapes abut for a sufficient overlap, as shown in the following figure:



Values

- `l_layerNames`

  Specifies the layers to which the constraint must be applied.

  Type: String (layer name), list of strings (layer and purpose names), or integers (layer numbers)

- `x_minWidth`

  The minimum permissible orthogonal width of shapes.

  Type: Integer

Parameters

None

Applies To

- Trunk constraints
- Twig constraints
- Strap constraints

Example

```
'("minWidth" '("Metal1") 0.14)
```

Sets the minimum width constraint to `0.14`.

```
'("minWidth" '("Metal1" "Metal2" "Metal3") 0.11)
```

Sets the minimum width constraint for the specified layers to `0.14`.

Related Topics

Defining Modgen Topology Settings Using the Array Assistant

numStrands

Definition	Specifies the number of strands to be used to route the topology object.
Values	■ `nil` Layer specification is not required. ■ *`x_numStrands`* Specifies the number of strands to be used to route the topology object. *Type:* Integer
Parameters	None
Applies To	■ Trunk constraints ■ Twig constraints ■ Strap constraints

Example

```
'("numStrands" nil 3)
```

Specifies that three strands must be used to route pins in the given topology object.

Related Topics

Defining Modgen Topology Settings Using the Array Assistant

pinCover

Definition

Defines parameters to control how vias must cover pins.

Values

- `nil`

  Layer specification is not required.

- *`g_pinCoverStatus`*

  Specifies whether the tap must cover the pin area.

  Type: Boolean (`t` or `nil`) or Integer (`0` or `1`)

  Default: `100`

Parameters

- **`pinCoverTapViaPercent`**

  Specifies the percentage of pin shape to be covered by the tapping via on all the layers. The percentage is measured in the direction perpendicular to the trunk.

  Type: Integer

  Valid Values: `0` through `100`

  Default: `100`

- **`pinCoverTapLowerViaPercent`**

  Specifies the percentage of a pin shape to be covered by the lower layers of the tapping vias. This value applies to all layers expect the top layer. The percentage is measured in the direction perpendicular to the trunk.

  Type: Integer

  Valid Values: `0` through `100`

  Default: `100`

- **`pinCoverViaModeTrunkOver`**

  Specifies the via mode to be applied when a trunk or pin is covered by a via stack. Valid values are:

  - `matchTrunk`: Covers the width of the trunk.
  - `full`: Covers the entire pin shape.
  - `fullExceptTop`: Covers the entire trunk on the top via layer and the entire pin on the lower via layers.

  Type: Enum

  Valid Values: `(matchTrunk, full, fullExceptTop)`

  Default: `matchTrunk`

- **`pinCoverViaModePercentTrunkOver`**

  Specifies the percentage of trunk or pin to be covered by the via stack. The percentage is measured in the direction perpendicular to the trunk. This option can be used only when `pinCoverViaModeTrunkOver` is set to `full` or `fullExceptTop`.

  Type: Integer

  Valid Values: `0` through `100`

  Default: `100`

Applies To

- Twig constraints

Example

```
list("pinCover" nil 1
        list("pinCoverTapViaPercent" 70)
        list("pinCoverTapLowerViaPercent" 70)
        list("pinCoverViaModeTrunkOver" full)
        list("pinCoverViaModePercentTrunkOver" 80)
        ))
```

Specifies the `pinCover` parameters.

Related Topics

Defining Modgen Topology Settings Using the Array Assistant

properTwigTrunkOverlap

Definition	Specifies whether the twig must extend to cover the width of the trunk.
Values	■ `nil` Layer specification is not required. ■ *`g_overlapStatus`* Specifies whether the twig must extend to cover the width of the trunk. *Type:* Boolean (`t` or `nil`) or Integer (`0` or `1`)
Parameters	None
Applies To	■ Twig constraints

Example

```
'("properTwigTrunkOverlap" nil 1 )
'("properTwigTrunkOverlap" nil t )
```

The `properTwigTrunkOverlap` value is an integer in the first example, and a Boolean in the second example.

Related Topics

Defining Modgen Topology Settings Using the Array Assistant

strandSpacing

Definition	Specifies the required exact spacing between individual wire strands on the given layers.
Values	■ *`l_layerNames`* Specifies the layers to which the constraint must be applied. ■ *`x_strandSpacing`* Specifies the exact spacing between individual wire strands on the given layer. *Type:* Integer
Parameters	None
Applies To	■ Trunk constraints ■ Twig constraints ■ Strap constraints

Example

```
'("strandSpacing" "Metal1" 0.06)
```

Sets `0.06` user units as the required spacing between individual strands on layer `Metal1`.

Related Topics

Defining Modgen Topology Settings Using the Array Assistant

strandWidth

Definition	Specifies the required exact width of individual wire strands on the given layer.
Values	■ *`l_layerNames`* Specifies the layers to which the constraint must be applied. ■ *`x_strandWidth`* Specifies the width of individual wire strands on the given layers. *Type:* Integer
Parameters	None
Applies To	■ Trunk constraints ■ Twig constraints ■ Strap constraints

Example

```
'("strandWidth" "Metal1" 0.03)
```

Sets `0.03` user units as the required width of individual strands on layer `Metal1`.

Related Topics

Defining Modgen Topology Settings Using the Array Assistant

trunkAccessingNumCuts

Definition	Specifies the number of cuts a via must contain when connecting a trunk to a twig.
Values	■ *`l_layerNames`* Specifies the layers to which the constraint must be applied. *Type:* List of strings (layer and purpose names) or integers (layer numbers) ■ *`x_numCuts`* Number of cuts that the via must contain. *Type:* Integer
Parameters	None
Applies To	■ Twig constraints ■ Strap constraints

Example

```
'("trunkAccessingNumCuts" '("Metal1") 2)
```

Specifies that vias with two cuts can be used to connect trunks to twigs.

Related Topics

Defining Modgen Topology Settings Using the Array Assistant

validLayers

Definition

Defines a list of valid routing layers or layer-purpose pairs.

Values

- `nil`

  Layer specification is not required.

- *`tx_layer`*

  Lists the layers that can be used for routing. If you do not specify the purpose, then the constraint applies to all purposes.

  Type: String (layer name) or integer (layer number)

- (*`tx_layer tx_purpose`*)

  Lists the layer-purpose pairs that can be used for routing.

  Type: String (layer and purpose names) or integer (layer and purpose numbers)

Parameters

None

Applies To

- Trunk constraints
- Twig constraints
- Strap constraints

Examples

```
'("validLayers" nil '("Metal1" "Metal2" "Metal3"))
```

Specifies the valid layers for routing.

```
'("validLayers" nil '("Metal1 pin" "Metal2 drawing" "Metal3 pin"))
```

Specifies the valid layer-purpose pairs for routing.

Related Topics

Defining Modgen Topology Settings Using the Array Assistant

validStackLPPs

Definition	Specifies a list of layer-purpose pairs that can be used for routing. The router uses all the listed layer-purpose pairs for routing.
Values	■ `nil` Layer specification is not required. ■ *`l_validStackLPPs`* List of layer-purpose pairs that can be used for routing. *Type:* list
Parameters	None
Applies To	■ Trunk constraints ■ Twig constraints ■ Strap constraints

Example

```
'("validStackLPPs" nil '("Metal1 drawing" "Metal2 pin" "Metal3 drawing"))
```

Defines valid layer-purpose pairs for routing.

Related Topics

Defining Modgen Topology Settings Using the Array Assistant

viaControl

Definition

Specifies the via control parameters that define how vias must be positioned and aligned on trunks and twigs.

Values

- *`l_layerNames`*

  Specifies the layers to which the constraint must be applied.

- *`x_viaControlStatus`*

  Specifies whether the via control parameters can be specified.

Parameters

- **`viaControlOrient`**

  Specifies the default via orientation. The bounding box of the via cuts is aligned along the specified direction. For example, a via with two cuts is rendered top-down if the via orientation is set to `Vertical`.

  Type: String

  Valid Values: `(Horizontal, Vertical, None)`

  Default: `None`

- **`viaControlInline`**

  Prefers vias that are fully enclosed or in line with the wire.

  Type: Boolean

  Valid Values: `(1, 0)` or `(t, nil)`

  Default: `0`

- **`viaControlOffset`**

  Honors the via offset values specified.

  Type: Boolean

  Valid Values: `(1, 0)` or `(t, nil)`

  Default: `0` or `nil`

- **`viaControlExtensionOrient`**

  Specifies the preference of the via extension or enclosure orientations. For vias with multiple cuts, a higher preference is given to vias with cut boxes lining up in a certain direction. Therefore, a vertical-cut bounding box via may have a horizontal extension. This means that the extended metal portion over the cut bounding box in the horizontal direction is larger than that in the vertical direction.

  Type: String

  Valid Values: `(Horizontal, Vertical, None)`

  Default: `None`

- **`viaControlCutClass`**

  Specifies the width and height of the associated cut classes.

  Type: List

  Valid Values: A list of two floating-point numbers, for example `'(0.032 0.032)`.

  Default: `'(0 0)`

Applies To

- Twig constraints

Example

```
list("viaControl" list("M1" "M2" "M3") 1
    list(
        list("viaControlOrient" "horizontal")
        list("viaControlInline" 1)
        list("viaControlOffset" 1)
        list("viaControlExtensionOrient" "horizontal")
        list("viaControlCutClass" '(0.07 0.07))
        )
    )
```

Specifies the various via control parameters.

Related Topics

Defining Modgen Topology Settings Using the Array Assistant

viaPercent

Definition	Specifies the allowed wire width as a percentage of the width of the associated pin. The setting applies in the direction perpendicular to the trunk.
Values	■ `nil` Layer specification is not required. ■ *`x_viaPercent`* Specifies the allowed wire width as a percentage of the width of the associated pin. *Type:* Integer *Valid Values:* `0` to `100` *Default:* `100`
Parameters	None
Applies To	■ Twig constraints ■ Strap constraints

Example

```
'("viaPercent" nil 60)
```

Specifies the allowed wire width as `60` percent of the width of the associated pin.

Related Topics

Defining Modgen Topology Settings Using the Array Assistant

wirePercent

Definition	Specifies the allowed wire width as a percentage of the pin width. The value is calculated along the pin edge parallel to the direction of the trunk.
Values	■ `nil` Layer specification is not required. ■ *`x_wirePercent`* Specifies the wire width as a percentage of the pin width. *Type:* Integer
Parameters	None
Applies To	■ Twig constraints ■ Strap constraints

Example

```
'("wirePercent" nil 90)
```

Specifies that the allowed wire width is `90` percent of the width of the associated pin.

Related Topics

Defining Modgen Topology Settings Using the Array Assistant