Virtuoso Automated Standard Cell Placement and Routing Flow Guide

Product Version IC23.1
November 2023

Contents

1

Virtuoso Automated Standard Cell Placement and Routing

The Virtuoso automated standard cell placement and routing flow comprises a series of tasks to generate automatically placed and routed layouts.

The standard cell placement and routing solution in Virtuoso seamlessly integrates the Innovus placer (GigaPlacer– GP) and router NanoRoute (NR) for designs less than 2K instances that are not timing driven. The flow starts from a schematic design and generates the layout view from source. Then it goes through creating WSPs, creating placement rows, doing supply routing, IO pin planning, placement cell selection, and placement. At this point, placement is complete and the design is ready for routing. It uses the Pre-Route Browser to guide you in selecting the nets for routing and then route the nets. The routing results can be viewed in the Routing Results Browser.

In standard cell routing, there are different ways for you to generate WSPs. You can also route the design without them, relying on Innovus-created tracks instead.

The Verilog, LEF/DEF (EMH) flow is not supported. It can cause module problems where cells are being created when they already exist as leaf cells.

In the Virtuoso standard cell placement and routing flow, you use the Auto P&R assistant for standard cell placement and the Routing assistant for standard cell routing in the Layout MXL cockpit.

You can use environment variables to change the value of many aspects of your environment either for an individual design session or permanently until you change the value of the variable again.

Related Topics

Environment Setup for Automated Standard Cell Placement and Routing Flow

Virtuoso Automated Standard Cell Placement and Routing Flow

Routing Assistant User Interface for Standard Cell

Environment Setup for Automated Standard Cell Placement and Routing Flow

To run the automated standard cell placement and routing flow, ensure the following:

- **Virtuoso Release**
  - Standard Cell Routing: IC23.1 or later
  - Standard Cell Placement: IC23.1 or later
- **Innovus**

  Innovus 20.14 version or later must be available in the build path.
- **PDK Settings**
  - PDK settings are available either from the foundry or built from the technology file from the foundry
  - An RMSOA PDK matching layer stack of base PDK with non-colliding siteDef, NDR, and via names
- **Standard cells must have a PR boundary (layout and abstract views)**

  Use a library that has both the layout and abstract views for standard cells in the same library. This allows automatic remastering of layout views to abstracts in Innovus.
- **Abstract views for standard cells and blocks are recommended**
  - Site definitions or names
  - Antenna information
  - EDGETYPES (advanced node)
  - Blocks with IO pins not on the PR boundary must have detailed abstracts
- **Pcells in the design**

When Pcells exist in the design, the `CDS_ENABLE_EXP_PCELL` shell environment variable must be set to `true` and the Pcell cache saved.

- Supply and ground sigTypes are set on power and ground nets.
- Virtuoso Standard Cell placement is run on the design and the placement is LVS correct and DRC compliant as specified in the technology file.
- Pins are grouped or clustered in close proximity and that buses are typically used for pins between blocks.
- All pins are on a routing layer.
- The minimum number of wire widths do not conform to the `LEFDefaultRouteSpec` or `NonDefaultRules` (NDRs) rules.
- **License:**

Standard Cell Placement	Virtuoso Layout Suite MXL
Standard Cell Routing	Virtuoso Layout Suite MXL This license is limited to 2000 standard cell instances in the design, which allows use of Virtuoso Layout Suite MXL without checking out additional Innovus startup licenses. For designs featuring more than 2000 instances, one or more Innovus startup licenses are required. The Innovus license can be configured using the invsLicenseArgs environment variable. There might be other Innovus licenses required based on the chosen methodology, such as `Innovus_hfr_opt` for high-frequency routing (NRHF). The Innovus ENG100 (Encounter_NG100) license is needed until Innovus 23.1X is available for cdsFixedVias.

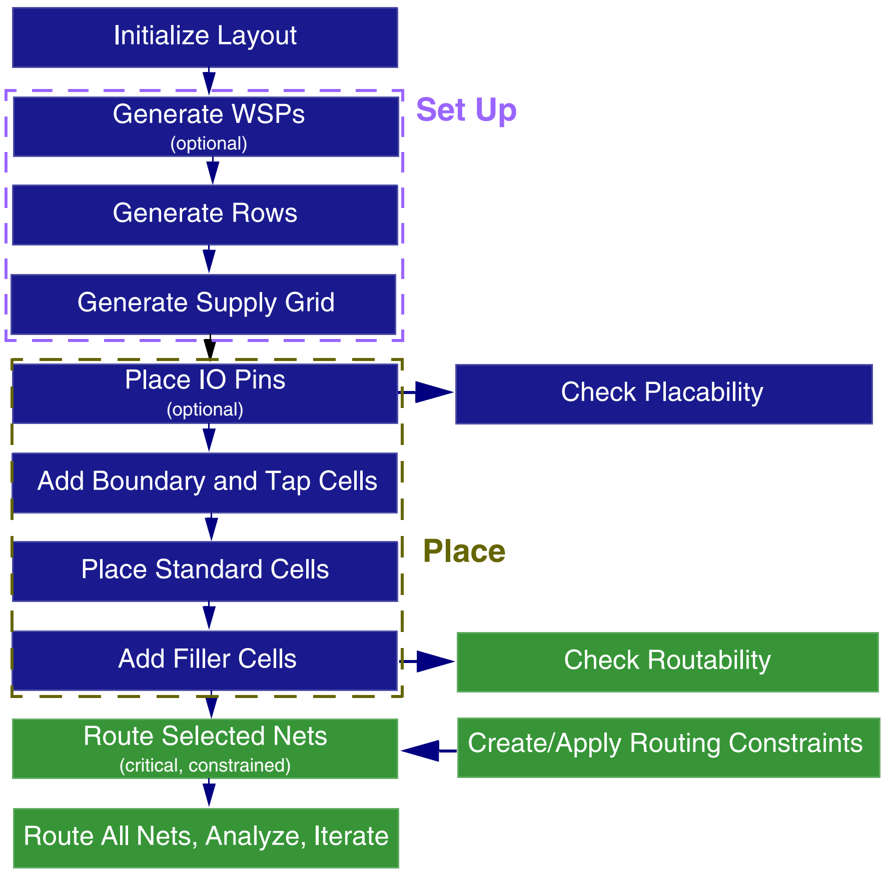
Related Topics

Virtuoso Automated Standard Cell Placement and Routing

Virtuoso Automated Standard Cell Placement and Routing Flow

Virtuoso Automated Standard Cell Placement and Routing Flow

The following diagram summarizes the Virtuoso automated standard cell placement and routing flow.



Flow Steps:

Standard Cell Placement:

Standard cell placement involves the following steps:

- Initializing the layout: The first step is layout generation, where information about the PR boundary, instances, nets, and pins is generated in the target layout as per the source schematic.
- Setting up the design for placement: In this step, you prepare the design for placement and routing by generating WSPs, rows, and supply grid.

- Placing Standard Cells: Here, you run the automated standard cell placer. This step involves placing IO pins and adding boundary and tap cells before running the placer, and adding filler cells after running the placer.

Standard Cell Routing:

The following steps summarize the standard cell routing flow.

- Generate Width Spacing Patterns: The first step in the routing technology flow is WSP generation. Information about the shapes on the layer, tracks, and patterns is generated in the design.
- Check Routability: In this step, you select the checks that should run before routing a design.
- Generate Supply Grid: The next step in the flow lets you generate power rails for VDD and GND nets.
- Run Automatic Routing: This step in the flow helps you to route all or selected nets.
- View and Analyze Routing Results: Finally, you can analyze the routing results from a single table using the Routing Results Browser. It shows the number of opens, shorts, wire length, DRC, vias, and so on.

Related Topics

Virtuoso Automated Standard Cell Placement and Routing

Environment Setup for Automated Standard Cell Placement and Routing Flow

Virtuoso Automated Standard Cell Routing

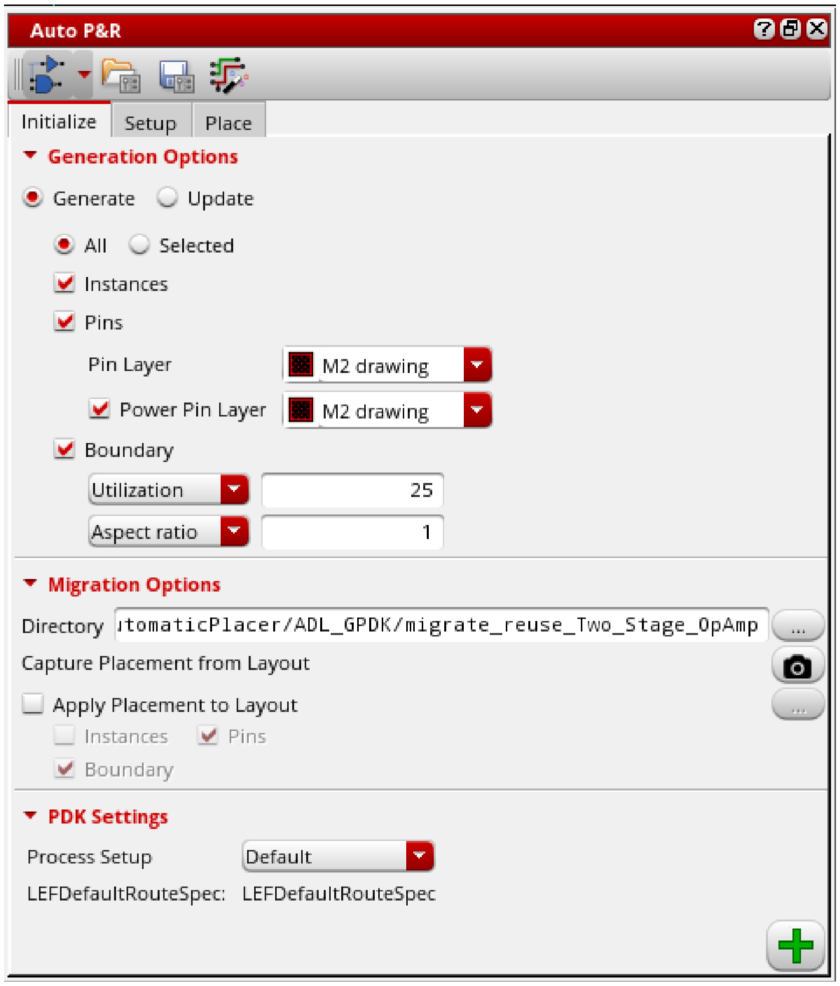
2

Initializing a Layout in the Auto P&R Standard Cell Flow

To initialize a layout view in the Auto P&R standard cell flow:

1. Open the *Initialize* tab of the Auto P&R assistant.

2. Select *Generate* to generate new objects in the layout canvas or *Update* to update the existing objects.


Auto P&R

Initialize | Setup | Place

Generation Options

[✓ Generate] Update

[✓ All] Selected

[✓] Instances

[✓] Pins

Pin Layer: M2 drawing

[✓] Power Pin Layer: M2 drawing

[✓] Boundary

Utilization: 25

Aspect ratio: 1

Migration Options

Directory: ıtomaticPlacer/ADL_GPDK/migrate_reuse_Two_Stage_OpAmp

Capture Placement from Layout

[ ] Apply Placement to Layout

[ ] Instances [✓] Pins

[✓] Boundary

PDK Settings

Process Setup: Default

LEFDefaultRouteSpec: LEFDefaultRouteSpec

3. Set the scope of layout generation to either *All* or *Selected*.

4. Select *Instances* to generate all instances from the source schematic.

5. Select *Pins to* generate pins during initialization.

6. Select *Power Pin Layer* to generate power pins.

7. Select an LPP from the *Pin layer* list that contains the standard cell pins to be generated. The default value is the first metal layer in the layer stack.

8. Select a layer from the *Power pin layer* list that contains the power and ground pins to be generated during initialization.

9. Select *Boundary* to generate a PR boundary as per the specified combination of *Utilization* and *Aspect Ratio* or *Width* and *Height*.

10. In Layout MXL, the *Migration Options* section is available, which lets you run the assisted flow of the Virtuoso® Custom Design Migration solution. For more information, see Setting Options for Custom Layout Design Migration.
11. In the *PDK Settings* section, select a process node from the *Process Setup* drop-down list.

    The process setup name is displayed below and the settings for the selected process node are loaded in the form. There is a direct mapping of this *Process Setup* in the Virtuoso Digital Implementation (VDI) GUI. The specified process setup triggers different placement and routing strategies that are built into Innovus.
12. Click *Generate* to generate the selected objects in the layout canvas.

Instances and pins are generated below the PR boundary.

Related Topics

Auto P&R Assistant User Interface for Standard Cell

Setting Up a Design in the Auto P&R Standard Cell Flow

Placing Standard Cells Automatically

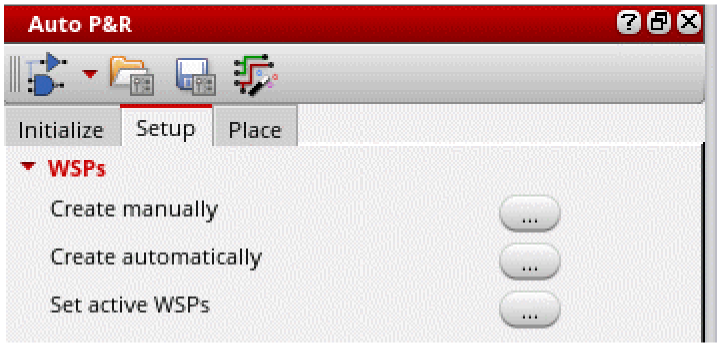
Setting Up a Design in the Auto P&R Standard Cell Flow

In the Auto P&R standard cell flow, after generating the PR boundary, instances, nets, and pins in the layout, the next step is to prepare the design for placement and routing.

Use the *Setup* tab of the Auto P&R assistant to configure the WSP settings, routing preferences, pin attributes, row generation options, and routing checks before running the standard cell placer.

To specify placement and routing settings for standard cells:

1. Open the *Setup* tab of the Auto P&R assistant.



2. Set up WSPs by selecting *Browse* beside one of the following options:

   ❑ *Create manually* to use the WSP Manager. After specifying the required settings, close the form.

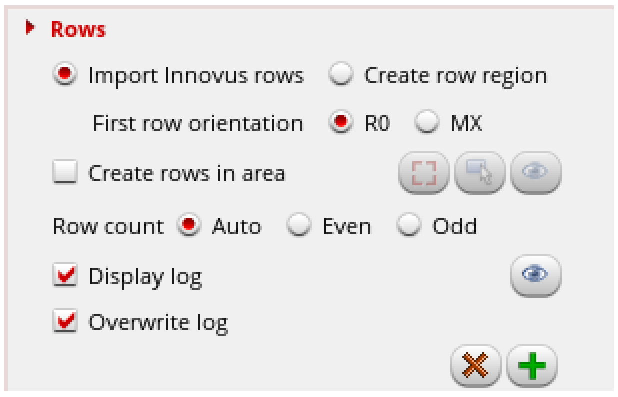
   ❑ *Create automatically* to use the *Setup* tab of the Routing assistant.

   ❑ *Set active WSPs* to use the Track Pattern assistant.

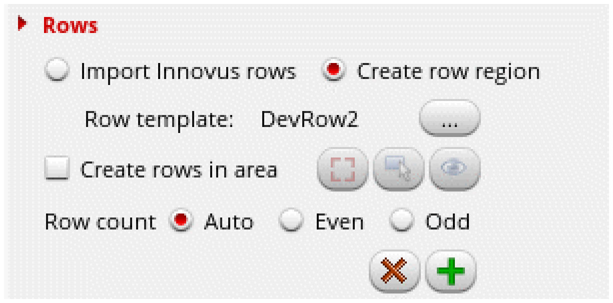
3. In the *Rows* section, specify how rows are to be created—imported from Innovus or created in Virtuoso.

   ❑ When set to *Import Innovus rows*, the tool imports the entire row region that either fits within the PR boundary or the region you draw with *Create rows in area*.

   Set *First row orientation* to the orientation of the bottom-most (first) row in the row region.


Rows
[✓ Import Innovus rows] Create row region
First row orientation [✓ R0] MX
☐ Create rows in area
Row count [✓ Auto] Even Odd
☑ Display log
☑ Overwrite log

- ❑ When *Rows* is set to *Create row region*, click the *Browse* button next to *Row template* and select a row template based on which rows are to be generated.



Rows
Import Innovus rows [✓ Create row region]
Row template: DevRow2
Create rows in area [ ]
Row count [✓ Auto] Even Odd

4. Select *Create rows in area* to enable the row region selection options.

   a. Set the row area to either the visible area ( ) or draw the required area ( ) in the design canvas.

   b. Select to toggle visibility of the area bbox in the design canvas.

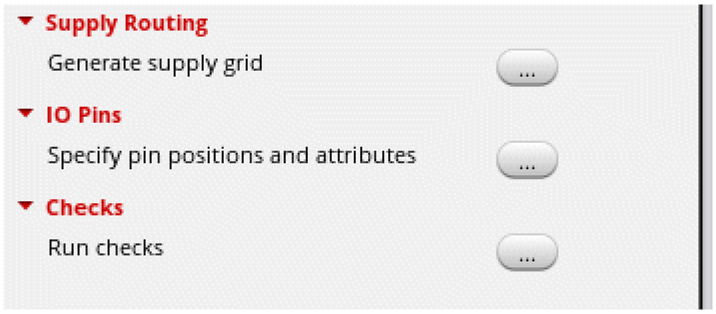
5. Select *Display log* to specify whether the setup log must be displayed in the CIW.

6. Select *Overwrite log* to overwrite the existing setup log. When this option is not selected, a new log file is created.

7. Click to delete any existing created rows.

8. Click to create a new row region.

9. Click *Browse* beside *Generate supply grid* to open the *Supply* tab of the Routing assistant, which lets you create a uniform width and pitch pattern-based WSP.



Supply Routing
Generate supply grid ...
IO Pins
Specify pin positions and attributes ...
Checks
Run checks ...

If the boundary cells have blockages on the power rails, select the *Add cell row routing* option on the *Route* tab of the Routing assistant. Do not create stripes on the standard cell power rails (cell row) layer with the supply router. Instead Innovus creates the stripes after the boundary cells are placed, and it respects the blockages in them.

10. Click *Browse* next to *Specify pin positions and attributes* to open the *Pin Planner* tab of the Pin Placement form, where you can specify pin constraints and pin attributes.

11. Click *Browse* beside *Run checks* to display the *Check* tab of the Routing assistant, where you can select the required pre-routing checks to be run during design placement.

Your design is now ready for placement.

Related Topics

Auto P&R Assistant User Interface for Standard Cell

Routing assistant - Setup tab

Routing assistant - Supply tab

Routing assistant - Check tab

Pin Planner tab of the Pin Placement form

Standard Cell Components Generated During Placement

Placement of standard cells has a unique set of requirements when compared to the placement of devices. In addition to standard cells, you require:

- **Boundary Cells**: At advanced nodes, adjacent cells (core cells) are placed in close proximity, which can lead to undesirable electrical effects. Adding boundary cells around core cells isolates the core cells from each other, and therefore prevents undesired effects.

  Boundary cells are represented by cells that have their component class set to `BOUNDARYCELL`. Component types are defined in the *Cells* table of the Configure Physical Hierarchy window—*Component Types* mode. After ensuring that the boundary cell definitions are in place, you can insert them between the core cells in your design.

- **Tap Cells**: In addition to boundary cells, you can insert either single-height or multi-height tap cells in the empty spaces between the standard cells. Tap cells are a set of contacts that are used to reduce latch-up effects between power and ground connections and the connections with wells or substrate contacts.

  Tap cells are represented by cells that have their component class set to `STDSUBCONT`. Component types are defined in the *Cells* table of the Configure Physical Hierarchy window—*Component Types* mode.

- **Filler Cells**: After standard cell placement, there might be some gaps or unfilled areas in layout designs. Unfilled areas might also result if the *Utilization* setting was selected on the *Initialize* tab when generating the design. Running a DRC check on such designs at this point could report several spacing violations. To avoid these violations, you must generate filler cells in the unfilled area between standard cells.

  Filler cells are assigned to a component type with their component class set to `FILLER`.

  Before inserting new filler cells, delete any existing filler cells. Otherwise, the placement of cells is prevented because of over congestion.

Related Topics

Auto P&R Assistant User Interface for Standard Cell

Placing Standard Cells Automatically

Adding Boundary Cells During Automated Standard Cell Placement

Adding Tap Cells During Automated Standard Cell Placement

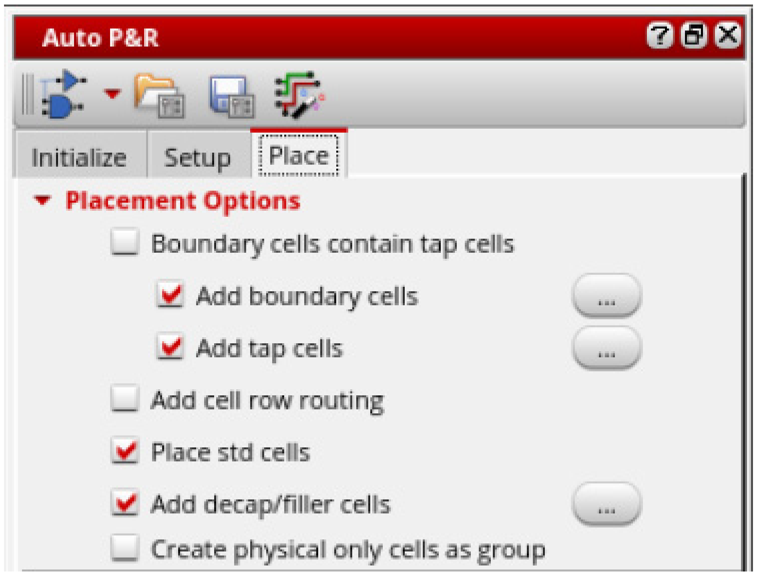
Adding Filler Cells During Automated Standard Cell Placement

Placing Standard Cells Automatically

In the Virtuoso standard cell automatic placement flow, you first initialize the layout and then set up the design for routing. You can then run the automated standard cell placer to place the standard cells in the design. While running the placer, you can add boundary cells and tap cells to prevent any undesired effect during routing.

To run the Virtuoso standard cell automatic placer:

1. Open the *Place* tab of the Auto P&R assistant.



- [ ] Boundary cells contain tap cells
- [x] Add boundary cells
- [x] Add tap cells
- [ ] Add cell row routing
- [x] Place std cells
- [x] Add decap/filler cells
- [ ] Create physical only cells as group

2. Insert boundary cells by doing one of the following:

   ❑ Select *Boundary cells contain tap cells* to insert boundary cells and tap cells automatically.

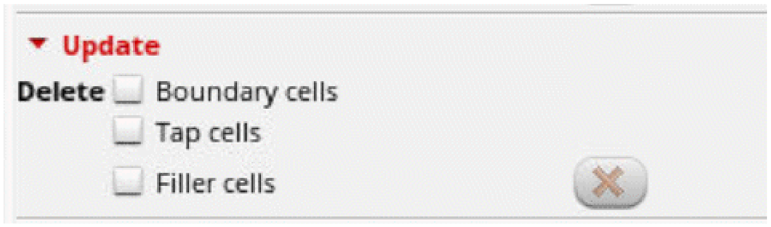
   ❑ Select *Add boundary cells* and click the *Browse* button beside it to open the Boundary Cells from.

   Use the options in the form to insert boundary cells. See Adding Boundary Cells During Automated Standard Cell Placement.

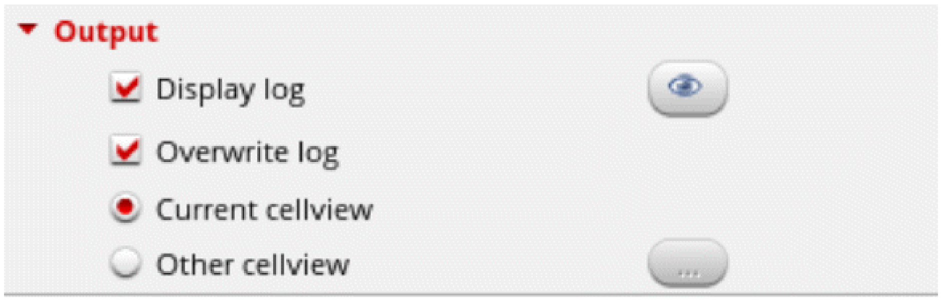
3. Select *Add tap cells* and click the *Browse* button beside it to open the Tap Cells from.

   Use the options in the form to add tap cells. See Adding Tap Cells During Automated Standard Cell Placement.

4. Select *Add cell row routing* to add follow-pin (cell row) routing and insert vias with correct color in Innovus.
5. Select *Place std cells* to add the command to place standard cells into the Tcl command script.
6. Select *Add decap/filler cells* and click the *Browse* button beside it to open the Filler Cells from.
7. Use the options in the form to add filler cells. See Adding Filler Cells During Automated Standard Cell Placement.
8. To insert decap cells, select only decap cells and then follow the procedure to add filler cells.
9. Select *Create physical only cells as group* to add physical-only instances (boundary, tap, and filler or decap) cells into a figGroup.
10. In the *Update* section, select the components to be deleted before generating new ones in the design.



11. In the *Output* section, specify the placement log settings.

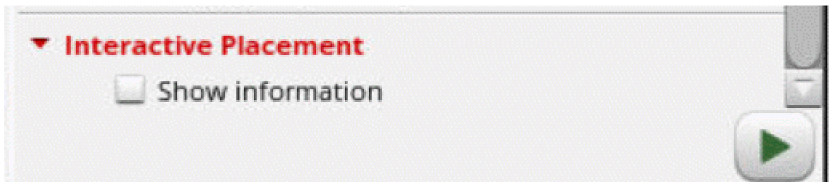


   a. Select *Display log* to specify whether the placement log must be displayed in the CIW.
   b. Click the view icon to display the contents of the log file in the CIW.
   c. Select *Overwrite log* to overwrite the existing placement log. When this option is not selected, a new log file is created.

   d. Specify where the placement results are to be stored: *Current view* or *Other view*.

   When set to *Other cellview*, click *Browse* and select a location from the Choose Source Layout Cellview form.

12. In the *Interactive Placement* section, select *Show information* to display information about placement in the CIW.



13. Click the *Run* ▶ button to run the Virtuoso standard cell automatic placer.

Related Topics

Auto P&R Assistant User Interface for Standard Cell

Standard Cell Components Generated During Placement

Adding Boundary Cells During Automated Standard Cell Placement

Adding Tap Cells During Automated Standard Cell Placement

Adding Filler Cells During Automated Standard Cell Placement

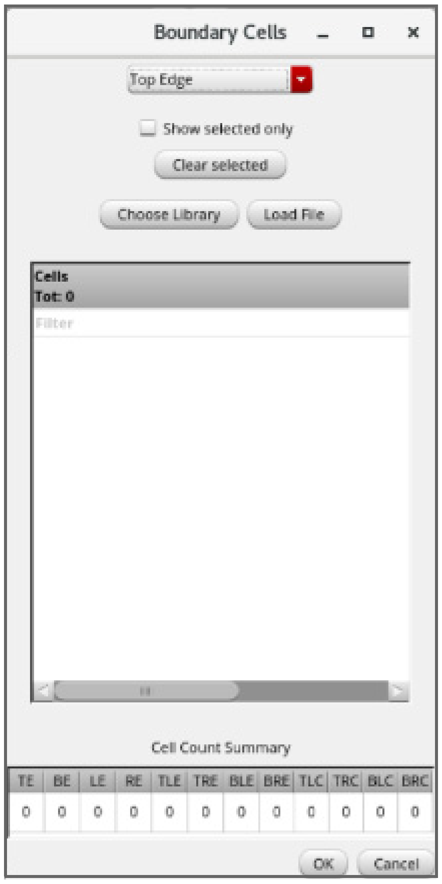
Adding Boundary Cells During Automated Standard Cell Placement

In standard cell designs, we adding boundary cells around core cells to isolate these core cells from each other, and therefore prevent any undesired effects. In the Virtuoso standard cell automatic placement and routing flow, boundary cells are added as part of the placement step.

To add boundary cells:

1. Open the *Place* tab of the Auto P&R assistant in Standard Cell mode
2. Select *Add boundary cells* and click the ellipses button.

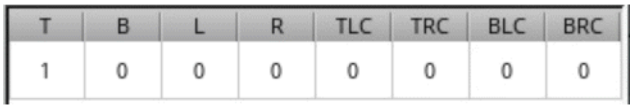
The Boundary Cells form appears.



TE	BE	LE	RE	TLE	TRE	BLE	BRE	TLC	TRC	BLC	BRC
0	0	0	0	0	0	0	0	0	0	0	0

3. From the drop-down list at the top, select the side on which to insert boundary cells.
4. In the *Cells* table, select the boundary cells to be inserted on the selected side.

   The boundary cells are added to the specified side. The *Filter* field supports regular expressions to help you find the cells if many cells are present. The table at the bottom shows the count of boundary cells. For example, if you have selected one boundary cell to be added to the *Top* edge, the table displays the following:



T	B	L	R	TLC	TRC	BLC	BRC
1	0	0	0	0	0	0	0

5. Repeat the above steps to add boundary cells to other edges.
6. Click *OK*.

Boundary cells are generated as specified when you run the automated standard cell placer.

Related Topics

Auto P&R Assistant User Interface for Standard Cell

Standard Cell Components Generated During Placement

Placing Standard Cells Automatically

Adding Tap Cells During Automated Standard Cell Placement

Adding Filler Cells During Automated Standard Cell Placement

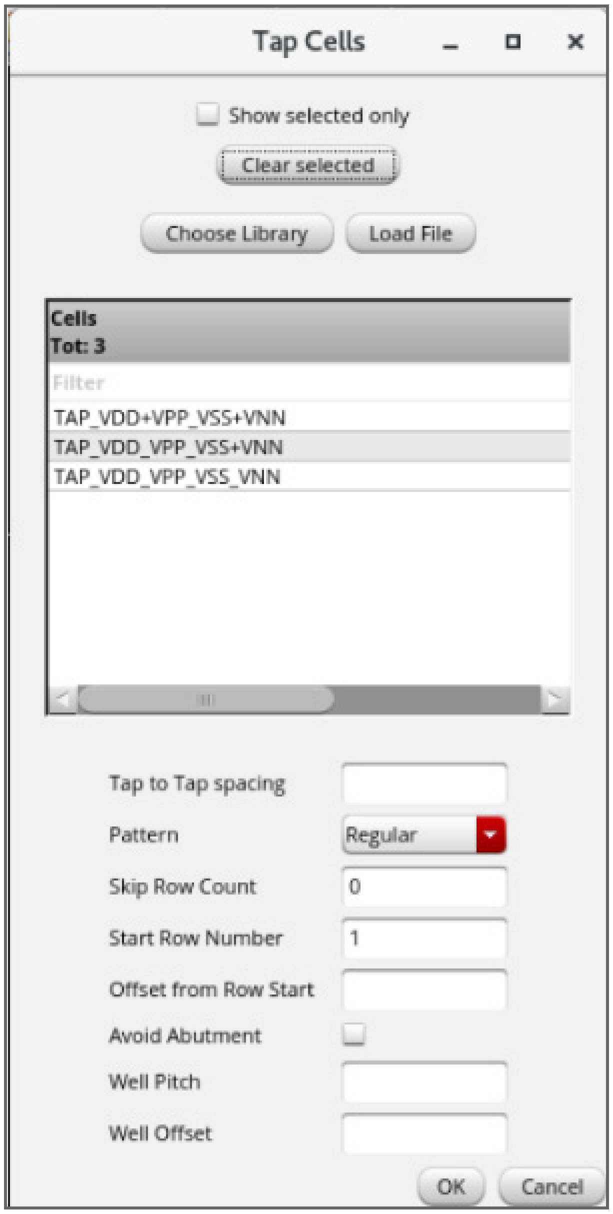
Adding Tap Cells During Automated Standard Cell Placement

In the Virtuoso standard cell automatic placement and routing flow, adding tap cells helps reduce latch-up effects between power and ground connections and the connections with wells or substrate contacts. Tap cells are inserted as part of the placement step.

To specify tap cell settings:

1. Open the *Place* tab of the Auto P&R assistant in Standard Cell mode.
2. Select *Add tap cells* and click the ellipsis button.

The Tap Cells form appears.



TAP_VDD+VPP_VSS+VNN
TAP_VDD_VPP_VSS+VNN
TAP_VDD_VPP_VSS_VNN

Tap to Tap spacing	
Pattern	Regular
Skip Row Count	0
Start Row Number	1
Offset from Row Start	
Avoid Abutment	☐
Well Pitch	
Well Offset	

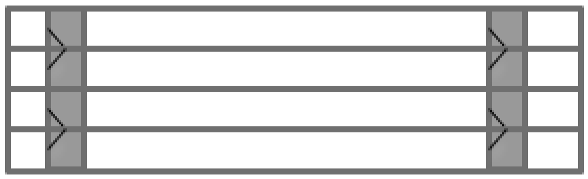
3. From the *Cells* list, select the required cells to be used as tap cells. These are the cells with their component class set to `STDSUBCONT`.

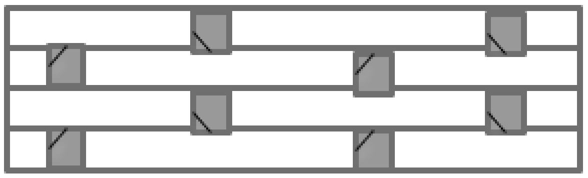
   ❑ You can filter cells by typing the required substring in the *Filter* box. The *Filter* field supports regular expressions.

   ❑ To clear all filters, select *Clear selected*.

   ❑ Select *Show selected Only* to hide the cells that are not selected.

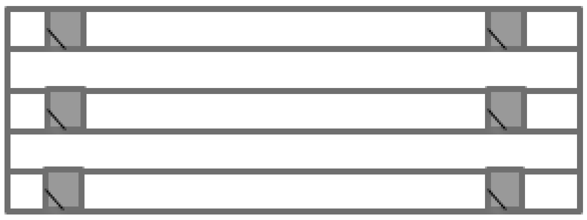
4. In *Tap to Tap spacing,* specify the spacing to be maintained between adjacent tap cells.
5. From the *Pattern* list, select the pattern in which tap cells are to be inserted: *Regular* or *Checker Board*.


Pattern: Regular

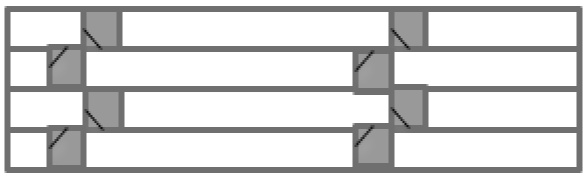

Pattern: Checker Board

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

6. Set *Skip Row Count* to the number rows to be skipped while inserting tap cells. The top row cannot be skipped. In the following example, alternate rows are skipped.



7. In *Start Row Number*, specify the row number from which tap cell insertion must start.
8. Specify the *Offset from Row Start*, which is the initial offset for the first tap cell in each row.
9. Select *Avoid Abutment* to specify that the tap cells must not be abutted vertically. By default, tap cells are abutted.



10. In *Well Pitch*, specify the pitch or the minimum spacing between standard cells and the tap cell wall.
11. In *Well Offset*, specify the offset of the tap cell wall from their adjoining standard cells.
12. Click *OK* to close the form and go back to the *Place* tab of the Auto P&R assistant.

Tap cells are added when you run the automated standard cell placer.

Related Topics

Auto P&R Assistant User Interface for Standard Cell

Standard Cell Components Generated During Placement

Placing Standard Cells Automatically

Adding Boundary Cells During Automated Standard Cell Placement

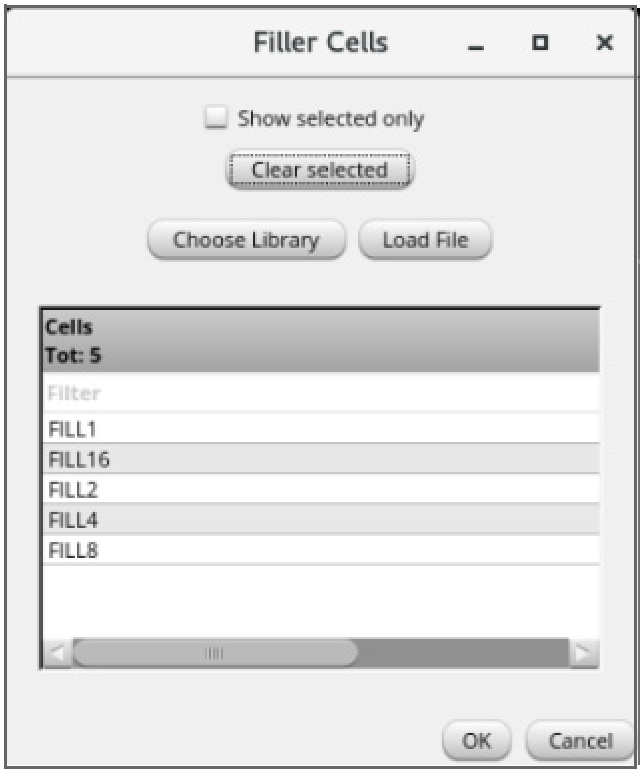
Adding Filler Cells During Automated Standard Cell Placement

Adding Filler Cells During Automated Standard Cell Placement

Use the Filler Cells form to select the filler cells to be inserted after running the placer. To specify filler cell settings:

1. Open the *Place* tab of the Auto P&R assistant in Standard Cell mode.
2. Select *Add decap/filler cells* and click *Browse*.

The Filler Cells form appears.



3. From the *Cells* list, select the required cells to be used as filler cells. These are the cells with their component class set to `FILLER`.
    - ❑ To filter the cells list, specify the keywords in the *Filter* box. The *Filter* field supports regular expressions.
    - ❑ To clear all filters, select *Clear Selected*.
    - ❑ Select *Show selected only* to hide the cells that are not selected.
4. Click *OK* to close the form and go back to the *Place* tab of the Auto P&R assistant.

Filler cells settings are saved and are generated after running the automated standard cell placer.

Related Topics

Auto P&R Assistant User Interface for Standard Cell

Standard Cell Components Generated During Placement

Placing Standard Cells Automatically

Adding Boundary Cells During Automated Standard Cell Placement

Adding Tap Cells During Automated Standard Cell Placement

3

Virtuoso Automated Standard Cell Routing

The Virtuoso standard cell routing solution seamlessly integrates the Innovus NanoRoute™ router in the Virtuoso Studio design environment. The flow starts from a schematic design from which a layout view is generated. You can then import IO pin and boundary information from an existing layout and step through the creation of Innovus-compatible row regions before completing power routing and the placement of tap cells and standard cells.

The Verilog, LEF/DEF flow is not supported. It can cause module problems where cells are being created when they already exist as leaf cells.

In standard cell routing, there are different ways for you to generate WSPs. You can also route the design without them, relying on Innovus created tracks instead.

Related Topics

Routing Assistant User Interface for Standard Cell

Configuring Standard Cell Router Settings

Generating Width Spacing Patterns for Standard Cell Routing

Checking Layout Routability after Running Standard Cell Placer

Generating a Supply Grid

Running Signal Routing for Standard Cells

Viewing and Analyzing Standard Cell Routing Results

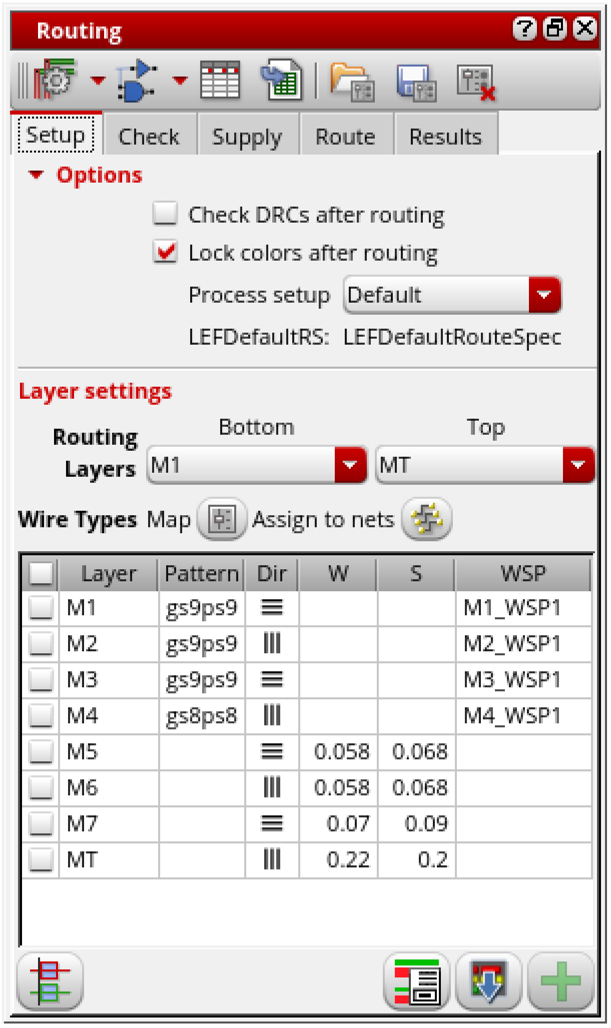
Configuring Standard Cell Router Settings

After the standard cell placement, use the Routing Assistant to configure the various routing options for standard cell routing before you run the router. To configure router settings:

1. Open a design in Layout MXL.
2. Choose *Window – Assistants – Routing*.

   Alternatively, right-click anywhere on the layout window menu bar and choose *Assistants – Routing*.

   The *Setup* tab of the Routing assistant is displayed.



☐ Check DRCs after routing
☑ Lock colors after routing
Process setup: Default
LEFDefaultRS: LEFDefaultRouteSpec

Layer settings

Routing Layers — Bottom: M1 — Top: MT

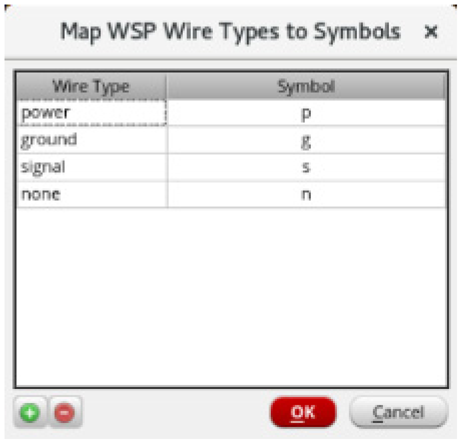
Wire Types Map — Assign to nets

Layer	Pattern	Dir	W	S	WSP
M1	gs9ps9	≡			M1_WSP1
M2	gs9ps9	‖‖‖			M2_WSP1
M3	gs9ps9	≡			M3_WSP1
M4	gs8ps8	‖‖‖			M4_WSP1
M5		≡	0.058	0.068	
M6		‖‖‖	0.058	0.068	
M7		≡	0.07	0.09	
MT		‖‖‖	0.22	0.2	

3. Select *Check DRCs after routing* to automatically run design rule checks after routing.

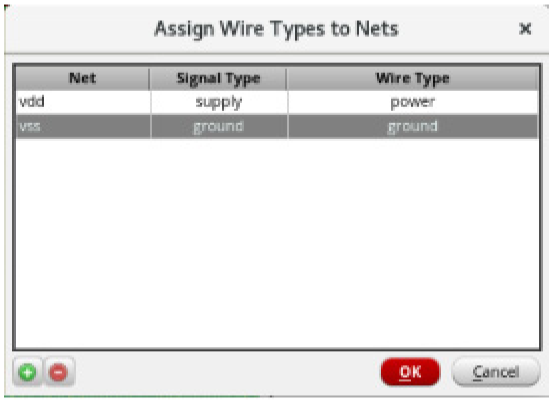
4. Select *Lock colors after routing* to automatically color-lock the shapes after routing.
5. Select the appropriate process node from the *Process setup* drop-down list. *Default* is selected if a process node does not exist.

   If the RMSOA PDK has multiple techLEFs, an additional Multi-tech LEF option is displayed. The different foundry rules are available as options in the drop-down list.
6. Choose the bottom and top preferred routing layers from the *Bottom* and *Top* drop-down lists. Specifying valid routing layers updates the WSPs visible in the table and specifies which layers the router should use for routing.
7. Select the layers for which you want to configure the router settings. You can either select all layers or one or more of them.
8. Click *Wire Types Map* to open the Map WSP Wire Types to Symbols form, which shows the wire type-to-symbol mapping information. You can add custom wire types and map them to unique symbols.



Wire Type	Symbol
power	p
ground	g
signal	s
none	n

9. Click *Assign to Nets* to open the Assign Wire Types to Nets form, which shows a table for wireTypes to nets assignment.



Net	Signal Type	Wire Type
vdd	supply	power
vss	ground	ground

10. Once the router settings are configured, you can generate width spacing patterns (WSPs).

Related Topics

Routing Assistant User Interface for Standard Cell

Generating Width Spacing Patterns for Standard Cell Routing

Checking Layout Routability after Running Standard Cell Placer

Map WSP Wire Types to Symbols Form

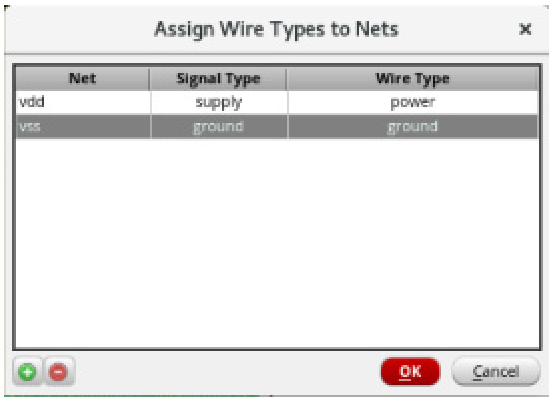
Assigning Wire Types to Nets

Assign Wire Types to Nets Form

Assigning Wire Types to Nets

The Assign Wire Types to Nets form is used to set `sigType` for the supply router and to know which tracks to create stripes for nets. To assign wire types to nets:

1. Click *Assign to Nets* button in the Setup tab of the Routing Assistant. It opens the Assign Wire Types to Nets form with a table that assigns wireTypes to nets.



Net	Signal Type	Wire Type
vdd	supply	power
vss	ground	ground

2. Click the *+* button to add a new wire type and net assignment.
3. Specify the net names as `VDD` and `VSS` in the *Net* column.
4. Specify the signal type as `power` and `ground` in the *Signal Type* column.

5. Select the wire type as `power` or `ground` in the *Wire Type* column. You can assign multiple wireTypes to a net.
6. Click *OK*.

Related Topics

Assign Wire Types to Nets Form

Configuring Standard Cell Router Settings

Routing Assistant User Interface for Standard Cell

Generating Width Spacing Patterns for Standard Cell Routing

In automated Routing Technology, there are three methods for generating WSPs.

- Using the WSP Manager
- Importing WSPs from a design
- Generating WSPs automatically

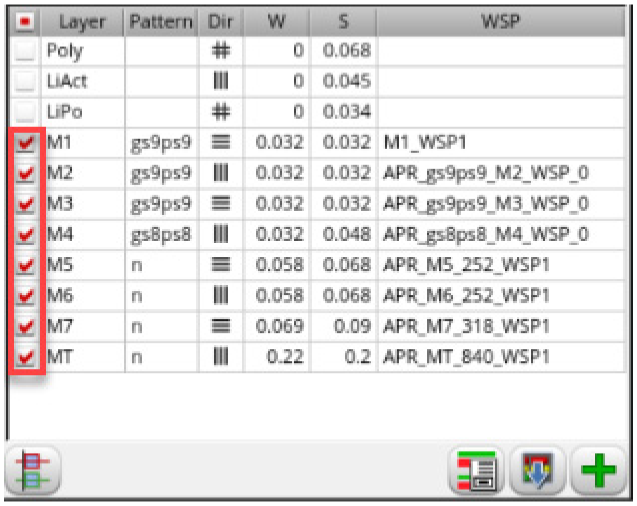
This topic covers the method of generating WSPs automatically. For information on generating WSPs using WSP Manager and importing WSPs, see Creating and Modifying WSPs and Importing WSPs from Another Design.

To automatically generate WSPs:

1. Open a design in Layout MXL.
2. Choose *Window – Assistants – Routing*.

   Alternatively, right-click anywhere on the layout window menu bar and choose *Assistants – Routing*. The Routing Assistant is displayed.
3. In the *Setup* tab, select the bottom and top routing layers from the *Bottom* and *Top* drop-down list. Specifying valid routing layers updates the WSPs visible in the table and specifies which layers the router should use for routing.
4. Select the required routing layers by clicking the check box in the layer table.WSPs are generated only on the selected layers. You can select all layers or specific ones. You can

also select one or more layers by clicking the check box next to the layer name. The check box selection indicates for which layers you want to generate WSPs.



	Layer	Pattern	Dir	W	S	WSP
	Poly		#	0	0.068	
	LiAct		|||	0	0.045	
	LiPo		#	0	0.034	
✓	M1	gs9ps9	≡	0.032	0.032	M1_WSP1
✓	M2	gs9ps9	|||	0.032	0.032	APR_gs9ps9_M2_WSP_0
✓	M3	gs9ps9	≡	0.032	0.032	APR_gs9ps9_M3_WSP_0
✓	M4	gs8ps8	|||	0.032	0.048	APR_gs8ps8_M4_WSP_0
✓	M5	n	≡	0.058	0.068	APR_M5_252_WSP1
✓	M6	n	|||	0.058	0.068	APR_M6_252_WSP1
✓	M7	n	≡	0.069	0.09	APR_M7_318_WSP1
✓	MT	n	|||	0.22	0.2	APR_MT_840_WSP1

Multiple active WSP definitions are not allowed on the same layer. Multiple widths are supported in the same WSP (coincident tracks) but not multiple WSP definitions. This is because, in such cases, it is difficult to assume that all the effective tracks are DRC clean in terms of color and spacing.

5. Leave the *Pattern* column of the layer table blank for the `M5` layer. The *Filter* field supports regular expressions to help you find the cells if many cells are present. If the standard cell power rail layer does not match a uniform width and spacing track pattern, it is appropriate to either use add_tracks with width and spacing parameters or use the Wire Assistant Derive functionality to create the WSPs for that layer.

   The WSPs are automatically generated from the tracks that Innovus generates when the *Pattern* field is left blank. If there are existing WSPs, the Innovus tracks are not generated. Ensure that all the WSPs are inactive in the Track Pattern assistant before doing this step.

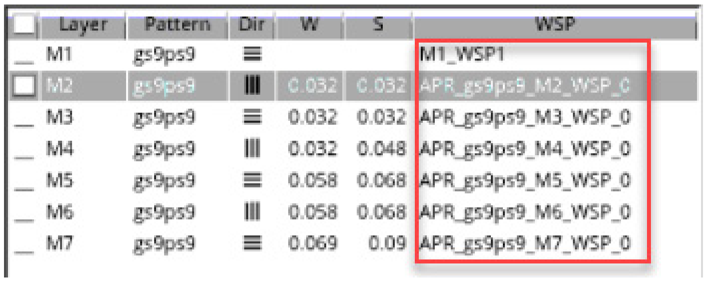
   If the *Pattern* field is not blank, WSPs are generated based on the pattern with the WSP pattern generator. The *Pattern* field determines whether WSPs are generated from Innovus or from a pattern.

   WSPs generated from Innovus do not have a wireType assigned. They can be used as a starting point in WSP Manager and the wireType can be added there.

6. Specify a pattern for example, `gs9ps9`, for metal layers in the *Pattern* column of the layer table. The pattern `gs9ps9` means that the track wireTypes are `ground`, `signal`, `signal`, `signal`, `signal`, `signal`, `signal`, `signal`, `signal`, `signal`, `power`, `signal`, `signal`, `signal`, `signal`, `signal`, `signal`, `signal`, `signal`,`signal`. It generates WSPs based on the pattern with the WSP pattern generator.

The Map WSP Wire Types to Symbols Form informs you which character represents the wireType in the WSP that is to be created. The particular character or symbol is used in the pattern string.

7. Click *Auto-generate WSPs* at the bottom of the Routing assistant. WSPs are automatically generated for the selected metal layers.



Layer	Pattern	Dir	W	S	WSP
M1	gs9ps9	≡			M1_WSP1
M2	gs9ps9	‖‖	0.032	0.032	APR_gs9ps9_M2_WSP_0
M3	gs9ps9	≡	0.032	0.032	APR_gs9ps9_M3_WSP_0
M4	gs9ps9	‖‖	0.032	0.048	APR_gs9ps9_M4_WSP_0
M5	gs9ps9	≡	0.058	0.068	APR_gs9ps9_M5_WSP_0
M6	gs9ps9	‖‖	0.058	0.068	APR_gs9ps9_M6_WSP_0
M7	gs9ps9	≡	0.069	0.09	APR_gs9ps9_M7_WSP_0

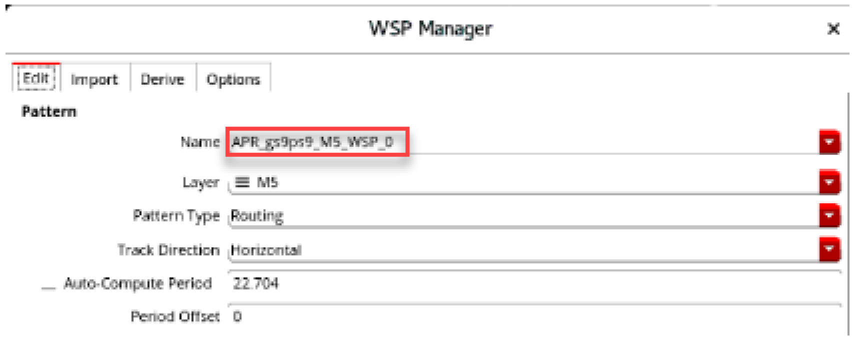
8. Click *Snap pins to WSPs* and check if the IO pins can snap to the WSPs.

9. Select any layer in the layer table to view its WSP attributes.

   Multiple active WSP definitions are not allowed on the same layer. However, multiple widths in the same WSP (coincident tracks) are supported but not multiple WSP definitions. The is because you cannot assume all the effective tracks are DRC clean, for example, in terms of color and spacing.

10. Click the *Show WSP Manager* button at the bottom of the Routing assistant.

    The WSP attributes of the selected layer are displayed.



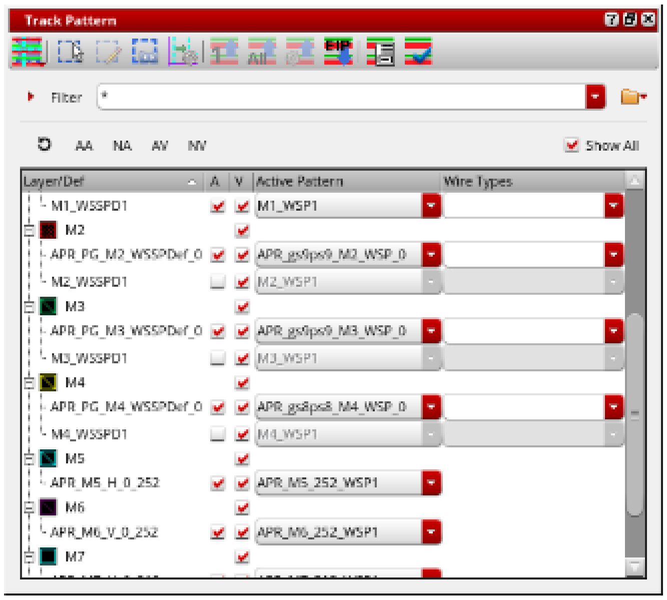
Pattern

Name	APR_gs9ps9_M5_WSP_0
Layer	≡ M5
Pattern Type	Routing
Track Direction	Horizontal
Auto-Compute Period	22.704
Period Offset	0

11. Choose *Window – Assistant – Track Pattern* from the layout window menu bar to open the Track Pattern assistant.

12. Click the *Show All* check box.

You can see the pattern-based auto-generated WSPs prefixed with `APR_`.



Layer/Def	A	V	Active Pattern	Wire Types
M1_WSSPD1	✓	✓	M1_WSP1	
M2		✓		
APR_PG_M2_WSSPDef_0	✓	✓	APR_gs9ps9_M2_WSP_0	
M2_WSSPD1		✓	M2_WSP1	
M3		✓		
APR_PG_M3_WSSPDef_0	✓	✓	APR_gs9ps9_M3_WSP_0	
M3_WSSPD1		✓	M3_WSP1	
M4		✓		
APR_PG_M4_WSSPDef_0	✓	✓	APR_gs8ps8_M4_WSP_0	
M4_WSSPD1		✓	M4_WSP1	
M5		✓		
APR_M5_H_0_252	✓	✓	APR_M5_252_WSP1	
M6		✓		
APR_M6_V_0_252	✓	✓	APR_M6_252_WSP1	
M7		✓		

Related Topics

Configuring Standard Cell Router Settings

Checking Layout Routability after Running Standard Cell Placer

Generating a Supply Grid

Running Signal Routing for Standard Cells

Viewing and Analyzing Standard Cell Routing Results

Routing Assistant User Interface for Standard Cell

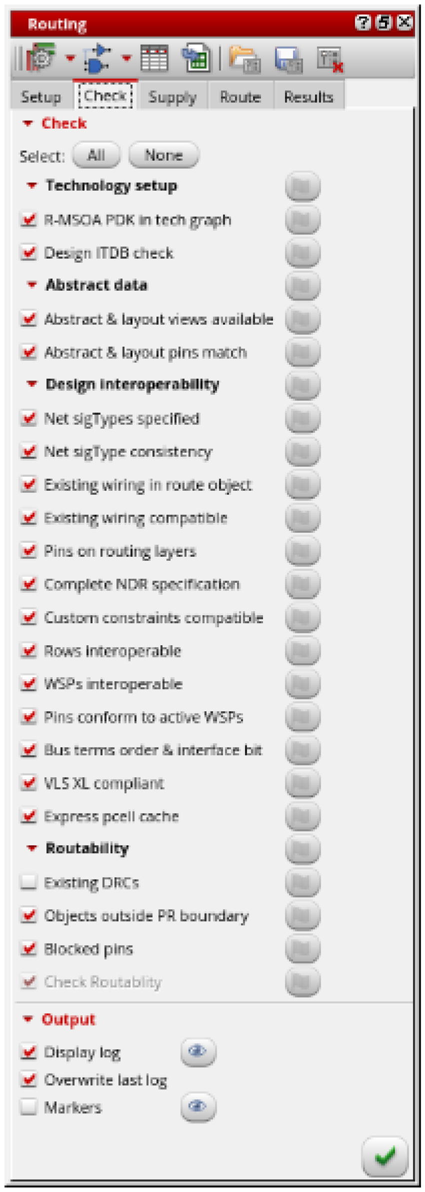
Checking Layout Routability after Running Standard Cell Placer

You can run pre-routing checks to detect design issues before routing a design. Running the checks lets you identify potential violations or issues that you might run into while routing the design or objects that might cause trouble for the router later in the flow. You can fix these issues before running the router.

To check the routability of a design:

1. Open a design in Layout MXL.
2. Choose *Window – Assistants – Routing*.

   Alternatively, right-click anywhere on the layout window menu bar and choose *Assistants – Routing*.
3. In the Routing assistant, click the *Check* tab.



Routing

Setup | Check | Supply | Route | Results

Check

Select: All None

Technology setup
- [✓] R-MSOA PDK in tech graph
- [✓] Design ITDB check

Abstract data
- [✓] Abstract & layout views available
- [✓] Abstract & layout pins match

Design interoperability
- [✓] Net sigTypes specified
- [✓] Net sigType consistency
- [✓] Existing wiring in route object
- [✓] Existing wiring compatible
- [✓] Pins on routing layers
- [✓] Complete NDR specification
- [✓] Custom constraints compatible
- [✓] Rows interoperable
- [✓] WSPs interoperable
- [✓] Pins conform to active WSPs
- [✓] Bus terms order & interface bit
- [✓] VLS XL compliant
- [✓] Express pcell cache

Routability
- [ ] Existing DRCs
- [✓] Objects outside PR boundary
- [✓] Blocked pins
- [✓] Check Routablity

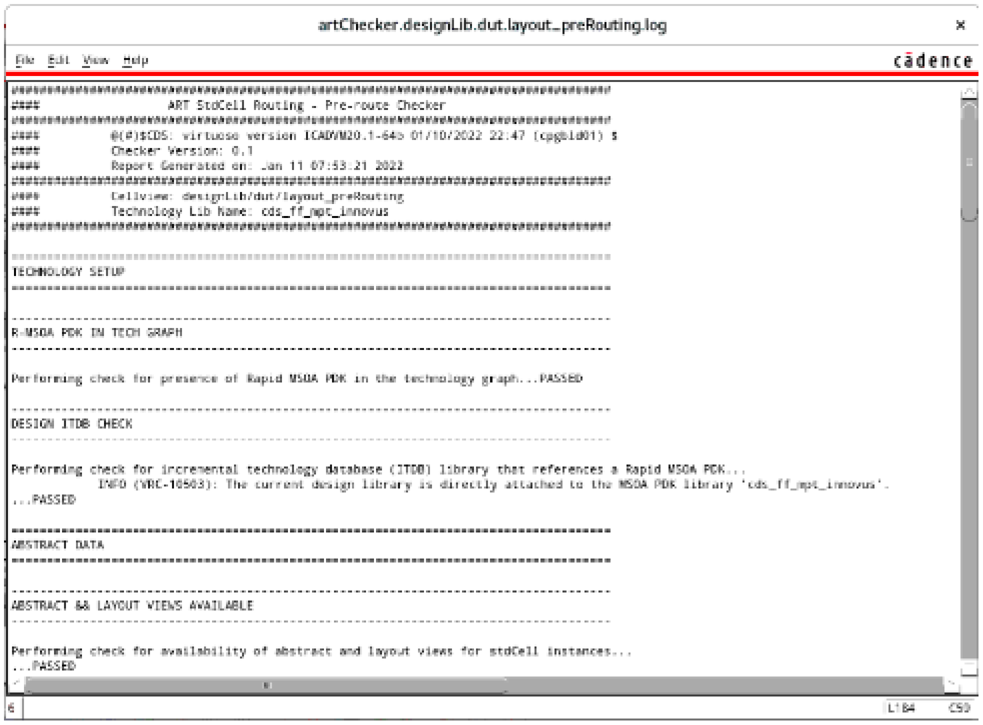
Output
- [✓] Display log
- [✓] Overwrite last log
- [ ] Markers

4. Click *All* to select all pre-routing checks or select one or more checks by clicking the check box next to each check.

5. In the *Output* section, specify the routing checks log settings.

   a. Select *Display log* to specify whether the routing log must be displayed.

   b. Click the *Display existing log file* button next to the *Display log* option to view the log file.You can use this log file to check for any issues after the pre-route checks are run.



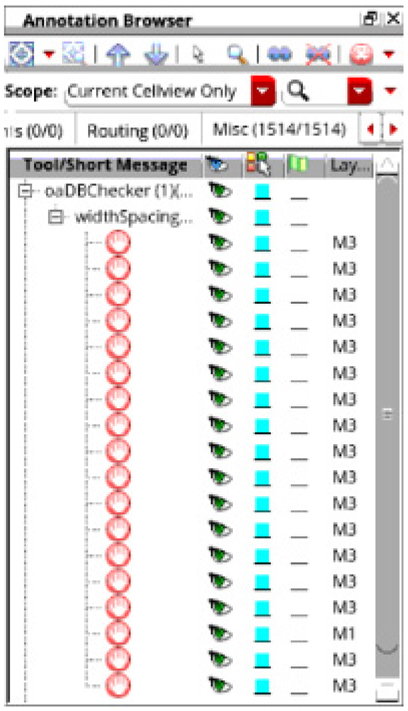
```
################################################################################
####                    ART StdCell Routing - Pre-route Checker
################################################################################
####            @(#)$CDS: virtuoso version ICADVM20.1-64b 01/10/2022 22:47 (cpgbld01) $
####            Checker Version: 0.1
####            Report Generated on: Jan 11 07:53:21 2022
################################################################################
####            Cellview: designLib/dut/layout_preRouting
####            Technology Lib Name: cds_ff_mpt_innovus
################################################################################

================================================================================
TECHNOLOGY SETUP
================================================================================

--------------------------------------------------------------------------------
R-MSOA PDK IN TECH GRAPH
--------------------------------------------------------------------------------

Performing check for presence of Rapid MSOA PDK in the technology graph...PASSED

--------------------------------------------------------------------------------
DESIGN ITDB CHECK
--------------------------------------------------------------------------------

Performing check for incremental technology database (ITDB) library that references a Rapid MSOA PDK...
            INFO (VRC-10503): The current design library is directly attached to the MSOA PDK library 'cds_ff_mpt_innovus'.
...PASSED

================================================================================
ABSTRACT DATA
================================================================================

--------------------------------------------------------------------------------
ABSTRACT && LAYOUT VIEWS AVAILABLE
--------------------------------------------------------------------------------

Performing check for availability of abstract and layout views for stdCell instances...
...PASSED
```

   Log file provides additional information with clickable links to show the issue in the layout. You can also view the issues in the Annotation Browser using the *Markers* option.

   c. Select *Overwrite log* to overwrite the existing routing log. When this option is not selected, a new log file is created.

   d. Select *Markers* to specify whether the markers must be displayed in the Annotation Browser.

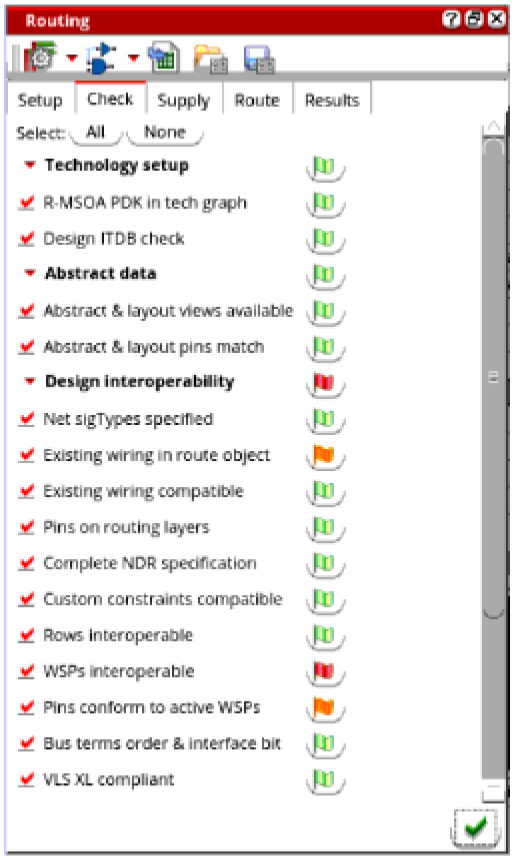
e. Click the *Show Annotation Browser* button next to the Markers option to view the issues reported for the pre-route checks. The issues are displayed in the *Misc* tab of the Annotation Browser.



Tool/Short Message	Lay...
oaDBChecker (1)(...	
widthSpacing...	
	M3
	M3
	M3
	M3
	M3
	M3
	M3
	M3
	M3
	M3
	M3
	M3
	M3
	M3
	M3
	M1
	M3
	M3

6. Click *Run pre-route checks* ✓.

Once the checks are run, the status flags appear in green, orange, and red. A green flag indicates that the check was passed, orange indicates a warning, and red indicates an error. Ensure that no red flags appear. This means that the design is correctly setup.



Technology setup	🟩
✓ R-MSOA PDK in tech graph	🟩
✓ Design ITDB check	🟩
Abstract data	🟩
✓ Abstract & layout views available	🟩
✓ Abstract & layout pins match	🟩
Design interoperability	🟥
✓ Net sigTypes specified	🟩
✓ Existing wiring in route object	🟧
✓ Existing wiring compatible	🟩
✓ Pins on routing layers	🟩
✓ Complete NDR specification	🟩
✓ Custom constraints compatible	🟩
✓ Rows interoperable	🟩
✓ WSPs interoperable	🟥
✓ Pins conform to active WSPs	🟧
✓ Bus terms order & interface bit	🟩
✓ VLS XL compliant	🟩

Clicking the flag button takes you to the location in the log file where that check result was reported.

Related Topics

Configuring Standard Cell Router Settings

Generating Width Spacing Patterns for Standard Cell Routing

Generating a Supply Grid

Running Signal Routing for Standard Cells

Viewing and Analyzing Standard Cell Routing Results

Virtuoso Pre-Route Browser

Routing Assistant User Interface for Standard Cell

Generating a Supply Grid

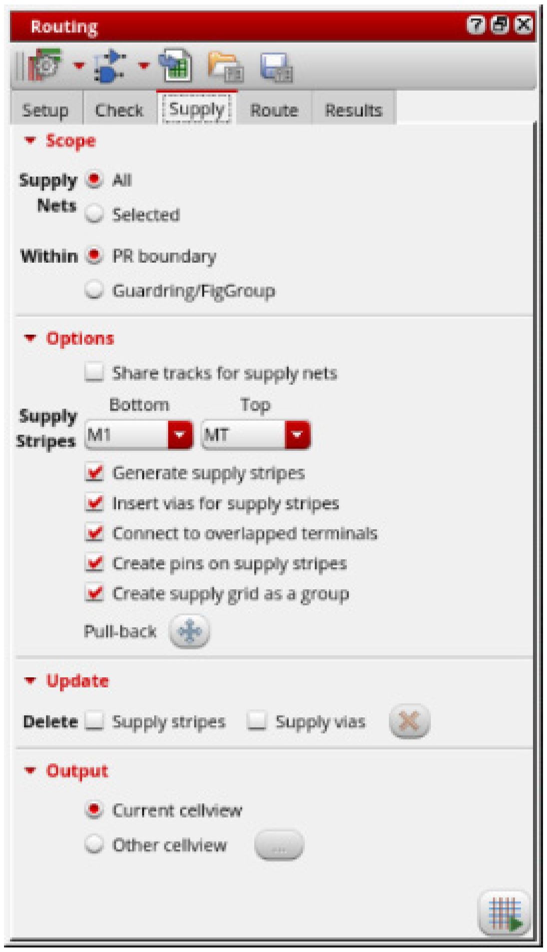
In certain standard cell libraries, boundary cells have blockages. In such case, the flow should be modified not to use rails in the row region and to place boundary cells before routing. Generally, a supply grid should be created before the standard cell placement so that placement can prevent routing from interfering with access to standard cell pins.

To generate power rails of VDD and GND nets:

1. Open a design in Layout MXL.
2. Choose *Window – Assistants – Routing*.

   Alternatively, right-click anywhere on the layout window menu bar and choose *Assistants – Routing*.

3. In the Routing assistant, click the *Supply* tab.



Scope

Supply Nets: (● All) ○ Selected

Within: (● PR boundary) ○ Guardring/FigGroup

Options

☐ Share tracks for supply nets

Supply Stripes: Bottom M1, Top MT

- ✓ Generate supply stripes
- ✓ Insert vias for supply stripes
- ✓ Connect to overlapped terminals
- ✓ Create pins on supply stripes
- ✓ Create supply grid as a group

Pull-back

Update

Delete: ☐ Supply stripes ☐ Supply vias

Output

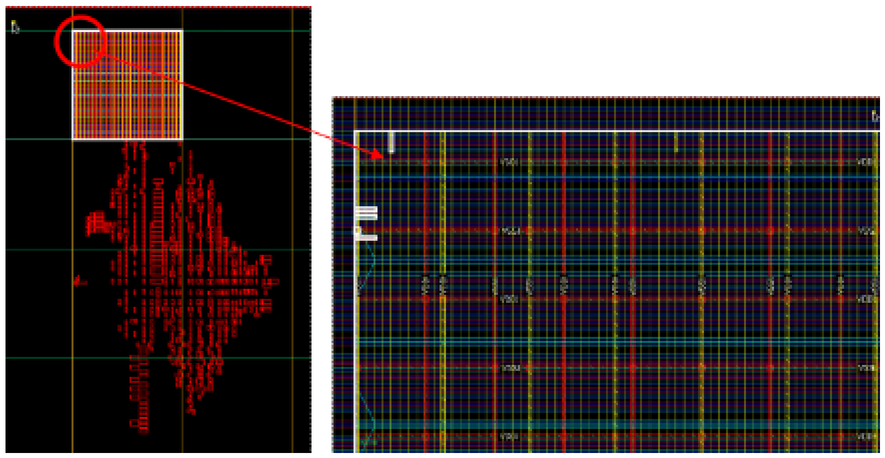
(● Current cellview) ○ Other cellview

4. Click *All* to define the scope of supply nets to route. You can also select to route *Selected* nets.
5. Click *Guardring/FigGroup* to route everything within a guard ring or figGroup.
6. Select the bottom and top layers for supply stripes from the *Supply Stripes Bottom* and *Top* drop-down list. It generates stripes only on tracks with power and ground wire types that are assigned to nets that have their `sigTypes` set to supply and ground.

   Automatic generation of WSPs is the simplest method to specify the wire types for the tracks. See Assigning Wire Types to Nets to find out how nets with `sigTypes` are assigned to wire types.
7. Select *Generate supply stripes*.
8. Select *Insert vias for supply stripes*.

9. Click *Run power route* at the lower-right corner of the *Supply* tab.

   You can see power stripes and vias generated inside the PR boundary.



Related Topics

Configuring Standard Cell Router Settings

Generating Width Spacing Patterns for Standard Cell Routing

Checking Layout Routability after Running Standard Cell Placer

Running Signal Routing for Standard Cells

Viewing and Analyzing Standard Cell Routing Results

Routing Assistant User Interface for Standard Cell

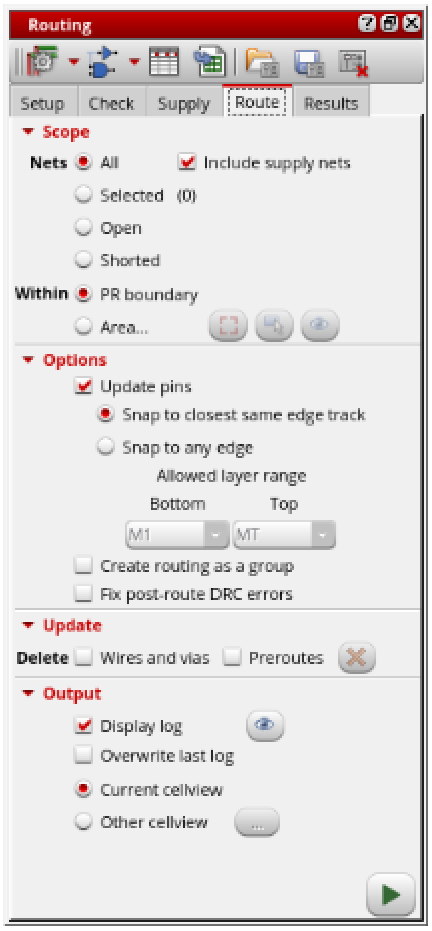
Running Signal Routing for Standard Cells

To run signal routing:

1. Open a design in Layout MXL.
2. Choose *Window – Assistants – Routing*.

   Alternatively, right-click anywhere on the layout window menu bar and choose *Assistants – Routing*.

3. In the Routing assistant, click the *Route* tab.



Routing

Setup | Check | Supply | Route | Results

Scope

Nets [✓ All] [✓ Include supply nets]
Selected (0)
Open
Shorted

Within [✓ PR boundary]
Area...

Options

[✓ Update pins]
[✓ Snap to closest same edge track]
Snap to any edge

Allowed layer range
Bottom: M1 Top: MT

Create routing as a group
Fix post-route DRC errors

Update

Delete: Wires and vias, Preroutes

Output

[✓ Display log]
Overwrite last log
[✓ Current cellview]
Other cellview

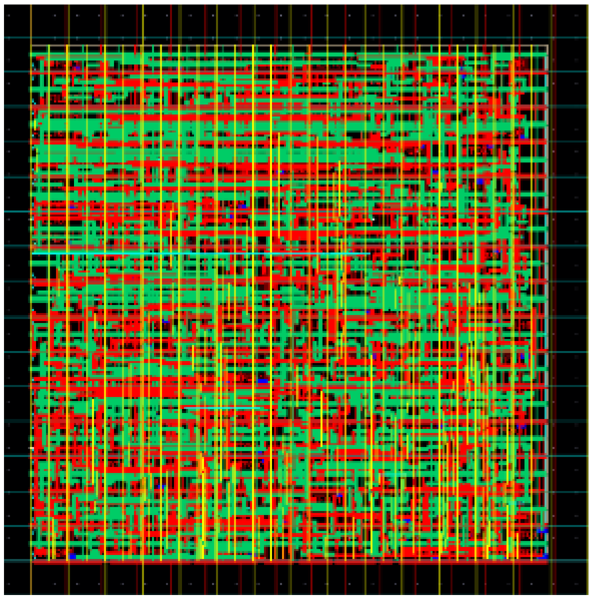
4. Click *Selected* to define the scope of nets to route. You can also select to route *All*, *Open*, or *Shorted* nets.

5. Select *Include Supply Nets* to route the `tieHi` and `tieLo` nets.

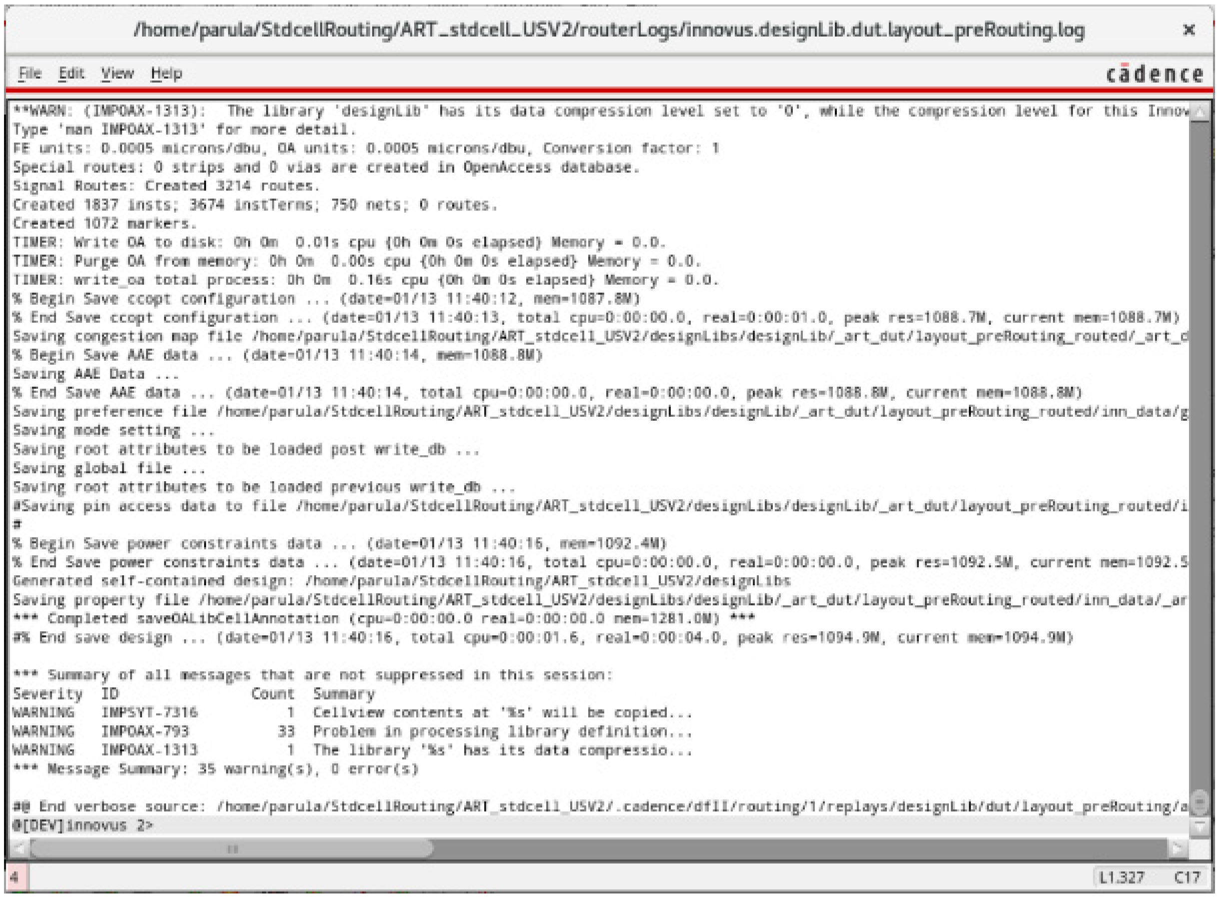
6. Deselect the *Update Pins* option. This option should be enabled only if WSPs are not used.

7. Click *Run signal route* ▶ at the lower-right corner of the *Route* tab.

The selected nets are routed.



While the router is running, the *Run signal route button changes to a* Stop ■ button. A directory called `routerLogs` is created in the run directory.


/home/parula/StdcellRouting/ART_stdcell_USV2/routerLogs/innovus.designLib.dut.layout_preRouting.log
File
Edit
View
Help
cādence
4
L1.327
C17

```
**WARN: (IMPOAX-1313):  The library 'designLib' has its data compression level set to '0', while the compression level for this Innov
Type 'man IMPOAX-1313' for more detail.
FE units: 0.0005 microns/dbu, OA units: 0.0005 microns/dbu, Conversion factor: 1
Special routes: 0 strips and 0 vias are created in OpenAccess database.
Signal Routes: Created 3214 routes.
Created 1837 insts; 3674 instTerms; 750 nets; 0 routes.
Created 1072 markers.
TIMER: Write OA to disk: 0h 0m  0.01s cpu {0h 0m 0s elapsed} Memory = 0.0.
TIMER: Purge OA from memory: 0h 0m  0.00s cpu {0h 0m 0s elapsed} Memory = 0.0.
TIMER: write_oa total process: 0h 0m  0.16s cpu {0h 0m 0s elapsed} Memory = 0.0.
% Begin Save ccopt configuration ... (date=01/13 11:40:12, mem=1087.8M)
% End Save ccopt configuration ... (date=01/13 11:40:13, total cpu=0:00:00.0, real=0:00:01.0, peak res=1088.7M, current mem=1088.7M)
Saving congestion map file /home/parula/StdcellRouting/ART_stdcell_USV2/designLibs/designLib/_art_dut/layout_preRouting_routed/_art_d
% Begin Save AAE data ... (date=01/13 11:40:14, mem=1088.8M)
Saving AAE Data ...
% End Save AAE data ... (date=01/13 11:40:14, total cpu=0:00:00.0, real=0:00:00.0, peak res=1088.8M, current mem=1088.8M)
Saving preference file /home/parula/StdcellRouting/ART_stdcell_USV2/designLibs/designLib/_art_dut/layout_preRouting_routed/inn_data/g
Saving mode setting ...
Saving root attributes to be loaded post write_db ...
Saving global file ...
Saving root attributes to be loaded previous write_db ...
#Saving pin access data to file /home/parula/StdcellRouting/ART_stdcell_USV2/designLibs/designLib/_art_dut/layout_preRouting_routed/i
#
% Begin Save power constraints data ... (date=01/13 11:40:16, mem=1092.4M)
% End Save power constraints data ... (date=01/13 11:40:16, total cpu=0:00:00.0, real=0:00:00.0, peak res=1092.5M, current mem=1092.5
Generated self-contained design: /home/parula/StdcellRouting/ART_stdcell_USV2/designLibs
Saving property file /home/parula/StdcellRouting/ART_stdcell_USV2/designLibs/designLib/_art_dut/layout_preRouting_routed/inn_data/_ar
*** Completed saveOALibCellAnnotation (cpu=0:00:00.0 real=0:00:00.0 mem=1281.0M) ***
#% End save design ... (date=01/13 11:40:16, total cpu=0:00:01.6, real=0:00:04.0, peak res=1094.9M, current mem=1094.9M)

*** Summary of all messages that are not suppressed in this session:
Severity  ID              Count  Summary
WARNING   IMPSYT-7316         1  Cellview contents at '%s' will be copied...
WARNING   IMPOAX-793         33  Problem in processing library definition...
WARNING   IMPOAX-1313         1  The library '%s' has its data compressio...
*** Message Summary: 35 warning(s), 0 error(s)

#@ End verbose source: /home/parula/StdcellRouting/ART_stdcell_USV2/.cadence/dfII/routing/1/replays/designLib/dut/layout_preRouting/a
@[DEV]innovus 2>
```

Routing errors are reported in the CIW, which might be hidden underneath other windows. The CIW can be raised automatically by setting the following environment variable:

```
envSetVal("ui" "raiseCIWonError" 'boolean t)
```

8. In the *Update* section of the *Route* tab, select the *Delete Wires and vias*.
9. Click the *Delete* button to delete the routed wires and vias for the selected nets.

   The routed data is deleted.

Related Topics

Configuring Standard Cell Router Settings

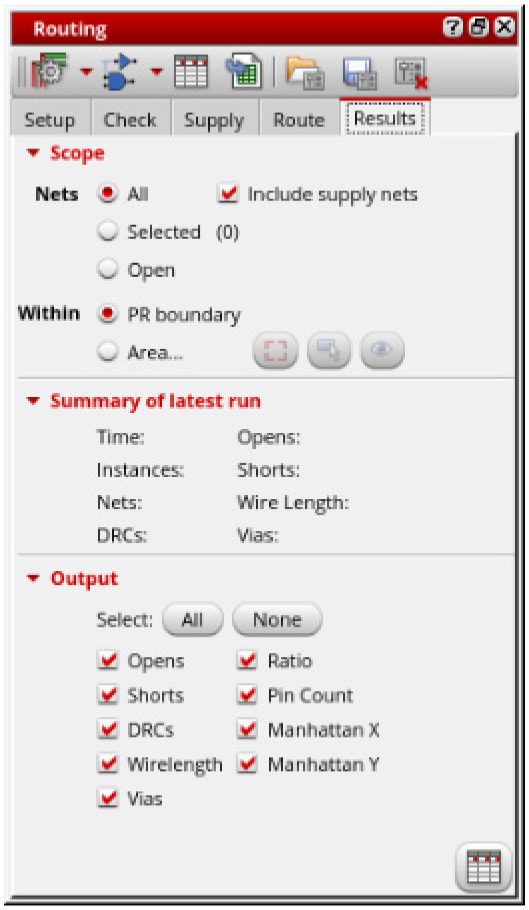
Generating Width Spacing Patterns for Standard Cell Routing

Viewing and Analyzing Standard Cell Routing Results

You can analyze the routing results using the Routing Results Browser. It shows the number of opens, shorts, wire length, DRC, vias, and so on.

To view and analyze the results:

1. Click the *Results* tab in the Routing Assistant.



Scope

Nets: [✓ All] [✓ Include supply nets] Selected (0) Open

Within: [✓ PR boundary] Area...

Summary of latest run

Time:	Opens:
Instances:	Shorts:
Nets:	Wire Length:
DRCs:	Vias:

Output

Select: All None

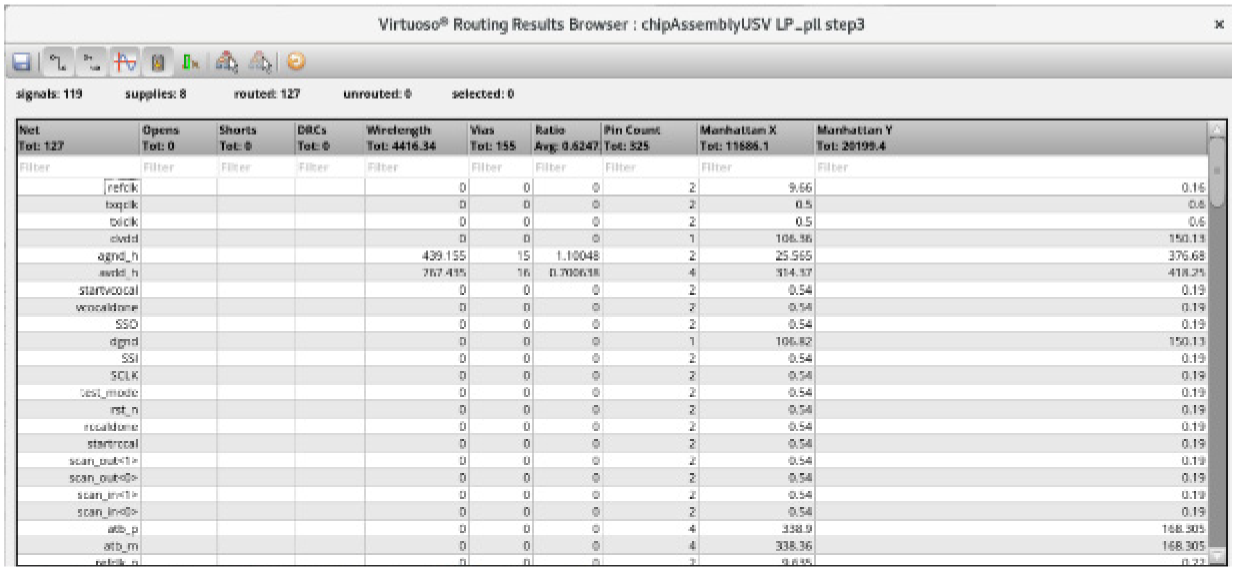
✓ Opens	✓ Ratio
✓ Shorts	✓ Pin Count
✓ DRCs	✓ Manhattan X
✓ Wirelength	✓ Manhattan Y
✓ Vias	

2. Select the scope of nets to analyze the routing result. You can either select *All*, *Selected*, or *Open* nets.
3. Select *Supply Nets* to see the routing results of the power and ground nets.
4. Select *All* to display all the output columns in the Routing Results Browser.

   You can also select specific outputs for which you want the columns to be displayed in the Routing Results Browser.
5. Click *Show results browser* at the bottom right corner.

   The Routing Results Browser is displayed. Examine the spreadsheet and observe the wirelength, via count, and ratio (routed length and ideal length). The ratio can be calculated with MST, Steiner, or Spine ideal routes.

   You can also see the total number of routed nets, opens, shorts and details of various violations.


Virtuoso® Routing Results Browser : chipAssemblyUSV LP_pll step3

signals: 119 supplies: 8 routed: 127 unrouted: 0 selected: 0

Net Tot: 127	Opens Tot: 0	Shorts Tot: 0	DRCs Tot: 0	Wirelength Tot: 4416.34	Vias Tot: 155	Ratio Avg: 0.6247	Pin Count Tot: 525	Manhattan X Tot: 11686.1	Manhattan Y Tot: 20199.4
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
refclk				0	0	0	2	9.66	0.16
txqclk				0	0	0	2	0.5	0.6
txclk				0	0	0	2	0.5	0.6
clvdd				0	0	0	1	106.36	150.13
agnd_h				439.155	15	1.10048	2	25.565	376.68
avdd_h				767.415	16	0.700618	4	314.37	418.25
startvcocal				0	0	0	2	0.54	0.19
vcocaldone				0	0	0	2	0.54	0.19
SSO				0	0	0	2	0.54	0.19
dgnd				0	0	0	1	106.82	150.13
SSI				0	0	0	2	0.54	0.19
SCLK				0	0	0	2	0.54	0.19
test_mode				0	0	0	2	0.54	0.19
rst_n				0	0	0	2	0.54	0.19
rccaldone				0	0	0	2	0.54	0.19
startrccal				0	0	0	2	0.54	0.19
scan_out<1>				0	0	0	2	0.54	0.19
scan_out<0>				0	0	0	2	0.54	0.19
scan_in<1>				0	0	0	2	0.54	0.19
scan_in<0>				0	0	0	2	0.54	0.19
atb_p				0	0	0	4	338.9	168.305
atb_m				0	0	0	4	338.36	168.305
refclk_n				0	0	0	2	9.635	0.22

Related Topics

[Generating Width Spacing Patterns for Standard Cell Routing]()

[Checking Layout Routability after Running Standard Cell Placer]()

[Generating a Supply Grid]()

[Running Signal Routing for Standard Cells]()

Routing Assistant User Interface for Standard Cell

Virtuoso Routing Results Browser

4

Standard Cell Placement and Routing Environment Variables

The standard cell placement and routing environment variables are used to set default values for various standard cell placement and routing options in the *Auto P&R* and the *Routing* assistants.

Only the public environment variables are documented and supported for public use. All other standard cell routing environment variables, regardless of their name or prefix, are private and undocumented and are subject to change at any time.

The following list provides the names of the Automated Standard Cell Placement and Routing Flow environment variables.

Standard Cell Placement

advNode	init_boundaryAspectRatioOrHeight
init_boundaryAspectRatioVal	init_boundaryHeightVal
init_boundaryUtilizationOrWidth	init_boundaryUtilizationVal
init_boundaryWidthVal	init_createPowerPins
init_generateBoundary	init_generateInstances
init_generatePins	init_mode
init_scope	init_useSourceLayout
init_useSourceLayoutBoundary	init_useSourceLayoutInstances
init_useSourceLayoutPins	physOnlyFigGroupPrefix
place_addBoundaryCells	place_addCellRowRouting
place_addDecapFillerCells	place_addTapCells
	place_boundaryCellsContainTapCells
place_createPhysOnlyAsAGroup	place_defaultPlacedView

place_deleteBoundaryCells	place_deleteFillerCells
place_deleteTapCells	place_displayLog
place_overwriteLog	place_placeStdCells
place_placedLocation	place_showInformation
postPlaceTrigger	prePlaceTrigger
remasterLayoutLibs	runDir
	setup_Innovus1stRowOrientation
setup_createRowRegionInArea	setup_displayLog
setup_overwriteLog	setup_rowCount
setup_rowCreation	

Standard Cell Routing

FB1RouteBlockageLayer	abstractViewName	alternativeBoundaryLayer
check_displayLog	check_existingDRCs	check_generateMarkers
check_overwriteLog	checkerLogDir	checkerLogPrefix
coverTermPins	createNewFigGroupsPerRun	disableInvsWSPGen
fixAllPreroutes	invsLicenseArgs	layoutViewName
ldrsCG	multiTechLEFOverride	omitRedundantPatchShapes
omitTrimLayers	postInitCmds	postPlacementCmds
postRouteCmds	postRouteTrigger	preBndyTapCmds
preInitCmds	prePlacementCmds	preRouteCmds
preRouteTrigger	preserveFillerPlaceStatus	results_nets
results_netsWithin	results_supplyNets	route_createRoutingAsAGroup
route_defaultRoutedView	route_deletePreroutes	route_deleteWiresAndVias
route_displayLog	route_fixPostRouteDRCErrors	route_nets
route_netsWithin	route_overwriteLog	route_routedLOC
route_saveRoutingOnly	route_supplyNets	route_updatePinOption

route_updatePins	routerLogPrefix	runDir
separateStdCellLibs	setPinFixedWithinBoundary	setup_checkDRCsAfterRouting
setup_lockColorsAfterRouting	signalRouteFigGroupPrefix	siteDefHeight
siteDefSymmetricInR90	siteDefSymmetricInX	siteDefSymmetricInY
siteDefWidth	supply_connectToTerminals	supply_createGridAsGroup
supply_createPinLabel	supply_createPins	supply_createPinsOnEnds
supply_defaultRoutedView	supply_deleteStripes	supply_deleteVias
supply_genSupplyStripes	supply_generateStaples	supply_IgnoreBoundaryTracks
supply_IgnoreBoundaryVias	supply_insertTrim	supply_insertVias
supply_nets	supply_netsWithin	supply_pinLayerSet
supply_routedLOC	supply_saveRoutingOnly	supply_shareTracks
supply_useExisitingPGTracks	tracksCmds	useMultipleTapCells
useStylusMode		

FB1RouteBlockageLayer

`APR.stdcell.route FB1RouteBlockageLayer string "`*`layername`*`"`

Description

Specifies the layers option for `create_route_blockage` command in the Tcl command file generated for Innovus. If empty, the `create_route_blockage` command is not added.

The default is `FB1`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "FB1RouteBlockageLayer")
envSetVal("APR.stdcell.route" "FB1RouteBlockageLayer" 'string "layer1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

FB1SpecialDRCRegionLayer

`APR.stdcell.route FB1SpecialDRCRegionLayer string "region`*`layername`*`"`

Description

Creates a donut of FB1 region around a region of rows if it is significantly smaller than the prBoundary to apply FB1 rules outside.

The default is `""`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "FB1SpecialDRCRegionLayer")
envSetVal("APR.stdcell.route" "FB1SpecialDRCRegionLayer" 'string "FB1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

abstractViewName

`APR.stdcell.route abstractViewName string "`*`abstractname`*`"`

Description

Specifies the name of the abstract cellview of Innovus. The default is `abstract`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "abstractViewName")
envSetVal("APR.stdcell.route" "abstractViewName" 'string "view1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

alternativeBoundaryLayer

```
APR.stdcell.route alternativeBoundaryLayer string "layername"
```

Description

Specifies the alternative boundary layer if the PR boundary does not exist. This value is used to calculate the bounding box of a cellview when the bBox is unavailable.

The default is `""`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "alternativeBoundaryLayer")
envSetVal("APR.stdcell.route" "alternativeBoundaryLayer" 'string
"fboundarylayer_1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

check_displayLog

```
APR.stdcell.route check_displayLog boolean { t | nil }
```

Description

Controls the display of the checker log window once the checks are run. When set to `nil`, the log window is not displayed.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Check* tab
Field	*Display Log*

Examples

```
envGetVal("APR.stdcell.route" "check_displayLog")
envSetVal("APR.stdcell.route" "check_displayLog" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

check_existingDRCs

```
APR.stdcell.route check_existingDRCs boolean { t | nil }
```

Description

Specifies whether to run design rule checks for DRD.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Check* tab
Field	*Existing DRCs*

Examples

```
envGetVal("APR.stdcell.route" "check_existingDRCs")
envSetVal("APR.stdcell.route" "check_existingDRCs" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

check_generateMarkers

```
APR.stdcell.route check_generateMarkers boolean { t | nil }
```

Description

Controls the generation of markers for errors. These error markers can be viewed in the *Misc* tab of Annotation Browser. When set to `t`, the markers are generated.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Check* tab
Field	*Markers*

Examples

```
envGetVal("APR.stdcell.route" "check_generateMarkers")
envSetVal("APR.stdcell.route" "check_generateMarkers" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

check_overwriteLog

`APR.stdcell.route check_overwriteLog boolean { t | nil }`

Description

Controls the overwriting of the last log file. When set to `nil`, the existing log file is retained.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Check* tab
Field	*Overwrite last log*

Examples

```
envGetVal("APR.stdcell.route" "check_overwriteLog")
envSetVal("APR.stdcell.route" "check_overwriteLog" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

checkerLogDir

```
APR.stdcell.route checkerLogDir string "directory"
```

Description

Specifies the path to the directory that saves the checker log file from the *Check* tab of the standard cell router.

The default directory is `./checkerLogs`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "checkerLogDir")
envSetVal("APR.stdcell.route" "checkerLogDir" 'string "log1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

checkerLogPrefix

`APR.stdcell.route checkerLogPrefix string "`*`logprefixname`*`"`

Description

Specifies the prefix for the checker log file names. The checker log files follow the pattern of *`checkerLogPrefix.libName.cellName.viewName`*`.log`.

The default is `artChecker`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "checkerLogPrefix")
envSetVal("APR.stdcell.route" "checkerLogPrefix" 'string "deviceChecker")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

coverTermPins

```
APR.stdcell.route coverTermPins boolean { t | nil }
```

Description

Extends the wire to completely cover the IO pins. The standard cell router routes to the center of the pin with 0-extent wires.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "coverTermPins")
envSetVal("APR.stdcell.route" "coverTermPins" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

createNewFigGroupsPerRun

```
APR.stdcell.route createNewFigGroupsPerRun boolean { t | nil }
```

Description

Creates a new fig group for the routing results. The results of each routing run is created in a different figGroup.

The default is `nil`, which means that the routing results are copied into the same figGroup along with the other routing results.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "createNewFigGroupsPerRun")
envSetVal("APR.stdcell.route" "createNewFigGroupsPerRun" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

disableInvsWSPGen

```
APR.stdcell.route disableInvsWSPGen boolean { t | nil }
```

Description

Disables Innovus derived tracks. When set to `t`, the Innovus derived tracks are considered by the router to generate WSPs.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "disableInvsWSPGen")
envSetVal("APR.stdcell.route" "disableInvsWSPGen" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

fixAllPreroutes

```
APR.stdcell.route fixAllPreroutes boolean { t | nil }
```

Description

Fixes pre-routes for the current routing run. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "fixAllPreroutes")
envSetVal("APR.stdcell.route" "fixAllPreroutes" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

invsLicenseArgs

`APR.stdcell.route invsLicenseArgs string "`*`licensename`*`"`

Description

Specifies the name of the Innovus license.

The default is `layout`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "invsLicenseArgs")
envSetVal("APR.stdcell.route" "invsLicenseArgs" 'string "licenseA")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

layoutViewName

```
APR.stdcell.route layoutViewName string "layoutname"
```

Description

Specifies the name of the Innovus layout cellview.

The default is `layout`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "layoutViewName")
envSetVal("APR.stdcell.route" "layoutViewName" 'string "layout1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

ldrsCG

```
APR.stdcell.route ldrsCG string "CGname"
```

Description

Specifies the name of the override `LEFDefaultRouteSpec` constraint group name.

The default is `""`. This means no constraint group name is specified.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "ldrsCG")
envSetVal("APR.stdcell.route" "ldrsCG" 'string "constraint1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

multiTechLEFOverride

```
APR.stdcell.route multiTechLEFOverride string "list_of_CGnames"
```

Description

Specifies the constraint group name that overrides `foundryRule`, `LEFDefaultRouteSpec` and `LEFSpecialRouteSpec`. The value is parsed and passed to Innovus as `init_oa_foundry_rule`, `init_oa_default_rule`, `init_oa_special_rule`. It overrides what is specified in the GUI.

The default is an empty string `""`. This means a constraint group name is not specified.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "multiTechLEFOverride")
```

```
envSetVal("APR.stdcell.route" "multiTechLEFOverride" 'string "foundry_innovus_1
LEFDefaultRouteSpec_1 LEFSpecialRouteSpec_1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

omitRedundantPatchShapes

```
APR.stdcell.route omitRedundantPatchShapes boolean { t | nil }
```

Description

Omits synchronization of redundant patch shapes from the Innovus routed view. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "omitRedundantPatchShapes")
envSetVal("APR.stdcell.route" "omitRedundantPatchShapes" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

omitTrimLayers

```
APR.stdcell.route omitTrimLayers boolean { t | nil }
```

Description

Omits trim layers and bridge metal from syncing routing results back from Innovus.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "omitTrimLayers")
envSetVal("APR.stdcell.route" "omitTrimLayers" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

physOnlyFigGroupPrefix

`APR.stdcell.place physOnlyFigGroupPrefix string "`*`GroupPrefixname`*`"`

Description

Specifies a unique figGroup prefix for the name of the figGroup.

The default is `place`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.place" "physOnlyFigGroupPrefix")
envSetVal("APR.stdcell.place" "physOnlyFigGroupPrefix" 'string "place1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

postInitCmds

```
APR.stdcell.route postInitCmds string "filename"
```

Description

Specifies the path to the post-initialization hook file that is sourced after design initialization stage. The file can be placed anywhere and can be specified as an absolute path or a relative path from the current working directory.

The default is `""`. This means a filename is not specified.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "postInitCmds")
envSetVal("APR.stdcell.route" "postInitCmds" 'string "postinit1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

postPlacementCmds

`APR.stdcell.route postPlacementCmds string "`*`filename`*`"`

Description

Specifies the path to the post-placement hook file that is sourced after the placement stage. The file can be placed anywhere and can be specified as an absolute path or a relative path from the current working directory.

The default is `""`. This means a filename is not specified.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "postPlacementCmds")
envSetVal("APR.stdcell.route" "postPlacementCmds" 'string "postPlace1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

postRouteCmds

```
APR.stdcell.route postRouteCmds string "filename"
```

Description

Specifies the post-route hook file that is sourced after the routing stage. The file can be placed anywhere and can be specified as an absolute path or a relative path from the current working directory.

The default is `""`. This means a filename is not specified.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "postRouteCmds")
envSetVal("APR.stdcell.route" "postRouteCmds" 'string "postRoute1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

postRouteTrigger

`APR.stdcell.route postRouteTrigger string "`*`procedurename`*`"`

Description

Specifies SKILL procedure to be executed after routing has completed.

The default is `""`, which means no additional processing is done.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "postRouteTrigger")
envSetVal("APR.stdcell.route" "postRouteTrigger" 'string "postRouteProcedure")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

preBndyTapCmds

```
APR.stdcell.route preBndyTapCmds string "tapCmdNames"
```

Description

Sources the Tcl plugin to be sourced before running the `add_endcaps` and `add_well_taps` commands. The default is an empty string `""`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "preBndyTapCmds")
envSetVal("APR.stdcell.route" "preBndyTapCmds" 'string "foundry_innovus_1
LEFDefaultRouteSpec_1 ")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

preInitCmds

`APR.stdcell.route preInitCmds string "`*`filename`*`"`

Description

Specifies the path to the pre-initialization hook file that is sourced before design initialization stage. The file can be placed anywhere and can be specified as an absolute path or a relative path from the current working directory.

The default is `""`. This means a filename is not specified.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "preInitCmds")
envSetVal("APR.stdcell.route" "preInitCmds" 'string "preinit1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

prePlacementCmds

```
APR.stdcell.route prePlacementCmds string "filename"
```

Description

Specifies the path to the pre-placement hook file that is sourced before the placement stage. The file can be placed anywhere and can be specified as an absolute path or a relative path from the current working directory.

The default is `""`. This means a filename is not specified.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "prePlacementCmds")
envSetVal("APR.stdcell.route" "prePlacementCmds" 'string "prePlace1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

preRouteCmds

```
APR.stdcell.route preRouteCmds string "filename"
```

Description

Specifies the path to the pre-route hook file that is sourced before the routing stage. The file can be placed anywhere and can be specified as an absolute path or a relative path from the current working directory.

The default is `""`. This means a filename is not specified.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "preRouteCmds")
envSetVal("APR.stdcell.route" "preRouteCmds" 'string "route1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

preRouteTrigger

```
APR.stdcell.route preRouteTrigger string "procedurename"
```

Description

Specifies SKILL procedure to be executed before routing.

The default is `""`, which means no additional processing is done.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "preRouteTrigger")
envSetVal("APR.stdcell.route" "preRouteTrigger" 'string "preRouteProcedure")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

preserveFillerPlaceStatus

```
APR.stdcell.route preserveFillerPlaceStatus boolean { t | nil }
```

Description

Preserves the placement status of the filler cells in the scratch view instead of changing it to fixed. It can be changed in Innovus. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "preserveFillerPlaceStatus")
envSetVal("APR.stdcell.route" "preserveFillerPlaceStatus" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

remasterLayoutLibs

```
APR.stdcell.place remasterLayoutLibs string "libname"
```

Description

Specifies a library that contains layout views. It is similar to `separateStdCellLibs`, except that it specifies layout library names instead of abstract library names if layout and abstract views are not in the same library. It is used for the final `remaster_instance` when the data is synced back to virtuoso after the placement is run.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.place" "remasterLayoutLibs")
envSetVal("APR.stdcell.place" "remasterLayoutLibs" 'string "Layout1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

separateStdCellLibs

results_nets

```
APR.stdcell.route results_nets cyclic { "All" | "Selected" | "Open" }
```

Description

Specifies whether you want all nets, selected nets, or only open nets to be included in the results table.

The default is `"All"`.

GUI Equivalent

Command	Routing assistant – *Results* tab
Field	*Nets – All*, *Selected*, *Open*

Examples

```
envGetVal("APR.stdcell.route" "results_nets")
envSetVal("APR.stdcell.route" "results_nets" 'cyclic "Selected")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

results_netsWithin

```
APR.stdcell.route results_netsWithin cyclic { "PR boundary" | "Area" }
```

Description

Specifies whether to route everything inside the PR boundary or within an area.

The default is `"PR boundary"`.

GUI Equivalent

Command	Routing assistant – *Results* tab
Field	*Within*

Examples

```
envGetVal("APR.stdcell.route" "results_netsWithin")
envSetVal("APR.stdcell.route" "results_netsWithin" 'cyclic "Area")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

results_supplyNets

```
APR.stdcell.route results_supplyNets boolean { t | nil }
```

Description

Includes power and ground in the results table. When set to `nil`, power and ground nets are not included in the results table.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Results* tab
Field	*Supply nets*

Examples

```
envGetVal("APR.stdcell.route" "results_supplyNets")
envSetVal("APR.stdcell.route" "results_supplyNets" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

route_createRoutingAsAGroup

```
APR.stdcell.route route_createRoutingAsAGroup boolean { t | nil }
```

Description

Specifies whether or not to create routing as a figGroup. When set to `t`, the routing is created as a figGroup.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Create routing as a group*

Examples

```
envGetVal("APR.stdcell.route" "route_createRoutingAsAGroup")
envSetVal("APR.stdcell.route" "route_createRoutingAsAGroup" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

signalRouteFigGroupPrefix

route_defaultRoutedView

`APR.stdcell.route route_defaultRoutedView string "`*`defaultViewName`*`"`

Description

Specifies the default name for the routed view if signal routing is written to another cellview.

The default is `layout.routed`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Output – Other cellview*

Examples

```
envGetVal("APR.stdcell.route" "route_defaultRoutedView")
envSetVal("APR.stdcell.route" "route_defaultRoutedView" 'string "route2")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

route_deletePreroutes

```
APR.stdcell.route route_deletePreroutes boolean { t | nil }
```

Description

Specifies that the pre-routed wires and vias that are manually created are automatically deleted.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Delete – Preroutes*

Examples

```
envGetVal("APR.stdcell.route" " route_deletePreroutes")
envSetVal("APR.stdcell.route" " route_deletePreroutes" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

route_deleteWiresAndVias

```
APR.stdcell.route route_deleteWiresAndVias boolean { t | nil }
```

Description

Specifies that wires, vias, and shield lines created by the router are automatically deleted.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Delete – Wires and vias*

Examples

```
envGetVal("APR.stdcell.route" " route_deleteWiresAndVias")
envSetVal("APR.stdcell.route" " route_deleteWiresAndVias" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

route_displayLog

```
APR.stdcell.route route_displayLog boolean { t | nil }
```

Description

Controls the display of the log window when standard cell routing is run. When set to `nil`, the log window is not displayed.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Display Log*

Examples

```
envGetVal("APR.stdcell.route" "route_displayLog")
envSetVal("APR.stdcell.route" "route_displayLog" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

route_fixPostRouteDRCErrors

```
APR.stdcell.route route_fixPostRouteDRCErrors boolean { t | nil }
```

Description

Runs `fix_errors` command to resolve existing shorts and spacing violations after running the `route_design` command. The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Fix post-route DRC errors*

Examples

```
envGetVal("APR.stdcell.route" "route_fixPostRouteDRCErrors")
envSetVal("APR.stdcell.route" "route_fixPostRouteDRCErrors" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

route_nets

```
APR.stdcell.route route_nets cyclic { "All" | "Selected" | "Open" | "Shorted }
```

Description

Specifies whether you want to route all nets, selected nets, or only nets with opens or shorts.

The default is `"All"`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Nets – All*, *Selected*, *Open, Shorted*

Examples

```
envGetVal("APR.stdcell.route" "route_nets")
envSetVal("APR.stdcell.route" "route_nets" 'cyclic "Selected")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

route_netsWithin

```
APR.stdcell.route route_netsWithin cyclic { "PR boundary" | "FigGroup" | "Area"}
```

Description

Specifies whether to route everything inside the PR boundary or only within a specified figGroup or area.

The default is `"PR boundary"`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Within*

Examples

```
envGetVal("APR.stdcell.route" "route_netsWithin")
envSetVal("APR.stdcell.route" "route_netsWithin" 'cyclic "FigGroup")
envSetVal("APR.stdcell.route" "route_netsWithin" 'cyclic "Area")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

route_overwriteLog

```
APR.stdcell.route route_overwriteLog boolean { t | nil }
```

Description

Specifies whether to overwrite or keep the existing log. When set to `t`, the existing log is overwritten.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Overwrite last log*

Examples

```
envGetVal("APR.stdcell.route" "route_overwriteLog")
envSetVal("APR.stdcell.route" "route_overwriteLog" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

route_routedLOC

```
APR.stdcell.route route_routedLOC cyclic { "Current cellview" | "Other cellview" }
```

Description

Specifies whether to write the signal routing to the current cellview or to a different cellview.

The default is `"Current cellview"`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Current cellview, Other cellview*

Examples

```
envGetVal("APR.stdcell.route" "route_routedLOC")
envSetVal("APR.stdcell.route" "route_routedLOC" 'cyclic "Other cellview")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

route_saveRoutingOnly

```
APR.stdcell.route route_saveRoutingOnly boolean { t | nil }
```

Description

Specifies whether or not to save standard cell routing settings.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Save routing only*

Examples

```
envGetVal("APR.stdcell.route" "route_saveRoutingOnly")
envSetVal("APR.stdcell.route" "route_saveRoutingOnly" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

route_supplyNets

```
APR.stdcell.route route_supplyNets boolean { t | nil }
```

Description

Specifies whether to route power and ground (`tieHi` and `tieLo`) nets. When set to `nil`, the power and ground nets are not routed.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Supply Nets*

Examples

```
envGetVal("APR.stdcell.route" "route_supplyNets")
envSetVal("APR.stdcell.route" "route_supplyNets" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

route_updatePinOption

```
APR.stdcell.route route_updatePinOption cyclic { "Snap to closest same edge track"
     | "Snap to any edge" }
```

Description

Specifies whether to snap IO pins to the closest track on the same layer and same edge or to any edge on any layer within the specified layer range. This variable is effective only if `route_updatePins` is set to `t`.

The default is `"Snap to closest same edge track"`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Update Pins – Snap to closest same edge track*, *Snap to any edge*

Examples

```
envGetVal("APR.stdcell.route" "route_updatePinOption")
envSetVal("APR.stdcell.route" "route_updatePinOption" 'cyclic "Snap to any edge")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

route_updatePins

```
APR.stdcell.route route_updatePins boolean { t | nil }
```

Description

Controls pin placement in Innovus. When set to `nil`, the pin placement does not happen.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Update Pins*

Examples

```
envGetVal("APR.stdcell.route" "route_updatePins")
envSetVal("APR.stdcell.route" "route_updatePins" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

routerLogPrefix

`APR.stdcell.route routerLogPrefix string "`*`logprefixname`*`"`

Description

Specifies the prefix for the router log file name. The router log files follow the pattern of *`routerLogPrefix.libName.cellName.viewName`*`.logxx` where xx is a number.

The default is `innovus`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "routerLogPrefix")
envSetVal("APR.stdcell.route" "routerLogPrefix" 'string "inn")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

runDir

`APR.stdcell.route runDir string "`*`dirname`*`"`

Description

Specifies the path to the Innovus router log directory. The default directory is `./routerLogs`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "runDir")
envSetVal("APR.stdcell.route" "runDir" 'string "run1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

separateStdCellLibs

`APR.stdcell.route separateStdCellLibs string "`*`libname`*`"`

Description

Specifies a library that contains abstract views. If the abstract views are not in the same library as the layout views and not in any library in the technology graph then this environment variable must be set. If no abstracts are available then the `cellType` of the layout view must be set to `core` or the `INVS_QAI_CELL_OVERRIDE` cellview property must be set.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "separateStdCellLibs")
envSetVal("APR.stdcell.route" "separateStdCellLibs" 'string "run1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

setPinFixedWithinBoundary

```
APR.stdcell.route setPinFixedWithinBoundary boolean { t | nil }
```

Description

Changes the placement status of IO pins to fixed if they are contained within the PR boundary. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "setPinFixedWithinBoundary")
envSetVal("APR.stdcell.route" "setPinFixedWithinBoundary" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

setup_checkDRCsAfterRouting

```
APR.stdcell.route setup_checkDRCsAfterRouting boolean { t | nil }
```

Description

Specifies whether to automatically run design rule checks after routing. When set to `t`, the design rule checks are run automatically.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Setup* tab
Field	*Check DRCs after routing*

Examples

```
envGetVal("APR.stdcell.route" "setup_checkDRCsAfterRouting")
envSetVal("APR.stdcell.route" "setup_checkDRCsAfterRouting" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

setup_lockColorsAfterRouting

```
APR.stdcell.route setup_lockColorAfterRouting boolean { t | nil }
```

Description

Specifies whether to automatically lock colored shapes after routing. If the process has trim shapes that cuts locked color shapes, then the GUI option is selected and disabled. If the process does not have trim shapes, the GUI option is unselected and disabled. If, the process has trim shapes and it cuts unlocked color shapes, it is off by default but is enabled.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Setup* tab
Field	*Lock colors after routing*

Examples

```
envGetVal("APR.stdcell.route" "setup_lockColorAfterRouting")
envSetVal("APR.stdcell.route" "setup_lockColorAfterRouting" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

signalRouteFigGroupPrefix

```
APR.stdcell.route signalRouteFigGroupPrefix string "figGroupPrefixName"
```

Description

Specifies the prefix used to create the figGroup to store signal routing figs when the *Create routing as a group* option is selected in the *Route* tab.

The default is `route`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "signalRouteFigGroupPrefix")
envSetVal("APR.stdcell.route" "signalRouteFigGroupPrefix" 'string "fig1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

route_createRoutingAsAGroup

siteDefHeight

```
APR.stdcell.route siteDefHeight float float_number
```

Description

Specifies the height of the siteDef used to create inter-operable rows. The value is specified in user units. If the value is not specified, the width is calculated as the minimum height of standard cell instances.

The default is `-1`. This means no value is specified.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "siteDefHeight")
envSetVal("APR.stdcell.route" "siteDefHeight" 'float 1.0)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

siteDefSymmetricInR90

```
APR.stdcell.route siteDefSymmetricInR90 boolean { t | nil }
```

Description

Specifies whether the siteDef used to create inter-operable rows is symmetric in the R90 direction.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "siteDefSymmetricInR90")
envSetVal("APR.stdcell.route" "siteDefSymmetricInR90" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

siteDefSymmetricInX

```
APR.stdcell.route siteDefSymmetricInX boolean { t | nil }
```

Description

Specifies whether the siteDef used to create inter-operable rows is symmetric in the X direction.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "siteDefSymmetricInX")
envSetVal("APR.stdcell.route" "siteDefSymmetricInX" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

siteDefSymmetricInY

```
APR.stdcell.route siteDefSymmetricInY boolean { t | nil }
```

Description

Specifies whether the siteDef used to create inter-operable rows is symmetric in the Y direction.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "siteDefSymmetricInY")
envSetVal("APR.stdcell.route" "siteDefSymmetricInY" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

siteDefWidth

```
APR.stdcell.route siteDefWidth float float_number
```

Description

Specifies the width of the siteDef used to create interoperable rows. The value is specified in user units. If the value is not specified, the width is calculated as the greatest common factor of the widths of the standard cell instances.

The default is `-1`. This means no value is specified.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "siteDefWidth")
envSetVal("APR.stdcell.route" "siteDefWidth" 'float 1.0)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_connectToTerminals

```
APR.stdcell.route supply_connectToTerminals boolean { t | nil }
```

Description

Specifies whether or not the supply router should connect to IO pins and guard rings.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Connect to overlapped terminals*

Examples

```
envGetVal("APR.stdcell.route" "supply_connectToTerminals")
envSetVal("APR.stdcell.route" "supply_connectToTerminals" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_createGridAsGroup

```
APR.stdcell.route supply_createGridAsGroup boolean { t | nil }
```

Description

Creates the supply grid as a figGroup.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Create supply grid as a group*

Examples

```
envGetVal("APR.stdcell.route" "supply_createGridAsGroup")
envSetVal("APR.stdcell.route" "supply_createGridAsGroup" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_createPinLabel

```
APR.stdcell.route supply_createPinLabel boolean { t | nil }
```

Description

Creates labels on pins when the *Pins Create* option is selected in the *Supply* tab of the Routing assistant.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Create label*

Examples

```
envGetVal("APR.stdcell.route" "supply_createPinLabel")
envSetVal("APR.stdcell.route" "supply_createPinLabel" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_createPins

```
APR.stdcell.route supply_createPins boolean { t | nil }
```

Description

Creates a pin instead of a pathSeg for the supply stripe.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Create pins on supply stripes*

Examples

```
envGetVal("APR.stdcell.route" "supply_createPins")
envSetVal("APR.stdcell.route" "supply_createPins" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_createPinsOnEnds

```
APR.stdcell.route supply_createPinsOnEnds boolean { t | nil }
```

Description

Creates pins on the ends of stripes instead of one long pin when the *Pins Create* option is selected in the *Supply* tab of the Routing assistant.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Create on ends*

Examples

```
envGetVal("APR.stdcell.route" "supply_createPinsOnEnds")
envSetVal("APR.stdcell.route" "supply_createPinsOnEnds" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_createPinsOnPinPurpose

```
APR.stdcell.route supply_createPinsOnPinPurpose boolean { t | nil }
```

Description

Creates supply routing pins on pin purpose instead of drawing purpose.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Create on pin purpose*

Examples

```
envGetVal("APR.stdcell.route" "supply_createPinsOnPinPurpose")
envSetVal("APR.stdcell.route" "supply_createPinsOnPinPurpose" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_defaultRoutedCellExpression

```
APR.stdcell.route supply_defaultRoutedCellExpression string "viewName"
```

Description

Specifies the output to the other cell view. The cell is created as an instance in the design library. You can search for the cell name in the design library with `<cellname>_proute`.

The default is `%_proute`, where the symbol `%` is <*cellname*>.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Specify Routed Cellview*

Examples

```
envGetVal("APR.stdcell.route" "supply_defaultRoutedCellExpression")
envSetVal("APR.stdcell.route" "supply_defaultRoutedCellExpression" 'string
"dut_proute")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_defaultRoutedView

```
APR.stdcell.route supply_defaultRoutedView string "viewName"
```

Description

Specifies the routed view name if supply routing is written to another cellview.

The default is `layout.routed`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Output – Other cellview*

Examples

```
envGetVal("APR.stdcell.route" "supply_defaultRoutedView")
envSetVal("APR.stdcell.route" "supply_defaultRoutedView" 'string "routed1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_defaultWireCGOverride

`APR.stdcell.route supply_defaultWireCGOverride string "`*`WireCGName`*`"`

Description

Lets you specify a different default wire constraint group while the supply router is running.

The default is `""`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Output – Other cellview*

Examples

```
envGetVal("APR.stdcell.route" "supply_defaultWireCGOverride")
envSetVal("APR.stdcell.route" "supply_defaultWireCGOverride" 'string
"virtuosoDefaultSetup")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_deleteStripes

```
APR.stdcell.route supply_deleteStripes boolean { t | nil }
```

Description

Specifies that the stripes generated by the supply router are automatically deleted before power routing.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Delete – Supply stripes*

Examples

```
envGetVal("APR.stdcell.route" "supply_deleteStripes")
envSetVal("APR.stdcell.route" "supply_deleteStripes" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_deleteVias

```
APR.stdcell.route supply_deleteVias boolean { t | nil }
```

Description

Specifies that the vias generated by the supply router are automatically deleted before power routing.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Delete – Supply vias*

Examples

```
envGetVal("APR.stdcell.route" "supply_deleteVias")
envSetVal("APR.stdcell.route" "supply_deleteVias" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_genSupplyStripes

```
APR.stdcell.route supply_genSupplyStripes boolean { t | nil }
```

Description

Specifies that supply stripes should be generated. When set to `nil`, the supply stripes are not generated.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Generate supply stripes*

Examples

```
envGetVal("APR.stdcell.route" "supply_genSupplyStripes")
envSetVal("APR.stdcell.route" "supply_genSupplyStripes" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_generateStaples

```
APR.stdcell.route supply_generateStaples boolean { t | nil }
```

Description

Enables supply stapling.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "supply_generateStaples")
envSetVal("APR.stdcell.route" "supply_generateStaples" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_IgnoreBoundaryTracks

```
APR.stdcell.route supply_IgnoreBoundaryTracks boolean { t | nil }
```

Description

Ignores generating stripes on tracks that are on the PR boundary.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Ignore boundary tracks*

Examples

```
envGetVal("APR.stdcell.route" "supply_IgnoreBoundaryTracks")
envSetVal("APR.stdcell.route" "supply_IgnoreBoundaryTracks" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_IgnoreBoundaryVias

```
APR.stdcell.route supply_IgnoreBoundaryVias boolean { t | nil }
```

Description

Ignores generating vias on tracks that are on the PR boundary.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Ignore boundary vias*

Examples

```
envGetVal("APR.stdcell.route" "supply_IgnoreBoundaryVias")
envSetVal("APR.stdcell.route" "supply_IgnoreBoundaryVias" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_insertTrim

```
APR.stdcell.route supply_insertTrim boolean { t | nil }
```

Description

Inserts trims between the intersection of the layer above and below the via in the supply grid to fix DRC errors.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Insert trim to fix DRCs*

Examples

```
envGetVal("APR.stdcell.route" "supply_insertTrim")
envSetVal("APR.stdcell.route" "supply_insertTrim" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_insertVias

```
APR.stdcell.route supply_insertVias boolean { t | nil }
```

Description

Specifies that vias are inserted between the intersection of the layer above and below the via in the supply grid.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Insert vias for supply stripes*

Examples

```
envGetVal("APR.stdcell.route" "supply_insertVias")
envSetVal("APR.stdcell.route" "supply_insertVias" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_nets

```
APR.stdcell.route supply_nets cyclic { "All" | "Selected" }
```

Description

Specifies whether to route all or selected supply nets.

The default is `"All"`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Supply Nets – All*, *Selected*

Examples

```
envGetVal("APR.stdcell.route" "supply_nets")
envSetVal("APR.stdcell.route" "supply_nets" 'cyclic "Selected")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_netsWithin

```
APR.stdcell.route supply_netsWithin cyclic { "PR boundary" | "Guardring/FigGroup" }
```

Description

Specifies whether to route everything inside the PR boundary or only within a guard ring or figGroup.

The default is `"PR boundary"`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Within – PR boundary*, *Guardring/FigGroup*

Examples

```
envGetVal("APR.stdcell.route" "supply_netsWithin")
envSetVal("APR.stdcell.route" "supply_netsWithin" 'cyclic "Guardring/FigGroup")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_pinLayers

`APR.stdcell.route supply_pinLayers string "`*`pinLayer`*`"`

Description

Lets you specify a list of layers to create pins on for supply routing.

The default is `""`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Use Selected Layers*

Examples

```
envGetVal("APR.stdcell.route" "supply_pinLayers")
envSetVal("APR.stdcell.route" "supply_pinLayers" 'string "metal1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_pinLayerSet

```
APR.stdcell.route supply_pinLayerSet cyclic { "Use all supply stripe layers" | "Use
     selected layers" }
```

Description

Specifies the top and bottom layers of the supply stripes layer range that pins should be created on.

The default is `"Use all supply stripe layers"`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Pins create*

Examples

```
envGetVal("APR.stdcell.route" "supply_pinLayerSet")
envSetVal("APR.stdcell.route" "supply_pinLayerSet" 'cyclic "Use selected layers")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_routedLOC

```
APR.stdcell.route supply_routedLOC cyclic { "Current cellview" | "Other cellview" }
```

Description

Specifies whether to write the supply routing to the current cellview or to a different cellview.

The default is `"Current cellview"`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Current cellview, Other cellview*

Examples

```
envGetVal("APR.stdcell.route" "supply_routedLOC")
envSetVal("APR.stdcell.route" "supply_routedLOC" 'cyclic "Other cellview")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_saveRoutingOnly

```
APR.stdcell.route supply_saveRoutingOnly boolean { t | nil }
```

Description

Specifies whether to copy only the supply grid or all initial data and the supply grid to the new cellview.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Save routing only*

Examples

```
envGetVal("APR.stdcell.route" "supply_saveRoutingOnly")
envSetVal("APR.stdcell.route" "supply_saveRoutingOnly" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_shareTracks

```
APR.stdcell.route supply_shareTracks boolean { t | nil }
```

Description

Distinguishes upper and below areas for one WSP track by instance's pin name.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Share tracks for supply nets*

Examples

```
envGetVal("APR.stdcell.route" "supply_shareTracks")
envSetVal("APR.stdcell.route" "supply_shareTracks" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

supply_useExisitingPGTracks

```
APR.stdcell.route supply_useExisitingPGTracks boolean { t | nil }
```

Description

Lists layers in Supply Stripes that have PG tracks.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Supply* tab
Field	*Only use layers with WSP P/G tracks*

Examples

```
envGetVal("APR.stdcell.route" "supply_useExisitingPGTracks")
envSetVal("APR.stdcell.route" "supply_useExisitingPGTracks" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

tracksCmds

`APR.stdcell.route tracksCmds string "`*`filename`*`"`

Description

Specifies the Tcl plugin hook file for custom `add_tracks` settings that is sourced before the placement stage. The file can be placed anywhere and can be specified with the absolute path or the relative path to the current working directory. The default is `""`.

Note: The plugin is skipped if one or more 0-width active WSPs exists on metal layers in the design. This indicates that they were imported from Innovus-generated tracks in a previous step. A message is displayed to notify whether the plugin was sourced or skipped.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "tracksCmds")
envSetVal("APR.stdcell.route" "tracksCmds" 'string "track1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

useMultipleTapCells

```
APR.stdcell.route useMultipleTapCells cyclic { "disable" | "enable" | "auto" }
```

Description

Changes the output to support multiple tap cells.

The default is `"auto"`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "useMultipleTapCells")
envSetVal("APR.stdcell.route" "useMultipleTapCells" 'cyclic "enable")
envSetVal("APR.stdcell.route" "useMultipleTapCells" 'cyclic "disable")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

useStylusMode

```
APR.stdcell.route useStylusMode boolean { t | nil }
```

Description

Controls whether to use stylus mode for the Tcl plugin syntax. If set to `t`, stylus mode is used. Otherwise, legacy mode is used.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.route" "useStylusMode")
envSetVal("APR.stdcell.route" "useStylusMode" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

Standard Cell Placement Environment Variables

- init_boundaryAspectRatioOrHeight
- init_boundaryAspectRatioVal
- init_boundaryHeightVal
- init_boundaryUtilizationOrWidth
- init_boundaryUtilizationVal
- init_boundaryWidthVal
- init_generateBoundary
- init_generateInstances
- init_useSourceLayout
- init_useSourceLayoutBoundary
- init_useSourceLayoutInstances
- init_useSourceLayoutPins
- place_addBoundaryCells
- place_addCellRowRouting
- place_addDecapFillerCells
- place_addTapCells
- place_boundaryCellsContainTapCells
- place_defaultPlacedView
- place_deleteBoundaryCells
- place_deleteFillerCells
- place_deleteTapCells
- place_displayLog
- place_overwriteLog
- place_placedLocation
- place_placeStdCells

- place_showInformation
- setup_createRowRegionInArea
- setup_Innovus1stRowOrientation
- setup_rowCreation

advNode

```
APR.stdcell.place advNode boolean { t | nil }
```

Description

Controls the GUI behavior. For example, some standard cells have well taps inside boundary cells and must place boundary and tap cells together. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.place" "advNode")
envSetVal("APR.stdcell.place" "advNode" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

init_boundaryAspectRatioOrHeight

```
APR.stdcell.place init_boundaryAspectRatioOrHeight cyclic { "Aspect ratio" |
     "Height" }
```

Description

Specifies whether the aspect ratio or the height is used as the basis for determining the PR boundary.

The default is `"Aspect ratio"`.

GUI Equivalent

Command	Auto P&R assistant – *Initialize* tab
Field	*Boundary*

Examples

```
envGetVal("APR.stdcell.place" "init_boundaryAspectRatioOrHeight")
envSetVal("APR.stdcell.place" "init_boundaryAspectRatioOrHeight" 'cyclic
"Height")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

init_boundaryAspectRatioVal

`APR.stdcell.place init_boundaryAspectRatioVal float` *`float_number`*

Description

Specifies the width-to-height ratio of the PR boundary.

The default value is `1` indicates a square boundary. An aspect ratio of `0.5` specifies a boundary twice as high as it is wide. A value of `2` specifies a boundary twice as wide as its height.

GUI Equivalent

Command	Auto P&R assistant – *Initialize* tab
Field	*Boundary*

Examples

```
envGetVal("APR.stdcell.place" "init_boundaryAspectRatioVal")
envSetVal("APR.stdcell.place" "init_boundaryAspectRatioVal" 'float 0.5)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

init_boundaryHeightVal

`APR.stdcell.place init_boundaryHeightVal float` *`float_number`*

Description

Specifies the height of the PR boundary.

The default value is `10.0`.

GUI Equivalent

Command	Auto P&R assistant – *Initialize* tab
Field	*Boundary*

Examples

```
envGetVal("APR.stdcell.place" "init_boundaryHeightVal")
envSetVal("APR.stdcell.place" "init_boundaryHeightVal" 'float 5.0)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

init_boundaryUtilizationOrWidth

```
APR.stdcell.place init_boundaryUtilizationOrWidth cyclic { "Utilization" | "Width"
      }
```

Description

Specifies whether the PR boundary is to be calculated based on the utilization percentage or the width.

The default is `"Utilization"`.

GUI Equivalent

Command	Auto P&R assistant – *Initialize* tab
Field	*Boundary*

Examples

```
envGetVal("APR.stdcell.place" "init_boundaryUtilizationOrWidth")
envSetVal("APR.stdcell.place" "init_boundaryUtilizationOrWidth" 'cyclic "Width")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

init_boundaryUtilizationVal

```
APR.stdcell.place init_boundaryUtilizationVal float float_number
```

Description

Specifies the area utilization to be used when generating the PR boundary. It is the ratio of the combined area of the logical cells to the area of the PR boundary expressed as a percentage.

The default value is `25.0`.

GUI Equivalent

Command	Auto P&R assistant – *Initialize* tab
Field	*Boundary*

Examples

```
envGetVal("APR.stdcell.place" "init_boundaryUtilizationVal")
envSetVal("APR.stdcell.place" "init_boundaryUtilizationVal" 'float 15.0)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

init_boundaryWidthVal

```
APR.stdcell.place init_boundaryWidthVal float float_number
```

Description

Specifies the width of the PR boundary.

The default value is `10.0`.

GUI Equivalent

Command	Auto P&R assistant – *Initialize* tab
Field	*Boundary*

Examples

```
envGetVal("APR.stdcell.place" "init_boundaryWidthVal")
envSetVal("APR.stdcell.place" "init_boundaryWidthVal" 'float 5.0)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

init_createPowerPins

```
APR.stdcell.place init_createPowerPins boolean { t | nil }
```

Description

Controls whether IO power and ground pins are created during initialization of the layout view. The default is `t`.

GUI Equivalent

Command	Auto P&R assistant – *Initialize* tab
Field	*Power pin layer*

Examples

```
envGetVal("APR.stdcell.place" "init_createPowerPins")
envSetVal("APR.stdcell.place" "init_createPowerPins" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

init_generateBoundary

```
APR.stdcell.place init_generateBoundary boolean { t | nil }
```

Description

Generates a PR boundary.

The default is `nil`.

GUI Equivalent

Command	Auto P&R assistant – *Initialize* tab
Field	*Boundary*

Examples

```
envGetVal("APR.stdcell.place" "init_generateBoundary")
envSetVal("APR.stdcell.place" "init_generateBoundary" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

init_generateInstances

```
APR.stdcell.place init_generateInstances boolean { t | nil }
```

Description

Generates all the instances in the schematic that do not have any ignore properties attached to them.

The default is `t`.

GUI Equivalent

Command	Auto P&R assistant – *Initialize* tab
Field	*Instances*

Examples

```
envGetVal("APR.stdcell.place" "init_generateInstances")
envSetVal("APR.stdcell.place" "init_generateInstances" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

init_generatePins

```
APR.stdcell.place init_generatePins boolean { t | nil }
```

Description

Specifies that signal IO pins and power IO pins on a layer are to be generated.

The default is `t`.

GUI Equivalent

Command	Auto P&R assistant – *Initialize* tab
Field	*Pins*

Examples

```
envGetVal("APR.stdcell.place" "init_generatePins")
envSetVal("APR.stdcell.place" "init_generatePins" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

init_mode

```
APR.stdcell.place init_mode cyclic { "Generate" | "Update" }
```

Description

Specifies a standard cell generation mode. The available options are:

- `Generate`: Generates new standard cells in the layout design.
- `Update`: Updates existing standard cells in the layout design.

The default value is `Generate.`

GUI Equivalent

Command	Auto P&R assistant – *Initialize* tab
Field	*Generation Options – Generate, Update*

Examples

```
envGetVal("APR.stdcell.place" "init_mode")
envSetVal("APR.stdcell.place" "init_mode" 'cyclic "Update")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

init_scope

```
APR.stdcell.place init_scope cyclic { "All" | "Selected" }
```

Description

Specifies the scope of layout generation.

- `All`: Generates all standard cells present in the source.
- `Selected`: Generates only the selected standard cells.

The default value is `All`.

GUI Equivalent

Command	Auto P&R assistant – *Initialize* tab
Field	*Generation Options – All, Selected*

Examples

```
envGetVal("APR.stdcell.place" "init_scope")
envSetVal("APR.stdcell.place" "init_scope" 'cyclic "Selected")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

init_useSourceLayout

```
APR.stdcell.place init_useSourceLayout boolean { t | nil }
```

Description

Specifies that a source layout is to be reused during layout generation. When set to t, you can further specify which components from the source layout are to be reused by using the `init_useSourceLayoutBoundary`, `init_useSourceLayoutInstancePositions`, and `init_useSourceLayoutPins` environment variables.

The default is `nil`.

GUI Equivalent

Command	Auto P&R assistant – *Initialize* tab
Field	*Reuse Options – Use Source Layout*

Examples

```
envGetVal("APR.stdcell.place" "init_useSourceLayout")
envSetVal("APR.stdcell.place" "init_useSourceLayout" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

init_useSourceLayoutBoundary

init_useSourceLayoutInstances

init_useSourceLayoutPins

init_useSourceLayoutBoundary

```
APR.stdcell.place init_useSourceLayoutBoundary boolean { t | nil }
```

Description

Adjusts the PR boundary in the current cellview as per its setting in the source cellview. This environment variable is honored only when `init_useSourceLayout` is set to t.

The default is `nil`.

GUI Equivalent

Command	Auto P&R assistant – *Initialize* tab
Field	*Reuse Options – Use Source Layout – Boundary*

Examples

```
envGetVal("APR.stdcell.place" "init_useSourceLayoutBoundary")
envSetVal("APR.stdcell.place" "init_useSourceLayoutBoundary" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

init_useSourceLayout

init_useSourceLayoutInstances

init_useSourceLayoutPins

init_useSourceLayoutInstances

```
APR.stdcell.place init_useSourceLayoutInstances boolean { t | nil }
```

Description

Updates the positions of standard cells, custom cells, and macros in the current cellview as per the source cellview. This environment variable is honored only when `init_useSourceLayout` is set to t.

The default is `nil`.

GUI Equivalent

Command	Auto P&R assistant – *Initialize* tab
Field	*Reuse Options – Use Source Layout – Instance Positions*

Examples

```
envGetVal("APR.stdcell.place" "init_useSourceLayoutInstances")
envSetVal("APR.stdcell.place" "init_useSourceLayoutInstances" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

init_useSourceLayout

init_useSourceLayoutBoundary

init_useSourceLayoutPins

init_useSourceLayoutPins

```
APR.stdcell.place init_useSourceLayoutPins boolean { t | nil }
```

Description

Places pins in the current cellview as per their positions in the source cellview. This environment variable is honored only when `init_useSourceLayout` is set to t.

The default is `nil`.

GUI Equivalent

Command	Auto P&R assistant – *Initialize* tab
Field	*Reuse Options – Use Source Layout – Instance Positions*

Examples

```
envGetVal("APR.stdcell.place" "init_useSourceLayoutPins")
envSetVal("APR.stdcell.place" "init_useSourceLayoutPins" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

init_useSourceLayout

init_useSourceLayoutInstances

init_useSourceLayoutBoundary

place_addBoundaryCells

```
APR.stdcell.place place_addBoundaryCells boolean { t | nil }
```

Description

Adds boundary cells around core cells.

The default is `t`.

GUI Equivalent

Command	Auto P&R assistant – *Place* tab
Field	*Placement Options – Add boundary cells*

Examples

```
envGetVal("APR.stdcell.place" "place_addBoundaryCells")
envSetVal("APR.stdcell.place" "place_addBoundaryCells" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

place_addCellRowRouting

```
APR.stdcell.place place_addCellRowRouting boolean { t | nil }
```

Description

Inserts standard cell power routing stripes for the layers that have standard cell power pins and connects to the layer above using vias. The inserted vias are color correct in Innovus.

The default is `nil`.

GUI Equivalent

Command	Auto P&R assistant – *Place* tab
Field	*Placement Options – Add cell row routing*

Examples

```
envGetVal("APR.stdcell.place" "place_addCellRowRouting")
envSetVal("APR.stdcell.place" "place_addCellRowRouting" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

place_addDecapFillerCells

```
APR.stdcell.place place_addDecapFillerCells boolean { t | nil }
```

Description

Adds decap/filler cells into the design. Select decap cells first and place them. You can then follow it up with selected filler cells for the remaining spaces.

The default is `t`.

GUI Equivalent

Command	Auto P&R assistant – *Place* tab
Field	*Placement Options – Add decap/filler cells*

Examples

```
envGetVal("APR.stdcell.place" "place_addDecapFillerCells")
envSetVal("APR.stdcell.place" "place_addDecapFillerCells" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

place_addTapCells

```
APR.stdcell.place place_addTapCells boolean { t | nil }
```

Description

Adds single-height or multi-height tap cells in the empty spaces between the standard cells.

The default is `t`.

GUI Equivalent

Command	Auto P&R assistant – *Place* tab
Field	*Placement Options – Add tap cells*

Examples

```
envGetVal("APR.stdcell.place" "place_addTapCells")
envSetVal("APR.stdcell.place" "place_addTapCells" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

place_boundaryCellsContainTapCells

```
APR.stdcell.place place_boundaryCellsContainTapCells boolean { t | nil }
```

Description

Disables the independent selection of *Add boundary cells* and *Add tap cells* options and makes them both selected. This is to ensure that boundary cells are not fixed from a previous run so they cannot be swapped with tap cells as needed when tap cells are placed. If done in the same run, boundary cells do not have a fixed or locked status when tap cells are added.

The default is `nil`.

GUI Equivalent

Command	Auto P&R assistant – *Initialize* tab
Field	*Reuse Options – Use Source Layout – Instance Positions*

Examples

```
envGetVal("APR.stdcell.place" "place_boundaryCellsContainTapCells")
envSetVal("APR.stdcell.place" "place_boundaryCellsContainTapCells" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

place_createPhysOnlyAsAGroup

```
APR.stdcell.place place_createPhysOnlyAsAGroup boolean { t | nil }
```

Description

Controls whether the physical-only cells are added to a figGroup. The default is `nil`.

GUI Equivalent

Command	Auto P&R assistant – *Place* tab
Field	*Create physical only cells as group*

Examples

```
envGetVal("APR.stdcell.place" "place_createPhysOnlyAsAGroup")
envSetVal("APR.stdcell.place" "place_createPhysOnlyAsAGroup" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

place_defaultPlacedView

`APR.stdcell.place place_defaultPlacedView string "`*`placementView`*`"`

Description

Specifies the default view name when the results of placement are written to a different view.

The default is `"layout.placed"`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.place" "place_defaultPlacedView")
envSetVal("APR.stdcell.place" "place_defaultPlacedView" 'string "layout.placed1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

place_deleteBoundaryCells

```
APR.stdcell.place place_deleteBoundaryCells boolean { t | nil }
```

Description

Deletes the boundary cells in the design.

The default is `nil`.

GUI Equivalent

Command	Auto P&R assistant – *Place* tab
Field	*Update – Delete Boundary cells*

Examples

```
envGetVal("APR.stdcell.place" "place_deleteBoundaryCells")
envSetVal("APR.stdcell.place" "place_deleteBoundaryCells" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

place_deleteFillerCells

```
APR.stdcell.place place_deleteFillerCells boolean { t | nil }
```

Description

Deletes the filler cells in the design.

The default is `nil`.

GUI Equivalent

Command	Auto P&R assistant – *Place* tab
Field	*Update – Delete Filler cells*

Examples

```
envGetVal("APR.stdcell.place" "place_deleteFillerCells")
envSetVal("APR.stdcell.place" "place_deleteFillerCells" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

place_deleteTapCells

```
APR.stdcell.place place_deleteTapCells boolean { t | nil }
```

Description

Deletes the tap cells in the design.

The default is `nil`.

GUI Equivalent

Command	Auto P&R assistant – *Place* tab
Field	*Update – Delete Tap cells*

Examples

```
envGetVal("APR.stdcell.place" "place_deleteTapCells")
envSetVal("APR.stdcell.place" "place_deleteTapCells" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

place_displayLog

```
APR.stdcell.place place_displayLog boolean { t | nil }
```

Description

Controls the display of the placement log.

The default is `t`.

GUI Equivalent

Command	Auto P&R assistant – *Place* tab
Field	*Output – Display Log*

Examples

```
envGetVal("APR.stdcell.place" "place_displayLog")
envSetVal("APR.stdcell.place" "place_displayLog" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

place_overwriteLog

```
APR.stdcell.place place_overwriteLog boolean { t | nil }
```

Description

Controls the overwriting of the last log file. When set to `nil`, the existing log file is retained and data is not updated.

The default is `t`.

GUI Equivalent

Command	Auto P&R assistant – *Place* tab
Field	*Output – Overwrite log*

Examples

```
envGetVal("APR.stdcell.place" "place_overwriteLog")
envSetVal("APR.stdcell.place" "place_overwriteLog" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

place_placedLocation

```
APR.stdcell.place place_placedLocation cyclic { "Current cellview" | "Other
     cellview" }
```

Description

Specifies whether to write the output of the cell placement to the current cellview or a different cellview.

The default is `"Current cellview"`.

GUI Equivalent

Command	Auto P&R assistant – *Place* tab
Field	*Output – Current cellview*, *Other cellview*

Examples

```
envGetVal("APR.stdcell.place" "place_placedLocation")
envSetVal("APR.stdcell.place" "place_placedLocation" 'cyclic "Other cellview")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

place_placeStdCells

```
APR.stdcell.place place_placeStdCells boolean { t | nil }
```

Description

Places the standard cells in the design.

The default is `nil`.

GUI Equivalent

Command	Auto P&R assistant – *Place* tab
Field	*Placement Options – Place std cells*

Examples

```
envGetVal("APR.stdcell.place" "place_placeStdCells")
envSetVal("APR.stdcell.place" "place_placeStdCells" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

place_showInformation

```
APR.stdcell.place place_showInformation boolean { t | nil }
```

Description

Displays information about the placement.

The default is `nil`.

GUI Equivalent

Command	Auto P&R assistant – *Place* tab
Field	*Interactive Placement – Show information*

Examples

```
envGetVal("APR.stdcell.place" "place_showInformation")
envSetVal("APR.stdcell.place" "place_showInformation" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

postPlaceTrigger

`APR.stdcell.place postPlaceTrigger string "`*`cmdName`*`"`

Description

Specifies a SKILL command that is executed after placement. The default is an empty string `""`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.place" "postPlaceTrigger")
envSetVal("APR.stdcell.place" "postPlaceTrigger" 'string "cmd1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

prePlaceTrigger

```
APR.stdcell.place prePlaceTrigger string "cmdName"
```

Description

Specifies a SKILL command that is executed before placement. The default is an empty string `""`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.place" "prePlaceTrigger")
envSetVal("APR.stdcell.place" "prePlaceTrigger" 'string "cmd1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

runDir

`APR.stdcell.place runDir string "`*`directoryName`*`"`

Description

Specifies a directory to place the placer log files. The default is `./placerLogs`.

GUI Equivalent

None

Examples

```
envGetVal("APR.stdcell.place" "runDir")
envSetVal("APR.stdcell.place" "runDir" 'string "dir1")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

setup_createRowRegionInArea

```
APR.stdcell.place setup_createRowRegionInArea boolean { t | nil }
```

Description

Specifies the area within which a row region must be created. If a region is not specified, the entire PR boundary is considered for creating the row region.

The default is `nil`.

GUI Equivalent

Command	Auto P&R assistant – *Setup* tab
Field	*Rows – Create row region – Create rows in area*

Examples

```
envGetVal("APR.stdcell.place" "setup_createRowRegionInArea")
envSetVal("APR.stdcell.place" "setup_createRowRegionInArea" 'boolean t)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

setup_displayLog

```
APR.stdcell.place setup_displayLog boolean { t | nil }
```

Description

Specifies whether to display the row generation log file after the placement setup is done. The default is `t`.

GUI Equivalent

Command	Auto P&R assistant – *Setup* tab
Field	*Display log*

Examples

```
envGetVal("APR.stdcell.place" "setup_displayLog")
envSetVal("APR.stdcell.place" "setup_displayLog" 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

setup_Innovus1stRowOrientation

```
APR.stdcell.place setup_Innovus1stRowOrientation cyclic { "R0" | "MX" }
```

Description

Specifies the orientation of the bottom-most (first) row in the row region when generating rows in Innovus.

The default is `"R0"`.

GUI Equivalent

Command	Auto P&R assistant – *Place* tab
Field	*Rows – First row orientation*

Examples

```
envGetVal("APR.stdcell.place" "setup_Innovus1stRowOrientation")
envSetVal("APR.stdcell.place" "setup_Innovus1stRowOrientation" 'cyclic "MX")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

setup_overwriteLog

```
APR.stdcell.place setup_overwriteLog boolean { t | nil }
```

Description

Specifies whether the row generation log file must be overwritten. The default is `t`.

GUI Equivalent

Command	Auto P&R assistant – *Setup* tab
Field	*Overwrite log*.

Examples

```
envGetVal("APR.stdcell.place" "setup_overwriteLog ")
envSetVal("APR.stdcell.place" "setup_overwriteLog " 'boolean nil)
```

Related Topics

Standard Cell Placement and Routing Environment Variables

setup_rowCount

```
APR.stdcell.place setup_rowCount cyclic { "Auto" | "Even" | "Odd"}
```

Description

Specifies whether to fit as many rows as possible, or an even number of rows, or an odd numbers within the PR boundary or drawn area. The default is `"Auto"`.

GUI Equivalent

Command	Auto P&R assistant – *Setup* tab
Field	*Row Count*

Examples

```
envGetVal("APR.stdcell.place" "setup_rowCount")
envSetVal("APR.stdcell.place" "setup_rowCount" 'cyclic "Even")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

setup_rowCreation

```
APR.stdcell.place setup_rowCreation cyclic { "Import Innovus rows" | "Create row
     region" }
```

Description

Specifies whether to create rows in Innovus or from Virtuoso as a row region.

- `Import Innovus rows`: imports rows from Innovus.
- `Create row region`: creates a row region in Virtuoso using an existing row template.

The default is `"Import Innovus rows"`.

GUI Equivalent

Command	Auto P&R assistant – *Place* tab
Field	*Rows – Import Innovus rows, Create row region*

Examples

```
envGetVal("APR.stdcell.place" "setup_rowCreation")
envSetVal("APR.stdcell.place" "setup_rowCreation" 'cyclic "Create row region")
```

Related Topics

Standard Cell Placement and Routing Environment Variables

5
Automated Standard Cell Placement and Routing User Interface

In the Virtuoso standard cell placement and routing flow, you use the Auto P&R assistant for standard cell placement and the Routing assistant for standard cell routing in the Layout MXL cockpit.

The Auto P&R assistant is the integrated, automatic placement and routing solution available in Virtuoso. Use the Auto P&R assistant to initialize, set up, place, and route the standard cells in the layout design automatically as per your requirements.

The Routing assistant has the same look-and-feel for device, standard cell, and chip assembly routing with a common toolbar for all routing types. The Routing assistant is a dockable assistant pane that provides various options to let you perform tasks related to various routing types.

This topic lists the standard cell placement and routing assistant and forms.

Standard Cell Placement and Routing Assistants	
Auto P&R Assistant User Interface for Standard Cell	
Routing Assistant User Interface for Standard Cell	
Standard Cell Placement Forms (Auto P&R Assistant)	
Boundary Cells Form	Choose Source Layout Cellview Form
Filler Cells Form	Load Option Presets Form
Save Option Presets Form	Select Row Template Form
Tap Cells Form	
Standard Cell Routing Forms (Routing Assistant)	
Assign Wire Types to Nets Form	Map WSP Wire Types to Symbols Form
Pull Back and Offset Values Form	

Related Topics

Routing Assistant

Accessing Routing Assistant

Virtuoso Routing Constraint Manager

Opening Routing Constraint Manager

Auto P&R Assistant User Interface for Standard Cell

The Auto P&R assistant is the integrated, automatic placement and routing solution available in Virtuoso. Use the Auto P&R assistant to initialize, set up, place, and route the standard cells in the layout design automatically as per your requirements.

The Auto P&R assistant has the following components:

Routing Assistant Toolbar	Lets you access the buttons to complete the steps of the routing flow.
Routing Assistant Tabs	Lets you specify the options for running the selected routing type.
Auto P&R Command Buttons	Lets you access the buttons on each tab to compete a routing task.

Auto P&R Assistant Toolbar

The following table lists the functions of the different buttons on the Auto P&R assistant toolbar:

Icon	Command	Description
	Change placer type	Changes the type of placement you want to run on the design. The two placement types are: *Device and Stdcell*.
	Load preset options	Lets you load the placement options from an existing preset file.
	Save preset options	Lets you save the placement preset options to a file.

Icon	Command	Description
	Raise the Routing Assistant	Opens the Routing Assistant.

Auto P&R Assistant Tabs

The following table lists the functions of the different tabs in the Auto P&R assistant:

Tab	Description
Initialize	Generates layout representations of the schematic design components. Any existing components in the layout view are deleted and are regenerated from scratch.
Setup	Generates a row region and width spacing pattern (WSP), gate, and source and drain tracks. The rows are used for standard cell placement. The tracks are used for standard cell snapping and routing.
Place	Lets you customize placement settings and run the placer.

Initialize

The following table describes the fields available on the *Initialize* tab of the Auto P&R assistant for standard cell placement.

Field	Description
Generation Options	Lets you select the objects to be generated in the layout.
Generate	Generates the instances and pins in the schematic that do not have any ignore properties attached to them.
Instances	Generates all the instances in the schematic that do not have any ignore properties attached to them.
Pins	Generates signal and IO power pins.
Pin Layer	Specifies the layer in the target layout view for the signal pin from the schematic.

Field	**Description**
Power pin layer	Specifies the layer in the target layout view for the power or ground pin from the schematic. Deselecting the check box prevents power IO pin from being created, if you plan to create power IO pins with the supply router. The first metal layer in the layer stack is set as the default value.
Boundary	Generates a PR boundary.
Utilization	Specifies the percentage of area within the PR boundary that can be filled with objects. The default is 25%.
Aspect Ratio	Specifies the width-to-height ratio of the PR boundary. The default value is 1, which indicates a square boundary. An aspect ratio of 0.5 specifies a boundary twice as high as it is wide. A value of 2 specifies a boundary twice as wide as its height.
Migration Options	This section is available only in Layout MXL and is part of the assisted custom layout design migration flow. Use the options in this section to capture source data and apply it to a target layout.
Migration Directory	Specifies the path to the directory in which the captured source data is stored. The target layout references this location for applying the captured source data.
Capture Placement from Layout	Captures data from the source layout and stores it in the specified *Migration Directory.*
Apply Placement to Layout	Specifies the target layout to which the captured reuse information must be applied.
Layout Objects	Specifies the objects in the target layout to which reuse settings are to be applied. The available options are: ■ *Instances* ■ *Pins* ■ *Boundary* ■ *Constraints*
PDK Settings	Specifies the process setup and `LEFDefaultRouteSpec` name.
Process Setup	Specifies the process setup name to use for the Innovus process node.

Field	Description
Multi-tech LEF Foundry Rule	Lets you select a foundry rule for Innovus from the multi-tech LEFs in the RMSOA PDK. This option is available only for Rapid Mixed-Signal Open Access (RMSOA) multi-tech PDKs.
LEFDefaultRouteSpec	Displays the name of the *`LEFDefaultRouteSpec`* to be used in Innovus. This option works with the multi-tech LEF selection.

Setup

The following table describes the fields available on the *Setup* tab of the Auto P&R assistant.

Field	Description
WSPs	Specifies WSP settings.
Create manually	Opens the WSP Manager, which lets you create and modify WSPs, import WSPs from other designs, and generate WSPs from existing shapes in the layout canvas.
Create automatically	Opens the *Setup* tab of the Routing assistant. Use the *Layer* and *Pattern* columns in the table in this assistant to specify WSP attributes.
Set active WSPs	Opens the Track Pattern assistant. For WSSPDefs that have more than one allowed pattern, you can change the active pattern using the Track Pattern assistant. You can also change the active WSSPDef.
Rows	Specifies how rows and row regions are to be created.
Import Innovus rows	Imports rows from Innovus.
Create row region	Creates a row region in Virtuoso using an existing row template.
First row orientation	Specifies the orientation of the bottom-most (first) row in the row region. This option is displayed only when *Import Innovus rows* is selected.

Field	Description
Row template	Specifies the row template based on which rows are to be created. Clicking the *Browse* button opens the Select Row Template form. You can navigate to the required cellview and select a row template. This option is displayed only when *Create row region* is selected.
Create rows in area	Specifies the area within which a row must be created. The available icons are: ■ *Set area bbox to the visible area* : Sets the visible area of the design canvas as the row region. ■ *Draw the area bbox* : Lets you draw the row region on the design canvas. Use *Show/hide the area bbox* to display or hide the highlight around the row region. If a region is not specified, the entire PR boundary is considered for creating row region. This works for both Innovus-imported rows and row template-created rows.
Row count	Specifies whether to fit as many rows as possible, or an even number of rows, or an odd numbers within the PR boundary or drawn area.
Supply Routing	Creates a supply grid.
Generate Supply Grid	Opens the *Supply* tab for standard cell routing in the Routing assistant. Use this option when boundary cells have blockages. You can create a supply grid before running the placer to prevent routing from interfering with access to standard cell pins.
IO Pins	Specifies IO pin attributes.
Specify pin positions and attributes	Opens the *Pin Planner* tab of the Pin Placement form. Use the options in this form to specify pin constraints and pin placement attributes.
Checks	Runs pre-routing checks to detect design issues before routing the design.

Field	Description
Run checks	Opens the *Check* tab for standard cell routing in the Routing assistant. You can select the pre-routing checks to be run.

Place

The following table describes the fields available on the *Place* tab of the Auto P&R assistant.

Field	Description
Placement Options	Specifies the options for placement of standard cells. All placement options can be run individually or simultaneously if all are selected.
Boundary cells contain tap cells	Inserts boundary cells and tap cells together automatically. With this option selected, *Add boundary cells* and *Add tap cells* are automatically selected and disabled so that these commands cannot be run independently. As boundary and tap cells are added simultaneously, boundary cells do not have a fixed or locked status when tap cells are added. This ensures that boundary cells are not swapped with tap cells when tap cells are placed. If this option is not selected, boundary cells and tap cells can be added independently.
Add boundary cells	Adds boundary cells around core cells. The *Browse* button opens the Boundary Cells Form, which lets you select boundary cells to be inserted along each edge of the PR boundary.
Add tap cells	Adds single-height or multi-height tap cells in the empty spaces between the standard cells. The *Browse* button opens the Tap Cells Form, which lets you specify tap cell attributes. *Add tap cells* is not available if *Boundary cells contain tap cells* is selected.
Add cell row routing	Inserts follow-pin (cell row) routing for standard cell rails and vias with correct color in Innovus.
Place std cells	Adds the command to place standard cells into the Tcl command script.

Field	Description
Add decap/filler cells	Adds decap/filler cells into a design. The *Browse* button opens the Filler Cells Form, which lets you specify filler cell attributes.
Create physical only cells as group	Adds physical only instances (boundary, tap, and filler or decap) cells into a figGroup.
Update	Allows cells of the type selected to be deleted.
Boundary cells	Deletes the boundary cells in the design.
Tap cells	Deletes the tap cells in the design.
Filler cells	Deletes the filler cells in the design.
Output	Specifies settings to display the placement log and to save output to a different placed view.
Display log	Controls the display of the placement log in the CIW once the cells are placed.
Overwrite log	Specifies whether the previous log files are to be retained or overwritten by the new one. When the standard cell placer is run, three files are created in the `routerLogs` directory within the run directory—two log files with extensions `.log` and `.logv` and a `.cmd` file. The naming convention followed for these files is: `innovus.<libName>.<cellName>.<viewName>.{ cmd | log | logv }` When *Overwrite log* is switched off, the tool appends a `#` to the end of the three existing files. When switched on, the tool overwrites the highest `#` log file.
Current view	Writes the output of the cell placement to the current cellview.
Other view	Lets you save the output to a different view.
Interactive Placement	Refines the interactive placement settings.
Show Information	Displays information about the placement.

Auto P&R Command Buttons

The following table lists the functions of the different command buttons on the Auto P&R assistant:

Icon	Command	Description
➕	*Generate with specified options*	Generates width spacing patterns automatically on the selected layers.
▶	*Run GigaPlace*	Runs the standard cell placer.

Related Topics

Automated Standard Cell Placement and Routing User Interface

Initializing a Layout in the Auto P&R Standard Cell Flow

Routing Assistant User Interface for Standard Cell

The Routing Assistant has the following components:

Routing Assistant Toolbar	Lets you access the buttons to complete the steps of the routing flow.
Routing Assistant Tabs	Lets you specify the options for running the selected routing type.
Routing Assistant Command Buttons	Lets you access the buttons on each tab to compete a routing task.

Routing Assistant Toolbar

The following table lists the functions of the different buttons on the Routing assistant toolbar:

Icon	Command	Description
⚙	*Change routing mode*	Selects the routing mode. The two routing modes are *Automatic* and *Interactive*.

Icon	Command	Description
	Change routing type	Changes the type of routing you want to run on the design. The three routing types are: *Device*, *StdCell*, *Chip Assembly*.
	Raise the pre-routing browser	Provides net information prior to routing.
	Set router constraints	Opens the Routing Constraint Manager.
	Load preset options	Lets you load options from an existing preset file.
	Save preset options	Lets you save the Routing Assistant options to a file.
	Delete preset options	Lets you delete an existing preset file.

Routing Assistant Tabs

The following table lists the functions of the different tabs in the Routing assistant:

Tab	Description
Setup	Lets you specify the settings for running the router.
Check	Lets you select the checks that you want to run before routing the design.
Supply	Lets you run the power router. You need to have power and ground wireType tracks in your width spacing patterns. Usually, you run this routing before placement.
Route	Lets you specify the scope of signal routing.
Results	Lets you select the scope and the results column that should be displayed in Routing Results Browser.

The options in the Routing Assistant depend on the routing type in which the design is open. The three routing types are:

- **Device-level** flow helps users become familiar with the device-level placement and routing solution in Virtuoso for advanced nodes, with the focus on uniform designs.
- **Standard Cell** routing technology seamlessly integrates the NanoRoute™ router in the Virtuoso environment. It provides different ways for you to generate WSPs as well as route without them, relying on Innovus-created tracks.

- **Chip Assembly** routing technology targets top-level designs that have macro instances, I/O pads, and can also contain standard cell areas. It also addresses memory type designs using spine routing.

Setup

The following table describes the fields available on the *Setup* tab of the Routing assistant for standard cell routing.

Field	Description
Options	Lets you setup some general options for standard cell routing.
Check DRCs after routing	Automatically runs DRD design rule checks before routing. Environment variable: setup_checkDRCsAfterRouting
Lock Colors after routing	Locks colored shapes for routing. This option is set based on the technology file layer rules. If the process has trim shapes that cut only locked color shapes, than the option is selected by default and editable. However, if the process has no trim shapes, the option is deselected by default and not editable. If, trim shapes work on locked and unlocked shapes, the option is selected by default and editable. Environment variable: setup_lockColorsAfterRouting
Process setup	Selects the process node to be used for routing.
Layer Settings	Lets you specify the layers for which to generate WSPs.
Routing Layers	Specifies valid bottom and top routing layers. The bottom and top layer fields contain both frontside layers and backside layers as defined in the technology file. When a backside layer is specified in the bottom and top layer fields, the standard cell Nano Router is unable to run. As a result, the routing process is aborted and an error message is displayed in CIW.
Wire Types Map	Displays a form with a table that maps the wireType to the symbol to be used in the pattern specification in the table.

Field	Description
Wire Types Assign to nets	Displays a form with a table that assigns wireTypes to nets.
Layer table	Provides a list of layers on which WSPs are generated.
Layer	The name of the layer.
Pattern	Generates the pattern of WSP as per your requirement. For example, `gs3ns2g3` where `g` stands for ground, `s` for signal, `n` for Null, `p` for power, and so on.
Dir	The routing direction of the layer. The supported routing directions include *Horizontal*, *Vertical*, *Orthogonal*, and *Forbid*. *Forbid* implies that no direction is considered.
W	The width specified for the layer.
S	The spacing specified for the layer.
WSP	The width spacing patterns defined for the layer.

Check

The following table describes the fields available on the *Check* tab of the Routing assistant for standard cell routing.

Field	Description
Check	Provides a list of various checks that can be run for the routing type.
Select	Lets you specify the checks to be run. Click *All* or *None* to select or deselect all checks with a single click.
Technology setup	Lets you specify the checks related to the technology setup.
R-MSOA PDK in tech graph	Checks that there is a R-MSOA library in the technology graph.
Design ITDB check	Checks if the design library has an incremental technology database (ITDB) library that references a Rapid MSOA PDK.
Abstract data	Lets you specify the checks related to abstract data in the design.
Abstract & layout views available	Checks for availability of abstract and layout views for standard cell instances.

Field	Description
Abstract & layout pins match	Checks that all pins match between layout and abstract views for standard cell instances.
Design interoperability	Lets you specify the design interoperability checks between Innovus and Virtuoso.
Net sigTypes specified	Checks that signal nets at level 0 are not connected to power and ground nets at level 1.
Net sigTypes consistency	Checks that there is a power and ground sigType net in the layout.
Existing wiring in route object	Checks for pre-existing wires in the routing objects.
Existing wiring compatible	Checks for the incompatible wires and wire segments. It also checks the name of standard via variants. If the via variant name is invalid, a message is displayed for the invalid name in CIW.
Pins on routing layers	Checks whether there are pins on non-routing layers in the design.
Complete NDR specification	Checks for the completeness of constraint group non-default rules (NDR), such as valid layers, spacing values, and so on.
Custom constraints compatible	Checks the compatibility of custom constraints.
Rows interoperable	Checks the row interoperability to ensure that each active row has a siteDef.
WSPs interoperable	Checks for the interoperability of active WSPs of each layer.
Pins conform to active WSPs	Checks for the conformance of pins to active WSPs.
Bus terms order & interface bit	Checks bus annotation and reports any bus terminals (`busTerms`) that do not have ordering (`busOrder`) information. It also checks the status of interface bits.
VLS XL compliant	Checks whether or not the design is XL compliant. This means that the connectivity is XL compliant so that hierarchy is set up correctly between terms and instTerms throughout the design hierarchy.

Field	Description
Express pcell cache	Checks for the presence of a Pcell cache.
Routability	Lets you specify the routability checks.
Existing DRCs	Specifies whether to run design rule checks using DRD. Environment variable: check_existingDRCs
Objects outside PR boundary	Lets you check for the objects that are outside the PR boundary.
Blocked pins	Lets you check if there are any blocked pins in the design.
Output	Settings to display the output of routing checks.
Display Log	Controls the display of the checker log in the CIW once the checks are run. Environment variable: check_displayLog
Overwrite last log	Controls the overwriting of the last log file. When the option is deselected, the existing log file is retained. Environment variable: check_overwriteLog, checkerLogDir
Markers	Controls the generation of markers for errors. These error markers can be viewed in the *Misc* tab of the Annotation Browser. Environment variable: check_generateMarkers

Supply

The following table describes the fields available on the *Supply* tab of the Routing assistant for standard cell routing.

Field	Description
Scope	Defines the scope of supply routing.
Supply Nets	Specifies whether to route all or selected supply nets. Environment variable: supply_nets

Field	Description
Within	Specifies whether to route everything inside the PR boundary or only within a guard ring or figGroup, or specify an area. You can also create supply stripes inside a row region or wsp region. The available options are: *PR boundary*, *Guardring/FigGroup*, *Area*, and *WSP/Row*. Environment variable: supply_netsWithin
Options	Lets you specify the options for supply routing.
Only use layers with WSP P/G tracks	Provides a list of layers in the *Supply stripes* cyclic field for bottom and top layers that have power or ground tracks in the active WSP for that layer. Environment variable: supply_useExisitingPGTracks
Supply stripes	Specifies the top and bottom layers of the supply stripes layer range that pins should be created on.
Generate supply stripes	Generates stripes when supply routing is run. Environment variable: supply_genSupplyStripes
Insert vias for supply stripes	Inserts the vias between the intersection of the layer above and below the via. Environment variable: supply_insertVias
Connect to overlapped terminals	Specifies whether or not the supply router should connect to IO pins and guard rings. Environment variable: supply_connectToTerminals
Insert trim to fix DRCs	Inserts trims between the intersection of the layer above and below the via in the supply grid to fix DRC errors. Environment variable: supply_insertTrim
Share tracks for supply nets	Distinguishes upper and below areas for one WSP track by instance's pin name. Environment variable: supply_shareTracks
Create supply grid as a group	Creates the supply grid as a figGroup. Environment variable: supply_createGridAsGroup

Field	Description
Ignore boundary tracks	Ignores generating stripes on tracks that are on the PR boundary. Environment variable: supply_IgnoreBoundaryTracks
Ignore boundary vias	Ignores generating vias on tracks that are on the PR boundary. Environment variable: supply_IgnoreBoundaryVias
Assign to nets	Opens a form with a table that assigns wireTypes to nets.
Pull-back	Controls the spacing of power nets per layer from the PR boundary to avoid any DRC violation.
Pins	Lets you specify the supply routing options for pins in the design.
Create	Creates a pin instead of a pathSeg for the supply stripe. Environment variable: supply_createPins
Create label	Creates labels on pins when the *Pins Create* option is selected in the *Supply* tab of the Routing assistant. Environment variable: supply_createPinLabel
Use all supply stripe layers	Creates pins on the layers on which stripes are generated. Environment variable: supply_pinLayerSet
Use selected layers	Creates pins on only the layers that are selected. Environment variable: supply_pinLayerSet
Create on ends	Creates pins on the ends of stripes instead of one long pin when the *Pins Create* option is selected. Environment variable: supply_createPinsOnEnds
Create on pin purpose	Lets you use a pin as the layer purpose for the created pin. Environment variable: supply_createPinsOnPinPurpose
Update	Lets you select the supply routing options to be deleted.
Delete Supply stripes	Deletes the stripes generated by the supply router before power routing. Environment variable: supply_deleteStripes
Delete Supply vias	Deletes the vias generated by the supply router before power routing. Environment variable: supply_deleteVias

Field	Description
Output	Lets you specify the settings to display the routing results.
Current cellview	Write the output of the supply routing to the current cellview. Environment variable: supply_defaultRoutedView, supply_routedLOC
Other cellview	Lets you select a view name to write the output of the supply routing to another cellview. You can also specify a non-database existing name. The name of the cell is `$cell_proute` with `_proute` as the postfix to the cell name. The view name is always `layout`. Environment variable: supply_defaultRoutedView, supply_routedLOC, supply_defaultRoutedCellExpression
Save routing only	Specifies whether to copy only the supply grid or all initial data and the supply grid to the new cellview. Environment variable: supply_saveRoutingOnly

Route

The following table describes the fields available on the *Route* tab of the Routing assistant for standard cell routing type.

Field	Description
Scope	Defines the scope of signal routing.
Nets	Specifies whether you want to route all nets, selected nets, or nets with opens or shorts. Environment variable: route_nets
Include Supply Nets	Specifies whether to route power and ground (`tieHi` and `tieLo`) nets. Environment variable: route_supplyNets
Within	Specifies whether to route everything inside the PR boundary or area. Environment variable: route_netsWithin
Options	Lets you specify the options for signal routing.
Update pins	Controls whether the placement and snapping of IO pins will be done in Innovus. Environment variable: route_updatePins

Field	Description
Snap to closest same edge track	Snaps the IO pin to the closest track on the same layer and same edge. Environment variable: route_updatePinOption
Snap to any edge	Snaps the IO pin to snap to any edge on any layer within the specified layer range. Environment variable: route_updatePinOption
Allowed layer range	Specifies the layer range on which to snap the IO pin to any edge.
Create routing as a group	Specifies whether or not to create routing as a figGroup. Environment variable: route_createRoutingAsAGroup
Fix post-route DRC errors	Resolves existing shorts and spacing violations by removing violating routes and repairing connectivity on the target nets. Environment variable: route_fixPostRouteDRCErrors
Update	Lets you select the signal routing options to be deleted.
Delete Wires and vias	Deletes the wires, vias, and shield lines created by the router. Environment variable: route_deleteWiresAndVias
Delete Manual routing	Deletes the pre-routed wires and vias that are created manually. Environment variable: route_deletePreroutes
Output	
Display log	Controls the display of the Innovus routing log when signal routing is run. Environment variable: route_displayLog
Overview last log	Specifies whether to overwrite the last log or keep the existing one. When the option is selected, the existing log is overwritten. Environment variable: route_createRoutingAsAGroup
Current cellview	Write the output of the supply routing to the current cellview. Environment variable: route_routedLOC, route_defaultRoutedView
Other cellview	Lets you specify a view name to write the output of the signal routing to another cellview. Environment variable: route_routedLOC, route_defaultRoutedView

Field	Description
Save routing only	Specifies whether or not to save standard cell routing settings. Environment variable: route_saveRoutingOnly

Results

The following table describes the fields available on the *Results* tab of the Routing assistant for standard cell routing type.

Field	Description
Scope	Defines the scope of the routing results.
Nets	Specifies whether you want to show the routing result of all nets, selected nets, or only nets with opens. Environment variable: results_nets
Include supply Nets	Shows the routing result of power and ground nets. Environment variable: results_supplyNets
Within	Shows the routing result of nets inside the PR boundary or within an area. When the option is selected as *Area*, the available icons are: ■ *Set area bbox to visible area* : Sets the visible area of the design canvas as the region. ■ *Draw the area bbox* : Lets you draw the region on the design canvas. Use *show/hide the area bbox* to display or hide the highlight around the region. Environment variable: results_netsWithin
Summary of latest run	Displays the summary of the routing results for various parameters, such as *Time*, *Instances*, *Nets*, *DRCs*, *Opens*, *Shorts*, *Wire Length*, *Vias*.
Output	Specifies the output columns to be displayed in the Routing Results Browser

Field	Description
Select	Lets you select either all or none parameters for which output should be displayed. The parameters are: *Rule Violations*, *Symmetry Violations*, *Matched Length Violations*, *Shield Violations, Opens*, *Shorts*, *DRCs*, *Wirelength*, *Vias*, *Ratio*, *Pin count*, *Manhattan X*, and *Manhattan Y*.

Routing Assistant Command Buttons

The following table lists the functions of the different command buttons on the Routing assistant:

Icon	Command	Description
	Snap pins to WSPs	Snaps IO pins to width spacing patterns.
	Show WSP Manager	Displays WSP Manager.
	Import WSPs	Imports width spacing patterns to the current cellview.
➕	*Auto-generate WSPs*	Generates width spacing patterns automatically on the selected layers.
✓	*Run pre-route checks*	Runs pre-routing checks and saves the result to a log file.
	Run power route	Runs power routing.
▶	*Run signal router*	Runs signal routing.
	Show results browser	Displays the Virtuoso Routing Results Browser.

Related Topics

Map WSP Wire Types to Symbols Form

Assign Wire Types to Nets Form

Pull Back and Offset Values Form

Routing Assistant

Accessing the Routing Assistant

Virtuoso Routing Results Browser

Virtuoso Pre-Route Browser

Virtuoso Routing Constraint Manager

Assign Wire Types to Nets Form

The Assign Wire Types to Nets form displays the nets with the power and ground signal type attribute that exists in the design. It also lets you specify on-the-fly net name that does not exist in the design. This form is used to set the sigType and for the supply router to know on which tracks to create stripes for nets.

Column	Description
Net	Specifies the net name.
Signal Type	Specifies the signal type. The two default signal types are supply and ground.
Wire Type	Lets you select the wire type. You can assign multiple wireTypes to a net. The wireTypes can be specified as a string of comma delimited names. When a wireType is not assigned for all the nets in the table, a warning message appears stating that table settings should be considered when wireType is specified for all rows. However, when the wire types are specified for all rows, the warning message no longer appears. In this case, the supply Router considers the Assign Wire Types to Nets table.
+	Adds the net name.
-	Deletes the net name and the related wireTypes information.

When *All* option is selected the Routing Assistant, the Assign Wire Types to Nets form displays all the Nets from the Navigator assistant. However, when the *Selected* option is enabled, the intersection of nets in the Assign Wire Types to Nets form and the nets selected in the Navigator assistant are displayed. For example, all the nets mentioned below are power and ground signal nets.

- Assign Wire Types to Nets table nets: A, B, C, D, E, X, and Y
- All Nets in Navigator: A, B, C, D, and E
- Selected Nets in Navigator: C, D, and E

In this case, the nets that will be displayed in the Assign Wire Types to Nets form are mentioned in the table below.

Example	Net scope is *All*	Net Scope is *Selected*
Assign Wire Types to Nets table is setup well	A, B, C, D, and E	C, D, and E

Related Topics

Map WSP Wire Types to Symbols Form

Assigning Wire Types to Nets

Automated Standard Cell Placement and Routing User Interface

Routing Assistant User Interface for Standard Cell

Boundary Cells Form

The following table describes the fields available in the Boundary Cells form.

Field	Description
Alignment	Lets you specify the position of the rail in a placement area. The table at the bottom of the form lets you view the alignment of the selected boundary cells.
Show selected only	Clears the filter field and only shows selected cells.
Clear selected button	Deselects all cells.
Choose Library	Opens the Library Browser from which you can select the library that contains the required cells. A list of layout and abstract views in the selected library are displayed in the *Cells* pane.
Cells	Lists cells that have their component class set to `BOUNDARYCELL`. You can select the required boundary cells from the list. If by default no boundary cells are listed, click *Choose Library* and select a library from the Library Browser. A list of layout and abstract views in the selected library are displayed in the *Cells* pane.

Field	Description
Alignment table	Mentions a number against each position of the rail in a placement area.

Related Topics

Automated Standard Cell Placement and Routing User Interface

Auto P&R Assistant User Interface for Standard Cell

Choose Source Layout Cellview Form

Use the Choose Source Layout Cellview form to select the library, cell, and view names to be used as a source layout.

Field	Description
Library	Specifies the library name for the source layout.
Cell	Specifies the cell name for the source layout.
View	Specifies the view name for the source layout.

Related Topics

Automated Standard Cell Placement and Routing User Interface

Routing Assistant User Interface for Standard Cell

Auto P&R Assistant User Interface for Standard Cell

Filler Cells Form

The following table describes the fields available in the *Filler Cells* form.

Field	Description
Show selected only	Clears the filter field and only shows selected cells.
Clear selected	Deselects all cells.

Field	Description
Choose Library	Opens the Library Browser from which you can select the library that contains the required cells. A list of layout and abstract views in the selected library are displayed in the *Cells* pane.
Cells	Lists cells that have their component class set to `FILLER`. You can select the required filler cells from the list. If by default no filler cells are listed, click *Choose Library* and select a library from the Library Browser. A list of layout and abstract views in the selected library are displayed in the *Cells* pane.

Related Topics

Automated Standard Cell Placement and Routing User Interface

Auto P&R Assistant User Interface for Standard Cell

Load Option Presets Form

The following table describes the fields available in the Load Option Presets form.

Field	Description
Select a preset for stdCell placer type	Lists all the available preset files that define standard cell placement settings. You can select the required preset file and apply the settings to the current design.

Related Topics

Auto P&R Assistant User Interface for Standard Cell

Map WSP Wire Types to Symbols Form

Use the Map WSP Wire Types to Symbols form is used to create WSPs.

The four default symbols in the pattern (`p: power`, `g: ground`, `s: signal`, `n: none`) correspond to built-in wire types. These symbols are permanent and cannot be changed. However, you can add more with a different symbol to use in your patterns and the wireTypes. The symbols that you specify becomes the wireTypes in the WSP. For example, you can specify wire type `vdd` with a symbol `a` and `vss` with a symbol `b`. You can than use `a` and `b` in the table to represent wireTypes `vdd` and `vss`, respectively, in the *Pattern* column of the *Setup* tab.

Column	Description
Wire Type	Displays the wire type.
Symbol	Displays the symbol assigned to represent the wireType.

Related Topics

Automated Standard Cell Placement and Routing User Interface

Assign Wire Types to Nets Form

Routing Assistant User Interface for Standard Cell

Pull Back and Offset Values Form

Use the Pull Back and Offset Values form to specify the distance the stripes should be pulled inward from the PR boundary on the ends and which strips should not be created on the other two sides. This helps to avoid shorts or DRC spacing errors if the blocks are abutted. The pull back value can be different for each layer and each side, depending on the routing layer direction.

Column	Description
Load	Lets you select the pull back file to be loaded from the Open Pullback File dialog box. The pull back values are loaded from the text file.
Clear	Clears the information of the loaded pull back file.
Layer	Displays the layer name.

Column	Description
Direction	Displays the routing direction of the layer.
Left/Bottom/ Right/Top	Lets you specify the pull back value for each side of the routing direction. This value is read as per the direction. If a single value is provided, it is applicable for both sides.

Related Topics

Automated Standard Cell Placement and Routing User Interface

Routing Assistant User Interface for Standard Cell

Save Option Presets Form

The following table describes the fields available in the Save Option Presets form.

Field	Description
Select an existing preset name (overwrite) or enter a new name	Specifies the name of the preset file in which the current placement options are to be saved. If you can specify a new name to create a new preset file or an existing file name to overwrite the existing preset file.
Select path	Specifies the path in which the preset file is to be saved.

Related Topics

Auto P&R Assistant User Interface for Standard Cell

Select Row Template Form

The following table describes the fields available in the Select Row Template form.

Field	Description
Library	Specifies the library name for the row template.
Cell	Specifies the cell name for the row template.

Field	Description
View	Specifies the view name for the row template.
Select Row Template	Lists the existing row templates.
Details	Provides the property details of the selected row template.

Related Topics

Automated Standard Cell Placement and Routing User Interface

Auto P&R Assistant User Interface for Standard Cell

Tap Cells Form

The following table describes the fields available in the Tap Cells form.

Field	Description
Show selected only	Clears all filters field shows only selected cells.
Clear selected	Deselects all cells.
Choose Library	Opens the Library Browser from which you can select the library that contains the required cells. A list of layout and abstract views in the selected library are displayed in the *Cells* pane.
Cells	Lists cells that have their component class set to `STDSUBCONT`. You can select the required tap cells from the list. If, by default, no tap cells are listed, click *Choose Library* and select a library from the Library Browser. A list of layout and abstract views in the selected library are displayed in the *Cells* pane.
Tap to Tap spacing	Lets you specify the tap-to-tap spacing for tap cells. This value comes from the Design Rule Manual (DRM).
Pattern	Lets you select a pattern to place the tap cells. The options are: *Regular* and *Checkerboard*.
Skip Row Count	Specifies the rows that must be skipped during tap cell placement.
Start Row Number	Specifies the row number to place the tap cells.

Field	Description
Offset from Row Start	Specifies the initial offset of the first tap cell in each row.
Avoid Abutment	Specifies whether the tap cells must not be abutted vertically.
Well Pitch	Specifies the minimum spacing between individual components placed in the placement area.
Well Offset	Defines an offset from the specified alignment. This lets you leave a channel around the edge of the placement area, or align the rail slightly off-center. Both positive and negative offset values are honored.

Related Topics

Automated Standard Cell Placement and Routing User Interface

Auto P&R Assistant User Interface for Standard Cell