Virtuoso Automated Chip Assembly Routing Flow Guide

Product Version IC23.1
October 2023

Printed in the United States of America.

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

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

Cadence is committed to using respectful language in our code and communications. We are also active in the removal and replacement of inappropriate language from existing content. This product documentation may however contain material that is no longer considered appropriate but still reflects long-standing industry terminology. Such content will be addressed at a time when the related software can be updated without end-user impact.

Contents

1

Chip Assembly Routing

The Virtuoso© chip assembly routing flow lets you route a top-level design that has macro instances, I/O pads, devices, and standard cells. It consists of a series of tasks to quickly and automatically generate routed layouts that are constraint compliant and LVS correct, and follow DRCs as captured in the Virtuoso technology file.

This chip assembly routing flow helps circuit and layout designers who are familiar with routing flows on mature nodes for their designs as well as for CAD teams that support such designs. The target designs for this flow are top-level designs.

For chip assembly routing, you use the Routing assistant. This assistant is available in the Layout MXL cockpit.

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

Related Topics

Environment Setup for Chip Assembly Routing Flow

Routing Assistant

Chip Assembly Routing Configuration

Routing Assistant User Interface for Chip Assembly Routing Flow

Environment Setup for Chip Assembly Routing Flow

To run the chip assembly routing flow, ensure that you have access to the following:

- **Virtuoso Release**

  IC23.1or later

- **PDK Settings**

  Usually, PDK settings are available from the foundry.

- **Macro cell library with layout views**

  - A library that consists of layout macros, standard cells, and cell arrays with different sizes and aspect ratios.
  - If layout views are being instantiated, cover obstructions can be used.

- **License**

  The chip assembly routing solution requires Virtuoso Layout Suite MXL tier. The chip assembly router that is available for advanced nodes designs displays the options same as that for the standard cell routing in the Routing assistant. The chip assembly router options are not displayed for advanced node designs.

Related Topics

Chip Assembly Routing

Chip Assembly Routing Configuration

2

Routing Assistant

The Routing assistant is a simplified, consistent, flow-based, dockable assistant that provides routing options for all design types. It provides three routing modes - device, standard cell, and chip assembly with a common toolbar for all routing types.

The Routing assistant lets you:

- Select the routing mode and design type
- Load and save routing options
- Manage the routing flow through dedicated tabs
  - ❑ Lets you define the scope and options for each task and specifying the output
  - ❑ Lets you set up and run pre-routing checks, generate power grids, route the design, and view the routing results
  - ❑ Lets you load, save, and delete routing presets
  - ❑ Provides access to the Pre-Routing Browser
  - ❑ Provides access to the Routing Results Browser
- Display alerts and other information about the width spacing patterns you are creating and updating
- Access the integrated Routing Constraint Manager to manage all routing constraints

Related Topics

Accessing the Routing Assistant

Loading, Saving, and Deleting a Routing Preset

Routing Assistant User Interface for Chip Assembly Routing Flow

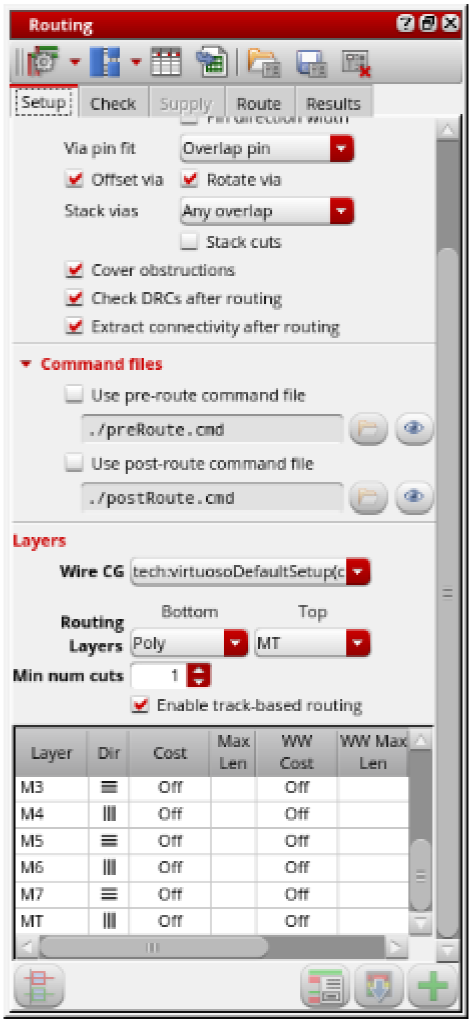
Accessing the Routing Assistant

To display the Routing assistant:

1. Start Virtuoso and launch Layout MXL.
2. Choose *Window – Assistants – Routing Assistant*.

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

   The Routing Assistant is added as a docked assistant to the current layout window. By default, the Routing Assistant is positioned to the right in the session window.



Via pin fit: Overlap pin

☑ Offset via ☑ Rotate via

Stack vias: Any overlap

□ Stack cuts

☑ Cover obstructions

☑ Check DRCs after routing

☑ Extract connectivity after routing

Command files

□ Use pre-route command file

./preRoute.cmd

□ Use post-route command file

./postRoute.cmd

Layers

Wire CG tech:virtuosoDefaultSetup(c

Routing Layers Bottom: Poly Top: MT

Min num cuts 1

☑ Enable track-based routing

Layer	Dir	Cost	Max Len	WW Cost	WW Max Len
M3	≡	Off		Off	
M4	‖‖	Off		Off	
M5	≡	Off		Off	
M6	‖‖	Off		Off	
M7	≡	Off		Off	
MT	‖‖	Off		Off	

To hide the Routing assistant, do one of the following:

- Click the *Hide* button in the Routing Assistant title bar.

- Right-click anywhere in the layout window menu bar and choose *Assistants – Routing*.
- Right-click anywhere in the assistant's title bar and choose *Routing*.

Related Topics

Routing Assistant

Loading, Saving, and Deleting a Routing Preset

Routing Assistant User Interface for Chip Assembly Routing Flow

Loading, Saving, and Deleting a Routing Preset

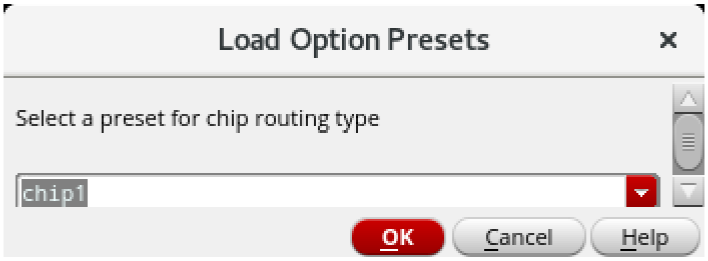
A preset is a set of predefined routing options and user override constraint values, which can be saved to a file. In the routing flow, the presets are used to reuse the settings and routing configurations that you had specified in the Routing assistant.

Loading a Preset

The *Load Preset options* button lets you load an already existing preset. To load a preset for specific router settings:

1. Click the *Load preset options* button on the Routing assistant toolbar.

   The Load Option Presets form is displayed.

   Load Option Presets

   Select a preset for chip routing type

   chip1

2. Select a preset for the routing type that is selected in the Routing assistant.
3. Click *OK*.

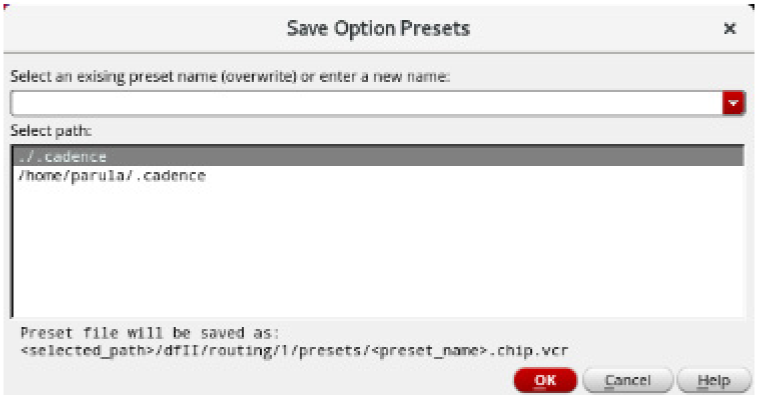
   The settings for the selected routing type in the selected preset file are loaded in to the Routing assistant.

Saving a Preset

The *Save preset options* button lets you save the routing settings variables to a preset file. To save specific router settings as a preset:

1. Specify the required Routing assistant settings as required by your design objectives.
2. Click the *Save preset options* button on the Routing assistant toolbar.

   The Save Option Presets form is displayed.



Select an exising preset name (overwrite) or enter a new name:

Select path:

```
./.cadence
/home/parula/.cadence
```

```
Preset file will be saved as:
<selected_path>/dfII/routing/1/presets/<preset_name>.chip.vcr
```

3. Select an existing preset name of enter a new name.
4. Select the path where you want to save the preset file.
5. Click *OK*.

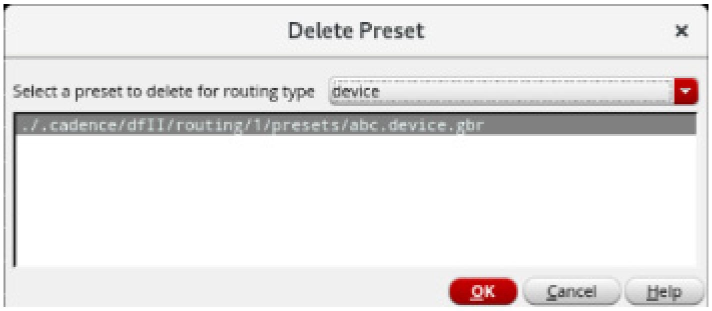
   The preset file is saved as `*filename*.chip.vcr`.

Deleting a Preset

The *Delete preset options* button lets you delete an existing chip assembly routing preset. To delete an existing preset:

1. Click the *Delete preset options* button on the Routing assistant toolbar.

The Delete Preset form is displayed.



Select a preset to delete for routing type: device

./.cadence/dfII/routing/1/presets/abc.device.gbr

2. Select the routing type for which you want to delete the preset.

   The presets for the selected routing type appears in the list box.

3. Select the preset that you want to delete for the selected routing type.

4. Click *OK*.

   The selected preset file is deleted.

Related Topics

Routing Assistant

Chip Assembly Routing

Accessing the Routing Assistant

Routing Assistant User Interface for Chip Assembly Routing Flow

Routing Assistant User Interface for Chip Assembly Routing Flow

The chip assembly router that is available for advanced nodes designs displays the options same as that for the standard cell routing in the Routing assistant. The chip assembly router options are not displayed for advanced node designs.

The Routing assistant has the following components, which are common for all routing types.:

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. The options in the Routing assistant depend on the selected routing type in the design: *device*, *stdcell*, *Chip Assembly*.
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*.
	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 to specify routing constraints such as process rule override, shielding, bus etcetera.
	Load preset options	Loads options from an existing preset file.
	Save preset options	Saves the Routing Assistant options to a file.
	Delete preset options	Deletes 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 to be run before routing the design.
Supply	This tab is disabled for chip assembly routing.

Tab	Description
Route	Lets you specify the scope of signal routing and runs the router.
Results	Lets you select the scope of results to be displayed in Routing Results Browser.

Let us now see the Chip Assembly routing options that are available in various tabs of the Routing assistant.

Setup

The following table describes the fields available on the *Setup* tab of the Routing assistant for chip assembly routing type.

Field	Description
Options	
Protect existing routing	Protects any existing routing in the design. Environment variable: setup_protectExistingRouting
Cut keepouts	Specifies whether blockages in the design should be cut out for pin access. Environment variable: setup_cutKeepouts
Taper	Specifies how the router should taper the connected pins, when needed. This option is selected by default. Environment variable: setup_taper
Up	Tapers the connected pins (neck up). This option is selected by default. Environment variable: setup_taperUp
Down	Tapers the connected pins (neck down). This option is selected by default. Environment variable: setup_taperDown
Pin direction width	Specifies the width of the edge accessible to the router instead of applying a width corresponding to the largest edge of the pin shape.This is for tapering (pin escape). Environment variable: setup_pinDirectionWidth

Field	Description
Via Pin fit	Specifies how a via should fit on pins and how vias are stacked. This option is deselected by default. The valid values are: ■ *Any fit* lets you use any option, as required by the design. ■ *Overlap pin* lets a via freely overlap with the pin boundary. This is the default value. ■ *Within pin* indicates that overlapping vias must fit entirely within the pin boundary. ■ *Centered* aligns the overlapping vias at the center point within the pin boundary. Environment variable: setup_pinFit
Offset via	Specifies whether the router should offset rectangular vias, such that its edges coincide with the wire outline. The vias can either be square or rectangular. This option is selected by default. Environment variable: setup_offsetVia
Rotate via	Rotates vias automatically, if needed by the router. Environment variable: setup_rotateVia
Stack vias	Select one of the following options to align stacked vias at the center point. ■ *Do not use* indicates that via stacking is disabled. ■ *Any overlap* stacks vias with both centered and off-centered overlaps. This is the default value. ■ *Exact* indicates that vias are only stacked center-to-center. Environment variable: setup_viaStackOption
Stack cuts	Stack vias with any overlap and allows cut shapes to overlap or spaces the via cuts according to the interlayer cut clearance rule. This option is unselected by default and available only if the *Stack* option is set to *Any overlap*. Environment variable: setup_viaStackCuts

Field	Description
Cover obstructions	Considers cover obstruction information in the design. This information is created by the Cover Obstruction Manager. Environment variable: setup_coverObstructions
Check DRCs after routing	Automatically runs DRD design rule checks after routing. Environment variable: setup_checkDRCsAfterRouting
Extract connectivity after routing	Runs the Layout XL extractor after routing is complete. Environment variable: setup_extractConnectivityAfterRouting
Command files	Specifies the command files for pre- and post- routing.
Use pre-route command file	Enables the use of a pre-route command file. The text field next to the option lets you specify a file to be used to define the advanced routing options. These options are for initialization and are used before the router starts. Environment variable: setup_preRouteCmdFileName, setup_usePreRouteCmdFile
Use post-route command file	Enables the use of a post-route command file. The text field next to the option lets you specify a file that can be used to define the advanced routing options. These options are used after routing is complete. Environment variable: setup_postRouteCmdFileName, setup_usePostRouteCmdFile
Layers	The options in this section let you specify the details for the layers to be used while routing the design.
Wire CG	Lets you select one of the available constraint groups, which may be defined in the technology file or at the design level.
Routing Layers	Lets you select the bottom and the top routing layers.
Min num cuts	Specifies the minimum number of cuts.
Enable track-based routing	Specifies whether or not the track patterns should be used for routing. Environment variable: setup_useTrackBasedRouting
Layer	Displays the name of the routing layer.
Dir	Sets the routing direction for the layer to one of the following values: *Horizontal*, *Vertical*, and *Orthogonal*.

Field	Description
Cost	Lets you select an option to limit the layer usage. The available options are: *Off*, *Low*, *Medium*, *High*, and *Forbid*. The default value is *Off*.
Max Len	Specifies the maximum length of the routing.
WW Cost	Lets you select the option to limit routing in the wrong direction. The available options are: ■ *Off* lets the router to go in both directions. ■ *Low* ■ *Medium* ■ *High* ■ *Forbid* restricts the router to go wrong way. The default value is *Off*.
WW Max Len	Specifies the maximum length for wrong-way routing.
Min W	Specifies the minimum layer width.
Max W	Specifies the maximum layer width.
Min S	Specifies the layer spacing.

Check

You can run pre-routing checks to identify design issues before routing the design. Running the checks lets you identify potential situations or objects that can cause a problem for the router later in the flow.

The following table describes the pre-routing checks available on the *Check* tab of the Routing assistant for chip assembly routing type.

Field	Description
Scope	
Nets	Specifies whether you want to route all nets or selected nets.
Include supply nets	Specifies whether to route power and ground (`tieHi` and `tieLo`) nets.
Check	

Field	Description
Select	Lets you select either all or no checks to be performed before routing the design. You can also specify selected checks.
Routing Tracks	Checks whether the tracks are available for routing.
Objects Outside PR Boundary	Checks if there are any objects outside the PR boundary.
Blocked Pins	Checks for any blocked pins before routing the design.
Off Grid Pins	Checks for any off-grid pins before routing the design.
Pin Width Violations	Checks for violations related to the width of a pin.
Bus Net Terminals	Checks the consistency of buses in terms of the number of terminals at source and at destination.
Existing DRCs	Specifies whether to run design rule checks for DRD. Environment variable: check_existingDRCs
Route Paths	Searches the layout and report any paths that exists and places a marker on them.
Output	
Display Log	Displays the checker log window once the checks are run. By default, the option is selected. Environment variable: check_displayLog
Overwrite last log	Overwrites the last log file. When the option is unselected, the existing log file is retained. By default, the option is selected. Environment variable: check_overwriteLog
Markers	Displays error markers on the *Misc* tab of the Annotation Browser. Environment variable: check_generateMarkers

Route

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

Field	Description
Scope	Defines the scope of routing.
Nets	Specifies whether you want to route all nets, selected nets, or only nets with opens or shorts. Environment variable: route_nets
Include supply nets	Includes power and ground (`tieHi` and `tieLo`) nets to the list of nets to be routed. Environment variable: route_supplyNets
Within	Specifies whether to route everything inside the PR boundary. Environment variable: route_netsWithin
Routing Options	Lets you specify the routing options for *Auto* routing.
Auto	Lets you specify the options for automatic routing types, such as global, detailed, and optimize.
Global route	Enables global routing, where routing constraints such as width and spacing per net are honored. Environment variable: route_globalRoute
passes	Specifies the number of passes for global routing. The default value is `3`. Environment variable: route_globalRoutePasses
Detail route	Enables detailed routing. During detailed routing, the automatic router uses the pathways suggested by global routing to route the connections in the design. Using the global router before the detailed router limits the search space and reduces the overall routing time. Environment variable: route_detailRoute
passes	Specifies the number of passes for detailed routing. Detailed routing passes are run after global routing. The default value is `25`. Environment variable: route_detailRoutePasses

Field	Description
Optimize route	Optimizes the routing results. Environment variable: route_optimizeRoute
passes	Specifies the number of passes for optimized routing. The default value is `3`. Environment variable: route_optimizeRoutePasses
Remove shorts & DRCs	Runs post-routing step to remove shorts and DRC errors. Environment variable: route_removeShortsAndDRCs
passes	Specifies the number of passes to remove shorts and DRC errors. Environment variable: route_removeShortsAndDRCsPasses
Generate shield	Enables the routing of shielding wires that are generated by the router. If the option is unselected, the shielding wires can still be routed as a post-route process. In such cases, the router considers the extra spacing required by shielding while routing the regular nets. Environment variable: route_generateShield
Tie shield	Enables the tying of shielding wires together to bring the connected shielding wires to the nearest power source, which can be part of a power pin, power ring, or trunk. Environment variable: route_tieShield
Update	
Remove jogs	Removes wire jogs considering connections between multiple layers through vias. This option is needed when the router or pre-routed wires might have unnecessary jogs.
Remove notches	Removes small notches between the wires and pins.
Remove Shorts	
Change vias to double cut	Replaces single-cut vias in the design with double-cut vias. This is inactive as of now.
Delete	Specifies whether you want to delete all types of routing objects or the selected objects.
Delete Spines	Deletes the spine wires that are created by the router. Environment variable: route_deleteSpines

Field	**Description**
Delete Wires and vias	Deletes the wires and vias that are created by the router. Environment variable: route_deleteWiresAndVias
Delete Preroutes	Deletes the pre-routed wires and vias that are manually created. Environment variable: route_deletePreroutes
Delete Shields	Deletes the shields that are created by the router. Environment variable: route_deleteShields
Delete Shield ties	Deletes the tie shields that are created by the router. Environment variable: route_deleteShieldTies
Output	
Display log	Displays the log window when the chip assembly router is run. Environment variable: route_displayLog
Current cellview	Writes the routing results to the current cellview. Environment variable: route_defaultRoutedView, route_routedLoc
Other cellview	Writes the routing results to another cellview. Click the ellipses button and select the cellview from the Specify Routed Cellview form. Environment variable: route_defaultRoutedView, route_routedLoc
Save routing only	Saves all routing objects, such as wires and vias, to the specified cellview. All other objects, such as instances or pcells for instance, are not saved. Environment variable: route_saveRoutingOnly

Results

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

Field	**Description**
Scope	Defines the scope of the routing results.

Field	Description
Nets	Specifies whether you want to show the routing result of all nets, selected nets, or only nets with opens and shorts. 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 or figGroup. 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*, and *Vias*.
Output	Specifies the output columns to be displayed in the Routing Results Browser
Select	Lets you select the 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 in the Routing assistant for chip assembly routing.

Icon	Command	Description
	Snap pins to WSPs	Unavailable for chip assembly routing.
	Show WSP Manager	Unavailable for chip assembly routing.
	Import WSPs	Unavailable for chip assembly routing.
+	*Auto-generate WSPs*	Unavailable for chip assembly routing.
✓	*Run pre-route checks*	Runs pre-routing checks and saves the result to a log file.

Icon	Command	Description
▶	*Run Signal router*	Runs signal routing.
▦	*Show results browser*	Displays the Virtuoso Routing Results Browser.

Related Topics

Routing Assistant

Accessing the Routing Assistant

Virtuoso Routing Constraint Manager

Opening Routing Constraint Manager

Routing Constraint Manager User Interface

Virtuoso Pre-Route Browser

Virtuoso Routing Results Browser

3

Chip Assembly Routing Configuration

Chip assembly routing is the preferred routing solution for designs that have macro instances, I/O pads, and also contains devices and standard cells. You can specify the chip assembly routing options by loading an existing preset or customize the routing settings using the Routing assistant. For more information, see Loading, Saving, and Deleting a Routing Preset.

The following steps summarize the Chip Assembly Routing flow to customize routing settings.

- **Configure the Router**

  The first step in the chip assembly routing flow is to set up the router.

- **Check Routability**

  In this step, you select the checks that should run before routing.

- **Set Routing Options for Automatic Mode**

  Next, you set routing options for automatic mode. This routing mode is appropriate for channel types of designs.

- **Generate Shields**

  Adds a shield for signal nets.

- **View and Analyze Routing Results**

  In this step, you analyze the routing results from a single table using the Routing Results Browser. It shows routing information such as the number of opens, shorts, wire lengths, design rule checks, and vias.

Related Topics

Chip Assembly Routing

Routing Assistant User Interface for Chip Assembly Routing Flow

Configuring Chip Assembly Routing Settings

Checking Layout Routability in Chip Assembly

Routing in Automatic Mode

Generating Shields

Viewing and Analyzing Chip Assembly Routing Results

Configuring Chip Assembly Routing Settings

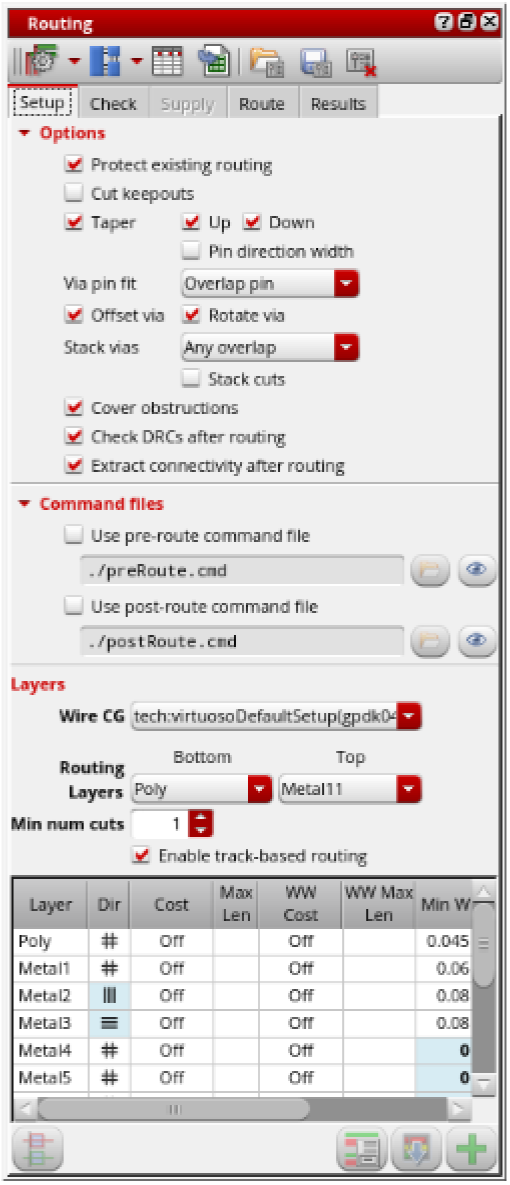
Before running the chip assembly router, you need to specify the, you can specify the various routing options made available in the Routing assistant for chip assembly routing. To configure router settings:

1. Open the required 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 appears.

3. Select Chip Assembly from the Change routing type drop-down list.



Options

☑ Protect existing routing

☐ Cut keepouts

☑ Taper ☑ Up ☑ Down

☐ Pin direction width

Via pin fit: Overlap pin

☑ Offset via ☑ Rotate via

Stack vias: Any overlap

☐ Stack cuts

☑ Cover obstructions

☑ Check DRCs after routing

☑ Extract connectivity after routing

Command files

☐ Use pre-route command file

./preRoute.cmd

☐ Use post-route command file

./postRoute.cmd

Layers

Wire CG: tech:virtuosoDefaultSetup(gpdk04

Routing Layers: Bottom: Poly Top: Metal11

Min num cuts: 1

☑ Enable track-based routing

Layer	Dir	Cost	Max Len	WW Cost	WW Max Len	Min W
Poly	#	Off		Off		0.045
Metal1	#	Off		Off		0.06
Metal2	|||	Off		Off		0.08
Metal3	≡	Off		Off		0.08
Metal4	#	Off		Off		0
Metal5	#	Off		Off		0

4. In the *Setup* tab, Select *Protect existing routing* to protect any existing routing in the design.

5. Select *Cut keepouts* to specify whether the blockages should be cut out for pin access.

6. Select *Taper* to specify how the router should connect to pins, *Taper Up* and *Taper Down*.

7. Specify how a via should fit on pins and how vias are stacked. The available options are: *Any fit*, *Overlap Pin*, *Via Within pin*, and *Centered*.

8. Select *Offset vias* to allow offset vias to the wires. The vias can either be square or rectangular.
9. Specify an option to align stacked vias at the center point. The available options are: *Do not use*, *Any Overlap*, and *Exact*.
10. Select *Stack cuts* to stack vias allowing cut shapes to overlap or space via cuts according to the inter-layer cut clearance rule. This option is available only if the *Stack* option is set to *Any overlap*.
11. Select *Cover obstructions* to consider the cover obstruction information in the design. This information is created by the Cover Obstruction Manager.
12. Specify the pre-route and post-route command files to be run before and after running the router in the *Use pre-route command file* and *Use post-route command file* fields.
13. Set the routing layer range using the *Bottom* and *Top* drop-down lists.
14. Specify the minimum number of cuts in the *Min num cuts* field.
15. Select *Enable track-based routing* to enable routing with track patterns.
16. Change the layer direction, cost, minimum width, minimum spacing, maximum length, and maximum width of the routing layer in the *Layers* table.

    Once the router settings are configured, you can run the pre-routing check.

Related Topics

Chip Assembly Routing

Routing Assistant User Interface for Chip Assembly Routing Flow

Checking Layout Routability in Chip Assembly

Routing in Automatic Mode

Generating Shields

Viewing and Analyzing Chip Assembly Routing Results

Checking Layout Routability in Chip Assembly

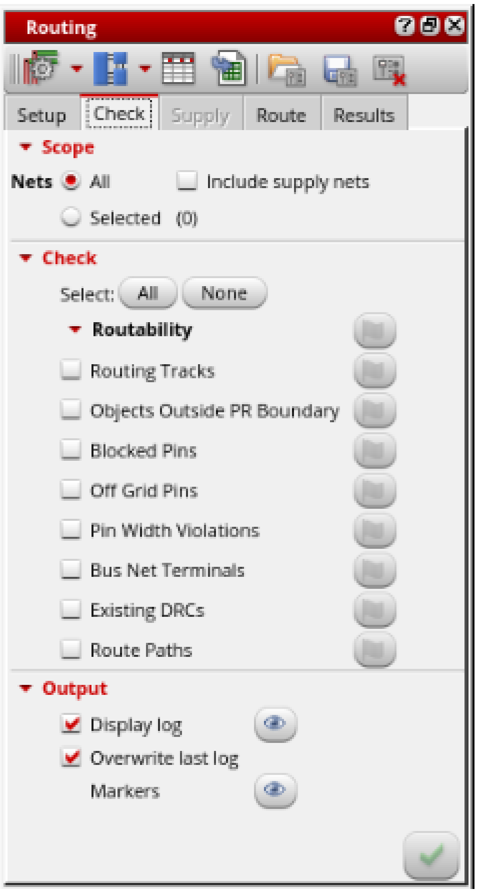
You can run pre-routing checks to detect design issues before routing a design. Running the checks lets you identify potential situations or objects that may cause trouble for the router later in the flow.

To check the routability of a design:

1. Open the 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 Routing assistant, click the *Check* tab.



Routing

Setup | Check | Supply | Route | Results

Scope

Nets ● All □ Include supply nets
○ Selected (0)

Check

Select: All None

Routability

□ Routing Tracks
□ Objects Outside PR Boundary
□ Blocked Pins
□ Off Grid Pins
□ Pin Width Violations
□ Bus Net Terminals
□ Existing DRCs
□ Route Paths

Output

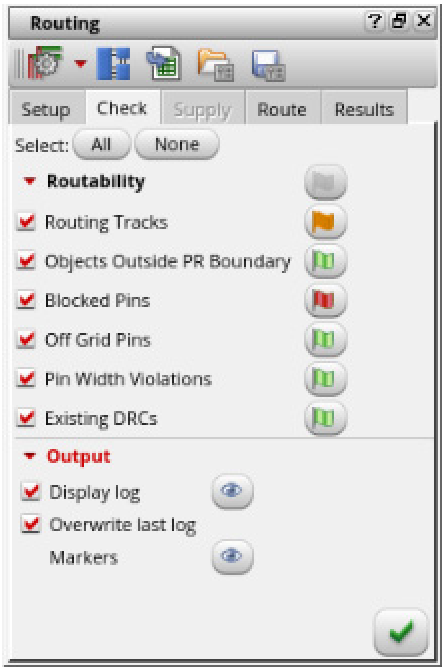
☑ Display log
☑ Overwrite last log
Markers

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

5. Deselect the *Routing Tracks* and *Existing DRCs* checks.

6. Click the *Run pre-route checks* ✓ button.

   Afteryou have run the checks, the status flags appear in green, orange, and red colors. 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 routable.



Routability

☑ Routing Tracks

☑ Objects Outside PR Boundary

☑ Blocked Pins

☑ Off Grid Pins

☑ Pin Width Violations

☑ Existing DRCs

Output

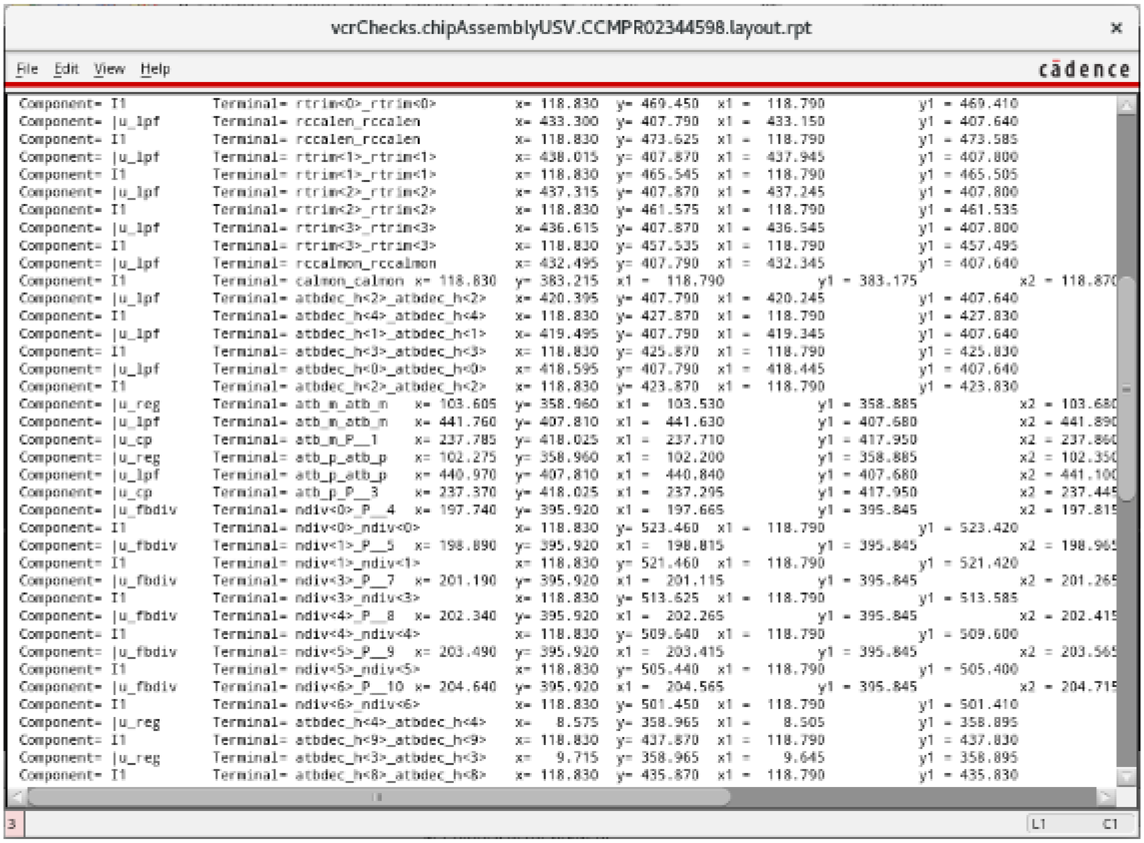
☑ Display log

☑ Overwrite last log

Markers

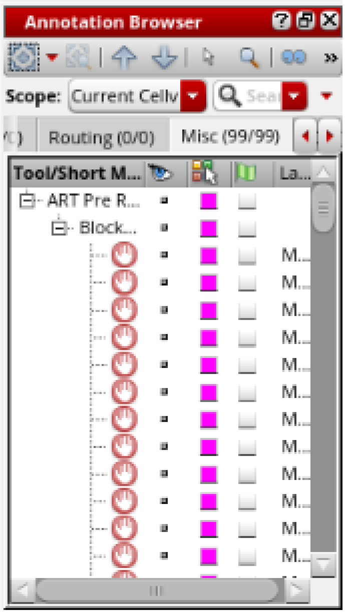
Clicking a flag button takes you to the location in the log file where that check was run.

7. Click the button next to the *Display log* option to view the log file and check for any issues.



```
Component= I1        Terminal= rtrim<0>_rtrim<0>          x= 118.830  y= 469.450  x1 =  118.790          y1 = 469.410
Component= |u_lpf    Terminal= rccalen_rccalen            x= 433.300  y= 407.790  x1 =  433.150          y1 = 407.640
Component= I1        Terminal= rccalen_rccalen            x= 118.830  y= 473.625  x1 =  118.790          y1 = 473.585
Component= |u_lpf    Terminal= rtrim<1>_rtrim<1>          x= 438.015  y= 407.870  x1 =  437.945          y1 = 407.800
Component= I1        Terminal= rtrim<1>_rtrim<1>          x= 118.830  y= 465.545  x1 =  118.790          y1 = 465.505
Component= |u_lpf    Terminal= rtrim<2>_rtrim<2>          x= 437.315  y= 407.870  x1 =  437.245          y1 = 407.800
Component= I1        Terminal= rtrim<2>_rtrim<2>          x= 118.830  y= 461.575  x1 =  118.790          y1 = 461.535
Component= |u_lpf    Terminal= rtrim<3>_rtrim<3>          x= 436.615  y= 407.870  x1 =  436.545          y1 = 407.800
Component= I1        Terminal= rtrim<3>_rtrim<3>          x= 118.830  y= 457.535  x1 =  118.790          y1 = 457.495
Component= |u_lpf    Terminal= rccalmon_rccalmon          x= 432.495  y= 407.790  x1 =  432.345          y1 = 407.640
Component= I1        Terminal= calmon_calmon  x= 118.830  y= 383.215  x1 =  118.790          y1 = 383.175          x2 = 118.870
Component= |u_lpf    Terminal= atbdec_h<2>_atbdec_h<2>    x= 420.395  y= 407.790  x1 =  420.245          y1 = 407.640
Component= I1        Terminal= atbdec_h<4>_atbdec_h<4>    x= 118.830  y= 427.870  x1 =  118.790          y1 = 427.830
Component= |u_lpf    Terminal= atbdec_h<1>_atbdec_h<1>    x= 419.495  y= 407.790  x1 =  419.345          y1 = 407.640
Component= I1        Terminal= atbdec_h<3>_atbdec_h<3>    x= 118.830  y= 425.870  x1 =  118.790          y1 = 425.830
Component= |u_lpf    Terminal= atbdec_h<0>_atbdec_h<0>    x= 418.595  y= 407.790  x1 =  418.445          y1 = 407.640
Component= I1        Terminal= atbdec_h<2>_atbdec_h<2>    x= 118.830  y= 423.870  x1 =  118.790          y1 = 423.830
Component= |u_reg    Terminal= atb_m_atb_m    x= 103.605  y= 358.960  x1 =  103.530          y1 = 358.885          x2 = 103.680
Component= |u_lpf    Terminal= atb_m_atb_m    x= 441.760  y= 407.810  x1 =  441.630          y1 = 407.680          x2 = 441.890
Component= |u_cp     Terminal= atb_m_P__1     x= 237.785  y= 418.025  x1 =  237.710          y1 = 417.950          x2 = 237.860
Component= |u_reg    Terminal= atb_p_atb_p    x= 102.275  y= 358.960  x1 =  102.200          y1 = 358.885          x2 = 102.350
Component= |u_lpf    Terminal= atb_p_atb_p    x= 440.970  y= 407.810  x1 =  440.840          y1 = 407.680          x2 = 441.100
Component= |u_cp     Terminal= atb_p_P__3     x= 237.370  y= 418.025  x1 =  237.295          y1 = 417.950          x2 = 237.445
Component= |u_fbdiv  Terminal= ndiv<0>_P__4  x= 197.740  y= 395.920  x1 =  197.665          y1 = 395.845          x2 = 197.815
Component= I1        Terminal= ndiv<0>_ndiv<0>            x= 118.830  y= 523.460  x1 =  118.790          y1 = 523.420
Component= |u_fbdiv  Terminal= ndiv<1>_P__5  x= 198.890  y= 395.920  x1 =  198.815          y1 = 395.845          x2 = 198.965
Component= I1        Terminal= ndiv<1>_ndiv<1>            x= 118.830  y= 521.460  x1 =  118.790          y1 = 521.420
Component= |u_fbdiv  Terminal= ndiv<3>_P__7  x= 201.190  y= 395.920  x1 =  201.115          y1 = 395.845          x2 = 201.265
Component= I1        Terminal= ndiv<3>_ndiv<3>            x= 118.830  y= 513.625  x1 =  118.790          y1 = 513.585
Component= |u_fbdiv  Terminal= ndiv<4>_P__8  x= 202.340  y= 395.920  x1 =  202.265          y1 = 395.845          x2 = 202.415
Component= I1        Terminal= ndiv<4>_ndiv<4>            x= 118.830  y= 509.640  x1 =  118.790          y1 = 509.600
Component= |u_fbdiv  Terminal= ndiv<5>_P__9  x= 203.490  y= 395.920  x1 =  203.415          y1 = 395.845          x2 = 203.565
Component= I1        Terminal= ndiv<5>_ndiv<5>            x= 118.830  y= 505.440  x1 =  118.790          y1 = 505.400
Component= |u_fbdiv  Terminal= ndiv<6>_P__10 x= 204.640  y= 395.920  x1 =  204.565          y1 = 395.845          x2 = 204.715
Component= I1        Terminal= ndiv<6>_ndiv<6>            x= 118.830  y= 501.450  x1 =  118.790          y1 = 501.410
Component= |u_reg    Terminal= atbdec_h<4>_atbdec_h<4>    x=   8.575  y= 358.965  x1 =    8.505          y1 = 358.895
Component= I1        Terminal= atbdec_h<9>_atbdec_h<9>    x= 118.830  y= 437.870  x1 =  118.790          y1 = 437.830
Component= |u_reg    Terminal= atbdec_h<3>_atbdec_h<3>    x=   9.715  y= 358.965  x1 =    9.645          y1 = 358.895
Component= I1        Terminal= atbdec_h<8>_atbdec_h<8>    x= 118.830  y= 435.870  x1 =  118.790          y1 = 435.830
```

8. To view the issues reported for the pre-route checks, click the button next to the *Markers* option. The issues are displayed in the *Misc* tab of the Annotation Browser.



Related Topics

Chip Assembly Routing

Routing Assistant User Interface for Chip Assembly Routing Flow

Configuring Chip Assembly Routing Settings

Routing in Automatic Mode

Generating Shields

Viewing and Analyzing Chip Assembly Routing Results

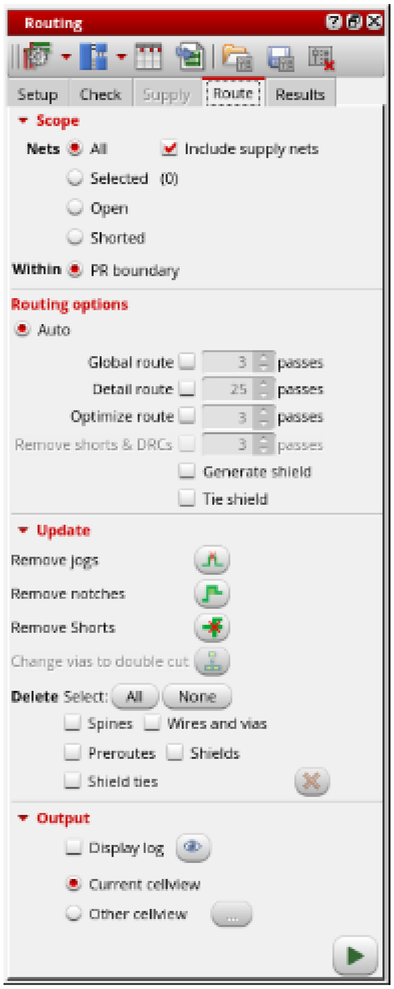
Routing in Automatic Mode

The *Auto* mode of Chip Assembly routing is the most appropriate routing style for channel type of designs. To obtain a clean result and high quality of results, global, detail, and optimize routing steps are all required, with 3, 10, and 3 passes, respectively. Depending on the design size, the PDK, the floorplan, and the constraints, the number of passes can be adjusted.

To run the router:

1. Open a design in Layout MXL.
2. Select the nets of interest from the Navigator assistant.
3. Choose *Window – Assistants – Routing*.

   Alternatively, right-click anywhere on the layout window menu bar and choose *Assistants – Routing*.
4. 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

Routing options

● Auto

Global route □ 3 passes
Detail route □ 25 passes
Optimize route □ 3 passes
Remove shorts & DRCs □ 3 passes
□ Generate shield
□ Tie shield

Update

Remove jogs
Remove notches
Remove Shorts
Change vias to double cut

Delete Select: All None
□ Spines □ Wires and vias
□ Preroutes □ Shields
□ Shield ties

Output

□ Display log
● Current cellview
○ Other cellview ...

5. Select the nets to route. You can either select *All*, *Selected*, *Open*, or *Shorted* nets.
6. Select *Include Supply Nets* to include power and ground nets to the list of nets to be routed.
7. Select *PR boundary* to run routing within the PR boundary.

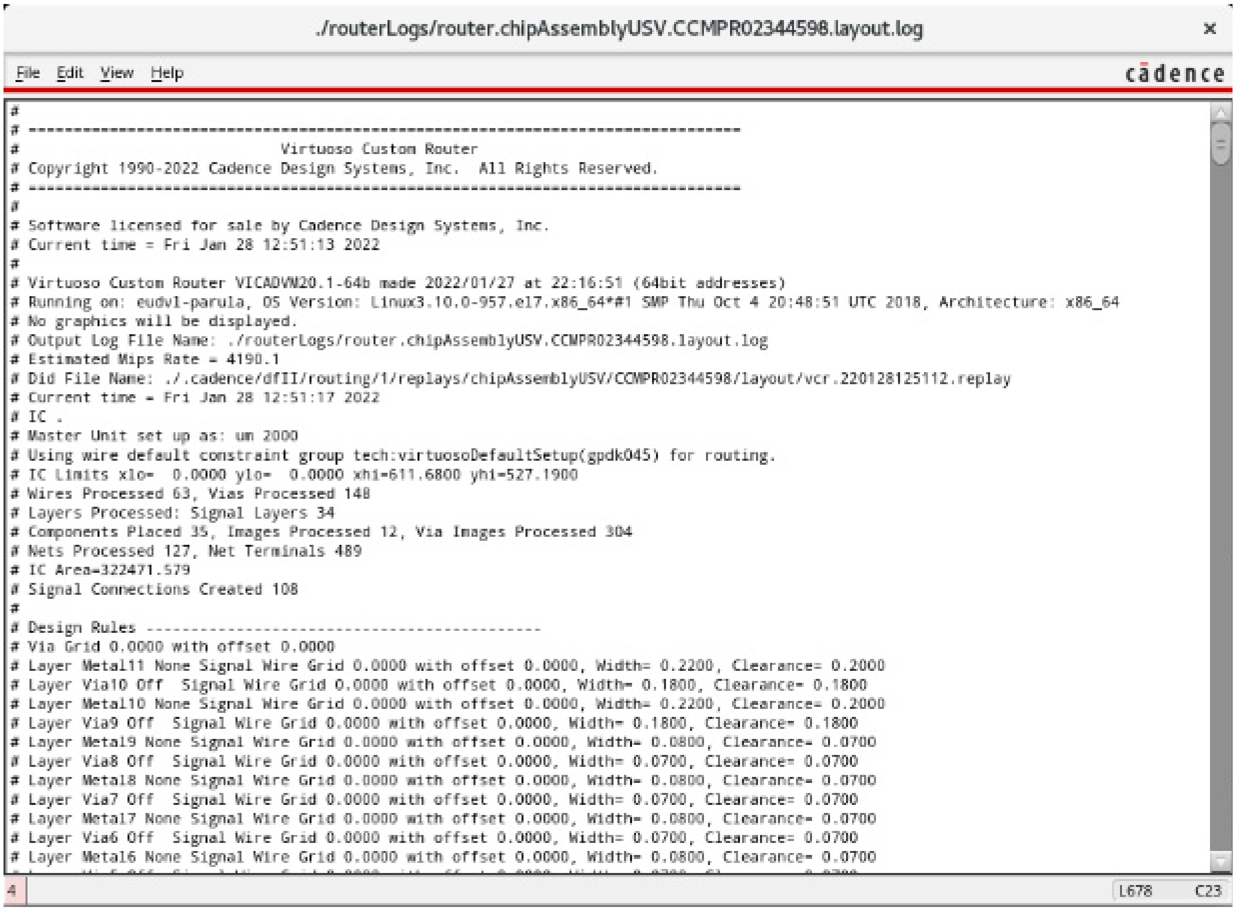
8. Select the routing option to be run. You can select *Global route*, *Detailed route*, *Optimize route*, or *Remove shorts & DRCs*.

9. Specify the required number of passes for selected router.

10. Specify a cellview to save the routing results. You can save the results either in the current view or specify another cellview.

11. Click *Run signal router* ▶ to run signal routing.

    The selected nets are routed. While the router is running, the *Stop* ■ button is displayed. A directory called `routerLogs` is created in the run directory. Clicking the *Stop* button cancels the current run without retrieving any routing information from the memory design.

    
    ./routerLogs/router.chipAssemblyUSV.CCMPR02344598.layout.log
×
File
Edit
View
Help
cādence
4
L678
C23

```
#
# --------------------------------------------------------------------------------
#                             Virtuoso Custom Router
# Copyright 1990-2022 Cadence Design Systems, Inc.  All Rights Reserved.
# --------------------------------------------------------------------------------
#
# Software licensed for sale by Cadence Design Systems, Inc.
# Current time = Fri Jan 28 12:51:13 2022
#
# Virtuoso Custom Router VICADVN20.1-64b made 2022/01/27 at 22:16:51 (64bit addresses)
# Running on: eudvl-parula, OS Version: Linux3.10.0-957.el7.x86_64*#1 SMP Thu Oct 4 20:48:51 UTC 2018, Architecture: x86_64
# No graphics will be displayed.
# Output Log File Name: ./routerLogs/router.chipAssemblyUSV.CCMPR02344598.layout.log
# Estimated Mips Rate = 4190.1
# Did File Name: ./.cadence/dfII/routing/1/replays/chipAssemblyUSV/CCMPR02344598/layout/vcr.220128125112.replay
# Current time = Fri Jan 28 12:51:17 2022
# IC .
# Master Unit set up as: um 2000
# Using wire default constraint group tech:virtuosoDefaultSetup(gpdk045) for routing.
# IC Limits xlo=  0.0000 ylo=  0.0000 xhi=611.6800 yhi=527.1900
# Wires Processed 63, Vias Processed 148
# Layers Processed: Signal Layers 34
# Components Placed 35, Images Processed 12, Via Images Processed 304
# Nets Processed 127, Net Terminals 489
# IC Area=322471.579
# Signal Connections Created 108
#
# Design Rules ---------------------------------------------
# Via Grid 0.0000 with offset 0.0000
# Layer Metal11 None Signal Wire Grid 0.0000 with offset 0.0000, Width= 0.2200, Clearance= 0.2000
# Layer Via10 Off  Signal Wire Grid 0.0000 with offset 0.0000, Width= 0.1800, Clearance= 0.1800
# Layer Metal10 None Signal Wire Grid 0.0000 with offset 0.0000, Width= 0.2200, Clearance= 0.2000
# Layer Via9 Off  Signal Wire Grid 0.0000 with offset 0.0000, Width= 0.1800, Clearance= 0.1800
# Layer Metal9 None Signal Wire Grid 0.0000 with offset 0.0000, Width= 0.0800, Clearance= 0.0700
# Layer Via8 Off  Signal Wire Grid 0.0000 with offset 0.0000, Width= 0.0700, Clearance= 0.0700
# Layer Metal8 None Signal Wire Grid 0.0000 with offset 0.0000, Width= 0.0800, Clearance= 0.0700
# Layer Via7 Off  Signal Wire Grid 0.0000 with offset 0.0000, Width= 0.0700, Clearance= 0.0700
# Layer Metal7 None Signal Wire Grid 0.0000 with offset 0.0000, Width= 0.0800, Clearance= 0.0700
# Layer Via6 Off  Signal Wire Grid 0.0000 with offset 0.0000, Width= 0.0700, Clearance= 0.0700
# Layer Metal6 None Signal Wire Grid 0.0000 with offset 0.0000, Width= 0.0800, Clearance= 0.0700
```

    Check the contents of the log file, especially the different route passes and observe the convergence, the runtime, and the overall stats. An essential section of the log window is *ROUTING HISTORY*, which displays the detailed runtime for each pass and the total runtime. Other statistics are also displayed, such as violations (conflicts), opens, and number of vias.

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:

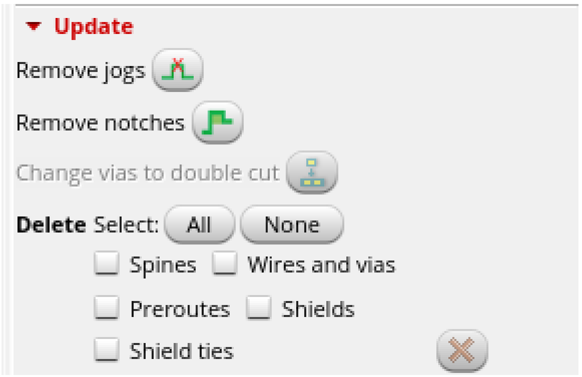
```
envSetVal("ui" "raiseCIWonError" 'boolean t)
```

Improving Routing Results

To improve routing results:

1. In the Routing assistant, click the *Route* tab.
2. Select *Remove Jogs* in the *Update* section of the *Route* tab.

   This option is needed when the router or pre-routed wires might have unnecessary jogs. You can run this command on the entire design or on the selected nets.
3. Select *Remove Notches* to fix the notches between the wires and pins.



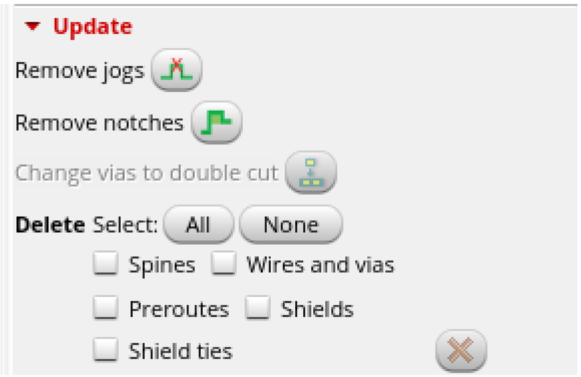
The improved routing results for the selected nets is displayed in the layout window.

Deleting Routed Nets

To delete the routing on nets:

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

2. In the *Update* section of the *Route* tab, click the *All* button to select all types of routing objects. You can also select to delete any of the *Spines*, *Wires and vias*, *Preroutes*, *Shields*, or *Shield ties*.



3. Click *Delete* to delete all the routing from the design.
4. To re-run signal routing, click *Run signal router* .

   The routing results for the selected nets is displayed in the layout window.

Related Topics

Chip Assembly Routing

Chip Assembly Routing Configuration

Routing Assistant User Interface for Chip Assembly Routing Flow

Configuring Chip Assembly Routing Settings

Checking Layout Routability in Chip Assembly

Generating Shields

Viewing and Analyzing Chip Assembly Routing Results

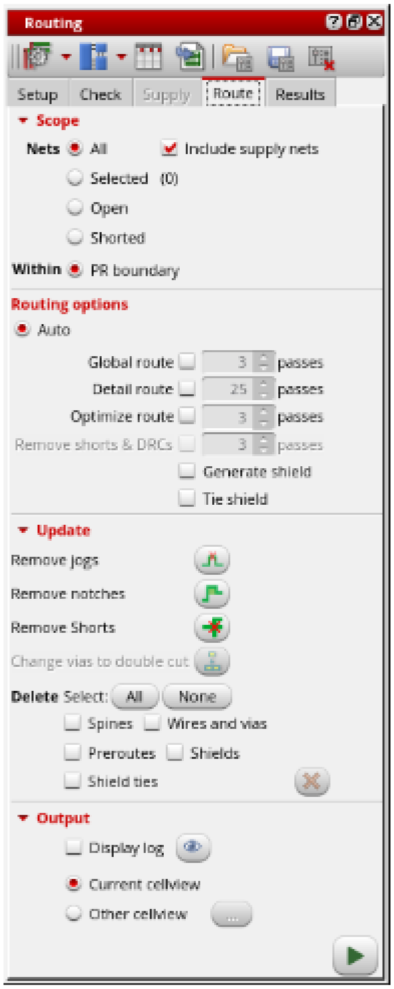
Generating Shields

The shield wires are around signal nets to improve signal integrity. The shields must be generate before routing the signal nets so that the router leaves additional clearance for the shield wires. You can generate shields only when the shield constraint is defined on the selected nets. A shield constraint can be created using Routing Constraint Manager. For more information, see Creating a Shield Constraint.

To generate shields:

1. Open a design in Layout MXL.
2. Select nets from the Navigator assistant.
3. Choose *Window – Assistants – Routing*.

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



Scope
Nets ● All ☑ Include supply nets
○ Selected (0)
○ Open
○ Shorted
Within ● PR boundary

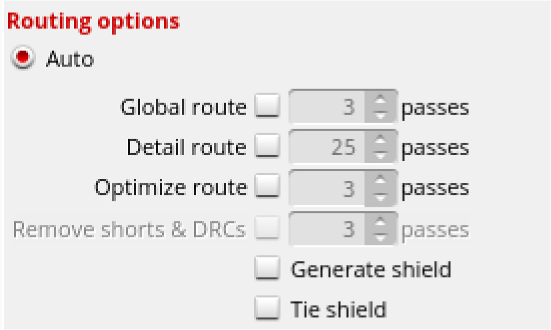
Routing options
● Auto
Global route □ 3 passes
Detail route □ 25 passes
Optimize route □ 3 passes
Remove shorts & DRCs □ 3 passes
□ Generate shield
□ Tie shield

□ Spines □ Wires and vias
□ Preroutes □ Shields
□ Shield ties

Output
□ Display log
● Current cellview
○ Other cellview

5. Select *Generate shield* in the *Routing options* section. If the option is unselected, the shielding wires can still be routed as a post-route process. In such case, the router takes care of the extra spacing required by the shielding while routing the regular nets.

6. Select *Tie shield*.



Routing options

● Auto

Global route □ 3 passes

Detail route □ 25 passes

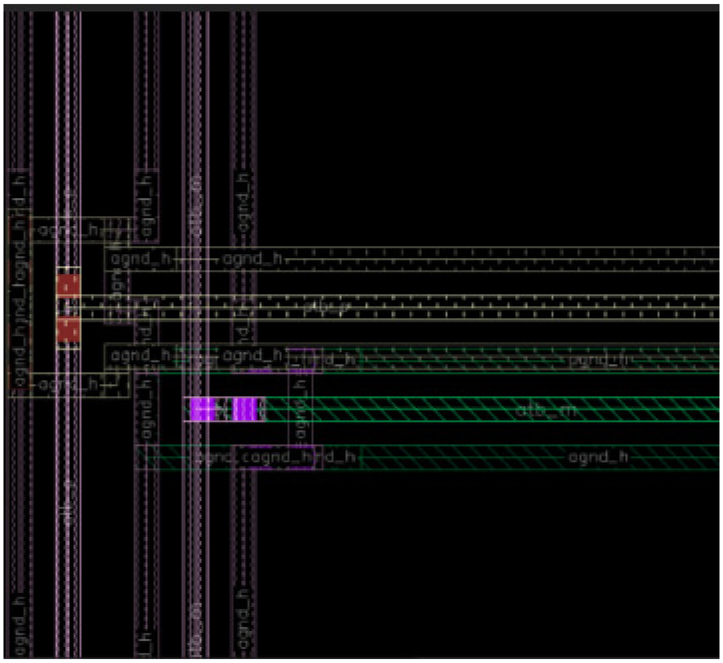
Optimize route □ 3 passes

Remove shorts & DRCs □ 3 passes

□ Generate shield

□ Tie shield

7. Click *Run signal router* ▶ to run routing. You can check the log file to view statistical information.



Related Topics

Chip Assembly Routing

Chip Assembly Routing Configuration

Routing Assistant User Interface for Chip Assembly Routing Flow

Configuring Chip Assembly Routing Settings

Checking Layout Routability in Chip Assembly

Routing in Automatic Mode

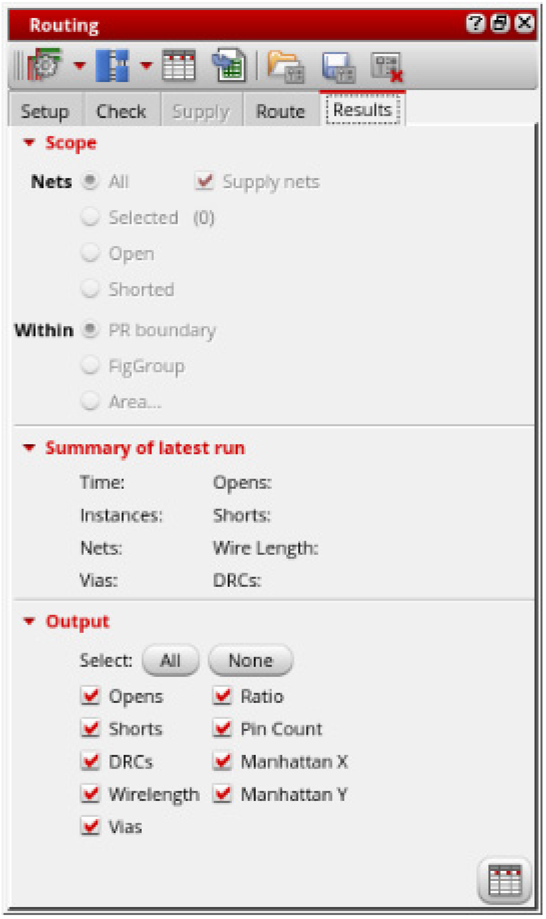
Viewing and Analyzing Chip Assembly Routing Results

Viewing and Analyzing Chip Assembly Routing Results

To view and analyze the results:

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. Click the *Results* tab in the Routing assistant.



Nets ● All ☑ Supply nets
○ Selected (0)
○ Open
○ Shorted

Within ● PR boundary
○ FigGroup
○ Area...

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

Select: All None

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

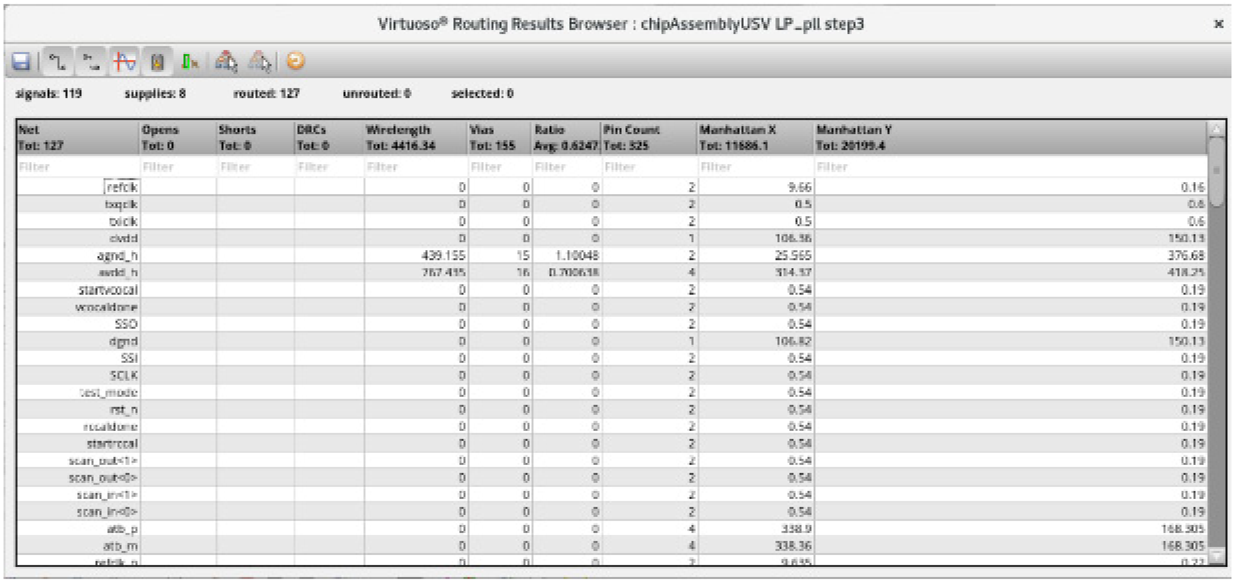
4. Select the scope of nets to analyze the routing results. You can either select *All*, *Selected*, *Open*, or *Closed* nets.

5. Select *Supply Nets* to see the results of the power and ground nets.

6. Select the area for which you want to view the routing results. You can select either PR boundary, a figGroup, or select an area in the design.

7. Select *All* in the *Output* section. You can also select one or more outputs by clicking the check box next to the output name. The selected output name column is displayed in the Routing Results Browser.

8. Click on the *Results Browser* button at the bottom right corner. The Routing Results Browser displays. You can see the total number of routed nets, opens, shorts and the 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
txiclk				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
rxcaldone				0	0	0	2	0.54	0.19
startrxcal				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

Chip Assembly Routing

Chip Assembly Routing Configuration

Routing Assistant User Interface for Chip Assembly Routing Flow

Configuring Chip Assembly Routing Settings

Checking Layout Routability in Chip Assembly

Routing in Automatic Mode

Generating Shields

Viewing and Analyzing Chip Assembly Routing Results

4

Virtuoso Routing Constraint Manager

The Virtuoso Routing Constraint Manager (RCM) provides a simplified and consolidated interface for managing routing constraints.

The following are some of the key features of the Routing Constraint Manager.

- Supports a prioritized set of constraints for routing
- Provides a simplified spreadsheet view of all routing constraints in the design
    - Lets you customize columns for constraint types
    - Provides ability to view by net or by constraint type
    - Provides ability to search, filter, and apply defaults
    - Lets you add and modify constraints
- Specifies additional parameters for specific constraints
    - Exposes parameter fields for a given constraint type to edit them in place
    - Lets you save and reuse settings for a constraint to apply to other nets or design
- Abstracts database and implementation details from user (for example, constraint group specifics)
- Stores constraints directly in OpenAccess as part of the design layout view
- Lets you start a constraint checker to visualize the constraint implementation status

Using RCM, you can filter the nets based on the constraints definition status, signal nets, power nets, and so on. You can create design-level constraint groups and apply them to specific design objects. You can create shielding constraint groups and apply them to specific nets in the design.

Related Topics

Opening Routing Constraint Manager

Routing Constraint Manager User Interface

Routing Assistant

Accessing the Routing Assistant

Opening Routing Constraint Manager

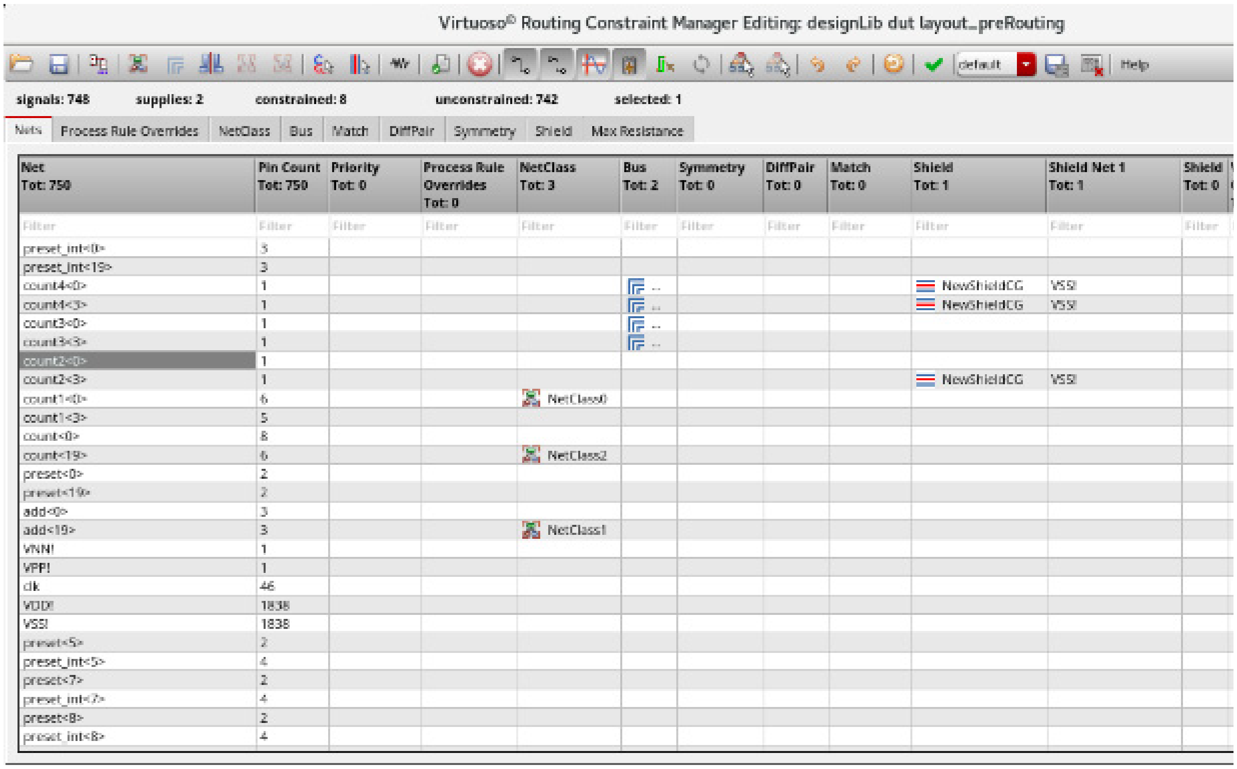
To open the Routing Constraint Manager:

1. Choose *Window – Assistants – Routing Assistant*.

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

2. Click *Set router constraints* on the Routing assistant toolbar.

   The Virtuoso Routing Constraint Manager is displayed.



signals: 748 supplies: 2 constrained: 8 unconstrained: 742 selected: 1

Net Tot: 750	Pin Count Tot: 750	Priority Tot: 0	Process Rule Overrides Tot: 0	NetClass Tot: 3	Bus Tot: 2	Symmetry Tot: 0	DiffPair Tot: 0	Match Tot: 0	Shield Tot: 1	Shield Net 1 Tot: 1	Shield Tot: 0
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
preset_int<0>	3										
preset_int<19>	3										
count4<0>	1				...				☰ NewShieldCG	VSS!	
count4<3>	1				...				☰ NewShieldCG	VSS!	
count3<0>	1				...						
count3<3>	1				...						
count2<0>	1										
count2<3>	1								☰ NewShieldCG	VSS!	
count1<0>	6			NetClass0							
count1<3>	5										
count<0>	8										
count<19>	6			NetClass2							
preset<0>	2										
preset<19>	2										
add<0>	3										
add<19>	3			NetClass1							
VNN!	1										
VPP!	1										
clk	46										
VDD!	1838										
VSS!	1838										
preset<5>	2										
preset_int<5>	4										
preset<7>	2										
preset_int<7>	4										
preset<8>	2										
preset_int<8>	4										

Related Topics

Virtuoso Routing Constraint Manager

Routing Constraint Manager User Interface

Use the Routing Constraint Manager to create, modify, and delete routing constraints. The Routing Constraint Manager has the following components:

Routing Constraint Manager Toolbar	Lets you create, save, and delete constraints. It also provides access to a number of constraint types (Group, Bus, Symmetry, DiffPair, Match) and allows you to specify your constraint configuration.
Routing Constraint Manager Summary Information	Displays the summary information of the nets in the Routing Constraint Manager.
Routing Constraint Manager Tabs	Lets you edit the routing constraints. In Routing Constraint Manager, there are eight tabs. These tabs include a number of options to set specific constraints. The supported routing constraints in Virtuoso are: Shielding, Diff Pair, Bus, Matched Length, Symmetry, Process Rule Overrides, Net Class.

Routing Constraint Manager Toolbar

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

Icon	Command	Description
📂	*Load Constraints*	Loads constraint options from an existing file.
💾	*Save Table Contents*	Saves the contents of the table for a given constraint in the csv or xml format.
	Edit Design Process Rule Overrides	Lets you edit the user-defined design default constraint group.
	NetClass	Creates a group (NetClass) constraint for all the selected nets.

Icon	Command	Description
	Bus	Creates a bus constraint for the selected nets.
	Symmetry	Creates a symmetry constraint for the selected nets.
	DiffPair	Creates a differential pair constraint for the selected nets.
	Match	Creates a matched length constraint for the selected nets.
	Apply Constraint Group	Applies a constraint group to the selected nets or groups.
	Apply Shield Constraint Group	Applies a parallel simple shielding constraint group to the selected nets or groups.
	Apply Max Resistance	Creates a maximum resistance constraint on a net.
	Routing Results	Displays the Routing Results Browser window.
	Remove	Deletes the constraint applied on the selected nets or groups.
	Show Constrained Nets	Displays only the constrained nets.
	Show Unconstrained Nets ON	Displays only the unconstrained nets.
	Show Signal Nets	Displays only the signal nets.
	Show Supply Nets	Displays only the power and ground nets.
	Snapshot Selected	Displays only the nets that are selected in the Navigator assistant.

Icon	Command	Description
⟳	*Synchronize Snapshot*	Synchronizes the net selection in Routing Constraint Manager with the ones that are currently selected in the Navigator assistant or layout canvas This sync option works in conjunction with the *Snapshot Selected* option. Once you have created a snapshot of the selected nets, the *Synchronize Snapshot* button updates the snapshot to match the nets that are selected in the Navigator assistant or layout canvas.
	Select All	Selects all the nets in the table.
	Deselect All	Deselects all the nets in the table.
↶	*Undo*	Reverses the action of the previous constraint editing operation. The bindkey associated with this is `Ctrl + X`.
↷	*Redo*	Reapplies a constraint editing operation that was reversed by an undo command. The bindkey associated with this is `Ctrl + Shit + Z`.
⟲	*Refresh Table*	Refreshes the constraints table.
✓	*Check Constraints*	Checks whether the specified constraints have been implemented correctly in the layout.
default ▾	*Column Visibility Presets*	Loads existing presets. This lets you decide which columns are relevant to be displayed
	Save Column Visibility Preset	Lets you create and save a column visibility preset.
	Delete Column Visibility Preset	Lets you delete a column visibility preset.

Routing Constraint Manager Summary Information

The following table lists the net summary information in Routing Constraint Manager:

Label	Description
Signals	Shows the number of signal nets.
Supplies	Shows the number of power and ground nets.
Constrained	Shows the number of constrained nets.
Unconstrained	Shows the number of unconstrained nets.
Selected	Shows the number of selected nets.

Routing Constraint Manager Tabs

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

Tab	Description
Nets	Provides an overview of the constraints applied on each net, including groupings of nets.
Process Rule Overrides	Displays all the constraint groups in the design. Each group can be expanded to see which nets the constraint groups are applied to.
NetClass	Displays different types of net groups.
Bus	Displays the bus groups in the design and their net constraints.
Match	Displays the match length groups in the design and their net constraints.
DiffPair	Displays the diffpair groups in the design and their net constraints.
Symmetry	Displays the symmetry groups in the design and their net constraints.
Shield	Displays all shield constraint groups. Each group expands to display which nets the shield applies to.

Max Resistance	Defines the maximum resistance permissible between two terminals or instance terminals or, between a terminal or instance terminal and all the terminals or instance terminals connected to it.

Nets

The following table describes the columns available in the table in the *Nets* tab of the Routing Constraint Manager.

Column	Description
Net	Name of the net.
Pin Count	Number of pins for each net.
Priority	Priority of the constraint applied to a net.
Net Process Rule Overrides	Rules that are applied to a net.
Group Process Rule Overrides	Rules that are currently applied to a net group. The two columns *Net Process Rule Overrides* and *Group Process Rule Overrides* behave identically to the existing *Process Rule Overrides column. Previously, constraint groups applied to groups could only be seen by opening a group editor.*
NetClass	Group constraint applied to a net.
Bus	Bus constraint applied to a net.
Symmetry	Symmetry constraint applied to a net.
DiffPair	Diffpair constraint applied to a net.
Match	Match constraint applied to a net.
Shield	Shield constraint applied to a net or a group of nets.
Shield Net 1	Name of the first net that is to be considered as the shield net. This shield net behaves identically to the original *Shield* column.
Shield Net 2	Name of the second net that is to be considered as the shield net. This shield net behaves identically to the original *Shield* column.
Via Config	Via configuration constraint group applied to a net or a group. The full functionality will be available in a future release.

Column	Description
Max Resistance	Maximum resistance constraint applied to a net.

Process Rule Overrides

The following table describes the columns available in the table in the *Process Rule Overrides* tab of the Routing Constraint Manager.

Column	Description
Process Rule Overrides	The constraint group applied to a net or a group of nets.
Via Config	The via configuration constraint group applied to a net or a group of nets. The full functionality will be available in a future release.
Dir	The preferred routing direction for the layer.
Vias	A valid vias constraint.
Cuts	The minimum number of cuts constraint (for each cut layer matching a valid via).
Bottom Layer	The name of the valid bottom routing layer.
Top Layer	The name of the valid top routing layer.
<Metal Layer> W Min	The minimum width of the metal layer.
<Metal Layer> W Max	The maximum width of the metal layer.
<Metal Layer > S	The minimum spacing of the metal layer.

NetClass

The following table describes the columns available in the table in the *NetClass* tab of the Routing Constraint Manager.

Column	Description
NetClass	The name of the constraint group applied to the nets.
Process Rule Overrides	The constraint group applied to a net or a group of nets.

Column	Description
Same Mask	Displays the mask name for the NetClass constraint.
Shield	The name of the shield constraint group applied to the group of nets (if one has been applied).
Via Config	The via configuration constraint group applied to a net or a group of nets. The full functionality will be available in a future release.
Dir	The preferred routing direction of the layer.
Vias	A valid vias constraint.
Cuts	The minimum number of cuts constraint (for each cut layer matching a valid via).
Bottom Layer	The name of the valid bottom routing layer.
Top Layer	The name of the valid top routing layer.
<Metal Layer> W Min	The minimum width of the metal layer.
<Metal Layer> W Max	The maximum width of the metal layer.
<Metal Layer S >	The minimum spacing of the metal layer.

Bus

The following table describes the columns available in the table in the *Bus* tab of the Routing Constraint Manager.

Column	Description
Bus	The name of the bus constraint applied to a net or a group of nets.
Process Rule Overrides	The constraint group applied to a net or a group of nets.
Shield	The name of the shield constraint group applied to the group of nets (if one has been applied).
Via Config	The via configuration constraint group applied to a net or a group. The full functionality will be available in a future release.
Dir	The preferred routing direction of the layer.
Vias	A valid vias constraint.

Column	Description
Cuts	The minimum number of cuts constraint (for each cut layer matching a valid via).
Bottom Layer	The name of the valid bottom routing layer.
Top Layer	The name of the valid top routing layer.
<Metal Layer> W Min	The minimum width of the metal layer.
<Metal Layer> W Max	The maximum width of the metal layer.
<Metal Layer S >	The minimum spacing of the metal layer.

Match

The following table describes the columns available in the table in the *Match* tab of the Routing Constraint Manager.

Column	Description
Match	The name of the matched length constraint applied to a net or a group of nets.
Process Rule Overrides	The constraint group applied to a net or a group of nets.
Pattern	The pattern used to lengthen wires for matched length routing. The different patterns are: *None*, *Accordion*, *RW Accordion*, *Trombone*, and *End Run*.
Match Per Layer	Length match checks performed per layer or over the entire length of the net.
Tolerance%	Absolute tolerance when matching net lengths is expressed as a percentage of the total length.
Shield	The name of the shield constraint group applied to the group of nets (if one has been applied).
Via Config	The via configuration constraint group applied to a net or a group of nets. The full functionality will be available in a future release.
Dir	The preferred routing direction of the layer.
Vias	A valid vias constraint.

Column	Description
Cuts	The minimum number of cuts constraint (for each cut layer matching a valid via).
Bottom Layer	The name of the valid bottom routing layer.
Top Layer	The name of the valid top routing layer.
<Metal Layer> W Min	The minimum width of the metal layer.
<Metal Layer> W Max	The maximum width of the metal layer.
<Metal Layer S >	The minimum spacing of the metal layer.

DiffPair

The following table describes the columns available in the table in the *DiffPair* tab of the Routing Constraint Manager.

Column	Description
DiffPair	The name of the differential pair constraint applied to a net or a group of nets.
Process Rule Overrides	The constraint group applied to a net or a group of nets.
Tolerance	Specifies the absolute tolerance when matching differential pair for violation detection.
Same Mask	Displays the mask name for the DiffPair constraint.
Shield	The name of the shield constraint group applied to the group of nets (if one has been applied).
Via Config	The via configuration constraint group applied to a net or a group of nets. The full functionality will be available in a future release.
Dir	The preferred routing direction for the layer.
Vias	A valid vias constraint.
Cuts	The minimum number of cuts constraint (for each cut layer matching a valid via).
Bottom Layer	The name of the valid bottom routing layer.
Top Layer	The name of the valid top routing layer.

Column	Description
<Metal Layer> W Min	The minimum width of the metal layer.
<Metal Layer> W Max	The maximum width of the metal layer.
<Metal Layer S >	The minimum spacing of the metal layer.

Symmetry

The following table describes the columns available in the table in the *Symmetry* tab of the Routing Constraint Manager.

Column	Description
Symmetry	The name of the symmetry constraint applied to a net or a group of nets.
Process Rule Overrides	The constraint group applied to a net or a group of nets.
Tolerance	Specifies the absolute tolerance when matching symmetries for violation detection.
Axis	The axis to be used for symmetry. The axis symmetry can be vertical and horizontal.
Same Mask	Displays the mask name for the symmetry constraint.
Shield	The name of the shield constraint group applied to the group of nets (if one has been applied).
Via Config	The via configuration constraint group applied to a net or a group of nets. The full functionality will be available in a future release.
Dir	The preferred routing direction for the layer.
Vias	A valid vias constraint.
Cuts	The minimum number of cuts constraint (for each cut layer matching a valid via).
Bottom Layer	The name of the valid routing bottom layer.
Top Layer	The name of the valid routing top layer.
<Metal Layer> W Min	The minimum width of the metal layer.
<Metal Layer> W Max	The maximum width of the metal layer.

Column	Description
<Metal Layer S >	The minimum spacing of the metal layer.

Shield

The following table describes the columns available in the table in the *Shield* tab of the Routing Constraint Manager.

Column	Description
Shield	The name of the shielding constraint group applied to a net or a group of nets.
Shield Type	Determines the type of shielding to route around a net. The available options are: ■ *None*: No shield. ■ *Parallel*: Planar shield wires parallel to the signal wire. ■ *Tandem*: Shield wires on a given layer above and below the signal wire. ■ *Coaxial*: Both tandem and parallel shields to surround the signal wire on all four sides The default value is *Parallel*.
Group Shield Type	Displays the type of group shielding.
Enclosure Style	Adds shield wires all around the vias in the preferred direction.
Tolerance%	Absolute tolerance when matching shield lengths is expressed as a percentage of total coverage.
Valid Shield Layers	List of layers that are valid for the shielding routing.
Share Shields	Determines whether shield routes can be shared by nets.
Use Existing Shapes	Determines whether existing shapes can be used as shielding.
Shield Terminal	Determines whether shield terminal has to be connected or not.
Minimum Length Enabled	Prevents shielding of any signal wire that is shorter than the given length. If selected, it enables the *Minimum Length* column.
Minimum Length	Specifies a minimum value for which shielding wires are added.

Column	Description
Tie Shield	Determines whether ties should be added to tie the new shield wires to the shield nets that they belong to.
Tie Frequency Enabled	Displays whether the tie frequency is enabled.
Tie Frequency	Specifies the maximum distance between ties that must be inserted to tie the new shield wires to their respective shield nets.
Shield Redundant Via	Adds redundant vias to tie shield wires to shield nets at every location where the shield wires overlap their respective existing power or ground rails.
Via Config	The via configuration constraint group applied to a net or a group. The full functionality will be available in a future release.
Dir	The preferred routing direction of the layer.
Vias	Corresponds to a valid vias constraint.
Cuts	The minimum number of cuts constraints (for each cut layer matching a valid via).
Bottom Layer	The name of the valid routing bottom layer.
Top Layer	The name of the valid routing top layer.
<Metal Layer> W Min	The minimum width of the metal layer.
<Metal Layer> S	The minimum spacing of the metal layer.

Max Resistance

The following table describes the columns available in the table in the *Max Resistance* tab of the Routing Constraint Manager.

Column	Description
Net	The net name on which the maximum resistance constraint is defined.
To Inst Term	The from-to terminals of the net on which the maximum resistance constraint should apply.
Max Resistance	The value of the maximum resistance.

Routing Constraint Manager Context-Sensitive Menus

The context-sensitive menus in the Routing Constraint Manager are displayed when you right-click a net or a constraint that appears on a net. These menus have options that are dependent on the Routing Constraint Manager tab, column, and the item selected. The options on the context-sensitive menus can include creating a new constraint, adding to an existing constraint, removing from a constraint, or deleting an existing constraint depending on the item selected. The create new or add options appear in the net, group, rules, and shield columns. The delete and remove options appear if an existing group or a constraint group is selected.

Related Topics

Opening Routing Constraint Manager

Virtuoso Routing Constraint Manager

Manage Constraints

Design Process Rule Override Editor Form

Process Rule Override Editor Form

Max Resistance Editor Form

Process Rule Override Editor: Bus Form

Process Rule Override Editor: DiffPair Form

Process Rule Override Editor: Match Form

Process Rule Override Editor: NetClass Form

Process Rule Override Editor: Symmetry Form

Shield Process Rule Override Editor Form

Constraint Checker Form

Manage Constraints

A simple way to define a constraint is to consider it as an objective that can be used to influence the possible implementation, placement, or routing in a system. Constraints are therefore requirements that need to be followed during design implementation.

By understanding constraints in your design hierarchy, you can determine specification limits for design objects.

A constraint can have parameters that are pre-defined name-value pairs that further define the design intent and the status of that constraint.

You can manage the routing constraints using Virtuoso Routing Constraint Manager.

Related Topics

Virtuoso Routing Constraint Manager

Opening Routing Constraint Manager

Routing Constraint Manager User Interface

Viewing and Hiding Columns in Routing Constraint Manager

You can customize the columns that should be displayed in each tab of the Routing Constraint Manager. To do this:

1. Right-click any column name in any of the tabs.

   The column header context menu appears.

2. Place the mouse pointer on the *Columns* header. The Column sub-menu appears.

3. Choose the column that you want to hide or display.

4. Click *Save Column Visibility Preset* to save the customized view in a preset file.

   You can see that the saved preset file appears in the list of saved presets in the *Column Visibility Presets* list.

Related Topics

Saving and Deleting Column Visibility Preset File

Saving and Deleting Column Visibility Preset File

A preset is a set of predefined settings, which can be saved to a file. The presets can be used to reuse the column visibility settings that you had specified in the Routing Constraint Manager.

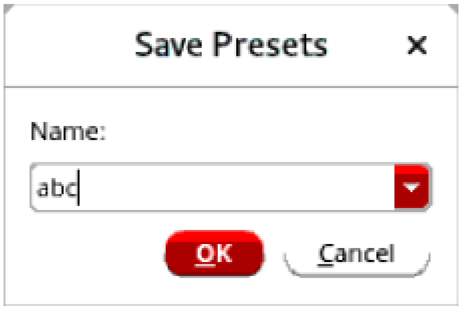
Saving Column Visibility Preset

The *Save Column Visibility Preset* button in the Routing Constraint Manager lets you save the columns settings, which should be displayed in each tab of the Routing Constraint Manager, to a file. When saving column presets, you can modify columns of all the tabs and save them to a single preset file or create different preset files for each.

To save the column visibility preset:

1. Modify the column visibility in each tab as required by your design objectives.
2. Click *Save Table Content* 📂 on the *Routing Constraint Manager* toolbar.

   The Save Presets form appears.



Save Presets

Name: abc

3. Select the file to which you want to save the existing column visibility settings or specify a new file name.
4. Click *OK*.

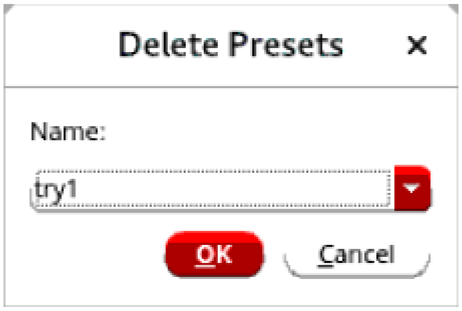
   The preset file is saved and appears in the *Column Visibility Presets* list on the Routing Constraint Manager toolbar.

Deleting a Column Visibility Preset

The *Delete Column Visibility Preset* button lets you delete an existing column visibility preset. To delete an existing preset:

1. Click *Delete Column Visibility Preset* on the Routing Constraint Manager toolbar.

   The Delete Presets form appears.



Delete Presets ×

Name: try1

2. Select the preset that you want to delete.
3. Click *OK*.

   The selected preset file is deleted and is no longer displayed in the *Column Visibility Presets* list on the Routing Constraint Manager toolbar

Related Topics

Viewing and Hiding Columns in Routing Constraint Manager

Loading Routing Constraints

Routing Constraint Manager lets you load constraints from the specified `library:cell:view`. To load constraints:

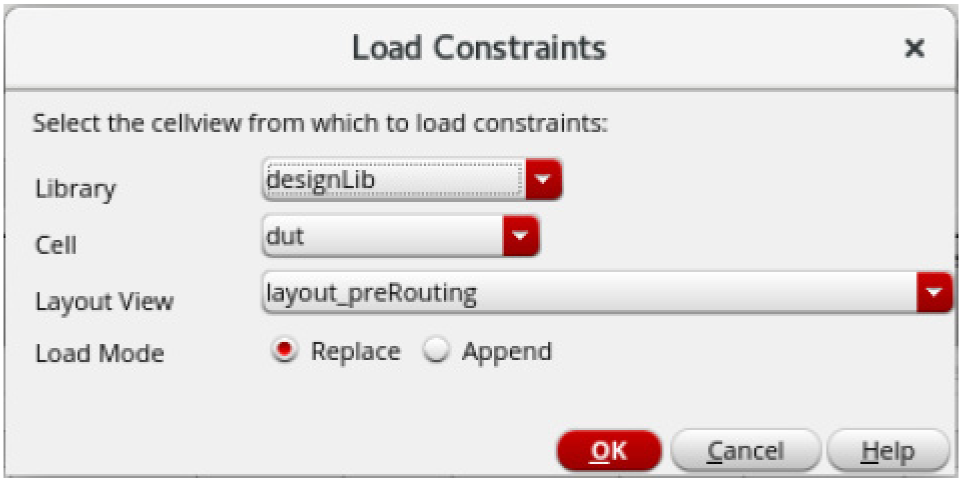
1. Open a design in Layout MXL.
2. Choose *Window – Assistants – Routing*.

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

   The Routing assistant is displayed.
3. Click *Routing Constraint Manager* on the Routing assistant toolbar.

4. Click *Load Constraints* on the *Routing Constraint Manager* toolbar.

   The Load Constraints form is displayed.



Load Constraints

Select the cellview from which to load constraints:

Library: designLib

Cell: dut

Layout View: layout_preRouting

Load Mode: ● Replace ○ Append

5. Choose a `library:cell:view` from the *Library*, *Cell*, and *Layout View* fields, respectively.
6. Select the import mode. You can either select *Replace* or *Append*.
7. Click *OK*.

The constraints from the selected `library_cell_view` are loaded to the Routing Constraint Manager.

Related Topics

Opening Routing Constraint Manager

Routing Constraint Manager User Interface

Load Constraints Form

Creating and Deleting a Constraint Group

A constraint group is a collection of constraints and process rules created for a technology database or a specific design. Constraint groups can be created, edited, and deleted using the Routing Constraint Manager and can be applied to nets, specific constraints (bus and netclass for instance), and designs.

Constraint groups exist in particular technology databases and can be made up of multiple sub-constraint groups.

For a constraint group (and the process rules contained in them) to be applied to an object, design, or technology database, it must be a member of the default constraint group for that design or technology.

Creating a Constraint Group

To create a constraint group:

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

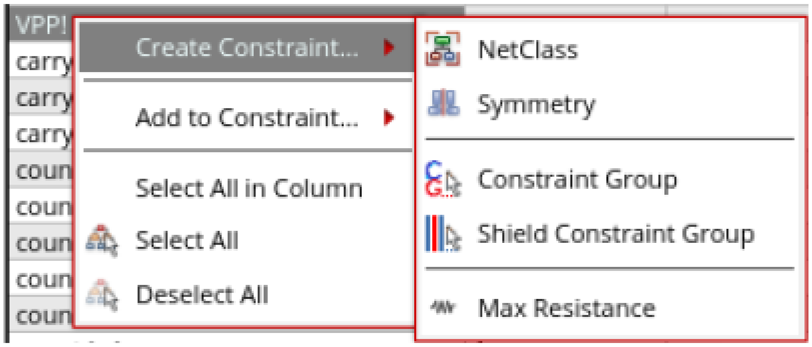
   Alternatively, right-click anywhere on the layout window menu bar and choose *Assistants – Routing*.

   The Routing assistant is displayed.
3. Click *Routing Constraint Manager* on the Routing assistant toolbar.
4. Select few nets in the *Net* column of the *Nets* tab in the Routing Constraint Manager.
5. Click the Apply Constraint Group button on the Routing Constraint Manager toolbar.
6. Alternatively, right-click a net in the *Nets* column of the constraints table.

   A menu is displayed.

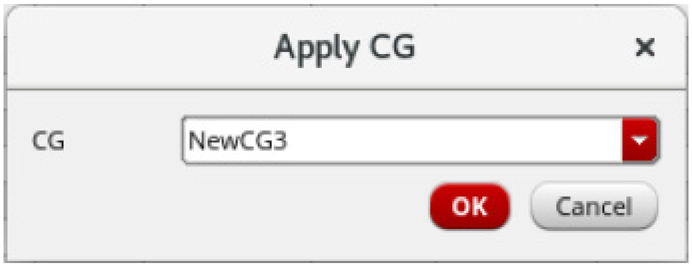
   a. Hover over to *Create Constraint*.

      A submenu is displayed.



   b. Choose *Constraint Group*.

The *Apply CG* form appears.



CG: NewCG3

7. Specify the name of the constraint group that you want to create in the *CG* field.
8. Click *OK*.

   A new constraint group is added to the selected net and displayed in the *Process Rule Overrides* column of the constraints table.

Deleting a Constraint Group

To delete a constraint group:

1. Right-click the constraint group name that you want to delete from the *Process Rule Overrides* column.
2. Click *Delete Constraint Group*.

   The constraint group is deleted.

Related Topics

Loading Routing Constraints

Adding and Removing a Net from a Constraint Group

Editing a Constraint

Selecting Nets in Navigator

Creating a Shield Constraint

Setting up a Constraint on a Net

Adding and Removing a Net from a Constraint Group

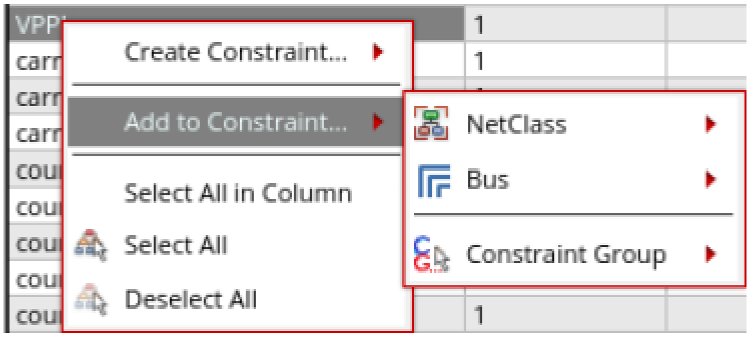
You can use this section to add a net to a constraint group and remove a net from a constraint group.

Adding a Net to a Constraint Group

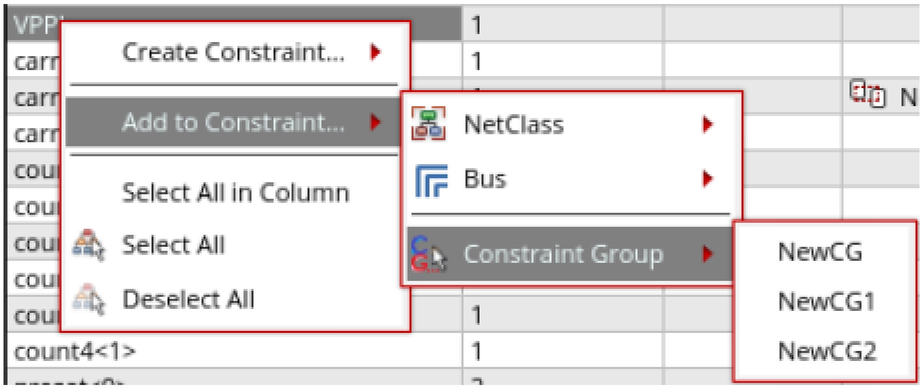
To add a net to a constraint group:

1. Right-click a net in the *Net* column. A shortcut menu is displayed.
2. Hover over to *Add to Constraint*.

   A sub menu is displayed.

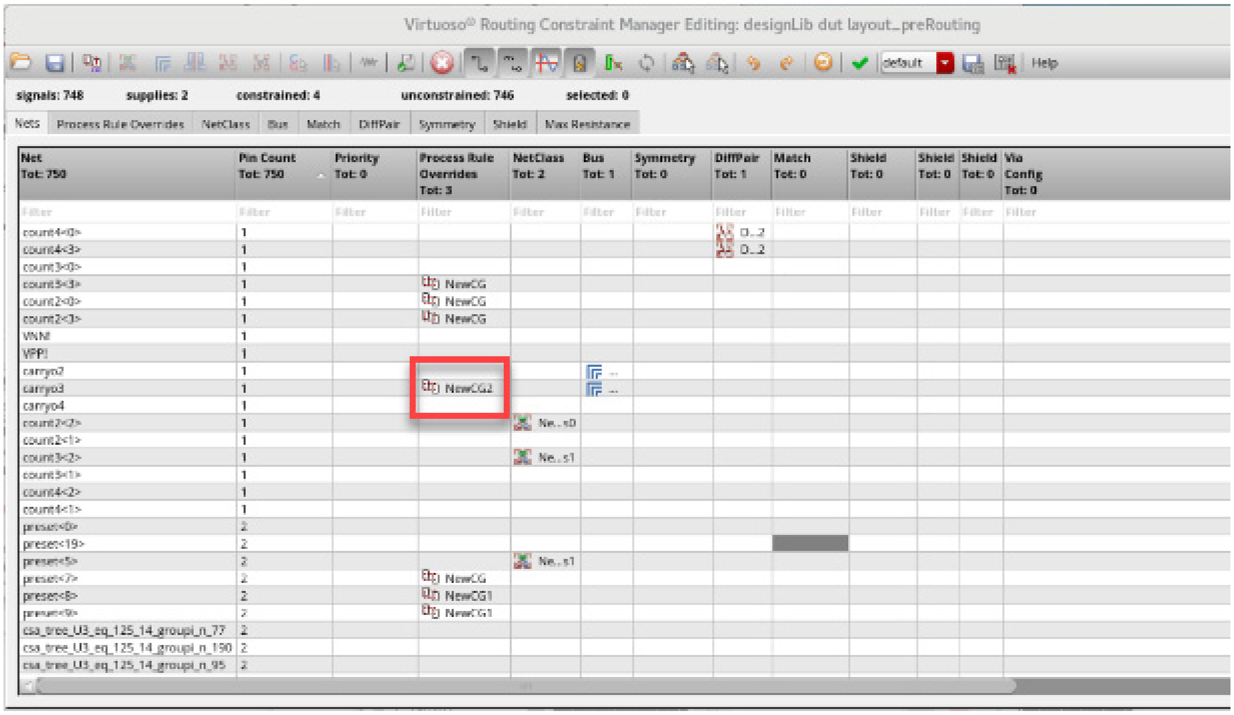


3. Hover over to *Constraint Group*.



4. Click the name of the constraint group to which you want to add the net.

The name of the constraint group appears in the Process Rule Overrides column for the selected net.



Virtuoso® Routing Constraint Manager Editing: designLib dut layout_preRouting

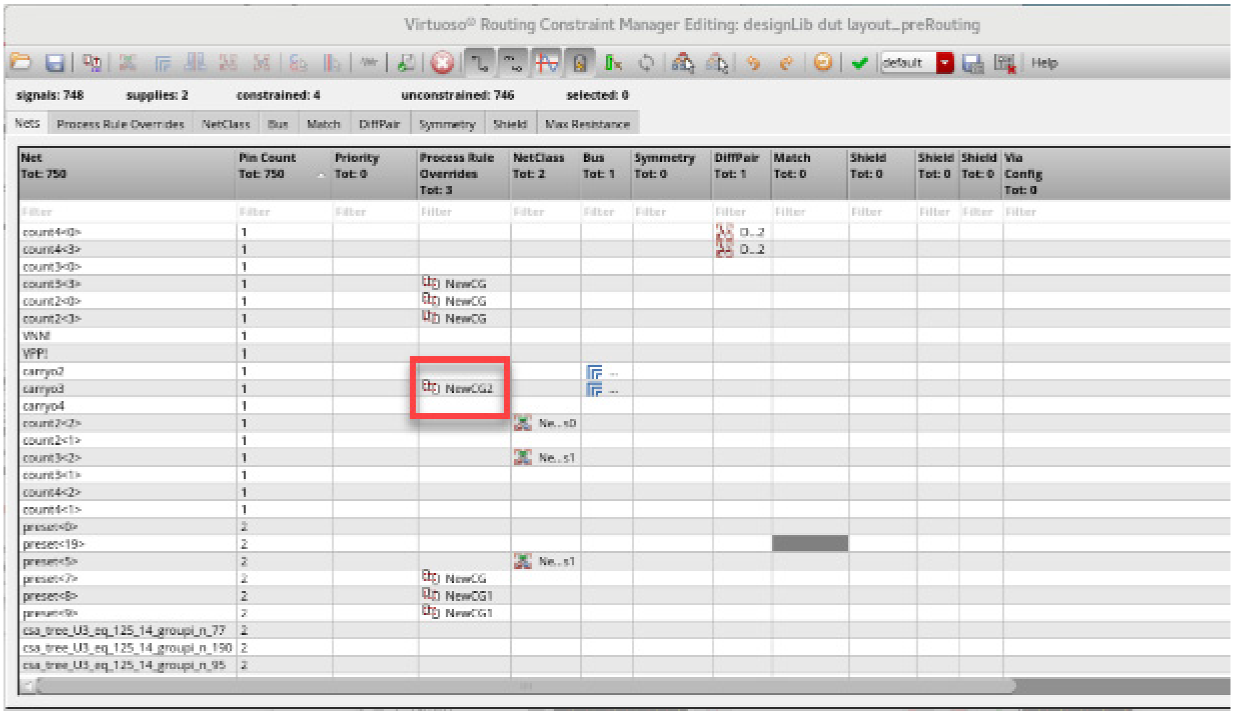
signals: 748 supplies: 2 constrained: 4 unconstrained: 746 selected: 0

Net Tot: 750	Pin Count Tot: 750	Priority Tot: 0	Process Rule Overrides Tot: 3	NetClass Tot: 2	Bus Tot: 1	Symmetry Tot: 0	DiffPair Tot: 1	Match Tot: 0	Shield Tot: 0	Shield Tot: 0	Shield Tot: 0	Via Config Tot: 0
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
count4<0>	1						D...2					
count4<3>	1						D...2					
count3<0>	1											
count5<3>	1		NewCG									
count2<0>	1		NewCG									
count2<3>	1		NewCG									
VNNI	1											
VPPI	1											
carryo2	1				...							
carryo3	1		NewCG2		...							
carryo4	1											
count7<2>	1			Ne...s0								
count2<1>	1											
count3<2>	1			Ne...s1								
count5<1>	1											
count4<2>	1											
count4<1>	1											
preset<0>	2											
preset<19>	2											
preset<5>	2			Ne...s1								
preset<7>	2		NewCG									
preset<8>	2		NewCG1									
preset<9>	2		NewCG1									
csa_tree_U3_eq_125_14_groupi_n_77	2											
csa_tree_U3_eq_125_14_groupi_n_190	2											
csa_tree_U3_eq_125_14_groupi_n_95	2											

Removing a Net from a Constraint Group

To remove a net from a constraint group:

1. Right-click the constraint group name in the *Process Rule Overrides* column next to the net name that you want to remove from the group.



Virtuoso® Routing Constraint Manager Editing: designLib dut layout_preRouting
default
Help

signals: 748 supplies: 2 constrained: 4 unconstrained: 746 selected: 0

Net Tot: 750	Pin Count Tot: 750	Priority Tot: 0	Process Rule Overrides Tot: 3	NetClass Tot: 2	Bus Tot: 1	Symmetry Tot: 0	DiffPair Tot: 1	Match Tot: 0	Shield Tot: 0	Shield Tot: 0	Shield Tot: 0	Via Config Tot: 0
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
count4<0>	1						D..2					
count4<3>	1						D..2					
count3<0>	1											
count5<3>	1		NewCG									
count2<0>	1		NewCG									
count2<3>	1		NewCG									
VNNI	1											
VPPI	1											
carryo2	1				...							
carryo3	1		NewCG2		...							
carryo4	1											
count7<2>	1			Ne...s0								
count2<1>	1											
count3<2>	1			Ne...s1								
count5<1>	1											
count4<2>	1											
count4<1>	1											
preset<0>	2											
preset<19>	2											
preset<5>	2			Ne...s1								
preset<7>	2		NewCG									
preset<8>	2		NewCG1									
preset<9>	2		NewCG1									
csa_tree_U3_eq_125_14_groupi_n_77	2											
csa_tree_U3_eq_125_14_groupi_n_190	2											
csa_tree_U3_eq_125_14_groupi_n_95	2											

2. Click *Remove from Constraint Group*.

   The net name is removed from the constraint group and the constraint group name disappears from the *Process Rule Overrides* column.

Related Topics

Loading Routing Constraints

Creating and Deleting a Constraint Group

Editing a Constraint

Selecting Nets in Navigator

Creating a Shield Constraint

Setting up a Constraint on a Net

Setting up a Constraint on a Net

The Routing Constraint Manager lets you establish the design needs, save them as constraints, and share those constraints across specification and implementation to drive the accelerated layout solution with reduced errors. A constraint-driven design preserves the design intent by enabling efficient design collaboration. Using Routing Constraint Manager, you can create following constraints:

- NetClass
- Bus
- Symmetry
- DiffPair
- Match
- Shield

To set up a constraint on a net:

1. Choose *Window – Assistants – Routing*.

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

   The Routing assistant is displayed.

2. Click *Routing Constraint Manager* on the Routing assistant toolbar.

   The Routing Constraint Manager displays.

3. Select a net from the *Net* column on the Nets tab.

   The constraints are activated depending on the number of nets selected. For example, you cannot add more than two nets to a DiffPair constraint group.

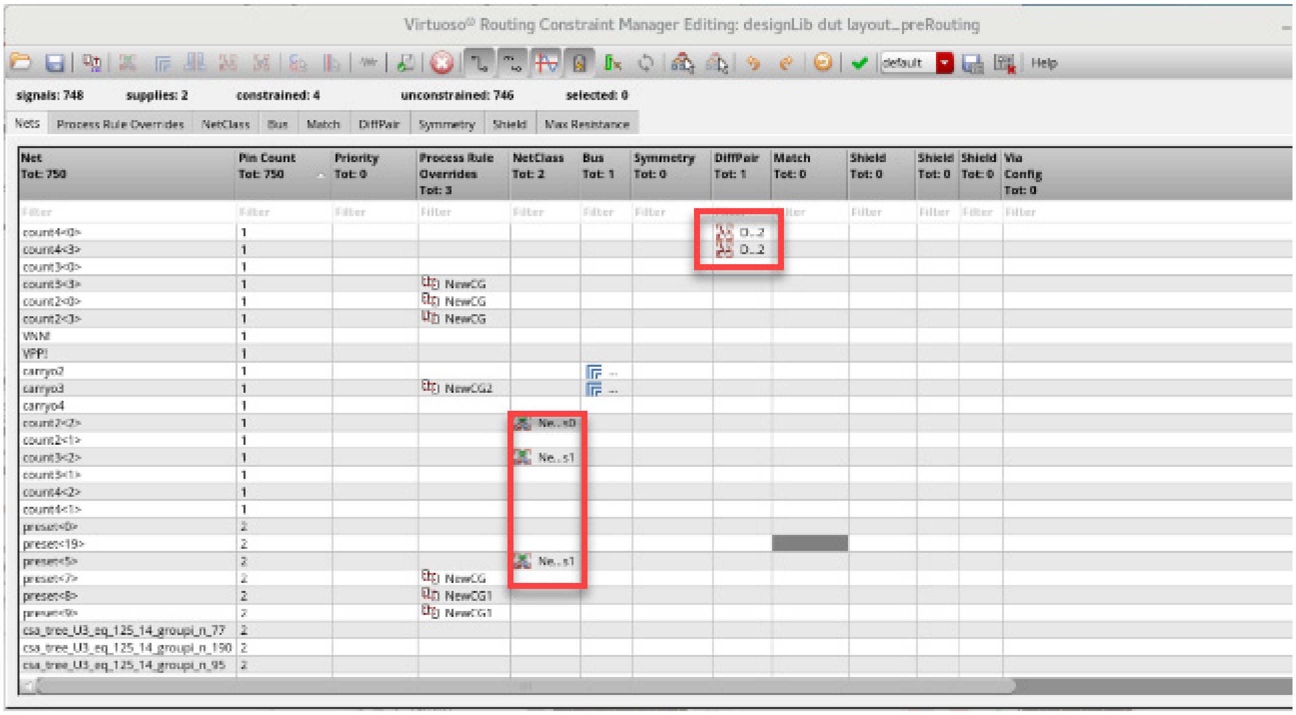
   If you select a single net, all constraint icons are active except Bus, DiffPair and Match. The Bus, DiffPair, and Match constraints require at least two nets to be selected.

4. Click the button for the constraint that you want to create. For example:.

   - ❑ To create a NetClass constraint, click .
   - ❑ To create a bus constraint, click .
   - ❑ To create a Symmetry constraint, click .
   - ❑ To create a DiffPair constraint, click .

- ❑ To create a Match Length constraint, click .

The created constraint appears in the specific constraint column of the constraints table.

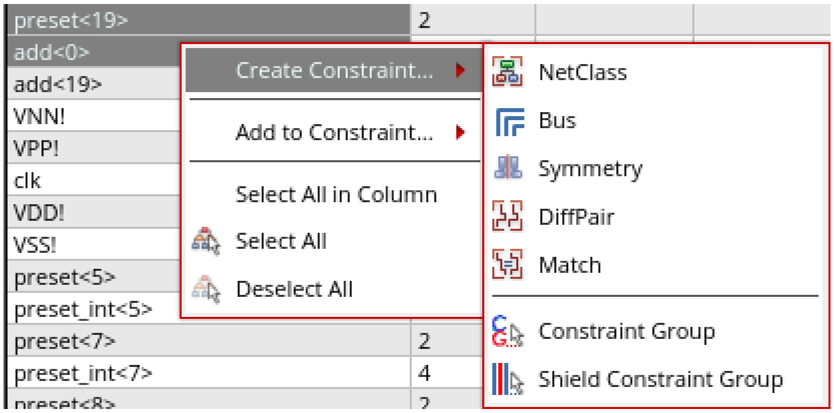


signals: 748   supplies: 2   constrained: 4   unconstrained: 746   selected: 0

Net Tot: 750	Pin Count Tot: 750	Priority Tot: 0	Process Rule Overrides Tot: 3	NetClass Tot: 2	Bus Tot: 1	Symmetry Tot: 0	DiffPair Tot: 1	Match Tot: 0	Shield Tot: 0	Shield Tot: 0	Shield Tot: 0	Via Config Tot: 0
Filter	Filter	Filter	Filter	Filter	Filter	Filter		Filter	Filter	Filter	Filter	Filter
count4<0>	1						0...2					
count4<3>	1						0...2					
count3<0>	1											
count5<3>	1		NewCG									
count2<0>	1		NewCG									
count2<3>	1		NewCG									
VNN!	1											
VPP!	1											
carryo2	1				...							
carryo3	1		NewCG2		...							
carryo4	1											
count2<2>	1			Ne...s0								
count2<1>	1											
count3<2>	1			Ne...s1								
count5<1>	1											
count4<2>	1											
count4<1>	1											
preset<0>	2											
preset<19>	2											
preset<5>	2			Ne...s1								
preset<7>	2		NewCG									
preset<8>	2		NewCG1									
preset<9>	2		NewCG1									
csa_tree_U3_eq_125_14_groupi_n_77	2											
csa_tree_U3_eq_125_14_groupi_n_190	2											
csa_tree_U3_eq_125_14_groupi_n_95	2											

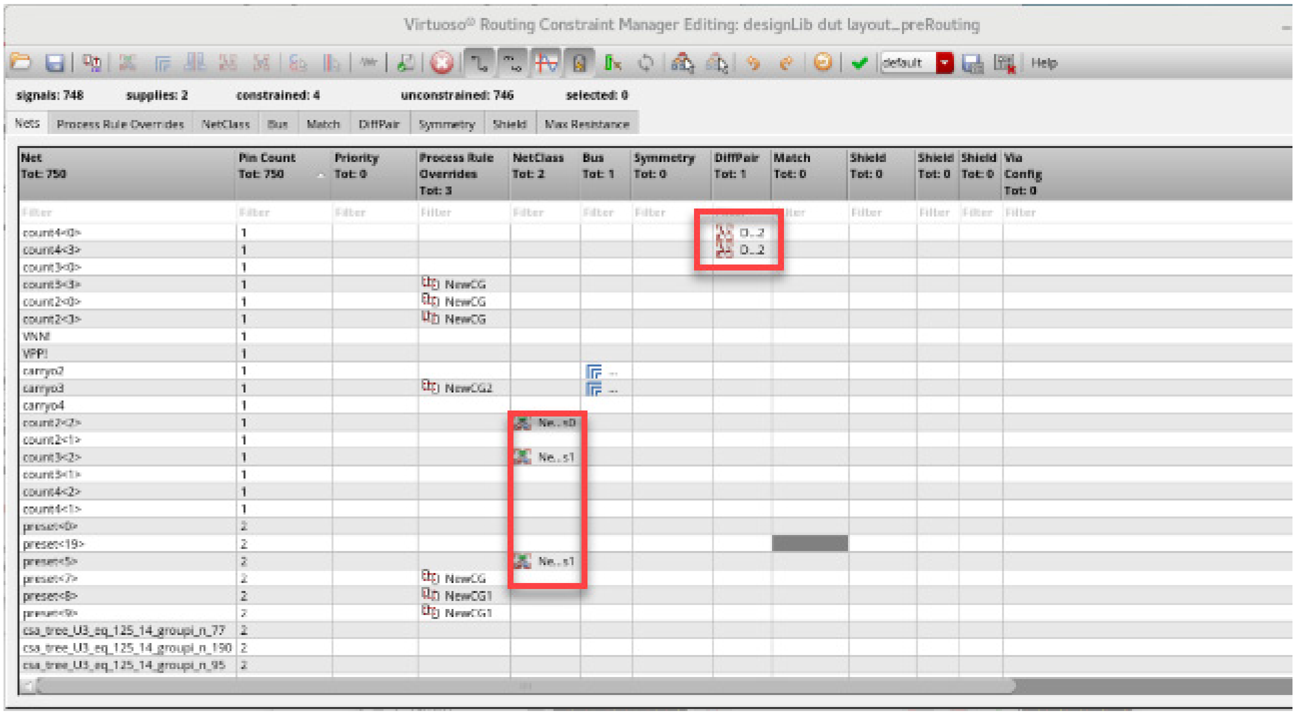
An alternate method to set up a constraint is:

1. Right-click the selected net. A menu is displayed.
2. Place the mouse cursor on the Create Constraint.



3. Click the constraint name that you want to create. For example, to create the netclass constraint, click *NetClass*.

The created constraint appears in the specific constraint column of the constraints table.



Virtuoso® Routing Constraint Manager Editing: designLib dut layout_preRouting

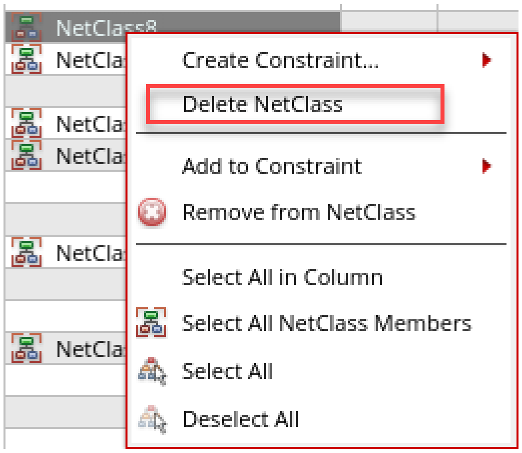
signals: 748 supplies: 2 constrained: 4 unconstrained: 746 selected: 0

Net Tot: 750	Pin Count Tot: 750	Priority Tot: 0	Process Rule Overrides Tot: 3	NetClass Tot: 2	Bus Tot: 1	Symmetry Tot: 0	DiffPair Tot: 1	Match Tot: 0	Shield Tot: 0	Shield Tot: 0	Shield Tot: 0	Via Config Tot: 0
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
count4<0>	1						0..2					
count4<3>	1						0..2					
count3<0>	1											
count3<3>	1		NewCG									
count2<0>	1		NewCG									
count2<3>	1		NewCG									
VNN!	1											
VPP!	1											
carryo2	1				...							
carryo3	1		NewCG2		...							
carryo4	1											
count2<2>	1			Ne...s0								
count2<1>	1											
count3<2>	1			Ne...s1								
count3<1>	1											
count4<2>	1											
count4<1>	1											
preset<0>	2											
preset<19>	2											
preset<5>	2			Ne...s1								
preset<7>	2		NewCG									
preset<8>	2		NewCG1									
preset<9>	2		NewCG1									
csa_tree_U3_eq_125_14_groupi_n_77	2											
csa_tree_U3_eq_125_14_groupi_n_190	2											
csa_tree_U3_eq_125_14_groupi_n_95	2											

Deleting a Constraint

To delete a constraint:

1. Right-click the constraint name that you want to delete.
2. Click *Delete <constraint name>*. For example, to delete a NetClass constraint, click *Delete NetClass*.

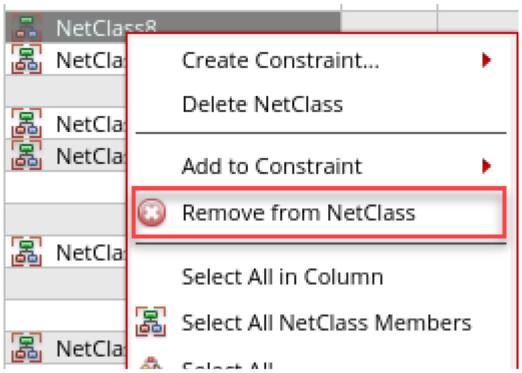


The constraint is deleted.

Removing a Net from a Constraint

To remove a net from a constraint:

1. Right-click the constraint name next to the net name that you want to remove form the constraint.
2. Click *Remove from <constraint name>*. For example, to remove a net from the NetClass constraint, click *Remove from NetClass*.



The net is removed from the NetClass constraint.

Related Topics

Loading Routing Constraints

Creating and Deleting a Constraint Group

Editing a Constraint

Selecting Nets in Navigator

Creating a Shield Constraint

Adding and Removing a Net from a Constraint Group

Creating a Shield Constraint

To create a shield constraint and set the associated shielding net:

1. Choose *Window – Assistants – Routing*.

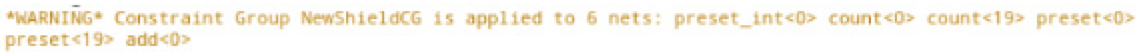
   Alternatively, right-click anywhere on the layout window menu bar and choose *Assistants – Routing*.

   The Routing assistant is displayed.

2. Click *Routing Constraint Manager* on the Routing assistant toolbar.

3. Select the net on which you want to add the shield constraint from the *Net* column in the *Nets* tab.

   If you select multiple nets and apply a shielding constraint, a consistency check warning message is displayed in the CIW.



```
*WARNING* Constraint Group NewShieldCG is applied to 6 nets: preset_int<0> count<0> count<19> preset<0>
preset<19> add<0>
```

4. Click *Apply Shield Constraint Group* on the Routing Constraint Manager toolbar. Alternatively, right-click and choose *Add to Constraint – Shield Constraint Group – <ShieldCGname>*.

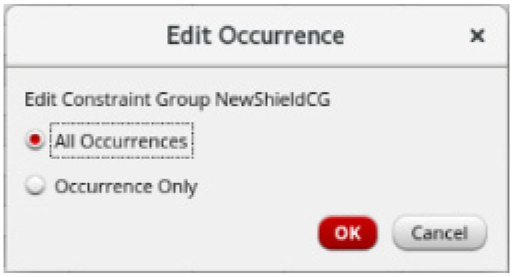
   You can also right-click and choose *Create Constraint...– Shield Constraint Group*. This lets you to either use the existing shield constraint group or create a new one.

   The shield constraint is added to the selected nets. You can now adjust the shield nets.

   Note: Before shielding is created, ensure that geometric power mesh shapes exist. Therefore, use supply routing before routing with shields or manually create trunks on power ground nets.

5. To adjust shield nets, double-click the shield constraint group on a net.

   The Edit Occurrence form appears. You can either edit the shield constraint group for all occurrences or for selected occurrence.



Edit Occurrence

Edit Constraint Group NewShieldCG

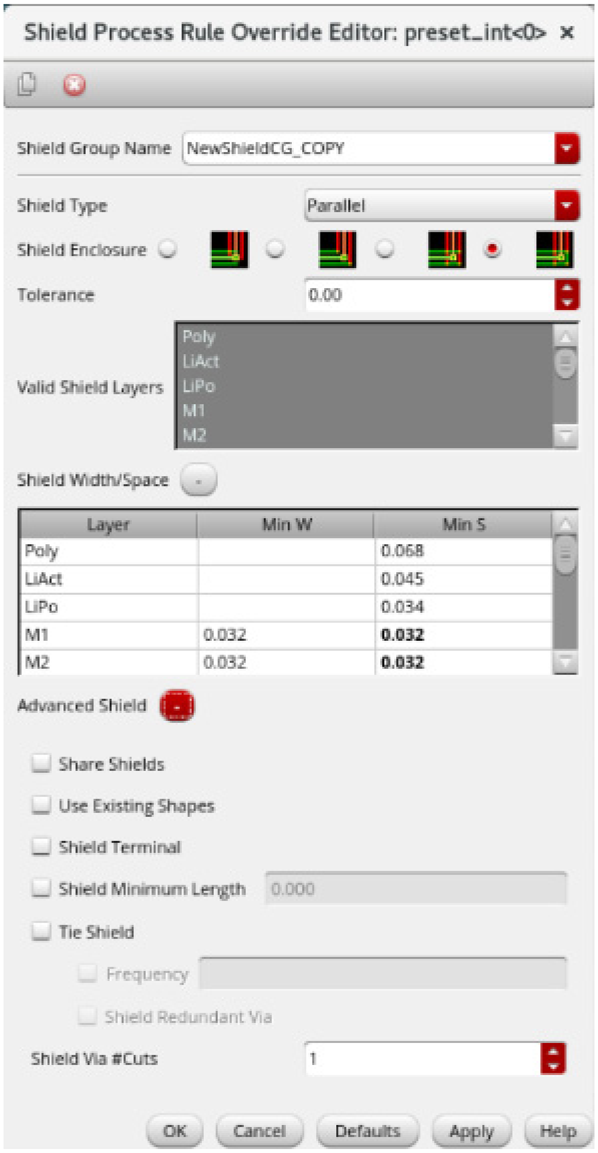
● All Occurrences

○ Occurrence Only

6. Ensure that *All Occurrences* is selected. This is the default option. Selecting *All Occurrences* affects all constraints that are using this shield constraint.

7. Click *OK*.

   The Shield Process Rule Override Editor form appears.



Shield Process Rule Override Editor: preset_int<0>

Shield Group Name: NewShieldCG_COPY

Shield Type: Parallel

Shield Enclosure: ○ ○ ○ ●

Tolerance: 0.00

Valid Shield Layers: Poly, LiAct, LiPo, M1, M2

Shield Width/Space

Layer	Min W	Min S
Poly		0.068
LiAct		0.045
LiPo		0.034
M1	0.032	**0.032**
M2	0.032	**0.032**

Advanced Shield

□ Share Shields

□ Use Existing Shapes

□ Shield Terminal

□ Shield Minimum Length 0.000

□ Tie Shield

□ Frequency

□ Shield Redundant Via

Shield Via #Cuts: 1

8. Expand *Shield Width/Space* and the *Advanced Shield* sections to view the options.

9. Change the minimum width and minimum spacing to `0.1um` for the metal layers.

10. Click *OK*.

    You have defined the shield constraint on the selected nets and adjusted the shield nets.

Related Topics

Loading Routing Constraints

Creating and Deleting a Constraint Group

Editing a Constraint

Selecting Nets in Navigator

Setting up a Constraint on a Net

Adding and Removing a Net from a Constraint Group

Shield Process Rule Override Editor Form

Creating a Bus Constraint and Routing the Buses

To create a Bus constraint:

1. Choose *Window – Assistants – Routing*.

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

   The Routing assistant is displayed.

2. Click *Routing Constraint Manager* on the Routing assistant toolbar.

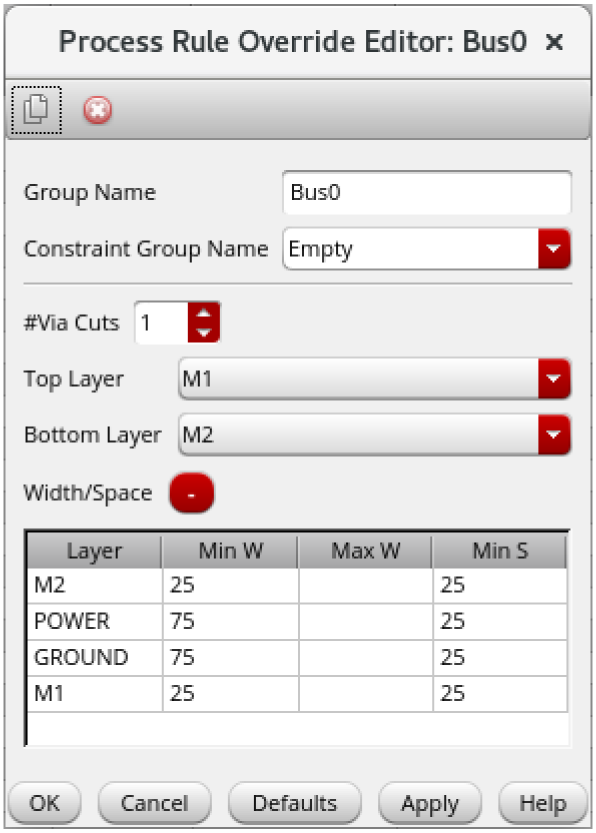
3. Select the net on which you want to add the bus constraint from the *Net* column in the *Nets* tab.

4. Click the Bus button on the Routing Constraint Manager toolbar. Alternatively, right-click and choose *Create Constraint...– Bus*.

   The bus constraint is added to the selected nets.

5. To adjust bus constraint created on the selected nets, double-click the bus constraint on a net.

The Process Rule Override Editor: Bus form appears.



Group Name: Bus0

Constraint Group Name: Empty

#Via Cuts: 1

Top Layer: M1

Bottom Layer: M2

Width/Space: -

Layer	Min W	Max W	Min S
M2	25		25
POWER	75		25
GROUND	75		25
M1	25		25

6. Select top and bottom layers from the *Top Layer* and *Bottom Layer* fields, respectively.
7. Repeat the previous steps to add a bus constraint on other nets.
8. Click the *Bus* tab in the Routing Constraint Manager and select the available buses.
9. Right-click and choose *Select All Bus Members*.

   All the nets in the available buses are selected.
10. Click the *Run signal router* button in the *Route* tab of the Routing Assistant. Ensure that the routing type is selected as *Chip*.

    You can see the bus routing result for all the nets.

Related Topics

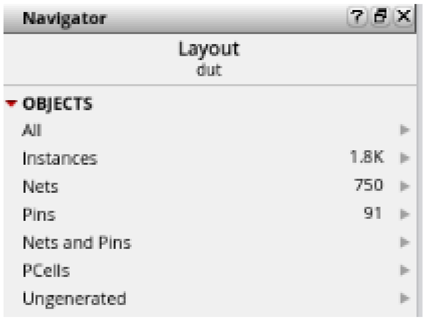
Loading Routing Constraints

Creating and Deleting a Constraint Group

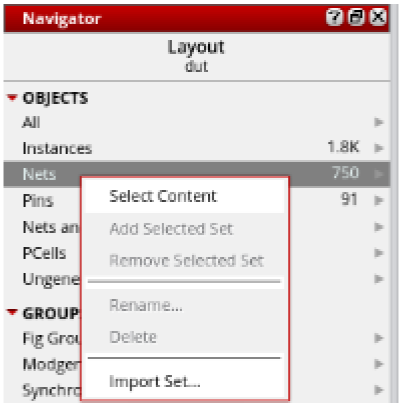
Selecting Nets in Navigator

Another way to select nets and constraint is using the Navigator assistant. To do this:

1. Right-click *Nets* in Navigator.



2. Click *Select Content*.



3. Choose *Window – Assistants – Routing*.

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

The Routing assistant is displayed.

4. Click *Routing Constraint Manager* on the Routing assistant toolbar.

   The Routing Constraint Manager displays.

5. Click the *Snapshot Selected Off* button on the Routing Constraint Manager toolbar.

   After selecting the *Snapshot Selected ON* button, the Routing Constraint Manager displays the nets selected in Navigator without having to filter or search in the *Net* tab.

6. Now, select another set of nets to add to the selected set in Routing Constraint Manager.
7. Right-click Nets in Navigator.
8. Click *Select Content*.
9. Click *Synchronize Snapshot* button on the Routing Constraint Manager toolbar.

   This adds the nets in the navigator to the selected set.

Related Topics

Loading Routing Constraints

Creating and Deleting a Constraint Group

Editing a Constraint

Setting up a Constraint on a Net

Creating a Shield Constraint

Adding and Removing a Net from a Constraint Group

Creating a Bus Constraint and Routing the Buses

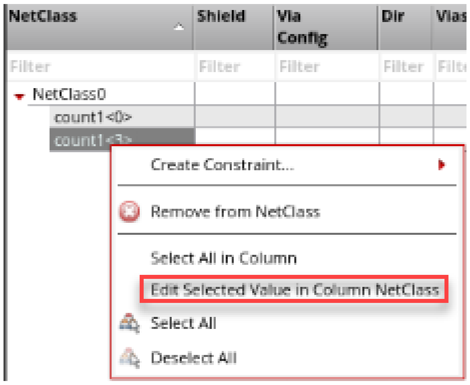
Editing a Constraint

All constraints can be edited from their assigned tabs in the Routing Constraint Manager. You can double click and bring up the Process Rule Override Editor form for a constraint. In this topic, let us consider editing a NetClass constraint. To edit a NetClass constraint:

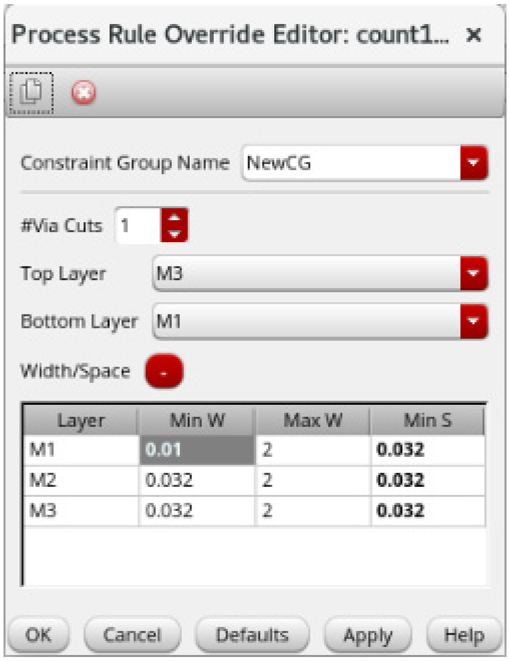
1. Double-click a NetClass constraint in the *NetClass* column.

   The Process Rule Override Editor:NetClass form appears without requiring a constraint group to be applied. *Empty* is displayed in the *Constraint Group Name* field in the form.

   Alternatively, right-click the constraint and then select *Edit Selected Value in Column NetClass*.



   The Process Rule Override Editor form appears. You can now create or edit the routing constraint.



Constraint Group Name: NewCG

#Via Cuts: 1

Top Layer: M3

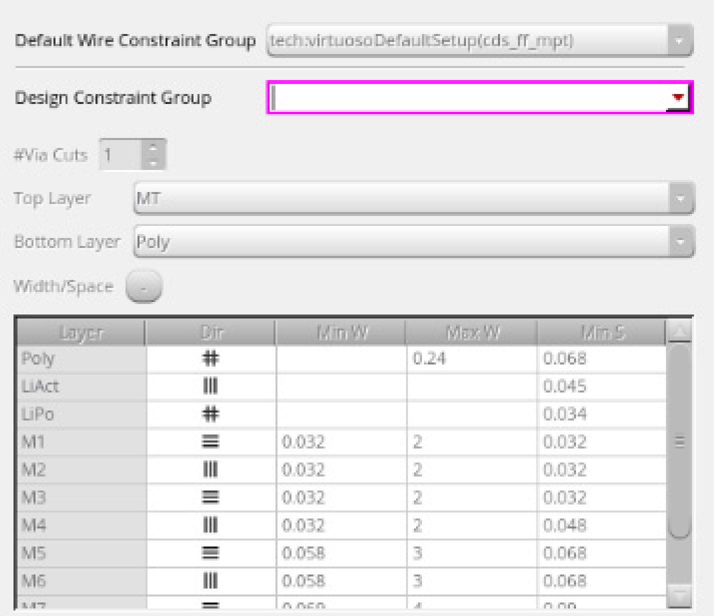
Bottom Layer: M1

Width/Space

Layer	Min W	Max W	Min S
M1	0.01	2	0.032
M2	0.032	2	0.032
M3	0.032	2	0.032

2. Change the group name without a constraint group. If you specify a group name that already exists elsewhere in the design or if the group name field is left blank, then the field

with the group name or constraint group name is highlighted and all other fields are disabled until you have specified a unique name.



Default Wire Constraint Group: tech:virtuosoDefaultSetup(cds_ff_mpt)

Design Constraint Group:

#Via Cuts: 1

Top Layer: MT

Bottom Layer: Poly

Width/Space

Layer	Dir	Min W	Max W	Min S
Poly	#		0.24	0.068
LiAct	|||			0.045
LiPo	#			0.034
M1	≡	0.032	2	0.032
M2	|||	0.032	2	0.032
M3	≡	0.032	2	0.032
M4	|||	0.032	2	0.048
M5	≡	0.058	3	0.068
M6	|||	0.058	3	0.068

3. Select the constraint group name. If you override constraint values then the constraint group changes from *Empty* to the next unique constraint group name, such as `NetClass2`.
4. Select a new value in the *Via Cuts* field.
5. Double-click and change the width value of a metal layer. After changing the value, the color of the cell changes for the modified value. This color change reflects the change from the default value.
6. Reset the width value to the default value. To do this:
    a. Right-click the column header in the table.
    b. From the shortcut menu, click *Reset column to default*.
7. Click *OK*.

   The changes made to the constraint are visible in the respective columns next to the net name.

Related Topics

Loading Routing Constraints

Creating and Deleting a Constraint Group

Selecting Nets in Navigator

Setting up a Constraint on a Net

Creating a Shield Constraint

Adding and Removing a Net from a Constraint Group

Process Rule Override Editor Form

Checks for Routing Constraints

The Routing Constraint Manager constraint checker lets you check constraints on routed nets. Some of the checks performed by the constraint checker are:

Shield Checks

- Perform shielding checks for the following topologies: parallel, tandem, and coaxial.
- Flag shielded nets forwhich the ratio of existing shields to ideal shields exceeds the specified tolerance.
- Flag shielded nets with the following errors. The errors reported with markers are:
    - Shield wires that violate `minWidth`
    - Regions where shielding is not placed at exactly gap spacing
    - Tandem shields that violate `tandemWidth`
    - Regions where the shielding is missing
    - Shield wires that violate valid shield and number of via shield cuts
        - Need to check if `validLayers` applies to the layers that are shielded, or the layers on which shields can be built
    - Nets shielded with the wrong net are considered unshielded.
    - No markers are generated if the ratio falls below the tolerance

Related Topics

Checking Routing Constraints

Routing Constraint Manager User Interface

Checking Routing Constraints

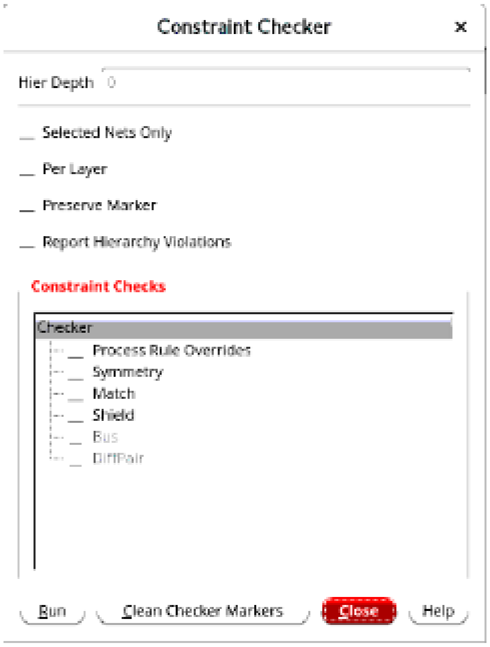
To check routing constraints:

1. In the Routing Assistant, click Routing Constraint Manager.

   The Routing Constraint Manager appears.

2. Click the Check Constraints ✓ button.

   The Constraint Checker form is displayed.



Constraint Checker

Hier Depth 0

☐ Selected Nets Only

☐ Per Layer

☐ Preserve Marker

☐ Report Hierarchy Violations

Constraint Checks

Checker
- ☐ Process Rule Overrides
- ☐ Symmetry
- ☐ Match
- ☐ Shield
- ☐ Bus
- ☐ DiffPair

3. Specify a value to adjust the depth of checks in the *Hier Depth* spin box.

4. Click *Selected Nets Only* if you want to check the routing constraints only on certain nets that are selected in the Routing Constraint Manager rather than all nets.

5. Select *Checks* to enable checks for all constraints.

6. From the *Constraint Checks* list, select the constraint for which you want to run the check. For example, to check for all shields in the layout, select *Shield*.

7. Click *Run*.

   You can see colored markers in the Routing Constraint Manager athat show the shield nets that have violations. The markers in the Routing Constraint Manager indicate that a constraint has a violation identified by the Constraint Checker. These markers are automatically updated in the Routing Constraint Manager if they are deleted outside of the Routing Constraint Manager, such as in the Annotation Browser or layout window.

   You can also see the information in the CIW along with the warning markers in Routing Constraint Manager. The routing violations are also reported in the Annotation Browser.

8. Click Close to close the Constraint Checker form.

9. In the Annotation Browser, open the *Constraints* tab.

   You can see that the shielded net is in violation because the tolerance is set to 0 in the shield constraint group. This leads to a situation where every single unshielded portion of the wire is reported as a violation.

10. Select any violation and see that it reports the missing pieces of shielding next to the pins.

11. To clean the warning marker in Routing Constraint Manager, click *Clean Checker Markers* at the bottom of the Constraint Checker form.

    This removes all the warning markers that were created.

Related Topics

Checks for Routing Constraints

Routing Constraint Manager User Interface

5

Chip Assembly Routing Flow Environment Variables

The chip assembly routing flow environment variables provides information on the names, descriptions, and graphical user interface equivalents. These environment variables are used to set default values for various chip assembly routing options in the Routing assistant.

You can set them either from CIW or load from cdsenv file through a SKILL function. For example, envLoadFile("~/.cdsenv").

Only the public environment variables are documented and supported for public use. All other chip assembly 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 chip assembly routing environment variables.

Chip Assembly Routing

check_displayLog	check_existingDRCs	check_generateMarkers
check_nets	check_overwriteLog	check_supplyNets
postRouteTrigger	preRouteTrigger	results_nets
results_netsWithin	results_supplyNets	route_defaultRoutedView
route_deletePreroutes	route_deleteShieldTies	route_deleteShields
route_deleteSpines	route_deleteWiresAndVias	route_detailRoute
route_detailRoutePasses	route_displayLog	route_generateShield
route_globalRoute	route_globalRoutePasses	route_nets
route_netsWithin	route_optimizeRoute	route_optimizeRoutePasses
route_overwriteLog	route_removeShortsAndDRCs	route_removeShortsAndDRCsPasses
route_routedLoc	route_saveRoutingOnly	route_supplyNets

route_tieShield

setup_checkDRCsAfterRouting

setup_coverObstructions

setup_cutKeepouts

setup_extractConnectivityAfterRouting

setup_insertTrim

setup_offsetVia

setup_pinDirectionWidth

setup_pinFit

setup_postRouteCmdFileName

setup_preRouteCmdFileName

setup_protectExistingRouting

setup_rotateVia

setup_taper

setup_taperDown

setup_taperUp

setup_usePostRouteCmdFile

setup_usePreRouteCmdFile

setup_useTrackBasedRouting

setup_viaStackCuts

setup_viaStackOption

Related Topics

Routing Assistant

Routing Assistant User Interface for Chip Assembly Routing Flow

Virtuoso Pre-Route Browser

check_displayLog

```
APR.chip.custom 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.chip.custom" "check_displayLog")
envSetVal("APR.chip.custom" "check_displayLog" 'boolean nil)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

check_existingDRCs

```
APR.chip.custom 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.chip.custom" "check_existingDRCs")
envSetVal("APR.chip.custom" "check_existingDRCs" 'boolean t)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

check_generateMarkers

```
APR.chip.custom check_generateMarkers boolean { t | nil }
```

Description

Displays the error markers in the *Misc* tab of the Annotation Browser.

The default is `nil`.

GUI Equivalent

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

Examples

```
envGetVal("APR.chip.custom" "check_generateMarkers")
envSetVal("APR.chip.custom" "check_generateMarkers" 'boolean t)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

check_nets

```
APR.chip.custom check_nets cyclic { "All" | "Selected" }
```

Description

Specifies whether to check all nets or selected nets. The default is "`All`".

GUI Equivalent

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

Examples

```
envGetVal("APR.chip.custom" "check_nets")
envSetVal("APR.chip.custom" "check_nets" 'cyclic "Selected")
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

check_overwriteLog

```
APR.chip.custom 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.chip.custom" "check_overwriteLog")
envSetVal("APR.chip.custom" "check_overwriteLog" 'boolean nil)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

check_supplyNets

```
APR.chip.custom check_supplyNets boolean { t | nil }
```

Description

Specifies whether to check power or ground (`tieHi` and `tieLo`) nets. When set to `t`, the power and ground nets are routed. The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Check* tab
Field	*Include supply nets*

Examples

```
envGetVal("APR.chip.custom" "check_supplyNets")
envSetVal("APR.chip.custom" "check_supplyNets" 'boolean t)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

postRouteTrigger

```
APR.chip.custom postRouteTrigger string "procedurename"
```

Description

Specifies the 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.chip.custom" "postRouteTrigger")
envSetVal("APR.chip.custom" "postRouteTrigger" 'string "postRouteProcedure")
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

preRouteTrigger

`APR.chip.custom preRouteTrigger string "`*`procedurename`*`"`

Description

Specifies the SKILL procedure to be executed before routing.

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

GUI Equivalent

None

Examples

```
envGetVal("APR.chip.custom" "preRouteTrigger")
envSetVal("APR.chip.custom" "preRouteTrigger" 'string "preRouteProcedure")
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

results_nets

```
APR.chip.custom results_nets cyclic { "All" | "Selected" | "Open" | "Shorted" }
```

Description

Specifies whether you want all nets, selected nets, or nets with opens and shorts to be included in the results table.

The default is `"All"`.

GUI Equivalent

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

Examples

```
envGetVal("APR.chip.custom" "results_nets")
envSetVal("APR.chip.custom" "results_nets" 'cyclic "Selected")
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

results_netsWithin

```
APR.chip.custom results_netsWithin cyclic { "PR boundary" | "FigGroup" | "Area" }
```

Description

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

The default is `"PR boundary"`.

GUI Equivalent

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

Examples

```
envGetVal("APR.chip.custom" "results_netsWithin")
envSetVal("APR.chip.custom" "results_netsWithin" 'cyclic "FigGroup")
envSetVal("APR.chip.custom" "results_netsWithin" 'cyclic "Area")
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

results_supplyNets

```
APR.chip.custom results_supplyNets boolean { t | nil }
```

Description

Includes power and ground nets 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.chip.custom" "results_supplyNets")
envSetVal("APR.chip.custom" "results_supplyNets" 'boolean t)
```

Related Topics

Routing Assistant User Interface for Chip Assembly Routing Flow

route_defaultRoutedView

```
APR.chip.custom route_defaultRoutedView string "defaultViewName"
```

Description

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

The default is `layout.routed`.

GUI Equivalent

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

Examples

```
envGetVal("APR.chip.custom" "route_defaultRoutedView")
envSetVal("APR.chip.custom" "route_defaultRoutedView" 'string "routed1")
```

Related Topics

Routing Assistant User Interface for Chip Assembly Routing Flow

route_deletePreroutes

```
APR.chip.custom route_deletePreroutes boolean { t | nil }
```

Description

Controls automatic deletion of the pre-routed wires and vias that are manually created.

The default is `nil`.

GUI Equivalent

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

Examples

```
envGetVal("APR.chip.custom" "route_deletePreroutes")
envSetVal("APR.chip.custom" "route_deletePreroutes" 'boolean t)
```

Related Topics

Routing Assistant User Interface for Chip Assembly Routing Flow

route_deleteShieldTies

```
APR.chip.custom route_deleteShieldTies boolean { t | nil }
```

Description

Controls automatic deletion of the tie shields that are created by the router.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Delete – Shield ties*

Examples

```
envGetVal("APR.chip.custom" "route_deleteShieldTies")
envSetVal("APR.chip.custom" "route_deleteShieldTies" 'boolean t)
```

Related Topics

Routing Assistant User Interface for Chip Assembly Routing Flow

route_deleteShields

```
APR.chip.custom route_deleteShields boolean { t | nil }
```

Description

Controls automatic deletion of the shields that are created by the router.

The default is `nil`.

GUI Equivalent

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

Examples

```
envGetVal("APR.chip.custom" "route_deleteShields")
envSetVal("APR.chip.custom" "route_deleteShields" 'boolean t)
```

Related Topics

Routing Assistant User Interface for Chip Assembly Routing Flow

route_deleteSpines

```
APR.chip.custom route_deleteSpines boolean { t | nil }
```

Description

Controls automatic deletion of the spine wires that are created by the router.

The default is `nil`.

GUI Equivalent

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

Examples

```
envGetVal("APR.chip.custom" "route_deleteSpines")
envSetVal("APR.chip.custom" "route_deleteSpines" 'boolean t)
```

Related Topics

Routing Assistant User Interface for Chip Assembly Routing Flow

route_deleteWiresAndVias

```
APR.chip.custom route_deleteWiresAndVias boolean { t | nil }
```

Description

Controls automatic deletion of the wires and vias that are created by the router.

The default is `nil`.

GUI Equivalent

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

Examples

```
envGetVal("APR.chip.custom" "route_deleteWiresAndVias")
envSetVal("APR.chip.custom" "route_deleteWiresAndVias" 'boolean t)
```

Related Topics

Routing Assistant User Interface for Chip Assembly Routing Flow

route_detailRoute

```
APR.chip.custom route_detailRoute boolean { t | nil }
```

Description

Specifies whether detailed routing should be enabled. If set to `t`, a value must be specified with the `route_detailRoutePasses` variable.

The default value is `nil`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Detail route*

Examples

```
envGetVal("APR.chip.custom" "route_detailRoute")
envSetVal("APR.chip.custom" "route_detailRoute" 'boolean nil)
```

Related Topics

Routing Assistant User Interface for Chip Assembly Routing Flow

route_detailRoutePasses

`APR.chip.custom route_detailRoutePasses int` *`number_of_passes`*

Description

Specifies the number of passes for detailed routing. This variable is effective only if the `route_detailRoute` variable is set to `t`.

The default value is `25`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Detail route – passes*

Examples

```
envGetVal("APR.chip.custom" "detailRoutePasses")
envSetVal("APR.chip.custom" "detailRoutePasses" 'int 15)
```

Related Topics

Routing Assistant User Interface for Chip Assembly Routing Flow

route_displayLog

```
APR.chip.custom route_displayLog boolean { t | nil }
```

Description

Controls the display of the log window when the chip assembly router 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.chip.custom" "route_displayLog")
envSetVal("APR.chip.custom" "route_displayLog" 'boolean nil)
```

Related Topics

Routing Assistant User Interface for Chip Assembly Routing Flow

route_generateShield

```
APR.chip.custom route_generateShield boolean { t | nil }
```

Description

Controls the generation of shielding wires during routing. When set to `t`, the shielding wires are automatically generated. For this, the shielding constraint should be available.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Generate shield*

Examples

```
envGetVal("APR.chip.custom" "route_generateShield")
envSetVal("APR.chip.custom" "route_generateShield" 'boolean t)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

route_globalRoute

```
APR.chip.custom route_globalRoute boolean { t | nil }
```

Description

Specifies whether global routing should be enabled. When set to `t`, a value must be specified with the `route_globalRoutePasses` variable.

The default value is `nil`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Global route*

Examples

```
envGetVal("APR.chip.custom" "route_globalRoute")
envSetVal("APR.chip.custom" "route_globalRoute" 'boolean t)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

route_globalRoutePasses

`APR.chip.custom globalRoutePasses int` *`number_of_passes`*

Description

Specifies the number of passes for global routing. This variable is effective only if the `route_globalRoute` variable is set to `t`.

The default value is `3`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Global route – passes*

Examples

```
envGetVal("APR.chip.custom" "globalRoutePasses")
```

```
envSetVal("APR.chip.custom" "globalRoutePasses" 'int 2)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

route_nets

```
APR.chip.custom route_nets cyclic { "All" | "Selected" | "Open" | "Shorted" }
```

Description

Specifies whether to route all nets, selected nets, or 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.chip.custom" "route_nets")
envSetVal("APR.chip.custom" "route_nets" 'cyclic "Open")
envSetVal("APR.chip.custom" "route_nets" 'cyclic "Shorted")
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

route_netsWithin

```
APR.chip.custom route_netsWithin cyclic { "PR boundary" | "FigGroup" | "Area"}
```

Description

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

The default is `"PR boundary"`.

GUI Equivalent

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

Examples

```
envGetVal("APR.chip.custom" "route_netsWithin")
envSetVal("APR.chip.custom" "route_netsWithin" 'cyclic "Guardring/FigGroup")
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

route_optimizeRoute

```
APR.chip.custom route_optimizeRoute boolean { t | nil }
```

Description

Specifies whether the post-route step to optimize routing results should be enabled. When set to `t`, a value must be specified with the `route_optimizeRoutePasses` variable.

The default value is `nil`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Optimize route*

Examples

```
envGetVal("APR.chip.custom" "route_optimizeRoute")
envSetVal("APR.chip.custom" "route_optimizeRoute" 'boolean t)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

route_optimizeRoutePasses

`APR.chip.custom optimizeRoutePasses int` *`number_of_passes`*

Description

Specifies the number of passes for optimize routing. This variable is effective only if the `route_optimizeRoute` variable is set to `t`.

The default value is `3`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Optimize route – passes*

Examples

```
envGetVal("APR.chip.custom" "optimizeRoutePasses")
```

```
envSetVal("APR.chip.custom" "optimizeRoutePasses" 'int 2)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

route_overwriteLog

```
APR.chip.custom route_overwriteLog boolean { t | nil }
```

Description

Specifies whether to overwrite or keep the existing log file. 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.chip.custom" "route_overwriteLog")
envSetVal("APR.chip.custom" "route_overwriteLog" 'boolean t)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

route_removeShortsAndDRCs

```
APR.chip.custom route_removeShortsAndDRCs boolean { t | nil }
```

Description

Specifies whether the post-routing step to remove shorts and DRC errors should be enabled. When set to `t`, a value must be specified with the `route_removeShortsAndDRCsPasses` variable.

The default value is `nil`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Remove shorts & DRCs*

Examples

```
envGetVal("APR.chip.custom" "route_removeShortsAndDRCs")
envSetVal("APR.chip.custom" "route_removeShortsAndDRCs" 'boolean t)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

route_removeShortsAndDRCsPasses

```
APR.chip.custom route_removeShortsAndDRCsPasses int number_of_passes
```

Description

Specifies the number of passes to remove shorts and DRC errors. This variable is effective only if the `route_removeShortsAndDRCs` variable is set to `t`.

The default value is `3`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Optimize Route – passes*

Examples

```
envGetVal("APR.chip.custom" "route_removeShortsAndDRCsPasses")
```

```
envSetVal("APR.chip.custom" "route_removeShortsAndDRCsPasses" 'int 2)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

route_routedLoc

```
APR.chip.custom route_routedLoc cyclic { "Current cellview" | "Other cellview" }
```

Description

Specifies whether to write the routing results 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.chip.custom" "route_routedLoc")
envSetVal("APR.chip.custom" "route_routedLoc" 'cyclic "Other cellview")
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

route_saveRoutingOnly

```
APR.chip.custom route_saveRoutingOnly boolean { t | nil }
```

Description

Specifies whether to save chip assembly routing to a cellview.

The default is `nil`.

GUI Equivalent

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

Examples

```
envGetVal("APR.chip.custom" "route_saveRoutingOnly")
envSetVal("APR.chip.custom" "route_saveRoutingOnly" 'boolean t)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

route_supplyNets

```
APR.chip.custom route_supplyNets boolean { t | nil }
```

Description

Specifies whether to route power or 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.chip.custom" "route_supplyNets")
envSetVal("APR.chip.custom" "route_supplyNets" 'boolean nil)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

route_tieShield

```
touting.chip.vcr route_tieShield boolean { t | nil }
```

Description

Controls tying of the shielding wires together to bring connected shielding wires to the nearest power source, which can be part of a power pin, power ring, or trunk. When set to `t`, the shielding wires are automatically tied.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Route* tab
Field	*Tie shield*

Examples

```
envGetVal("APR.chip.custom" " route_tieShield")
envSetVal("APR.chip.custom" " route_tieShield" 'boolean t)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

setup_checkDRCsAfterRouting

```
APR.chip.custom setup_checkDRCsAfterRouting boolean { t | nil }
```

Description

Specifies whether to automatically run design rule checks before routing. When set to `t`, the design rule checks are run automatically using the settings in the DRD Options form.

The default is `t`.

GUI Equivalent

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

Examples

```
envGetVal("APR.chip.custom" "setup_checkDRCsAfterRouting")
envSetVal("APR.chip.custom" "setup_checkDRCsAfterRouting" 'boolean nil)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

setup_coverObstructions

```
APR.chip.custom setup_coverObstructions boolean { t | nil }
```

Description

Considers the cover obstructions in the design. When set to `nil`, cover obstructions are ignored.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Setup* tab
Field	*Cover Obstructions*

Examples

```
envGetVal("APR.chip.custom" "setup_coverObstructions")
envSetVal("APR.chip.custom" "setup_coverObstructions" 'boolean nil)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

setup_cutKeepouts

```
APR.chip.custom setup_cutKeepouts boolean { t | nil }
```

Description

Specifies whether the router should cut keepouts for pin access. If set to `t`, the cut blockages around the pins are considered. The cutting distance is the `minSpacing` specified in the foundry constraint group.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Setup* tab
Field	*Cut Keepouts*

Examples

```
envGetVal("APR.chip.custom" "setup_cutKeepouts")
envSetVal("APR.chip.custom" "setup_cutKeepouts" 'boolean t)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

setup_extractConnectivityAfterRouting

```
APR.chip.custom setup_extractConnectivityAfterRouting boolean { t | nil }
```

Description

Species whether to run the Layout XL extractor after routing is complete. When set to `t`, the extractor is run automatically using the settings in the Layout Extractor Options form.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Setup* tab
Field	*Extract Connectivity after routing*

Examples

```
envGetVal("APR.chip.custom" "setup_extractConnectivityAfterRouting")
envSetVal("APR.chip.custom" "setup_extractConnectivityAfterRouting" 'boolean nil)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

setup_insertTrim

```
APR.chip.custom setup_insertTrim boolean { t | nil }
```

Description

Controls trim insertion to fix DRC errors.

The default is `t`.

GUI Equivalent

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

Examples

```
envGetVal("APR.chip.custom" " setup_insertTrim")
envSetVal("APR.chip.custom" " setup_insertTrim" 'boolean nil)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

setup_offsetVia

```
APR.chip.custom setup_offsetVia boolean { t | nil }
```

Description

Specifies whether the router should offset rectangular vias, such that the long dimension coincides with the wire outline.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Setup* tab
Field	*Offset Via*

Examples

```
envGetVal("APR.chip.custom" "setup_offsetVia")
envSetVal("APR.chip.custom" "setup_offsetVia" 'boolean nil)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

setup_pinDirectionWidth

```
APR.chip.custom setup_pinDirectionWidth boolean { t | nil }
```

Description

Controls the direction of the pins when the generated pin connections are tapered.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Setup* tab
Field	*Pin direction width*

Examples

```
envGetVal("APR.chip.custom" "setup_pinDirectionWidth")
envSetVal("APR.chip.custom" "setup_pinDirectionWidth" 'boolean t)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

setup_pinFit

```
APR.chip.custom setup_pinFit cyclic { "Off" | "Overlap pin" | "Within pin" |
     "Center" }
```

Description

Specifies the via placement style over the pins.

- `Off` Indicates that placement over the pins is disabled.
- `Overlap pin` vias can freely overlap with pin boundaries.
- `Within pin` overlapping vias must fit entirely within pin boundaries.
- `Center` the centers of overlapping vias must be within pin boundaries.

The default value is `Overlap pin.`

GUI Equivalent

Command	Routing assistant – *Setup* tab
Field	*Pin fit*

Examples

```
envGetVal("APR.chip.custom" "setup_pinFit")
envSetVal("APR.chip.custom" "setup_pinFit" 'cyclic "Off")
envSetVal("APR.chip.custom" "setup_pinFit" 'cyclic "Within pin")
envSetVal("APR.chip.custom" "setup_pinFit" 'cyclic "Center")
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

setup_postRouteCmdFileName

```
APR.chip.custom setup_postRouteCmdFileName string "filename"
```

Description

Specifies a file that can be used to define advanced routing options. These options are used after routing is complete.

The default is `./postRoute.cmd`.

GUI Equivalent

Command	Routing assistant – *Setup* tab
Field	*Use post-route command file* text field

Examples

```
envGetVal("APR.chip.custom" "setup_postRouteCmdFileName")
envSetVal("APR.chip.custom" "setup_postRouteCmdFileName" 'string "route.cmd")
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

setup_usePostRouteCmdFile

setup_preRouteCmdFileName

```
APR.chip.custom setup_preRouteCmdFileName string "filename"
```

Description

Specifies a file that can be used to define advanced routing options. These options are for initialization and are used before the router starts.

The default is `./preRoute.cmd`.

GUI Equivalent

Command	Routing assistant – *Setup* tab
Field	*Use pre-route command file* text field

Examples

```
envGetVal("APR.chip.custom" "setup_preRouteCmdFileName")
envSetVal("APR.chip.custom" "setup_preRouteCmdFileName" 'string "route_pre.cmd")
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

setup_usePreRouteCmdFile

setup_protectExistingRouting

```
APR.chip.custom setup_protectExistingRouting boolean { t | nil }
```

Description

Specifies whether the router should protect any existing wires during routing.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Setup* tab
Field	*Protect existing routing*

Examples

```
envGetVal("APR.chip.custom" "setup_protectExistingRouting")
envSetVal("APR.chip.custom" "setup_protectExistingRouting" 'boolean nil)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

setup_rotateVia

```
APR.chip.custom setup_rotateVia boolean { t | nil }
```

Description

Specifies whether the router can rotate vias during routing.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Setup* tab
Field	*Rotate via*

Examples

```
envGetVal("APR.chip.custom" "setup_rotateVia")
envSetVal("APR.chip.custom" "setup_rotateVia" 'boolean nil)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

setup_taper

```
APR.chip.custom setup_taper boolean { t | nil }
```

Description

Specifies whether or not the generated pin connections are tapered.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Setup* tab
Field	*Taper*

Examples

```
envGetVal("APR.chip.custom" "setup_taper")
envSetVal("APR.chip.custom" "setup_taper" 'boolean nil)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

setup_taperDown

setup_taperUp

setup_taperDown

```
APR.chip.custom setup_taperDown boolean { t | nil }
```

Description

Specifies whether or not the generated pin connections are tapered (neck down). To allow tapering in both directions, also specify the `setup_taperUp` variable.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Setup* tab
Field	*Down*

Examples

```
envGetVal("APR.chip.custom" "setup_taperDown")
envSetVal("APR.chip.custom" "setup_taperDown" 'boolean nil)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

setup_taper

setup_taperUp

```
APR.chip.custom setup_taperUp boolean { t | nil }
```

Description

Specifies whether or not the generated pin connections are tapered (neck up). To allow tapering in both directions, specify the `setup_taperDown` variable.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Setup* tab
Field	*Up*

Examples

```
envGetVal("APR.chip.custom" "setup_taperUp")
envSetVal("APR.chip.custom" "setup_taperUp" 'boolean nil)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

setup_taper

setup_usePostRouteCmdFile

```
APR.chip.custom setup_usePostRouteCmdFile boolean { t | nil }
```

Description

Controls the use of a post-route command file. When set to `t`, the post-route command file is used for advanced routing settings.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Setup* tab
Field	*Use post-route command file*

Examples

```
envGetVal("APR.chip.custom" "setup_usePostRouteCmdFile")
envSetVal("APR.chip.custom" "setup_usePostRouteCmdFile" 'boolean t)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

setup_postRouteCmdFileName

setup_usePreRouteCmdFile

```
APR.chip.custom setup_usePreRouteCmdFile boolean { t | nil }
```

Description

Controls the use of a pre-route command file. When set to `t`, the pre-route command file is used for advanced routing settings.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Setup* tab
Field	*Use pre-route command file*

Examples

```
envGetVal("APR.chip.custom" "setup_usePreRouteCmdFile")
envSetVal("APR.chip.custom" "setup_usePreRouteCmdFile" 'boolean t)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

setup_preRouteCmdFileName

setup_useTrackBasedRouting

```
APR.chip.custom setup_useTrackBasedRouting boolean { t | nil }
```

Description

Controls whether or not the track patterns should be used for routing. When set to `nil`, the track patterns are not considered for routing.

The default is `t`.

GUI Equivalent

Command	Routing assistant – *Setup* tab
Field	*Enable track-based routing*

Examples

```
envGetVal("APR.chip.custom" "setup_useTrackBasedRouting")
envSetVal("APR.chip.custom" "setup_useTrackBasedRouting" 'boolean nil)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

setup_viaStackCuts

```
APR.chip.custom setup_viaStackCuts boolean { t | nil }
```

Description

Enables inter-layer via cut stacking. This variable is effective only if the `setup_viaStackOption` variable is set to `Any overlap`.

The default is `nil`.

GUI Equivalent

Command	Routing assistant – *Setup* tab
Field	*Stack cuts*

Examples

```
envGetVal("APR.chip.custom" "setup_viaStackCuts")
envSetVal("APR.chip.custom" "setup_viaStackCuts" 'boolean t)
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

setup_viaStackOption

setup_viaStackOption

```
APR.chip.custom setup_viaStackOption cyclic { "Off" | "Any overlap" | "Exact" }
```

Description

Specifies the mode for inter-layer via stacking.

- `Off`: Indicates that via stacking is disabled.
- `Any overlap`: Indicates that vias are stacked with both centered and de-centered overlaps.
- `Exact`: Indicates that vias are only stacked center-to-center.

The default value is `Any overlap`.

GUI Equivalent

Command	Routing assistant – *Setup* tab
Field	*Stack vias*

Examples

```
envGetVal("APR.chip.custom" "setup_viaStackOption")
envSetVal("APR.chip.custom" "setup_viaStackOption" 'cyclic "Off")
envSetVal("APR.chip.custom" "setup_viaStackOption" 'cyclic "Exact")
```

Related Topics

Chip Assembly Routing Flow Environment Variables

Routing Assistant User Interface for Chip Assembly Routing Flow

setup_viaStackCuts

6

Routing-Related Forms

This topic lists the forms that are common for all routing types. These forms can either be invoked through the Routing assistant or forms that can be invoked from the Routing Constraint Manager.

Forms Related to Routing Assistant
Specify Routed Cellview
Virtuoso Pre-Route Browser
Virtuoso Routing Results Browser

Forms Related to Routing Constraint Manager
Axis Editor Form
Constraint Checker Form
Design Process Rule Override Editor Form
Load Constraints Form
Max Resistance Editor Form
Process Rule Override Editor: Bus Form
Process Rule Override Editor: DiffPair Form
Process Rule Override Editor: Match Form
Process Rule Override Editor: NetClass Form
Process Rule Override Editor: Symmetry Form
Shield Process Rule Override Editor Form

Related Topics

Routing Assistant

Virtuoso Routing Constraint Manager

Chip Assembly Routing

Chip Assembly Routing Configuration

Axis Editor Form

To open the Axis Editor form, click the *Axis* column in the *Symmetry* tab. This form lets you edit the axis direction.

Field	Description
Name	The name for the given axis constraint (multiple symmetries can use the same axis constraint).
Direction	The orientation of the axis of symmetry for the symmetry members. The valid values are *vertical* and *horizontal*.
Location	The distance from the design centerline (in x direction for vertical axis direction or y direction with horizontal axis direction) from which the symmetry plane extends.

Related Topics

Process Rule Override Editor: Symmetry Form

Routing Constraint Manager User Interface

Constraint Checker Form

To open the Constraint Checker form, click the *Check* button on the Routing Constraint Manager toolbar. This form lets you specify the options for constraint checking.

Field	Description
Hier Depth	Specifies a value to adjust the depth of checks. The checks are performed on the less than or equal to specified depth value.

Field	**Description**
Selected Nets Only	Performs the constraint checks only on the selected nets.
Per Layer	Checks for the matched length constraint.
Preserver Marker	Checks that the existing markers are not removed.
Report Hierarchy Violations	Checks the constraint down in the hierarchy and reports any existing violations.
Constraint Checks	Lets you select the constraints for which the checks are to be performed.
Checks	Enables all constraint checks when selected. The checks can be performed for the following constraints. ■ *Process Rule Overrides* ■ *Symmetry* ■ *Match* ■ *Shield* You can also perform checks on the selected constraints.

Related Topics

Opening Routing Constraint Manager

Virtuoso Routing Constraint Manager

Routing Constraint Manager User Interface

Checking Routing Constraints

Design Process Rule Override Editor Form

To open the Design CG Editor form, double-click a field in the *Rules* column on the *Rules* tab. The Design CG Editor form has the following components.

Toolbar	Lets you create, copy, and delete a constraint.
Fields	Lets you specify the options for design constraints

Toolbar

The following table lists the functions of the different buttons on the Design CG Editor form toolbar.

Icon	Command	Description
✚	*Create Constraint Group*	Lets you create a new constraint group.
❐	*Copy Constraint Group*	Lets you copy the constraint group.
⊗	*Delete Constraint Group*	Lets you delete the constraint group.

Fields

The Design CG Editor form lets you specify the options for rules constraints. The form contains the following fields.

Field	Description
Default Wire Constraint Group	Lets you select a wire constraint group.
Design Constraint Group	Lets you select a design constraint group.
#Via Cuts	Specifies the number of cuts for each cut layer matching a valid via.
Top Layer	Lets you select the preferred top routing layer.
Bottom Layer	Lets you select the preferred bottom routing layer.
Width/Space	Toggles to display the width and spacing layer table.
Layer	Displays the name of the layer.
Dir	Displays the preferred routing direction of the layer.
Min W	Lets you specify the minimum width to be used for the layer.
Max W	Lets you specify the maximum width to be used for the layer.
Min S	Lets you specify the minimum spacing to be used for the layer.

Related Topics

Opening Routing Constraint Manager

Virtuoso Routing Constraint Manager

Routing Constraint Manager User Interface

Load Constraints Form

Use the Load Constraints form to select the library, cell, and view names from which to load constraints.

Field	Description
Library	Specifies the library name from which to load constraints.
Cell	Specifies the cell name from which to load constraints.
View	Specifies the view name from which to load constraints.
Load Mode	Imports constraint groups from a specified cellview. The import mode can be specified as either *Replace* or *Append*. ■ *Replace*: overrides any already existing groups with same name. ■ *Append*: ignores constraint groups which already exist.

Related Topics

Loading Routing Constraints

Process Rule Override Editor Form

To open the Process Rule Override Editor form, double-click a constraint group. The Process Rule Override Editor form has the following components:

Toolbar	Lets you copy and delete a constraint.
Fields	Lets you specify the options for bus constraints.

Toolbar

The following table lists the functions of the different buttons on the Process Rule Override Editor form toolbar.

Icon	Command	Description
⧉	*Copy Constraint Group*	Lets you copy the constraint group.
⊗	*Delete Constraint Group*	Lets you delete the constraint group.

Fields

The Process Rule Override Editor form lets you specify the options for bus constraints.

Field	Description
Constraint Group Name	Lets you select a constraint group to use for seeding the values.
# Via Cuts	Specifies the number of cuts for each cut layer matching a valid via.
Top Layer	Lets you select the preferred top routing layer.
Bottom Layer	Lets you select the preferred bottom routing layer.
Width/Space	Toggles to display the width and spacing layer table.
Layer	Displays the name of the layer.
Min W	Lets you specify the minimum width to be used for the layer.
Max W	Lets you specify the maximum width to be used for the layer.
Min S	Lets you specify the minimum spacing to be used for the layer.

Related Topics

Opening Routing Constraint Manager

Virtuoso Routing Constraint Manager

Routing Constraint Manager User Interface

Max Resistance Editor Form

To open the Max Resistance Editor form, click the *Apply Max Resistance* button on the Routing Constraint Manager toolbar.

The following table describes the fields available in the Max Resistance Editor form.

Column	Description
From Inst Term	Displays the from-to terminals of the net on which the maximum resistance constraint should apply
To Inst Term	Displays the from-to terminals of the net on which the maximum resistance constraint should apply.
Max Resistance	Displays the value of the maximum resistance.

Related Topics

Opening Routing Constraint Manager

Virtuoso Routing Constraint Manager

Routing Constraint Manager User Interface

Process Rule Override Editor: Bus Form

To open the Process Rule Override: Bus form, double-click a bus constraint in the *Bus* column on the *Nets* tab. Alternatively, double-click a bus constraint in the *Bus* column on the *Bus* tab.

The Process Rule Override: Bus form has the following components:

Toolbar	Lets you copy and delete a constraint.
Fields	Lets you specify the options for bus constraints.

Toolbar

The following table lists the functions of the different buttons on the Process Rule Override: Bus form toolbar.

Icon	Command	Description
⧉	*Copy Constraint Group*	Lets you copy the constraint group.
⊗	*Delete Constraint Group*	Lets you delete the constraint group.

Fields

The Process Rule Override: Bus form lets you specify the options for bus constraints.

Field	Description
Group Name	Lets you specify the group name for the constraint.
Constraint Group Name	Lets you select a constraint group to use for seeding the values.
# Via Cuts	Specifies the number of cuts for each cut layer matching a valid via.
Top Layer	Lets you select the preferred top routing layer.
Bottom Layer	Lets you select the preferred bottom routing layer.
Width/Space	Toggles to display the width and spacing layer table.
Layer	Displays the name of the layer.
Min W	Lets you specify the minimum width to be used for the layer.
Max W	Lets you specify the maximum width to be used for the layer.
Min S	Lets you specify the minimum spacing to be used for the layer.

Related Topics

Opening Routing Constraint Manager

Virtuoso Routing Constraint Manager

Routing Constraint Manager User Interface

Process Rule Override Editor: DiffPair Form

To open the Process Rule Override Editor: DiffPair form, double-click a diffpair constraint in the *DiffPair* column on the *Nets* tab. Alternatively, double-click a diffpair constraint in the *DiffPair* column on the *DiffPair* tab.

The Process Rule Override Editor: DiffPair form has the following components:

Toolbar	Lets you copy and delete a constraint.
Fields	Lets you specify the options for bus constraints.

Toolbar

The following table lists the functions of the different buttons on the Process Rule Override Editor: DiffPair form toolbar.

Icon	Command	Description
⧉	*Copy Constraint Group*	Lets you copy the constraint group.
⊗	*Delete Constraint Group*	Lets you delete the constraint group.

Fields

The Process Rule Override Editor: DiffPair form lets you specify the options for bus constraints.

Field	Description
Group Name	Lets you specify the group name for the constraint.
Constraint Group Name	Lets you select a constraint group to use for seeding the values.
# Via Cuts	Specifies the number of cuts for each cut layer matching a valid via.
Tolerance	Lets you specify absolute tolerance when matching shield lengths.

Field	**Description**
Same Mask	Lets you select the same mask option. The same mask options are determined by the number of masks per layer defined in a given technology file. A minimum of `0` and a maximum of `3` options are available. An empty value in the *Same Mask* field indicates that the group has no same mask constraints. The value *any* appears as an option in the *Same Mask* field regardless of the number of masks defined in the technology file.
Top Layer	Lets you select the preferred top routing layer.
Bottom Layer	Lets you select the preferred bottom routing layer.
Width/Space	Toggles to display the width and spacing layer table.
Layer	Displays the name of the layer.
Min W	Lets you specify the minimum width to be used for the layer.
Max W	Lets you specify the maximum width to be used for the layer.
Min S	Lets you specify the minimum spacing to be used for the layer.

Related Topics

Opening Routing Constraint Manager

Virtuoso Routing Constraint Manager

Routing Constraint Manager User Interface

Process Rule Override Editor: Match Form

To open the Process Rule Override Editor: Match1 form, double-click a match constraint in the *Match* column on the *Nets* tab. Alternatively, double-click a match constraint in the *Match* column on the *Match* tab.

The Process Rule Override Editor: Match form has the following components.

Toolbar	Lets you copy and delete a constraint.
Fields	Lets you specify the options for match constraints.

Toolbar

The following table lists the functions of the different buttons on the Process Rule Override Editor: Match form toolbar.

Icon	Command	Description
⧉	*Copy Constraint Group*	Lets you copy the constraint group.
⊗	*Delete Constraint Group*	Lets you delete the constraint group.

Fields

The Process Rule Override Editor: Match form lets you specify the options for matched length constraints. The form contains the following fields.

Field	Description
Group Name	Lets you specify the group name for the matched length constraint.
Constraint Group Name	Lets you select a constraint group to use to seed the values.
#Via Cuts	Specifies the number of cuts for each cut layer matching a valid via.
Pattern	Lets you select the pattern used to lengthen wires for matched length routing. The available options are: *None*, *Accordion*, *RW Accordion*, *Trombone*, and *End Run*.
Match Per Layer	Determines the length match checks performed by layer or over the entire length of the net.
Match Tolerance	Specifies the absolute tolerance when matching net lengths is expressed as a percentage of total length. The Match Tolerance Value cannot be greater than 100%.
Top Layer	Lets you select the preferred top routing layer.
Bottom Layer	Lets you select the preferred bottom routing layer.
Width/Space	Toggles to display the width and spacing layer table.
Layer	Displays the name of the layer.
Min W	Lets you specify the minimum width to be used for the layer.

Field	Description
Max W	Lets you specify the maximum width to be used for the layer.
Min S	Lets you specify the minimum spacing to be used for the layer.

Related Topics

Opening Routing Constraint Manager

Virtuoso Routing Constraint Manager

Routing Constraint Manager User Interface

Process Rule Override Editor: NetClass Form

To open the Process Rule Override Editor: NetClass form, double-click a net class constraint that appears in the *Net* column on the *Nets* tab.

The Process Rule Override Editor: NetClass form has the following components:

Toolbar	Lets you copy and delete a constraint.
Fields	Lets you specify the options for bus constraints.

Toolbar

The following table lists the functions of the different buttons on the Process Rule Override Editor: Bus form toolbar.

Icon	Command	Description
⧉	*Copy Constraint Group*	Lets you copy the constraint group.
⊗	*Delete Constraint Group*	Lets you delete the constraint group.

Fields

The Process Rule Override Editor: NetClass form lets you specify the options for bus constraints.

Field	Description
Group Name	Lets you specify the group name for the constraint.
Constraint Group	Lets you select a constraint group to use for seeding the values.
# Via Cuts	Specifies the number of cuts for each cut layer matching a valid via.
Same Mask	Lets you select the same mask option. The same mask options are determined by the number of masks per layer defined in a given technology file. A minimum of `0` and a maximum of `3` options are available. An empty value in the *Same Mask* field indicates that the group has no same mask constraints. The value *any* appears as an option in the *Same Mask* field regardless of the number of masks defined in the technology file.
Top Layer	Lets you select the preferred top routing layer.
Bottom Layer	Lets you select the preferred bottom routing layer.
Width/Space	Toggles to display the width and spacing layer table.
Layer	Displays the name of the layer.
Min W	Lets you specify the minimum width to be used for the layer.
Max W	Lets you specify the maximum width to be used for the layer.
Min S	Lets you specify the minimum spacing to be used for the layer.

Related Topics

Opening Routing Constraint Manager

Virtuoso Routing Constraint Manager

Routing Constraint Manager User Interface

Process Rule Override Editor: Symmetry Form

To open the Process Rule Override Editor: Symmetry form, double-click a symmetry constraint in the *Symmetry* column on the *Nets* tab. Alternatively, double-click a symmetry constraint in the *Symmetry* column on the *Bus* tab.

The Process Rule Override Editor: Symmetry form has the following components:

Toolbar	Lets you copy and delete a constraint.
Fields	Lets you specify the options for bus constraints.

Toolbar

The following table lists the functions of the different buttons on the Process Rule Override Editor: Symmetry form toolbar.

Icon	Command	Description
⧉	*Copy Constraint Group*	Lets you copy the constraint group.
⊗	*Delete Constraint Group*	Lets you delete the constraint group.

Fields

The Process Rule Override Editor: Symmetry form lets you specify the options for bus constraints.

Field	Description
Group Name	Lets you specify the group name for the constraint.
Constraint Group	Lets you select a constraint group to use for seeding the values.
# Via Cuts	Specifies the number of cuts for each cut layer matching a valid via.
Tolerance	Lets you specify absolute tolerance when matching shield lengths.

Field	Description
Same Mask	Lets you select the same mask option. The same mask options are determined by the number of masks per layer defined in a given technology file. A minimum of `0` and a maximum of `3` options are available. An empty value in the *Same Mask* field indicates that the group has no same mask constraints. The value *any* appears as an option in the *Same Mask* field regardless of the number of masks defined in the technology file.
Top Layer	Lets you select the preferred top routing layer.
Bottom Layer	Lets you select the preferred bottom routing layer.
Width/Space	Toggles to display the width and spacing layer table.
Layer	Displays the name of the layer.
Min W	Lets you specify the minimum width to be used for the layer.
Max W	Lets you specify the maximum width to be used for the layer.
Min S	Lets you specify the minimum spacing to be used for the layer.

Related Topics

Opening Routing Constraint Manager

Virtuoso Routing Constraint Manager

Routing Constraint Manager User Interface

Shield Process Rule Override Editor Form

To open the Shield Process Rule Override Editor form, double-click a field in the *Shield* column on the *Nets* tab. Alternatively, double-click a field in the *Shield* column on the *Shield* tab.

The Shield Process Rule Override Editor form has the following components.

Toolbar	Lets you copy and delete a constraint.
Fields	Lets you specify the options for shield constraints.

Toolbar

The following table lists the functions of the different buttons on the Shield CG Editor form toolbar.

Icon	Command	Description
⧉	*Copy Constraint Group*	Lets you copy the constraint group.
⊗	*Delete Constraint Group*	Lets you delete the constraint group.

Fields

The Shield Process Rule Override Editor form lets you specify the options for Shielding constraints. The form contains the following fields.

Field	Description
Shield Group Name	Lets you select a pre-existing shielding constraint.
Shield Type	Lets you specify the type of shield to create for the given signal nets. The shielding types are: *None*, *Parallel*, *Tandem*, and *Coaxial*. ■ *None* No shield. ■ *Parallel* Planar shield wires parallel to the signal wire. ■ *Coaxial* Both tandem and parallel shields to surround the signal wire on four sides. ■ *Tandem* Shield wires on a given layer above and below the signal wire.
Shield Enclosure	Lets you select the enclosure type to add shield wires all around vias in the preferred direction. The shield enclosures are: *None*, *Center*, *Half*, and *Full*. The default is *Half*.
Group Shield Type	Lets you select a group shielding type. The group shielding types are *Surround* and *Interleaved*.
Tolerance	Lets you specify absolute tolerance when matching shield lengths.
Valid Shield Layers	Lets you select the layers to prevent shields from being added on the selected layers.
Shield Width/Space	Toggles to display the width and spacing layer table.

Field	**Description**
Layer	Displays the name of the layer.
Min W	Lets you specify the minimum width to be used for the layer.
Min S	Lets you specify the minimum spacing to be used for the layer.
Advanced Shield	
Share Shields	Lets you select whether or not shield routes can be shared by nets.
Use Existing Shapes	Lets you select whether or not to use the existing shapes.
Shield Terminal	Lets you select whether or not shield terminals should be connected.
Shield Minimum Length	Specifies whether or not to prevent shielding of any signal wire that is shorter than the given length.
Tie Shield	Specifies whether ties should be added to tie the new shield wires to the shield nets they belong to.
Frequency	Specifies the maximum distance between ties that must be inserted to tie the new shield wires to their respective shield nets.
Shield Redundant Via	Adds redundant vias to tie shield wires to shield nets at each location where the shield wires overlap their respective existing power and ground rails.
Shield Via #Cuts	The minimum cuts constraint for each cut layer matching a valid via for the shielding wires.

Related Topics

Opening Routing Constraint Manager

Virtuoso Routing Constraint Manager

Routing Constraint Manager User Interface

Specify Routed Cellview

The Specify Routed Cellview lets you specify a cellview that you want to route.

Field	Description
Library	Lets you select a library.
Cell	Lets you select a cell to be routed. You can also specify a non existing name.
View	Lets you select a view for the selected library and cell. You can also specify a non existing name.

Related Topics

Routing Assistant

Accessing the Routing Assistant

Routing Assistant User Interface for Chip Assembly Routing Flow

Virtuoso Pre-Route Browser

The Virtuoso Pre-Route Browser is a standalone router setup utility that is independent of the router and design style. It is used for identifying order and preferences of nets for routing.

The Pre-Route Browser has the following components:

Pre-Route Browser Toolbar	Lets you access the button to show the type of nets for which results are to be displayed.
Results Table	Displays the routing results for various nets.

Pre-Route Browser Toolbar

The following table describes the Pre-Route Browser toolbar buttons.

Icon	Command	Description
⇄	*Toggle Pre/Post Data*	Lets you toggle between the pre and post routing data.

Icon	Command	Description
	Save Table Content	Saves the content of the results table in the `Routing_results.xml` file.
	Load Columns	Lets you load data for custom columns. Pre-defined column data is not altered.
	Show Routed Nets	Toggles display of the routed nets in the design.
	Show Unrouted Nets	Toggles display of the open nets in the design.
	Show Nets With Pin Count = 1	Toggles display of the nets with pins count as one.
	Show Grouped Nets	Displays only the grouped nets.
	Snapshot Selected	Toggles display of the selected nets in the design. If the *Show Selected Nets Only* is switched ON then the table displays only the nets that are currently selected in the Navigator assistant or layout canvas.
	Synchronize Snapshot	Synchronizes the net selection in Routing Results Browser with the ones that are currently selected in the Navigator assistant or layout canvas This option Sync works in conjunction with the Snapshot Selected option. Once you have created a snapshot of selected nets, the *Synchronize Snapshot* button updates the snapshot to match what is now selected in the Navigator assistant or layout canvas.
	Select All	Selects all the nets that are displayed in the Routing Results Browser.
	Deselect All	Deselects all the nets that are displayed in the Routing Results Browser.
	Create Group/Add Nets	Creates a group for the selected nets.
	Delete Group/Remove Nets	Deletes the selected group or the group on the selected net.
	Rename Group	Lets you change the name of the net group.

Icon	Command	Description
↻	*Refresh Table*	Refreshes the results table in the Pre-Route Browser.
	Ideal Wire Length Computation	Lets you select the method or algorithm for calculating the ideal wire length. The different methods or algorithms of routing are: MST, Steiner, and Spine.

Results Table

The following table describes the columns displayed in Routing Results Browser.

Column	Description
Net	Displays the name of the net.
Group	Displays the name of the net group.
Sets	Displays general navigator net sets. It does not have mutual exclusivity requirement of groups.
Opens	Display the number of opens for each net in the design.
Shorts	Displays the number of shorts for each net in the design.
Trunks	Displays the number of trunks on a net.
Ideal Wirelength	Displays the ideal length of the wire.
Wirelength	Displays the actual length of the wire.
IO Pin Edge	Displays the edge of the IO pin on the net.
IO Pin Layer Unique	Displays the layer name for the net on which IO pin exists.
Pin Count	Displays the total number of pins for each net in the design.
Manhattan X	It is the x-distance of the union box of all the nets pin boxes.
Manhattan Y	It is the x-distance of the union box of all the nets pin boxes.

Related Topics

Routing Assistant

Accessing the Routing Assistant

Routing Assistant User Interface for Chip Assembly Routing Flow

Virtuoso Routing Results Browser

The Virtuoso Routing Results Browser shows routing information for each net in the design. It is a read-only dialog box. The table in the Routing Results Browser has all the same controls as Virtuoso Routing Constraint Manager in terms of sorting, filtering, and selection.

The Routing Results Browser has the following components:

Routing Results Browser Toolbar	Lets you access the button to show the type of nets for which results are to be displayed.
Results Table	Displays the routing results for various nets.

Routing Results Browser Toolbar

The following table describes the Routing Results Browser toolbar buttons.

Icon	Command	Description
	Save Table Content	Saves the content of the results table in the `Routing_results.xml` file.
	Show Routed Nets	Toggles display of the routed nets in the design.
	Show Unrouted Nets	Toggles display of the open nets in the design.
	Show Signal Nets	Toggles display of the signal nets in the design.
	Show Supply Nets	Toggles display of the supply or power nets in the design.
	Snapshot Selected	Toggles display of the selected nets in the design. If the *Snapshot Selected* is switched ON then the table displays only the nets that are currently selected in the Navigator assistant or layout canvas.

Icon	Command	Description
	Synchronize Snapshot	Synchronizes the net selection in Routing Results Browser with the ones that are currently selected in the Navigator assistant or layout canvas This option synchronizes in conjunction with the *Snapshot Selected* option. Once you have created a snapshot of selected nets, the *Synchronize Snapshot* button updates the snapshot to match what is now selected in the Navigator assistant or layout canvas.
	Select All	Selects all the nets that are displayed in the Routing Results Browser.
	Deselect All	Deselects all the nets that are displayed in the Routing Results Browser.
	Refresh Table	Refreshes the results table in the Routing Results Browser.
	Ideal Wire Length Computation	Lets you select the method or algorithm for calculating the ideal wire length. The different methods or algorithms of routing are: MST, Steiner, and Spine.

Results Table

The following table describes the columns displayed in Routing Results Browser.

Column	Description
Net	Displays the name of the net.
Rule Violations	Displays the number of rule violations.
Symmetry Violations	Displays the number of symmetry violations.
Matched Length Violations	Displays the number of matched length violations.
Shield Violations	Displays the number of shield violations.
Opens	Display the number of opens for each net in the design.

Shorts	Displays the number of shorts for each net in the design.
DRCs	Displays the number of design rule check errors for each net in the design.
Wirelength	Displays the total length of the routing for each net.
Vias	Displays the total number of vias used to route the net.
Ratio	Displays the ratio of the estimated routing length (spine style) versus the actual routing length.
Pin count	Displays the total number of pins for each net in the design.
Manhattan X	It is the x-distance of the union box of all the nets pin boxes.
Manhattan Y	It is the y-distance of the union box of all the nets pin boxes.

Related Topics

Routing Assistant

Accessing the Routing Assistant

Routing Assistant User Interface for Chip Assembly Routing Flow