IC Compiler™
Classic Router
User Guide

Version H-2013.03, March 2013

SYNOPSYS®

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at http://www.synopsys.com/Company/Pages/Trademarks.aspx.
All other product or company names may be trademarks of their respective owners.

Synopsys, Inc.
700 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

Copyright Statement for the Command-Line Editing Feature

Copyright Statement for the Line-Editing Library

Contents

Preface

This preface includes the following sections:

About This Guide

The Synopsys IC Compiler™ tool provides a complete netlist-to-GDSII or netlist-to-clock-tree-synthesis design solution, which combines proprietary design planning, physical synthesis, clock tree synthesis, and routing for logical and physical design implementations throughout the design flow.

This guide describes the usage of the classic router in the IC Compiler implementation and integration flow. The flow is described in the *IC Compiler Implementation User Guide*, which also describes the usage of Zroute in the flow. For information about defining the routing design rules, see the *IC Compiler Technology File and Routing Rules Reference Manual*.

Audience

This user guide is for design engineers who use the IC Compiler tool to implement designs.

To use the IC Compiler tool, you need to be skilled in physical design and design synthesis and be familiar with the following:

- Physical design principles
- The Linux or UNIX operating system
- The tool command language (Tcl)

Related Publications

For additional information about the IC Compiler tool, see the documentation on SolvNet at the following address:

https://solvnet.synopsys.com/DocsOnWeb

You might also want to see the documentation for the following related Synopsys products:

- Design Compiler®
- Milkyway Environment™
- IC Validator
- Hercules™

Release Notes

Information about new features, changes, enhancements, known limitations, and resolved Synopsys Technical Action Requests (STARs) is available in the *IC Compiler Release Notes* in SolvNet.

To see the *IC Compiler Release Notes*,

1. Go to the Download Center on SolvNet located at the following address:

   https://solvnet.synopsys.com/DownloadCenter

2. Select IC Compiler, and then select a release in the list that appears.

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
`Courier`	Indicates syntax, such as `write_file`.
`Courier italic`	Indicates a user-defined value in syntax, such as `write_file` *`design_list`*.
`Courier bold`	Indicates user input—text you type verbatim—in examples, such as `prompt>` **`write_file top`**
[ ]	Denotes optional arguments in syntax, such as `write_file [-format` *`fmt`*`]`
...	Indicates that arguments can be repeated as many times as needed, such as *`pin1 pin2 ... pinN`*
|	Indicates a choice among alternatives, such as `low | medium | high`
Ctrl+C	Indicates a keyboard combination, such as holding down the Ctrl key and pressing C.
\	Indicates a continuation of a command line.
/	Indicates levels of directory structure.
Edit > Copy	Indicates a path to a menu command, such as opening the Edit menu and choosing Copy.

Customer Support

Customer support is available through SolvNet online customer support and through contacting the Synopsys Technical Support Center.

Accessing SolvNet

SolvNet includes a knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. SolvNet also gives you access to a wide range of Synopsys online services including software downloads, documentation, and technical support.

To access SolvNet, go to the following address:

https://solvnet.synopsys.com

If prompted, enter your user name and password. If you do not have a Synopsys user name and password, follow the instructions to register with SolvNet.

If you need help using SolvNet, click HELP in the top-right menu bar.

Contacting the Synopsys Technical Support Center

If you have problems, questions, or suggestions, you can contact the Synopsys Technical Support Center in the following ways:

- Open a support case to your local support center online by signing in to SolvNet at https://solvnet.synopsys.com, clicking Support, and then clicking “Open A Support Case.”
- Send an e-mail message to your local support center.
  - E-mail support_center@synopsys.com from within North America.
  - Find other local support center e-mail addresses at http://www.synopsys.com/Support/GlobalSupportCenters/Pages
- Telephone your local support center.
  - Call (800) 245-8005 from within North America.
  - Find other local support center telephone numbers at http://www.synopsys.com/Support/GlobalSupportCenters/Pages

1

Introduction to the Classic Router

The classic router is the original IC Compiler router and can be used for technology nodes at or above 45 nm. It is the router in the IC Compiler-XP package and is an optional router in the IC Compiler, IC Compiler-DP, and IC Compiler-PC packages.

Note:

Use either the classic router or Zroute to route your design; do not mix the use of routers in your design flow. Although running Zroute on your design is possible after running the classic router, you should not run the classic router on your design after using Zroute. If you run the classic router after Zroute, the tool issues the following message:

```
Error: It is not recommended to use classic router commands after
Zroute commands. (RT-302)
```

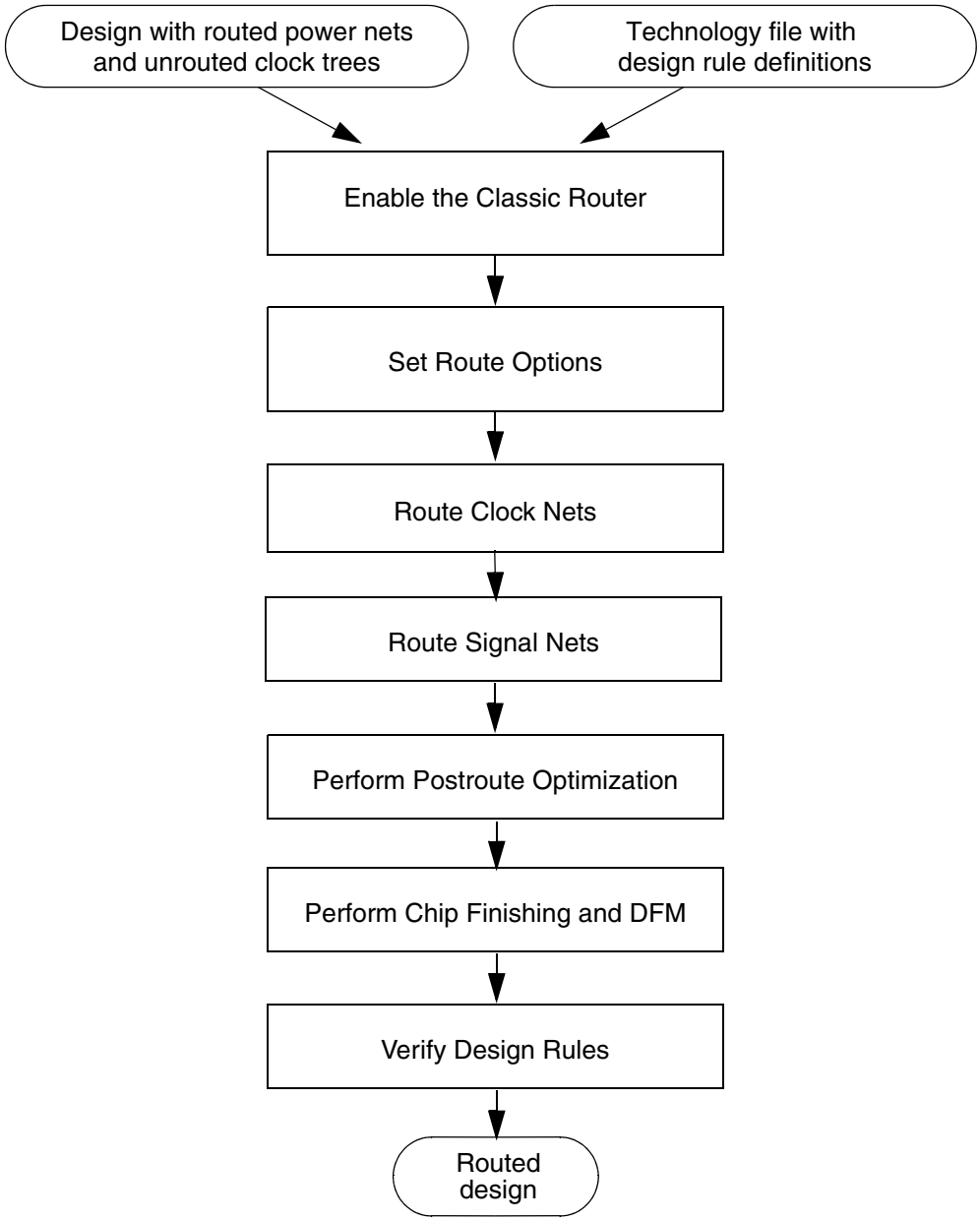
To check which router was used on your design, use the `is_zrt_routed_design` command, which returns `false` if the design was routed with the classic router and `true` if the design was routed with Zroute.

This chapter contains the following sections:

Basic Routing Flow

Figure 1-1 shows the basic routing flow, which includes clock routing, signal routing, chip finishing and DFM optimizations, and route verification.

Figure 1-1 Basic Routing Flow



Prerequisites for Routing

Before you can perform routing, your design must meet the following conditions:

- Power and ground nets have been routed after design planning and before placement.

  For more information, see the *IC Compiler Design Planning User Guide*.

- Clock tree synthesis and optimization have been performed.

  For more information, see Chapter 5, “Clock Tree Synthesis,” in the *IC Compiler Implementation User Guide*.

- Estimated congestion is acceptable.
- Estimated timing is acceptable (about 0 ns of slack).
- Estimated maximum capacitance and transition have no violations.

To verify that your design meets the last three prerequisites, you can check the routability of its placement as explained in the following section, “Checking Routability.”

Checking Routability

After placement is completed, you can have the tool check whether your design is ready for detail routing. The tool checks pin access points, cell instance wire tracks, pins out of boundaries, minimum grid and pin design rules, and blockages to make sure they meet design requirements. It creates an error file named after the top cell in your design (*top_cell_name*.err), with a list of violations that you should correct before performing detail routing.

To verify that your design is ready for detail routing, use the `check_routeability` command (or choose Route > Check Routability in the GUI).

After you run the `check_routeability` command, you can use the `report_error_coordinates` command to report the location of each error. You can also use the error browser to examine the errors in the GUI. For more information about the error browser, see the “Examining Routing and Verification Errors” section in Appendix A of the *IC Compiler Implementation User Guide* and the “Examining Routing and Verification Errors” topic in IC Compiler Help. To view IC Compiler Help, choose Help > IC Compiler Online Help in the GUI.

2

Setting Up for Routing

This chapter describes the setup tasks you must complete before running classic router commands. It contains the following sections:

Enabling the Classic Router

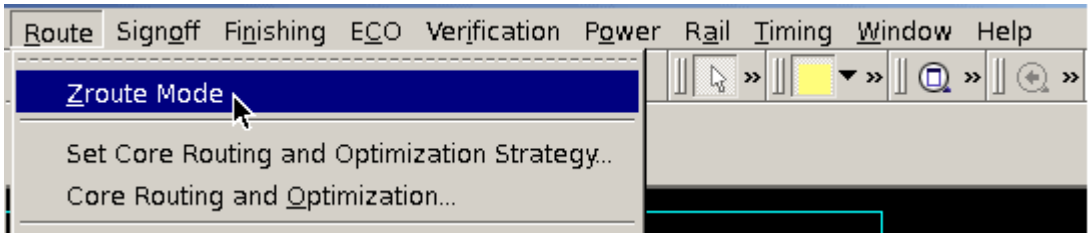
To enable the classic router, set the Zroute route mode option to `false`. The setting for the Zroute mode option is saved in the Milkyway design database.

If you are using the command line, enter the following command:

```
icc_shell> set_route_mode_options -zroute false
```

If you are using the GUI, deselect Zroute Mode in either the Route or Finishing menu, as shown in Figure 2-1.

Figure 2-1 Enabling the Classic Router in the GUI



Note:

If you are using the IC Compiler-XP package, the classic router is the default router. You do not need to enable it explicitly.

When you enable the classic router, the following commands use the classic router instead of Zroute:

- `estimate_fp_area`
- `report_congestion`
- `place_opt -congestion` (when the `placer_enable_enhanced_router` variable is `true`)
- `create_placement -congestion` (when the `placer_enable_enhanced_router` variable is `true`)
- `clock_opt`
- `route_opt`
- `optimize_clock_tree` (when run on a postroute design)
- `signoff_opt`

Note:

In addition to these commands that can use either Zroute or the classic router, there are several commands that are specific to the classic router.

Specifying Route Guides

You can create or remove route guides that do the following for a specific area of your design:

- Prevent routing for signal or prerouted nets, either completely or partially
- Change the wiring direction
- Control wiring density
- Fix violations

Note:

The blockage, pin, and via (BPV) extraction process also creates route guides. For more information about creating route guides during BPV, see the *Library Data Preparation for IC Compiler User Guide*.

To create a route guide, use the `create_route_guide` command (or choose Floorplan > Create Route Guide in the GUI). Table 2-1 defines the `create_route_guide` options. For more information about these options, see the man page.

Table 2-1 create_route_guide Command Options

Command option	Description
`-name` *name* (Name box in the GUI)	Specifies the name of the route guide.
`-repair_as_single_sbox` ("Repair as single SBox" check box in the GUI)	Enables fixing of violations.
`-no_signal_layers` *layers* ("No signal route on layers" check box and list in the GUI)	Prevents the routing of signal wires on the specified layers.
`-zero_min_spacing` ("Zero min-spacing" check box in the GUI)	Allows the routing of wires within the keepout margin of the route guide.
`-no_preroute_layers` *layers* ("No automatic preroutes on layers" check box and list in the GUI)	Prevents automatic preroutes on the specified layers.

Table 2-1 create_route_guide Command Options (Continued)

Command option	Description
`-preferred_direction_only_layers` *layers* (“Layers with preferred direction only” check box and list in the GUI)	Specifies the layers that must be routed in the preferred direction.
`-horizontal_track_utilization` *percentage* (Horizontal box in “Route track utilization” section in the GUI)	Specifies the allowable horizontal track utilization.
`-vertical_track_utilization` *percentage* (Vertical box in “Route track utilization” section in the GUI)	Specifies the allowable vertical track utilization.
`-coordinate` *rectangle* (Coordinates box in the GUI)	Specifies the coordinates for the route guide. In the GUI, you can type the coordinates in the dialog box or draw the rectangle in the layout view. You can also specify that the route guide should snap to the minimum grid, placement site, routing track (the default), middle routing track, or user grid. From the command line, the snapping is controlled by the `set_object_snap_type` command.
`-switch_preferred_direction` (“Switch preferred direction” check box in the GUI)	Switches the preferred wiring direction.

Note:

The classic router does not support the `-single_layer_routing` option. If you specify this option, the classic router ignores it.

To find route guides, use the `get_route_guides` command. For example, to get all the route guides in your design, enter

```
icc_shell> get_route_guides *
```

You can also find route guides by using a filter expression (`-filter` *expression*) or by using rectangular areas that encompass (`-within {{`*x1 y1*`} {`*x2 y2*`}}`) or touch (`-touch {{`*x1 y1*`} {`*x2 y2*`}}`) the guides.

To remove route guides, use the `remove_route_guide` command. You can remove a collection of route guides, such as that returned by the `get_route_guides` command; all route guides (`-all`); or a specific named route guide (`-name`).

For more information about the `get_route_guides` or `remove_route_guide` command, see the man page.

Defining Routing Blockages

A routing blockage defines a region where no routing is allowed on a specific layer. You define routing blockages by using the `create_routing_blockage` command. To define a routing blockage, you must specify

- The blockage layers to which the routing blockage applies

  Use the `-layers` option to specify the blockage layers. Valid values for the blockage layers are metal1Blockage-metal15Blockage, via1Blockage-via14Blockage, polyBlockage, and polyContBlockage. To query the Milkyway design library for the blockage layers, use the `get_layers -include_system` command.

  You can specify one or more blockage layers. If the routing blockage is defined on a metal blockage layer, it is a metal routing blockage. If the routing blockage is defined on a via blockage layer, it is a via routing blockage.

- The region of the blockage

  Use the `-bbox` option to specify the region for a rectangular routing blockage. Use the `-boundary` option to specify the region for a rectilinear routing blockage.

For example, to create rectangular routing blockages on the metal1 and via1 blockage layers, enter the following command:

```
icc_shell> create_routing_blockage \
   -layers {metal1Blockage via1Blockage} -bbox {x1 y1 x2 y2}
```

When the tool creates routing blockages, it assigns a name of RB_*id* to each routing blockage.

To find routing blockages, use the `get_routing_blockages` command. For example, to get all the routing blockages in your design, enter

```
icc_shell> get_routing_blockages *
```

You can also find routing blockages by using a filter expression (`-filter` *expression*) or by using rectangular areas that encompass (`-within {{`*x1 y1*`} {`*x2 y2*`}}`) or touch (`-touch {{`*x1 y1*`} {`*x2 y2*`}}`) the blockages.

To remove routing blockages, use the `remove_routing_blockage` command.

Setting the Preferred Routing Direction

Use the `set_preferred_routing_direction` command to reset and override the default preferred routing direction specified in the library or the design for a specific layer. The layer direction set with this command applies to the current design only.

The command syntax is

```
set_preferred_routing_direction
   -layers list_of_layers
   -direction horizontal | vertical
```

For example, to set the preferred routing direction to vertical for layer M5 and M7, enter

```
icc_shell> set_preferred_routing_direction -layers "M5 M7" \
   -direction vertical
```

Note:

Settings made with the `create_route_guide -switch_preferred_direction` command, which changes the preferred direction within the area that is covered by the route guide, override the settings made with the `set_preferred_routing_direction` command.

Use the `report_preferred_routing_direction` command to report the preferred routing direction for all the routing layers. The report lists the library and user-defined routing directions, as well as the direction that the tool will use. Example 2-1 shows an example report.

Example 2-1 Preferred Direction Report

```
****************************************
  Report : Layers
  Design : core_chip
  Version: Y-2006.06
  Date   : Thu Mar 23 03:43:06 2006
****************************************
         Layer Name         Library             Design              Tool understands
         metal1             Horizontal          Not Set             Horizontal
         metal2             Vertical            Not Set             Vertical
         metal3             Horizontal          Not Set             Horizontal
         metal4             Vertical            Not Set             Vertical
         metal5             Horizontal          Vertical            Vertical
         metal6             Vertical            Vertical            Vertical
```

Use the `remove_preferred_routing_direction` command to remove the user-defined directions for a specific layer from the design.

The command syntax is

```
remove_preferred_routing_direction -layers list_of_layers
```

Using Nondefault Routing Rules

The classic router supports the use of nondefault routing rules, both for routing and for shielding. Before you can use a nondefault routing rule, you must define it. The following sections describe how to define and apply nondefault routing rules.

Defining Nondefault Routing Rules

You define nondefault routing rules for specific nets by using the `define_routing_rule` command (or by choosing Route > Routing Setup > Define Routing Rule in the GUI). These rules define wire width and spacing rules and via types. Table 2-2 defines the `define_routing_rule` options. For more information about these options, see the man page.

Table 2-2 define_routing_rule Command Options

Command option	Description
rule_name (“Rule name” box in the GUI)	Specifies the name of the rule. This argument is required.

`-reference_rule_name`
`-default_reference_rule`
(“Reference rule” box in the GUI)
	Specifies the name of the reference rule.
	This option is required.

`-taper_level` (“Taper level” box in the GUI)	Specifies the tapering level. The default tapering level is 0.
`-taper_distance` (“Taper distance” in the GUI)	This option is not used by the classic router. To specify the tapering distance for the classic router, use the `droute_pinTaperLengthLimit` variable.
`-multiplier_width` (“Width multiplier” box in the GUI)	Specifies the layer width multiplier.
`-multiplier_spacing` (“Spacing Multiplier” box in the GUI)	Specifies the layer spacing multiplier.
`-shield` (“Specify shielding” check box in the GUI)	Defines shielding with default spacing and default width.

Table 2-2 define_routing_rule Command Options (Continued)

Command option	Description
`-snap_to_track` (“Snap shielding to track” check box in the GUI)	Snaps shielding wires to the track when selected. By default, snapping is not enabled.

`-widths`
`-spacings`
`-shield_widths`
`-shield_spacings`
(“Metal table” section in the GUI)
	Specifies the width and spacing rules. You can specify only a single width and spacing rule per layer.
	If you select “Specify shielding,” the Width and Spacing columns in the “Metal table” section in the GUI refer to shield width and shield spacing; otherwise, they refer to wire width and wire spacing.

`-via_cuts` (“Via table” section in the GUI)	Specifies the via types for the nondefault routing rule. You can specify only a single via type per layer.
`-cuts` (“Cut table” section in the GUI)	Specifies the general cut definitions for the nondefault routing rule. The tool automatically selects the via types based on the cut definition and the technology file. The classic router uses only the first selected via type.

For example, to define a nondefault routing rule named new_rule that uses the default routing rule as the reference rule and defines nondefault width and spacing rules for metal1 and metal4, enter the following command:

```
icc_shell> define_routing_rule new_rule -default_reference_rule \
   -widths {m1 0.8 m4 0.9} -spacings {m1 1.0 m4 1.0}
```

Applying Nondefault Routing Rules

The IC Compiler tool provides two commands for assigning nondefault routing rules to nets:

- `set_clock_tree_options`

  This command assigns nondefault routing rules to clock nets before clock tree synthesis.

- `set_net_routing_rule`

  This command assigns nondefault routing rules to signal nets and to clock nets after clock tree synthesis.

Applying Nondefault Routing Rules to Clock Nets

To assign a nondefault routing rule to clock nets, use the `-routing_rule` option of the `set_clock_tree_options` command.

Note:

This command works only before clock tree synthesis. If the clock tree has already been synthesized, use the `set_net_routing_rule` command to apply the nondefault routing rules, as described in "Applying Nondefault Routing Rules to Signal Nets" on page 2-9.

For example, to assign a shielding rule called shield_rule to the nets in the CLK clock tree before clock tree synthesis, enter the following command:

```
icc_shell> set_clock_tree_options -clock_trees CLK \
   -routing_rule shield_rule
```

By default, the shielding rule applies to all nets in the clock tree. To use the default routing rule for the nets closest to the sink pin, use the `-use_default_routing_for_sinks` option. The default routing rules are used for the specified number of clock tree levels closest to the clock sink. For more information about the `set_clock_tree_options` command, see Chapter 5, "Clock Tree Synthesis," in the *IC Compiler Implementation User Guide*.

Applying Nondefault Routing Rules to Signal Nets

To apply a nondefault routing rule to one or more nets, use the `set_net_routing_rule` command (or choose Route > Routing Setup > Set Net Routing Rule in the GUI). Table 2-3 defines the `set_net_routing_rule` options. For more information about these options, see the man page.

Table 2-3 set_net_routing_rule Command Options

Command option	Description
list_of_nets (Nets box in the GUI)	Specifies the nets to which to apply the rule. This argument is required.
-rule *rule_name* ("Routing rule" box in the GUI)	Specifies the name of the nondefault rule to apply. This option is required.
-top_layer_probe AnyPort | OutPort | AllPort ("Top layer probe mode" box in the GUI)	Specifies which ports to use for top-layer probing. The default is AnyPort.

Table 2-3 set_net_routing_rule Command Options (Continued)

Command option	Description
`-reroute normal | minorchange | freeze` (“Net re-routability” box in the GUI)	Specifies when to reroute nets. The default is `normal`.
`-timing_driven_spacing` (“Timing driven spacing” check box in the GUI)	When timing-driven spacing is enabled, the router adds space between critical nets and nets that are parallel to them to reduce the intralayer coupling capacitances. By default, timing-driven spacing is not enabled.

Reporting Nondefault Routing Rules

To report the nondefault routing rules for specific nets, use the `report_net_routing_rules` command.

```
icc_shell> report_net_routing_rules [get_nets *]
```

To output a Tcl script that contains the `define_routing_rule` commands used to define the nondefault routing rules for the specified nets, use the `-output` option when you run the `report_net_routing_rules` command.

The name of the nondefault routing rule for a net is stored in the `var_route_rule` net attribute. You can access the value of this attribute by using the `get_attribute` command.

To report the design-specific nondefault routing rules defined by the `define_routing_rule` command, use the `report_routing_rules` command. By default, this command reports all of the nondefault routing rules for the current design. To limit the report to a specific nondefault routing rule, specify the rule name as an argument to the command.

```
icc_shell> report_routing_rules rule_name
```

To output a Tcl script that contains the `define_routing_rule` commands used to define the specified nondefault routing rule or all nondefault routing rules for the design if you do not specify a routing rule, use the `-output` option when you run the `report_routing_rules` command.

Specifying the Routing Layers

The IC Compiler tool lets you specify which layers can be ignored for routing and which layers are to be ignored for RC and congestion estimation. You can also specify the routing layers to use for specific nets (net layer constraints).

By default, when you set a maximum routing layer, it is a hard constraint. You can change it to a soft constraint or you can allow the use of higher layers only for pin connections by setting the `hardMaxLayerConx` detail route option. This option applies to both ignored layers (`set_ignored_layers` command) and net layer constraints (`set_net_routing_layer_constraints` command).

By default, when you set a minimum routing layer, it is a soft constraint. You can change it to a hard constraint or you can allow the use of lower layers only for pin connections by setting the `hardMinLayerConx` detail route option. This option applies to both ignored layers (`set_ignored_layers` command) and net layer constraints (`set_net_routing_layer_constraints` command).

Specifying the Ignored Layers

To specify the ignored layers, use the `set_ignored_layers` command (or choose Route > Routing Setup > Set Ignored Layers in the GUI). By default, RC estimation and congestion analysis use the same layers as routing.

To specify the minimum and maximum routing layers, use the `-min_routing_layer` and `-max_routing_layer` options, respectively. If you use only these options, the same layers are used for routing, RC estimation, and congestion analysis. For example, to use layers M2 through M7 for routing, RC estimation, and congestion analysis, use the following command:

```
icc_shell> set_ignored_layers \
   -min_routing_layer M2 -max_routing_layer M7
```

To use fewer layers for RC estimation and congestion analysis than those used for routing, use the `-rc_congestion_ignored_layers` option to specify the layers to be ignored for RC estimation and congestion analysis. For example, to use layers M2 through M7 for routing and layers M3 through M7 for RC estimation and congestion analysis, use the following command:

```
icc_shell> set_ignored_layers \
   -min_routing_layer M2 -max_routing_layer M7 \
   -rc_congestion_ignored_layers {M1 M2 M8 M9}
```

To use more layers for RC estimation and congestion analysis than for routing, use the `remove_ignored_layers` command to remove the layer restriction for RC estimation and

congestion analysis. For example, to use layers M2 through M7 for routing and layers M2 through M8 for RC estimation and congestion analysis, use the following commands:

```
icc_shell> set_ignored_layers \
   -min_routing_layer M2 -max_routing_layer M7 \
icc_shell> remove_ignored_layers M8
```

To report the ignored layers, use the `report_ignored_layers` command.

Specifying Routing Layers for Specific Nets

To specify the routing layers for specific nets, use the `set_net_routing_layer_constraints` command (or choose Route > Routing Setup > Set Net Layer Constraints in the GUI).

To specify the minimum and maximum routing layers for specific nets, use the `-min_layer_name` and `-max_layer_name` options, respectively. For example, to use layers M2 through M7 for routing net n1, use the following command:

```
icc_shell> set_net_routing_layer_constraints [get_nets n1] \
   -min_layer_name M2 -max_layer_name M7
```

To report the routing layer constraints for specific nets, use the `report_net_routing_layer_constraints` command.

```
icc_shell> report_net_routing_layer_constraints [get_nets *]
```

To output a Tcl script that contains the `set_net_routing_layer_constraints` commands that are used to define the routing layer constraints for the specified nets, use the `-output` option when you run the `report_net_routing_layer_constraints` command.

Setting Routing Types

For debugging, you can set or change the routing types of wires, vias, or paths to indicate which classic router routing engine you want to route them with.

To specify a routing of any type, use the `set_route_type` command (or choose Route > Routing Setup > Set Route Type in the GUI).

- For signal nets, you can specify the following route types: `detail_route` or `user`.

  ```
  icc_shell> set_route_type -signal type objects
  ```

- For clock nets, you can specify the following route types: `ring`, `strap`, `tie_off`, or `user`.

  ```
  icc_shell> set_route_type -clock type objects
  ```

- For power and ground nets, you can specify the following route types: `ring`, `strap`, `tie_off`, `user`, `std_cell_pin_conn`, or `macro/IO_pin_conn`.

  ```
  icc_shell> set_route_type -pg type objects
  ```

To remove a routing type, use the `remove_route_by_type` command (or choose Route > Delete Route in the GUI).

Setting Routing Options

You can specify options that control how the classic router performs global routing, track assignment, and detail routing, as well as miscellaneous routing options.

- To set routing options, use the `set_route_options` command (or choose Route> Routing Setup > Set Route Options in the GUI). To reset the options to their default values, use `set_route_options -default` (or click Default in the GUI).
- To report the settings of all routing options, use the `report_route_options` command.
- To generate a Tcl script that contains the `set_route_options` commands that are used to define the current routing option settings, use the `-output` option when you run the `report_route_options` command.

Note:

You can set additional global route options by using the `set_groute_options` command. You can set additional detail route options by using the `set_droute_options` command. For more information about these commands, see the man pages.

Table 2-4 lists the routing options set by the `set_route_options` command.

Table 2-4 set_route_options Command Options

Option	Valid values	Description
Global routing options (Global Routing tab in the GUI)		
`-groute_skew_control` (“Skew control” check box in the GUI)	`true | false`	Enables (true) or disables (false) skew control during global routing. The default is `false`.
`-groute_skew_weight` (“Skew control Weight” box in the GUI)	*int* (must be between 1 and 10)	Specifies the weight associated with skew control. The default is 5.
`-groute_timing_driven` (“Timing driven” check box in the GUI)	`true | false`	Enables (true) or disables (false) timing-driven global routing. The default is `false`.
`-groute_timing_driven_weight` (“Timing driven Weight” box in the GUI)	*int* (must be between 1 and 7)	Specifies the weight associated with timing-driven global routing. The default is 4.
`-groute_congestion_weight` (“Congestion weight” box in the GUI)	*int* (must be between 1 and 12)	Specifies the weight associated with congestion-driven global routing. The default is 4.
`-groute_clock_routing` (“Clock routing” radio buttons in the GUI)	`normal | comb | balanced`	Specifies the global-routing clock topology. The default is `balanced`.
`-groute_incremental` (Incremental check box in the GUI)	`true | false`	Enables (true) or disables (false) incremental global routing. The default is `false`.

Table 2-4 set_route_options Command Options (Continued)

Option	Valid values	Description
Track assignment options (Track Assign tab in the GUI)		
`-track_assign_timing_driven` (“Timing driven” check box in the GUI)	`true | false`	Enables (true) or disables (false) timing-driven track assignment. The default is `false`.
`-track_assign_timing_driven_weight` (“Timing driven” Weight box in the GUI)	*int* (must be between 1 and 10)	Specifies the weight associated with timing-driven track assignment. The default is 1.
Detail routing options (Detail Routing tab in the GUI)		
`-droute_connect_tie_off` (“Connect tie off” check box in the GUI)	`true | false`	Enables (true) or disables (false) connection of tie-off nets during detail routing. The default is `true`.
`-droute_connect_open_nets` (“Connect open nets” check box in the GUI)	`true | false`	Enables (true) or disables (false) connection of open nets during detail routing. The default is `true`.
`-droute_reroute_user_wires` (“Reroute user wires” check box in the GUI)	`true | false`	Specifies whether the router can reroute user-created wires. The default is `false`.
`-droute_CTS_nets` (“Change CTS nets” radio buttons in the GUI)	`normal | minor_change_only`	Specifies whether only minor changes can be made to clock nets. The default is `minor_change_only`.
`-droute_single_row_column_via_array` (“Single row column via array” radio buttons in the GUI)	`center | optimize`	Specifies how to handle via arrays that consist of a single row and single column. The default is `center`.

Table 2-4 set_route_options Command Options (Continued)

Option	Valid values	Description
`-droute_stack_via_less_than_min_area` ("Stack via less than min area" radio buttons in the GUI)	`forbid` | `add_metal_stub`	Specifies how to handle stacked via arrays that do not meet the minimum area rule. The default is `add_metal_stub`.
`-droute_stack_via_less_than_min_area_cost` ("Stack via less than min area Cost" box in the GUI)	*int* (must be between 1 and 10)	Controls the cost associated with adding metal stubs. The default is 0.
Miscellaneous options (Miscellaneous tab in the GUI)		
`-poly_pin_access` ("Poly pin access" radio buttons in the GUI	`auto` | `off`	Controls poly pin access is handled. The default is `auto`.
`-drc_distance` ("DRC distance" radio buttons in the GUI)	`diagonal` | `manhattan`	Specifies how to measure distances for design rule checking. The default is `diagonal`.
`-same_net_notch` ("Same net notch" radio buttons in the GUI)	`ignore` | `check_and_fix`	Specifies whether to check the same net notch rule. The default is `ignore`.
`-fat_wire_check` ("Fat wire checking" radio buttons in the GUI)	`quick` | `merge_then_check`	Controls how the fat wire check is handled. The default is `merge_then_check`.
`-merge_fat_wire_on` ("Merge fat wire during" radio buttons in the GUI)	`preroute_only` | `preroute_signal` | `preroute_signal_blockage`	Controls when fat wires are merged. The default is `preroute_signal`.

Table 2-4 set_route_options Command Options (Continued)

Option	Valid values	Description
`-fat_blockage_as` ("Treat fat blockages as" radio buttons in the GUI)	`thin_wire` | `fat_wire`	Specifies how to treat fat blockages. The default is `fat_wire`.
`-wire_contact_eol_rule` ("Wire/Contact end of line rule" radio buttons in the GUI)	`ignore` | `check_and_fix`	Specifies whether check the end-of-line rule. The default is `ignore`.

Setting Signal Integrity Options

To improve signal integrity, you can enable crosstalk prevention during global routing and track assignment and enable crosstalk fixing during postroute optimization. To perform these signal integrity tasks when you run the `route_opt` command, you must use the `-xtalk_reduction` option.

For more information about the IC Compiler signal integrity features, see Chapter 8, "Signal Integrity," in the *IC Compiler Implementation User Guide*.

Enabling Crosstalk Prevention

By default, the classic router does not perform crosstalk reduction. If you enable signal integrity mode, the classic router performs crosstalk reduction during global routing and track assignment. To enable signal integrity mode, enter the following command:

```
icc_shell> set_si_options -route_xtalk_prevention true
```

The default crosstalk prevention threshold is 0.35 volts, which is probably too relaxed. If your design has crosstalk violations, use the `set_si_options -route_xtalk_prevention_threshold` command to lower the crosstalk prevention threshold during track assignment to a value between in the range of 0.25 to 0.35 volts. When you set the crosstalk prevention threshold, the tool automatically sets the `threshold_noise_ratio` common route option.

When you enable crosstalk prevention, the classic router avoids putting long, parallel wires on adjacent tracks during track assignment. To minimize noise, track assignment estimates the potential noise with a simplified crosstalk checker and reassigns wires to reduce the potential noise using the noise threshold.

Enabling Crosstalk Fixing

To minimize crosstalk-induced noise, you can specify the aggressors for a victim net. Crosstalk prevention during postroute optimization uses that information to minimize the crosstalk-induced noise from the aggressor nets.

To specify aggressors for a victim net, use the `set_net_aggressors` command (or choose Route > Routing Setup > Set Net Aggressors in the GUI). To specify the victim net, use the `-victim_net` option (or the “Victim net” box in the GUI). To specify the aggressor nets, use the `-aggressor_nets` option (or the “Aggressor nets” box in the GUI).

Enabling Distributed Routing

You can use distributed routing to decrease the runtime for the following routing commands:

- `route_opt`
- `route_group`
- `route_auto`
- `route_detail`
- `route_search_repair`
- `optimize_wire_via`
- `route_eco`
- `verify_route`
- `route_spreadwires`
- `insert_redundant_vias`
- `fix_lcc_hotspot`

Note:

You can also use distributed routing for the following power and ground routing commands: `create_power_straps` and `preroute_standard_cells`. These commands are described in the *IC Compiler Design Planning User Guide*.

To perform distributed routing, you must

1. Have the appropriate licenses

   To perform distributed routing using three or four CPUs, you must have the Galaxy-MultiRoute4 license. To perform distributed routing using more than four CPUs, you must have the Galaxy-MultiRoute8 license.

2. Set up the network

   To set up the network for distributed routing, run the `set_distributed_route` command (or choose File > Distributed Route Jobs > Set Distributed Route in the GUI). For more information about setting up the network, see the following section, "Setting Up the Distributed Routing Network."

3. Select the partitioning method

   Distributed routing divides the design into several routing partitions, based on size and congestion. These partitions are then routed on separate CPUs, in parallel, greatly reducing the overall runtime. One CPU is used to reassemble the partitions. For example, with four CPUs available, a typical runtime improvement would be a 3.5x reduction in overall routing time.

   By default, the classic router uses a two-pass partitioning method. If your design is very large, you might be able to reduce runtime by using the one-pass method instead. To use the one-pass method, set the `droute_enable_one_pass_partitioning` variable to 1.

4. Invoke distributed routing

   To enable distributed routing for all routing commands that support distributed routing, set the `droute_numCPUs` variable to a number greater than one before running the routing commands.

   To invoke distributed routing for a specific routing command, use the `-num_cpu` option to specify multiple CPUs when you run the routing command.

   Note:

   If you specify both the `droute_numCPUs` variable and the `-num_cpu` command option, the value specified by the `-num_cpu` option overrides the value specified by the `droute_numCPUs` variable. In addition, the number of CPUs specified must be less than or equal to the number of CPUs defined by the `set_distributed_route` command.

Setting Up the Distributed Routing Network

The classic router supports distributed routing operations either by using a package called *jp* to set up the communication between jobs on a network or by using resources under the control of the Load Sharing Facility (LSF) network. The following sections describe how to set up the network for distributed routing using each of these methods.

Setting Up Distributed Routing With jp

To set up distributed routing with the jp package,

1. If you are using processors on other machines, create a .rhosts file in your home directory.

   The .rhosts file specifies the remote machines that you can access by listing one entry per line. You might need to specify the complete host name, including the domain name.

   To determine the host name to put in the .rhosts file,

   a. Log in the machine you want to use by using the `rlogin` shell command.

   b. Determine the host name by using the `finger -l` shell command.

   To be safe, you can include both the simple and the complete host name for each remote machine, as shown in the following example .rhosts file.

   ```
   machineName1
   machineName1.domain.company.com
   machineName2
   machineName2.domain.company.com
   ```

2. Open the Milkyway design library, as described in Chapter 3, “Preparing the Design” in the *IC Compiler Implementation User Guide*.

3. Run the `set_distributed_route` command to specify the following:

   - The machines to use (`-jp_machines` option)

     You can specify up to 16 machines, including the current machine.

     If you are using only the current machine, you do not need to specify this option, because this is the default.

     For example, to use *machineName1*, *machineName2*, and *machineName3* in addition to the current machine, enter

     ```
     icc_shell> set_distributed_route \
        -jp_machines {machineName1 machineName2 machineName3}
     ```

- The maximum number of CPUs to use on each machine (`-jp_limit` option)

  For each machine for which you want to limit the number of CPUs used, specify the machine name and the maximum number of CPUs. The machine name must match a machine specified in the `-jp_machines` option. If you do not specify a limit for a machine, distributed routing will use all CPUs on that machine.

  For example, to use a maximum of two CPUs on each machine, enter

  ```
  icc_shell> set_distributed_route \
     -jp_limit {machineName1 2 machineName2 2 machineName3
  2}
  ```

- The location of the executable file for each machine (`-jp_bin` option)

  By default, distributed routing uses the location of the current icc_shell executable. If the executable file for a remote machine is in a location other than the default location, you must specify the machine name and the executable location for that machine. The machine name must match a machine specified in the `-jp_machines` option.

  For example, to specify the executable for *machineName1*, enter

  ```
  icc_shell> set_distributed_route \
     -jp_bin {machineName1 /machineName1/$SYNOPSYS/bin/}
  ```

  where $SYNOPSYS is the path to the installation directory.

- The location of the Milkyway design library (`-jp_lib` option)

  By default, distributed routing uses the location of the current Milkyway design library. If the design library for a remote machine is in a location other than the default location, you must specify the machine name and the design library location for that machine. The machine name must match a machine specified in the `-jp_machines` option.

Note:

When using the jp package, you can disconnect the network used for distributed routing by using the `set_distributed_route -jp_disconnect` option. This command disconnects all previously defined remote machines. You can also use the `close_distributed_route` command to close all of the sockets and shut down the daemons.

Setting Up Distributed Routing With LSF

To set up distributed routing with LSF, run the `set_distributed_route` command to

- Select LSF as the distributed routing method (`-lsf` option)
- Specify the LSF `bsub` command options to use when submitting jobs (`-lsf_advanced` option)

  By default, distributed routing with LSF does not use any `bsub` options. For more control over the job submissions, you can specify the `bsub` options to use. Common `bsub` options include `-q` (the queue to which to submit the jobs), `-m` (the machines to which to submit the jobs), and `-M` (the minimum memory required). For more information about the `bsub` command options, see the LSF documentation.

Reporting Distributed Route Jobs

To get information about a specific distributed route job, use the `report_distributed_route` command. You must use the `-job` option to identify the job to report. To get the job identification number, use a system command such as the UNIX `top` or `ps` command.

Cancelling Distributed Route Jobs

To cancel an active distributed route job, use the `remove_distributed_route` command. You must use the `-job` option to identify the job to cancel. To get the job identification number, use a system command such as the UNIX `top` or `ps` command.

3

Routing Clock and Signal Nets

This chapter describes how to use the classic router to perform routing tasks. It contains the following sections:

Routing Critical Nets

You can preroute a group of nets, such as clock nets, before routing the rest of the nets in the design. Taking this step can help you find timing problems. For example, you can route clock and bus pins to prerouted clock and bus wires and then perform timing analysis on these nets. If the timing results are satisfactory, you can route the remaining nets.

To preroute a group of nets, use the `route_group` command (or choose Route > Net Group Route in the GUI). To route all clock nets, use the `-all_clock_nets` option (or select the “All clock nets” radio button in the GUI); otherwise, you can specify a collection of nets to route by using the `-nets` option (or the “Specified nets” box in the GUI).

For example, to route all clock nets, enter

```
icc_shell> route_group -all_clock_nets
```

Table 3-1 lists the options for the `route_group` command.

Table 3-1 route_group Command Options

Option	Description
`-no_global` (“Global route” check box in the GUI)	Skips the global routing phase. By default, `route_group` performs global routing, track assignment, and detail routing.
`-no_track` (“Track assignment” check box in the GUI)	Skips the track assignment phase. By default, `route_group` performs global routing, track assignment, and detail routing.
`-no_detail` (“Detail route” check box in the GUI)	Skips the detail routing phase. By default, `route_group` performs global routing, track assignment, and detail routing.
`-nets` *`collection_of_nets`* (“Specified nets” radio button and text box in GUI)	Specifies the nets to route. Either this option or the `-all_clock_nets` option is required.
`-all_clock_nets` (“All clock nets” radio button in the GUI)	Routes all clock nets. Either this option or the `-nets` option is required.

Table 3-1 route_group Command Options (Continued)

Option	Description
`-dont_optimize_route_pattern` (“Optimize routing pattern” check box in the GUI)	Disables optimization of the routing pattern. By default, the routing pattern is optimized.
`-utilize_dangling_wires` (“Utilize dangling wires” check box in the GUI)	Enables reuse of existing dangling routes to fix open nets. By default, the router does not reuse existing dangling routes.
`-trim_antenna_of_user_wires` (“Trim wire segments of prerouted wires” check box in the GUI)	Enables trimming for wire segments of prerouted wires. By default, the router does not trim wire segments of prerouted wires.
`-num_cpus` *int* (“Clock routing” radio buttons in the GUI)	Specifies the number of CPUs to use for distributed routing. You must specify a value between 1 and 63. For information about setting up for distributed routing, see “Enabling Distributed Routing” on page 2-18. The default is 1.

Shielding Clock Nets

The classic router uses nondefault routing rules to define the shielding width and spacing. For information about nondefault routing rules, see “Using Nondefault Routing Rules” on page 2-7.

Note:

If you do not define the shielding spacing for clock nets, the classic router adds default spacing as shield spacing on clock nets.

After defining the nondefault routing rules and routing the clock nets, shield the clock nets by using the `create_auto_shield` command (or choosing Route > Create Auto Shield in the GUI). For information about shielding nets, see “Shielding Nets” on page 3-32.

Routing Signal Nets

Before you route the signal nets, all clock nets must be routed without violations.

You can route the signal nets by using one of the following methods:

- Use the `route_opt` core command

  The `route_opt` command performs global routing, track assignment, detail routing, search and repair, and postroute optimization.

- Use automatic routing (the `route_auto` basic command)

  The `route_auto` basic command performs global routing, track assignment, and detail routing.

  When you run `route_auto`, the classic router reads the design database before starting routing and updates the database when all routing steps are done. If you stop automatic routing before it performs detail routing, the classic router checks the input data when you restart routing with this command.

- Use the basic commands to perform the standalone routing tasks

  To perform global routing, use the `route_global` command. To perform track assignment, use the `route_track` command. To perform detail routing, use the `route_detail` command. To perform search and repair, use the `route_search_repair` command.

  When you run a standalone routing command, such as `route_global` or `route_detail`, the classic router reads in the design database at the beginning of each routing command and updates the database at the end of each command. The router does not check the input data. For example, if the track assignment step is skipped and you run detail routing directly, the classic router might generate bad routing results.

  If you need to customize your routing flow or you need to run a large design step-by-step, you might want to use the standalone routing commands instead of the `route_opt` command or automatic routing.

Routing Signal Nets by Using the route_opt Command

Before you use the `route_opt` command to route the signal nets, you must define the routing options, as described in “Setting Routing Options” on page 2-13 and set the `route_opt` routing and optimization strategy, as described in the following section. If logical DRC violations remain after running the `route_opt` command, you can use focal optimization to address these remaining violations, as described in “Performing Focal Optimization” on page 3-17.

Setting the Routing and Optimization Strategy

The IC Compiler tool provides strategy controls that you set to guide how routing and postroute optimizations are run.

To set the routing and optimization strategy, use the `set_route_opt_strategy` command (or choose Route > Set Core Routing and Optimization Strategy in the GUI).

Table 3-2 describes the `set_route_opt_strategy` command options. For more information, see the man page.

Table 3-2 set_route_opt_strategy Command Options

Command option	Valid values	Description
`-fix_hold_mode` (“Fix hold optimization stage” radio buttons in the GUI)	`all` | `route_base`	The stages for which hold optimization is performed. You can select prerouting, global routing, and detail routing (`all`) or global routing and detail routing (`route_base`). The default is `route_base`.
`-power_aware_optimization` (“Power aware optimization flow” check box in the GUI)	`true` | `false`	Controls whether postroute optimizations for setup, hold, and design rule constraints are power-aware. The default is `false`.
`-xtalk_reduction_loops` (“Maximum crosstalk reduction cycles” box in the GUI)	*int* (must be between -1 and 10)	The maximum number of crosstalk reduction optimization cycles. The default is 2.
`-search_repair_loops` (“Maximum initial route Search and Repair cycles” box in the GUI)	*int* (must be between 0 and 500)	The maximum number of initial routing search and repair iterations. The classic router default is 15.
`-eco_route_search_repair_loops` (“Maximum ECO Search and Repair cycles” box in the GUI)	*int* (must be between 0 and 500)	The maximum number of ECO search and repair iterations. The classic router default is 5.

Table 3-2 set_route_opt_strategy Command Options (Continued)

Command option	Valid values	Description
`-route_drc_threshold` ("Route violations threshold to trigger the reduction Search and Repair loop" box in the GUI)	*int*	The threshold number of routing violations before limiting search-and-repair to one iteration. The default is 3000.
`-enable_port_punching true | false`	`true` | `false`	Controls whether postroute optimizations can modify the logical topology of a net to increase the area available for buffer insertion to fix constraint violations. The default is `false`.

To report your settings, use the `report_route_opt_strategy` command.

Enabling Fixing of Hold Time Violations

The `route_opt` command fixes hold time violations on those clocks that have a `fix_hold` attribute of `true`.

To disable hold fixing, enter the following command to remove the `fix_hold` attribute from all clocks:

```
icc_shell> remove_attribute [get_clocks *] fix_hold
```

To enable hold fixing, use the `set_fix_hold` command to set the `fix_hold` attribute on the clocks on which to perform hold fixing.

For example, to enable hold fixing on clock clk, enter the following command:

```
icc_shell> set_fix_hold clk
```

To enable hold fixing on all clocks, enter the following command:

```
icc_shell> set_fix_hold [all_clocks]
```

Note:

If you enable hold fixing for a multicorner-multimode design, the tool considers the timing from driver to endpoint on a per-scenario basis and fixes hold violations accordingly.

You can specify a list of preferred buffers for hold fixing during postroute optimization by using the `set_prefer` and `set_fix_hold_options` commands. For example, the following commands instruct the tool to use cells BUF1 and BUF2 as the preferred cells during hold

fixing. Note that cells BUF1 and BUF2 can also be used to resolve setup and DRC violations.

```
icc_shell> set_prefer -min {BUF1 BUF2}
icc_shell> set_fix_hold_options -preferred_buffer
```

If you use the `set_dont_use` command to set the `dont_use` attribute on cells BUF1 and BUF2 as shown in the following script, the tool uses cells BUF1 and BUF2 to fix hold violations, but not setup and DRC violations.

```
icc_shell> set_dont_use {BUF1 BUF2}
icc_shell> set_prefer -min {BUF1 BUF2}
icc_shell> set_fix_hold_options -preferred_buffer
```

The IC Compiler tool also provides the following variables that control hold fixing:

- `routeopt_allow_min_buffer_with_size_only`

  By default, the tool cannot insert buffers during size-only optimization. When you set this variable to `true` and use the `-size_only` option when you run the `route_opt` command, the tool can insert buffers to fix hold violations.

- `routeopt_enable_aggressive_optimization`

  This variable enables additional, aggressive fixing of hold violations.

- `psyn_onroute_disable_hold_fix`

  This variable disables hold fixing, but preserves the minimum timing on clocks that have the `fix_hold` attribute.

Setting the Crosstalk Reduction Options

By default, the `route_opt` command does not perform crosstalk reduction. To perform crosstalk reduction,

1. Enable signal integrity mode by setting the `set_si_options -delta_delay` option to `true`.

   When you enable signal integrity mode, the `route_opt` command also performs signal-integrity-aware postroute optimization.

2. (Optional) Enable static noise fixing by setting the `set_si_options -static_noise` option to `true`.

   When you enable static noise fixing, the `route_opt` command also fixes static noise violations during postroute optimization.

3. Run the `route_opt` command with the `-xtalk_reduction` or `-only_xtalk_reduction` option.

   To perform both crosstalk reduction and postroute optimization, use the `-xtalk_reduction` option. To perform only crosstalk reduction and not postroute optimization, use the `-only_xtalk_reduction` option.

By default, when signal integrity mode is enabled and you run the `route_opt` command with the `-xtalk_reduction` or `-only_xtalk_reduction` option, the classic router performs a single iteration of crosstalk reduction on all nets with setup violations and, if enabled, static noise violations. You can increase the number of iterations by setting the `-xtalk_reduction_loops` option of the `set_route_opt_strategy` command. You can restrict the nets selected for crosstalk reduction by setting the `routeopt_xtalk_reduction_setup_threshold` variable.

The default method for reducing crosstalk is to increase the spacing between nets. To further reduce crosstalk and improve timing QoR, the tool can perform cell sizing by using footprint swapping on cells on the victim or aggressor nets in addition to increasing the spacing between nets. To enable cell sizing during crosstalk reduction, set the `routeopt_xtalk_reduction_cell_sizing` variable to `true` before running the `route_opt` command.

Using Scenario Compression for Multicorner-Multimode Designs

For multicorner-multimode designs that have fewer modes than scenarios, you can improve the capacity and runtime of postroute optimization without sacrificing QoR by enabling scenario compression. When you enable scenario compression, the original set of scenarios is compressed into a subset of dominant scenarios. In addition, the constraints of this dominant subset are modified to capture any violations from the nondominant scenarios. To identify the dominant scenarios used during compression, the tool appends a suffix of *__z__* to the scenario names.

To use scenario compression,

1. If your design is not yet routed, perform initial routing by running the `route_opt -initial_route_only` command.

   ```
   icc_shell> route_opt -initial_route_only
   ```

2. Enable scenario compression by running the `compress_scenarios` command.

   ```
   icc_shell> compress_scenarios
   ```

   By default, the tool determines the number of dominant scenarios. To specify a maximum number of dominant scenarios, use the `-max_compression` option with the `compress_scenarios` command.

3. Run the `route_opt` command one or more times to perform initial or incremental postroute optimization.

   Note:
   For the best runtime benefits, run the `route_opt` command at least two times.

   To run initial postroute optimization, use the `route_opt -skip_initial_route` command. To run incremental postroute optimization, use the `route_opt -incremental` command. You can use other `route_opt` options as necessary to configure the postroute optimization.

4. Disable scenario compression by running the `uncompress_scenarios` command.

   ```
   icc_shell> uncompress_scenarios
   ```

   Note:
   Scenario compression is supported only by the `route_opt` command with the `-skip_initial_route` or `-incremental` option. You must revert your scenarios back to the uncompressed state before running any other commands, including timing analysis commands.

Running the route_opt Command

To run routing and postroute optimization, use the `route_opt` command (or choose Route > Core Routing and Optimization in the GUI).

By default, the `route_opt` command performs the following tasks:

- Global routing

  By default, global routing is not timing-driven and does not do crosstalk prevention.

  To enable timing-driven global routing, use the `set_route_options -groute_timing_driven true` command (or choose Route > Routing Setup > Set Route Options in the GUI and select “Timing driven” in the Global Routing tab).

  To enable crosstalk-prevention mode, set the signal integrity options as described in “Setting Signal Integrity Options” on page 2-17, and use the `-xtalk_reduction` or `-only_xtalk_reduction` option when you run the `route_opt` command.

- Track assignment

  By default, track assignment is not timing-driven and does not do crosstalk prevention.

  To enable timing-driven track assignment, use the `set_route_options -track_assign_timing_driven true` command (or choose Route > Routing Setup > Set Route Options in the GUI and select “Timing driven” in the Track Assign tab).

  To enable crosstalk-prevention mode, set the signal integrity options as described in “Setting Signal Integrity Options” on page 2-17, and use the `-xtalk_reduction` or `-only_xtalk_reduction` option when you run the `route_opt` command.

- Detail routing

  To enable crosstalk-reduction mode, set the signal integrity options as described in “Setting Signal Integrity Options” on page 2-17, and use the `-xtalk_reduction` or `-only_xtalk_reduction` option when you run the `route_opt` command.

- Search and repair

- Postroute optimization

  By default, the IC Compiler tool uses medium effort during postroute optimization. You can change the effort level by using the `-effort` option. When you select high effort, the tool is more aggressive during optimization and runs three optimization loops.

  In addition to controlling the optimization effort, the effort setting also controls the type of sizing performed when you use the `-size_only` option.

  - During low effort optimization, the tool performs footprint swapping.
  - During medium effort optimization, the tool performs in-place size-only optimization.
  - During high effort optimization, the tool performs cell sizing.

You can also use variables to control the effort used for specific optimizations.

- To enable additional, aggressive fixing of hold time and maximum transition violations, set the `routeopt_enable_aggressive_optimization` variable to `true`.
- To restrict total negative slack optimization to size-only optimization, set the `routeopt_restrict_tns_to_size_only` variable to `true`.

For designs that contain block abstraction models, the tool performs top-level interface optimization during postroute optimization. For information about top-level interface optimization, see Chapter 10, “Using Hierarchical Models,” in the *IC Compiler Implementation User Guide*.

In addition to performing timing optimization, you can perform the following postroute optimizations: leakage power, signal integrity, wire length and via count reduction, and area recovery.

- To perform leakage-power optimization,

  1. For a multicorner-multimode design, use the `set_scenario_options` command to select the leakage scenarios.

     For more information about selecting the leakage scenarios, see Chapter 3, “Preparing the Design,” in the *IC Compiler Implementation User Guide*.

  2. (Optional) Enable power-aware postroute optimization by setting the `-power_aware_optimization` option of the `set_route_opt_strategy` command to `true`.

     By default, when power-aware postroute optimization is enabled, leakage power has a lower cost priority than setup, hold, and design rule constraints. You can change these cost priorities by setting the following variables to `true`: `routeopt_leakage_over_setup`, `routeopt_leakage_over_hold`, and `routeopt_leakage_over_drc`.

  3. Enable leakage-power optimization and recovery by using the `-power` option when you run the `route_opt` command.

  To perform only power recovery during incremental postroute optimization, use the `-only_power_recovery` option with the `route_opt -incremental` command.

- To perform signal integrity optimization,

  1. Set the crosstalk reduction options as described in “Setting the Crosstalk Reduction Options” on page 3-8.

  2. Use the `-xtalk_reduction` option when you run the `route_opt` command.

- To perform wire length and via count reduction optimization, use the `-optimize_wire_via` option when you run the `route_opt` command.
- To perform area recovery, use the `-area_recovery` option when you run the `route_opt` command.

  To perform only area recovery during incremental postroute optimization, use the `-only_area_recovery` option with the `route_opt` command.

By default, the postroute optimization step generates QoR reports before and after the postroute optimization. If you do not need the QoR reports that are generated after postroute optimization, you can reduce runtime by setting the `routeopt_skip_report_qor` variable to `true`. Setting this variable to `true` prevents generation of QoR reports after postroute optimization; this variable does not affect the generation of QoR reports before postroute optimization. To get more information about the optimization process, you can enable verbose reporting during the hold fixing, DRC fixing, and crosstalk reduction stages by setting the `routeopt_verbose` variable. This variable uses bitwise operation to enable various reporting capabilities. For details about setting this variable, see SolvNet article 032174.

If you want to analyze the routing results before performing postroute optimization or you want to use scenario compression during postroute optimization, first run the `route_opt` command without postroute optimization (`-initial_route_only` option), and then run the `route_opt` command with only postroute optimization. The `route_opt` command provides two options that perform only postroute optimization:

- `-skip_initial_route`

  This option performs two passes of postroute optimization and ECO routing. If signal integrity mode is enabled, the first pass does not perform crosstalk reduction or signal integrity optimization, but the second pass does.

- `-incremental`

  This option performs one pass of postroute optimization and ECO routing. If signal integrity mode is enabled, the tool performs crosstalk reduction and signal integrity optimization.

  When you use the `-incremental` option, you can restrict postroute optimization to one of the following optimization techniques:

  - Register-to-register optimization

    To restrict postroute optimization to use only register-to-register optimization, use the `-register_to_register` option with the `-incremental` option. By default, register-to-register optimization fixes setup, hold, and logical DRC violations. To restrict the optimization to a specific violation type, use the `-size_only`, `-only_hold_time`, or `-only_design_rule` option in addition to the `-incremental` and `-register_to_register` options.

- Sizing of weak drive cells

  To restrict postroute optimization to use only footprint-preservation sizing to increase the drive strength of weak cells, set the `routeopt_only_size_weak_drive_cells` variable to `true` before running the `route_opt -incremental` command.

  You can use this technique to generate a robust design with similar or better QoR that is less sensitive to variations.

To analyze the routing results before performing postroute optimization, use the following flow:

```
icc_shell> route_opt -initial_route_only
# analyze routing results
icc_shell> route_opt -skip_initial_route
```

To use scenario compression during postroute optimization, use the following flow:

```
icc_shell> route_opt -initial_route_only
icc_shell> compress_scenarios
icc_shell> route_opt -skip_initial_route
icc_shell> route_opt -incremental
icc_shell> uncompress_scenarios
```

For more information about using scenario compression, see "Using Scenario Compression for Multicorner-Multimode Designs" on page 3-8.

Table 3-3 describes the `route_opt` command options.

Table 3-3 route_opt Command Options

Command option	Description
`-effort low | medium | high` (Effort options in the GUI)	Specifies the optimization effort level. Higher effort levels provide more aggressive optimization at a runtime cost. The default is `medium`.
`-stage global |track | detail` ("Run optimization after" options in the GUI)	Specifies the last routing stage performed by the `route_opt` command before performing optimization. If you specify `global` or `detail`, only the specified stage is run. If you specify `track`, both global routing and track assignment are run. By default, the `route_opt` command performs optimization after running global routing, track assignment, and detail routing.

Table 3-3 route_opt Command Options (Continued)

Command option	Description
`-xtalk_reduction` (“Crosstalk reduction and SI optimization” option in the GUI)	Performs crosstalk reduction and signal integrity optimization. By default, the `route_opt` command does not perform crosstalk reduction or signal integrity optimization.
`-only_xtalk_reduction` (“Crosstalk reduction optimization only” option in the GUI)	Performs only crosstalk reduction (not signal integrity optimization). By default, the `route_opt` command does not perform crosstalk reduction or signal integrity optimization.
`-power` (“Perform power optimization” check box in the GUI)	Performs leakage-power optimization and recovery. When you use this option to perform leakage-power optimization, the tool uses low-effort leakage cell selection. By default, the `route_opt` command does not perform leakage-power optimization.
`-skip_initial_route` (“Skip initial routing” check box in the GUI)	Performs postroute optimization without routing.
`-initial_route_only` (“Initial routing only” check box in the GUI)	Performs only initial routing without postroute optimization. This option works in conjunction with the `-stage` option.
`-size_only` (“Sizing only optimization” check box in the GUI)	Resolves setup time violations by sizing only. To use register-to-register optimization to resolve only setup time violations during incremental postroute optimization, use this option with the `-incremental` and `-register_to_register` options.
`-optimize_wire_via` (“Optimize wire and via routing” check box in the GUI)	Performs wire and via optimization. By default, the `route_opt` command does not perform wire and via optimization.
`-area_recovery` (“Recover area for cells that are not on critical timing paths” check box in the GUI)	Recovers area for cells not on critical paths.

Table 3-3 route_opt Command Options (Continued)

Command option	Description
`-wire_size` (“Use wire sizing to fix setup time violations” check box in the GUI)	Determines whether wire sizing is used to fix setup time violations.
`-incremental` (“Incremental mode” check box in the GUI)	Performs incremental postroute optimization.
`-incremental -register_to_register` (“Register to register” check box in the GUI)	Performs only register-to-register optimization to resolve setup, hold, and logical DRC violations during incremental postroute optimization. You can restrict the optimization to a specific violation type by also using the `-size_only`, `-only_hold_time`, or `-only_design_rule` option.
`-incremental -only_wire_size` (“Timing optimization only with wire size” option in the GUI)	Performs only wire sizing to resolve setup time violations during incremental postroute optimization.
`-incremental -only_hold_time` (“Hold timing optimization only” option in the GUI)	Resolves only hold time violations during incremental postroute optimization. Use the `set_fix_hold` command to enable hold time fixing. To use only register-to-register optimization to resolve hold time violations during incremental postroute optimization, use this option with the `-register_to_register` option.
`-incremental -only_area_recovery` (“Area recovery only” option in the GUI)	Performs only area recovery during incremental postroute optimization.
`-incremental -only_design_rule` (“Fix design rules only” option in the GUI)	Performs only logical design rule fixing during incremental postroute optimization. By default, timing has a higher cost priority than design rule constraints. When you use this option, you can give a higher cost priority to design rule constraints by setting the `routeopt_drc_over_timing` variable to `true`. To use only register-to-register optimization to resolve logical design rule violations during incremental postroute optimization, use this option with the `-register_to_register` option.

Table 3-3 route_opt Command Options (Continued)

Command option	Description
`-incremental -only_power_recovery` (“Power recovery only” option in the GUI)	Performs only power recovery during incremental postroute optimization. When you use this option, the `-effort` option controls the cell-selection effort level for leakage-power optimization. By default, leakage power has a lower cost priority than setup, hold, and design rule constraints. You can change these cost priorities by setting the following variables to `true`: `routeopt_leakage_over_setup`, `routeopt_leakage_over_hold`, and `routeopt_leakage_over_drc`.
`-num_cpus` *number* (“Number of CPUs” box in the GUI)	Specifies the number of CPUs to use for distributed routing. You must specify a value between 1 and 63. For information about setting up for distributed routing, see “Enabling Distributed Routing” on page 2-18. The default is 1.

Saving Intermediate Results

To save intermediate `route_opt` results, enable checkpointing by setting the `routeopt_checkpoint` variable to `true`. When checkpointing is enabled, the `route_opt` command saves the design after each major stage. Table 3-4 shows the designs saved by `route_opt` checkpointing.

Table 3-4 Designs Saved By route_opt Checkpointing

Design name	Saved after
*design_name*_RT	Initial routing
*design_name*_ORPRT	Wire and via optimization
*design_name*_OR_*n*	Each postroute optimization pass
*design_name*_ECORT_*n*	Each ECO routing pass
*design_name*_RCRED	Crosstalk reduction
*design_name*_POR	Power optimization
*design_name*_PECORT	Power optimization ECO routing

Table 3-4 Designs Saved By route_opt Checkpointing (Continued)

Design name	Saved after
*design_name*_HOLD_PRE	Preroute-based hold optimization
*design_name*_HOLD	Global-route-based hold optimization
*design_name*_INCOR	Optimization after incremental `route_opt`
*design_name*_HOLD	ECO routing after incremental `route_opt`

Performing Focal Optimization

You can use the `focal_opt` command to perform aggressive, topology-based optimization on your postroute design that is focused on fixing either the setup, hold, or logical DRC violations that remain after the postroute optimization performed by the `route_opt` command. You can also use the `focal_opt` command to perform register-to-register optimization, crosstalk reduction, or final stage leakage recovery.

For each run, you must specify the focus of the optimization.

- To fix setup violations, use the `-setup_endpoints` option.

  To fix all setup violations, set this option to `all`. To fix specific setup violations, specify the endpoints by using an endpoint file or by providing a collection that contains the endpoints. For information about the format of the endpoint file, see “Endpoint File Format” on page 3-19.

- To fix hold violations, use the `-hold_endpoints` option.

  To fix all hold violations, set this option to `all`. To fix specific hold violations, specify the endpoints by using an endpoint file or by providing a collection that contains the endpoints. For information about the format of the endpoint file, see “Endpoint File Format” on page 3-19.

- To fix logical DRC violations on nets, use the `-drc_nets` option.

  To fix DRC violations on all nets, set this option to `all`. To fix DRC violations on specific nets, specify the nets by using a net file or by providing a collection that contains the nets. For information about the format of the net file, see “Net File Format” on page 3-19.

- To fix logical DRC violations on pins, use the `-drc_pins` option.

  To fix DRC violations on all pins, set this option to `all`. To fix DRC violations on specific pins, specify the pins by using an endpoint file or by providing a collection that contains

the pins. For information about the format of the endpoint file, see "Endpoint File Format" on page 3-19.

- To perform register-to-register optimization, use the `-register_to_register` option.

  By default, when you enable register-to-register optimization, the `focal_opt` command fixes setup, hold, and logical DRC violations. You can restrict the optimization to a specific violation type by using the `-setup_endpoints`, `-hold_endpoints`, `-drc_nets`, or `-drc_pins` option.

- To perform crosstalk reduction on specific nets, use the `-xtalk_reduction` option.

  Specify the nets on which to perform crosstalk reduction by using a net file or by providing a collection that contains the nets. For information about the format of the net file, see "Net File Format" on page 3-19.

- To perform final stage leakage recovery, use the `-power` option.

  When you perform final stage leakage recovery, you must enable exactly one leakage scenario. For information about leakage scenarios, see Chapter 3, "Preparing the Design," in the *IC Compiler Implementation User Guide*.

  For more information about final stage leakage recovery, see "Performing Final Stage Leakage-Power Recovery" on page 3-20.

By default, the `focal_opt` command uses medium effort to fix the remaining violations. When using medium effort, the `focal_opt` command explores more solution spaces than the `route_opt` command while fixing the violations, including solutions that exceed the density limit. If medium effort does not fix the remaining violations, you can use high effort (`-effort high` option), which explores even more solution spaces and uses runtime-expensive accurate signal integrity reestimation. Due to the additional runtime required for high effort, use this only when you have a few remaining violations to close. Note that the `-effort` option does not apply to final stage leakage recovery.

If your design has remaining violations, and you want to prioritize the fixing of those violations above all other cost priorities, you can use the `-prioritize` option. When you use the `-prioritize` option, the `focal_opt` command might violate other constraints to fix the prioritized violations. Note that the `-prioritize` option does not apply to crosstalk reduction or final stage leakage recovery.

If your design is very sensitive to postroute optimization changes, you can limit the optimizations to sizing-only optimizations by specifying the mode with the `-size_only_mode` option. When you specify this option you must select one of the following sizing modes: density-based sizing (`density`), in-place sizing (`in_place`), or footprint-preservation sizing (`footprint`). Note that the `-size_only_mode` option does not apply to crosstalk reduction or final stage leakage recovery.

If you find that the `focal_opt` command leaves unfixed violations, you can enable verbose reporting during optimization by setting the `routeopt_verbose` variable. This variable uses

bitwise operation to enable various reporting capabilities. For details about setting this variable, see SolvNet article 032174.

Endpoint File Format

To fix setup, hold, or logical DRC violations on specific endpoints, you can specify the endpoints in an endpoint file. This file can be either uncompressed or compressed in gzip format. When the file is a gzip file, it must have the .gz file extension.

The following formats are supported for specifying an endpoint:

- Endpoint only

  ```
  I_STACK_TOP/I3_STACK_MEM/Stack_Mem_reg_2__1_/D
  ```

- Endpoint with a slack value

  ```
  I_STACK_TOP/I3_STACK_MEM/Stack_Mem_reg_2__1_/D 0.58    0.44 r    -0.14
  ```

  By default, when you use this format with the `-setup_endpoints` or `-hold_endpoints` option, the tool uses the slack values specified in the endpoint file during focal optimization. To use the computed slack values instead, set the `focalopt_endpoint_margin` variable to `false`.

  When you use this format with the `-drc_pins` option, the tool ignores the slack information and uses only the pin information.

The best way to generate these lines is by editing the file generated by the `report_constraint` command.

Net File Format

To fix logical DRC violations on specific nets, you must specify the nets in a net file.

The following formats are supported for specifying a net:

- Net name only

  ```
  I_STACK_TOP/n342
  ```

- Net name with a violation

  ```
  I_STACK_TOP/n342 0.70           0.89          -0.19   (VIOLATED)
  ```

  When you use this format with the `-drc_nets` option, the tool ignores the slack information and uses only the net information.

The best way to generate these lines is by editing the file generated by the `report_constraint` command.

Performing Final Stage Leakage-Power Recovery

Final stage leakage-power recovery optimizes the leakage power based on the `cell_footprint` attribute of the library cells. Ensuring that your design is close to timing closure is a prerequisite for final stage leakage-power recovery.

The final stage leakage-power recovery step considers library cells to have the same footprint if the cells have

- The same operating conditions (P, V, and T)
- Logical equivalence
- The same physical boundary, width, and height, as defined in the FRAM view
- The same `cell_footprint` attribute

For information about preparing the logic libraries for final stage leakage-power recovery, see Chapter 3, “Preparing the Design,” in the *IC Compiler Implementation User Guide*.

To perform final stage leakage-power recovery on a postroute design,

1. For a multicorner-multimode design, use the `set_scenario_options` command to select the leakage scenario. You can select only a single leakage scenario.

   To ensure that you set only one leakage scenario, enter the following commands:

   ```
   icc_shell> set leakage_scenario S1
   icc_shell> set_scenario_options -leakage_power false \
      -scenarios [all_active_scenarios]
   icc_shell> set_scenario_options -leakage_power true \
      -scenarios $leakage_scenario
   ```

   If you do not have exactly one leakage scenario defined during final stage leakage-power recovery, the tool issues the PSYN-706 error message.

   ```
   Error: There are multiple scenarios or no scenario with
   set_scenario_options -leakage_power true. (PSYN-706)
   ```

   For more information about selecting the leakage scenarios, see Chapter 3, “Preparing the Design,” in the *IC Compiler Implementation User Guide*.

2. If the postroute design has setup, hold, or logical DRC violations, run the `focal_opt` command to perform focal optimization to fix these violations.

   For more information about the `focal_opt` command, see “Performing Focal Optimization” on page 3-17.

3. (Optional) Run the `verify_zrt_route` command to report the routing DRC violations that exist before final stage leakage-power recovery.

   Perform this step to validate the cell footprint in the physical libraries by comparing the DRC violations reported before and after final stage leakage-power recovery.

4. Run the `focal_opt -power` command to perform postroute focal optimization for leakage-power recovery.

   By default, this step uses footprint swapping to improve the leakage power on paths with positive slack. You can control the tradeoff between leakage-power recovery and timing QoR preservation by setting the `focalopt_power_critical_range` variable. This variable specifies the slack threshold for leakage-power recovery and timing QoR preservation. The `focal_opt -power` command performs leakage-power recovery only on cells whose worst slack is greater than the power critical range value and preserves the timing QoR such that the worst slack is greater than or equal to the power critical range value.

   When you set this variable to a positive number, the command preserves timing QoR with positive slack and performs less leakage-power recovery. When you set this variable to a negative number, the command performs more aggressive leakage-power recovery and allows timing degradation. For multicorner-multimode designs, the tool uses only the leakage scenario for leakage-power recovery, but it considers all active scenarios for timing QoR. Note that this command does not perform placement legalization or ECO routing after the leakage-power recovery.

   When you use the `-power` option, you cannot use any other options with the `focal_opt` command.

   When you run the `focal_opt -power` command, pay attention to the FOPT-28 and FOPT-29 messages in the log file.

   ```
   Information: Optimizing leakage power with 3 voltage thresholds
   (FOPT-028).
   Warning: Libcells cell_name1 and cell_name2 have same footprint but
   different pin geometry - eco route may be required after Leakage
   optimization (FOPT-029).
   ```

   These information and warning messages list the library cells that have issues with the `cell_footprint` attributes.

5. (Optional) Run the `verify_zrt_route` command to report the routing DRC violations that exist after final stage leakage-power recovery.

   Compare the reported violations to those reported before final stage leakage-power recovery. If the number of violations reported after running the `focal_opt -power` command is more than the number of violations reported before, it implies that the physical libraries have issues such as dissimilar pin shape that result in additional DRC violations.

The following example script shows how to perform final stage leakage-power recovery for a multicorner-multimode design after routing optimization:

```
# Set one scenario as the leakage corner
set_scenario_options -leakage_power false \
   -scenarios [all_active_scenarios]
set_scenario_options -leakage_power true -scenarios S1

# Fix setup, hold, and logical DRC violations
focal_opt -setup_endpoints all
focal_opt -hold_endpoints all
focal_opt -drc_nets all

# Perform final stage leakage-power recovery
focal_opt -power
```

Routing Signal Nets by Using Automatic Routing

Before you use automatic routing to route the signal nets, you must define the routing options as described in “Setting Routing Options” on page 2-13.

To run automatic routing, use the `route_auto` command (or choose Route > Auto Route in the GUI). By default, the `route_auto` command sequentially invokes global routing, track assignment, and detail routing.

Table 3-5 describes the `route_auto` command options. For more information, see the man page.

Table 3-5 route_auto Command Options

Command option	Description
`-no_global` (“Global route” check box in Run section in the GUI)	Skips global routing.
`-no_track` (“Track assignment” check box in Run section in the GUI)	Skips track assignment.
`-no_detail` (“Detail route” check box in Run section in the GUI)	Skips detail routing.
`-save_after_global_route` (“Global route” check box in Run section in the GUI)	Controls whether the design is saved after global routing. When specified, the tool saves the design in a CEL view named auto_GR.

Table 3-5 route_auto Command Options (Continued)

Command option	Description
`-save_after_track` (“Track assignment” check box in Run section in the GUI)	Controls whether the design is saved after track assignment. When specified, the tool saves the design in a CEL view named auto_TA.
`-save_after_detail_route` (“Detail route” check box in Run section in the GUI)	Controls whether the design is saved after detail routing. When specified, the tool saves the design in a CEL view named auto_DR.
`-search_repair_loop` *int* (“Maximum Search & Repair cycles” box in the GUI)	Specifies the maximum number of search and repair loops. You must specify a value between 0 and 500. The default is 0.
`-effort minimum | low | medium | high` (Effort radio buttons in the GUI)	Specifies the effort level for automatic routing. The default is `medium`.
`-num_cpus` *int* (“Number of CPUs” box in the GUI)	Specifies the number of CPUs to use for distributed routing. You must specify a value between 1 and 63. For information about setting up for distributed routing, see “Enabling Distributed Routing” on page 2-18. The default is 1.

Running Individual Routing Steps

After you have specified your routing options, you can debug your design or monitor each routing step by performing each routing step individually. Individual routing steps are explained in the following sections:

- Global Routing
- Track Assignment
- Detail Routing
- Search and Repair
- Wire and Via Optimization

Global Routing

Global routing maps general pathways through the design for each unrouted signal net and clock net.

To perform global routing, use the `route_global` command (or choose Route > Global Route in the GUI). By default, the classic router performs medium-effort global routing. To change the effort level, use the `-effort` option.

The global router performs the following tasks:

- Divides the chip into square units called global routing cells.

  The height and width of the global routing cells are determined by using the average height of the standard cells.

- Assigns nets to the global routing cells through which they pass.

  For each global routing cell, the routing capacity is calculated according to the blockages, pins, and routing tracks inside the cell. Although the nets are not assigned to the actual wire tracks during global routing, the number of nets assigned to each global routing cell is noted.

- Determines the demand for routing tracks, both horizontally and vertically, through each global routing cell, based on the locations of the cell pins and their respective unrouted connections.

  The global router considers spacing and wide-wire variable routing rules, as well as shielding variable routing rules, when calculating congestion.

- Reports the overflow for each metal layer.

  The overflow is the number of tracks still needed after the global router assigns nets to the available wire tracks in a global routing cell. It is calculated as the number of tracks needed minus the number of available tracks in a given direction through a global routing cell. The higher the overflow value, the worse the overflow. A negative overflow value, which is referred to as an underflow, indicates that all required routes can be accommodated. If your design has positive overflow values, you can analyze the congestion issues by using the congestion report and congestion map described in “Analyzing Congestion” on page 3-27.

Global Routing During Design Planning

During design planning you can perform prototype global routing by using the `route_zrt_global -effort minimum` command.

```
icc_shell> route_zrt_global -effort minimum
```

If you are using the hierarchical flow, set the `-plan_group_aware_routing` option of the `set_fp_flow_strategy` command to `true` to enable plan-group-aware global routing before running the `route_global` command. When you enable plan-group-aware global routing, the classic router routes all the nets in the design and preserves the hierarchy and pin constraints. You can increase the plan-group-aware routing speed by routing only the top-level nets. To do this, set the `-top_level_routing_only` option of the `set_fp_flow_strategy` command to `true`. When plan-group-aware global routing is not enabled, which is the default, the classic router ignores the plan groups and routes the design as flat.

```
icc_shell> set_fp_flow_strategy -plan_group_aware_routing true
icc_shell> route_global -effort minimum
```

Track Assignment

Before you perform detail routing, run track assignment to specify which tracks within each global routing cell are to be used for each net. Track assignment operates on the entire design at once; it can make long routes straight and reduce the number of vias, whereas the detail routing routes a small area at a time.

After track assignment finishes, all nets are routed, but not very carefully. There are many violations, particularly where the routing connects to pins. Detail routing works to correct those violations.

To perform track assignment, use the `route_track` command (or choose Route > Track Assignment in the GUI).

Detail Routing

Detail routing uses the general pathways suggested by global routing and track assignment to route the nets (paths and contacts).

To perform detail routing, use the `route_detail` command (or choose Route > Detail Route in the GUI).

By default, standalone detail routing

- Performs track assignment before detail routing

  To skip track assignment, use the `-track_assign skip` option.

- Does not have a runtime limit

  If you need to limit the runtime, specify the maximum number of minutes by using the `-run_time_limit` option.

- Uses one CPU

  If you are performing distributed routing, specify the number of CPUs by using the `-num_cpus` option. For information about setting up for distributed routing, see “Enabling Distributed Routing” on page 2-18.

- Does not perform any search and repair passes

  To change the maximum number of search and repair passes, use the `-search_repair_loop` option.

Search and Repair

During search-and-repair routing passes, the classic router searches for DRC violations and reroutes wires in an effort to avoid violations. The classic router does not add metal stubs on frozen nets to fix DRC violations, even when the nets have DRC violations.

To run standalone search-and-repair routing, use the `route_search_repair` command (or choose Route > Search-and-Repair Route in the GUI). By default, the `route_search_repair` command performs 50 search-and-repair passes. To change the maximum number of passes, use the `-loop` option.

If you need to limit the search-and-repair runtime, specify the maximum number of minutes by using the `-run_time_limit` option. You can reduce the turnaround time for running search-and-repair routing by using distributed routing. To use distributed routing, specify the number of CPUs by using the `-num_cpus` option. For information about setting up for distributed routing, see “Enabling Distributed Routing” on page 2-18.

For more information about the `route_search_repair` command, see the man page.

Wire and Via Optimization

To optimize wires and vias, use the `optimize_wire_via` command (or choose Route > Optimize Wire Via in the GUI). By default, the `optimize_wire_via` command performs 20 search-and-repair passes. To change the maximum number of search-and-repair passes, use the `-search_repair_loop` option.

If you need to limit the wire and via optimization runtime, specify the maximum number of minutes by using the `-run_time_limit` option. You can reduce the turnaround time for running wire and via optimization by using distributed routing. To use distributed routing, specify the number of CPUs by using the `-num_cpus` option. For information about setting up for distributed routing, see "Enabling Distributed Routing" on page 2-18.

For more information about the `optimize_wire_via` command, see the man page.

Analyzing Congestion

To help you analyze the congestion in your design, the tool can generate both an ASCII congestion report, as shown in Example 3-1 and a visual congestion map, as shown in Figure 3-1 on page 3-30. You can generate the congestion report and congestion map for each routing stage: global routing, track assignment, and detail routing. The more routing stages that have been completed, the more accurate the congestion information is; however, the harder it is to fix the congestion issues.

Generating a Congestion Report

To generate a congestion report, run the `report_congestion` command.

```
icc_shell> report_congestion
```

By default, this command generates a global route congestion report by running global routing and reports a summary of the overflow information for the entire design, as shown in Example 3-1.

Example 3-1 Default Global Route Congestion Report

```
Information: Reporting global route congestion data from Milkyway...

Both Dirs: Overflow = 39 Max = 8 (1 GRCs) GRCs = 14 (0.26%)
H routing: Overflow = 6 Max = 2 (2 GRCs) GRCs = 4 (0.07%)
V routing: Overflow = 33 Max = 8 (1 GRCs) GRCs = 10 (0.18%)
```

In the congestion report, the Overflow value is the total number of wires in the design that do not have a corresponding track available; the Max value corresponds to the highest number of overutilized wires in a single global routing cell; and the GRCs value is the total number of overcongested global routing cells in the design.

The generated congestion information, but not the global routing results, is stored in the Milkyway design database. To display the congestion report in the design database instead of regenerating a congestion report, use the `-no_reroute` option when you run the `report_congestion` command.

You can report additional information in the congestion report by using the options shown in Table 3-6.

Table 3-6 Generating Additional Congestion Information

Option	Description
`-grc_based`	Provides detailed information about the 10 worst global routing cells for each category: total, horizontal, and vertical.
`-grc_number`	Provides detailed information about the specified number of worst global routing cells for each category.
`-overflow_threshold`	Provides detailed information about the global routing cells that exceed the specified overflow threshold, up to a maximum of 10 global routing cells for each category.
`-by_layer`	Generates a congestion report for each routing layer. The type of report, summary or GRC-based, generated for each layer depends on the other options that you specify.

Example 3-2 shows an example of a congestion report with detailed information about the global routing cells. This same format is used when you specify the `-grc_based`, `-grc_number`, or `-overflow_threshold` options; the only difference is the number of global routing cells reported in each category.

Example 3-2 Global Route Congestion Report With Global Routing Cell Details

```
Information: Reporting global route congestion data from Milkyway...

There are 2704 GRCs (Global Route Cell) in this design
************************************************************
****************** GRC based congestion report ****************
************************************************************
 14 GRCs with overflow: 4 with H overflow and 10 with V overflow

++++++++ Top 10 congested GRCs ++++++++

 GRC {624345 676005 650175 701835}:
   H routing capacity/demand =  81/81 (occupy rate = 1.00),
   V routing capacity/demand =  54/62 (occupy rate = 1.15) with overflow 8
 GRC {624345 701835 650175 727665}:
   H routing capacity/demand =  74/74 (occupy rate = 1.00),
   V routing capacity/demand =  54/59 (occupy rate = 1.09) with overflow 5
...
 GRC {676005 391875 701835 417705}:
   H routing capacity/demand =  12/14 (occupy rate = 1.17),
   V routing capacity/demand =  3/3 (occupy rate = 1.00) with overflow 2

++++++++ Top 10 horizontally congested GRCs ++++++++
 GRC {650175 391875 676005 417705}:
   H routing capacity/demand =  12/14 (occupy rate = 1.17) with overflow 2
 GRC {676005 391875 701835 417705}:
   H routing capacity/demand =  12/14 (occupy rate = 1.17) with overflow 2
...
 GRC {727665 882645 753495 908475}:
   H routing capacity/demand =  91/91 (occupy rate = 1.00)

++++++++ Top 10 vertically congested GRCs ++++++++
 GRC {624345 676005 650175 701835}:
   V routing capacity/demand =  54/62 (occupy rate = 1.15) with overflow 8
 GRC {624345 701835 650175 727665}:
   V routing capacity/demand =  54/59 (occupy rate = 1.09) with overflow 5

1
```

You can generate a congestion report for a portion of the design by using the `-coordinates` option to specify a rectangular area or the `-polygon` option to specify a rectilinear area.

You can also generate congestion reports based on the track assignment and detail routing in the design by setting the `-routing_stage` option to `track` or `detail`, respectively. When you generate these reports, the tool does not run track assignment or detail routing; it uses the existing information in the design. If the track assignment or detail route information does not already exist in the design database, the tool issues an error message and does not generate a congestion report.

For more information about the `report_congestion` command, see the man page.

Generating a Congestion Map

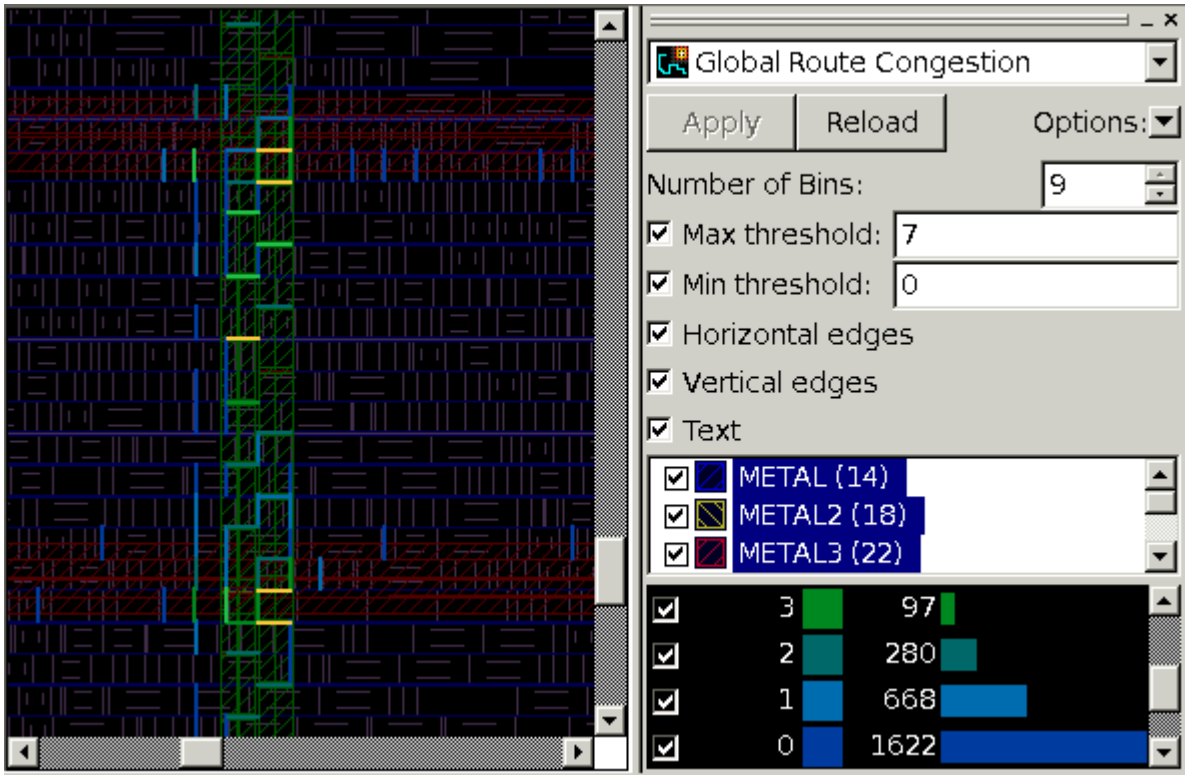
To display the global route congestion map, choose Route > Global Route Congestion Map in the GUI. If the design database contains global route congestion information, the tool generates the congestion map based on this information; otherwise, you must click Reload to generate the congestion map. When you click Reload, the tool opens a dialog box that contains the following command:

```
report_congestion -grc_based -by_layer -routing_stage global
```

When you click OK in this dialog box, the tool generates a new congestion map. If you want to use different options for the `report_congestion` command, you can modify this command before clicking OK.

Figure 3-1 shows an example of a congestion map.

Figure 3-1 Global Route Congestion Map



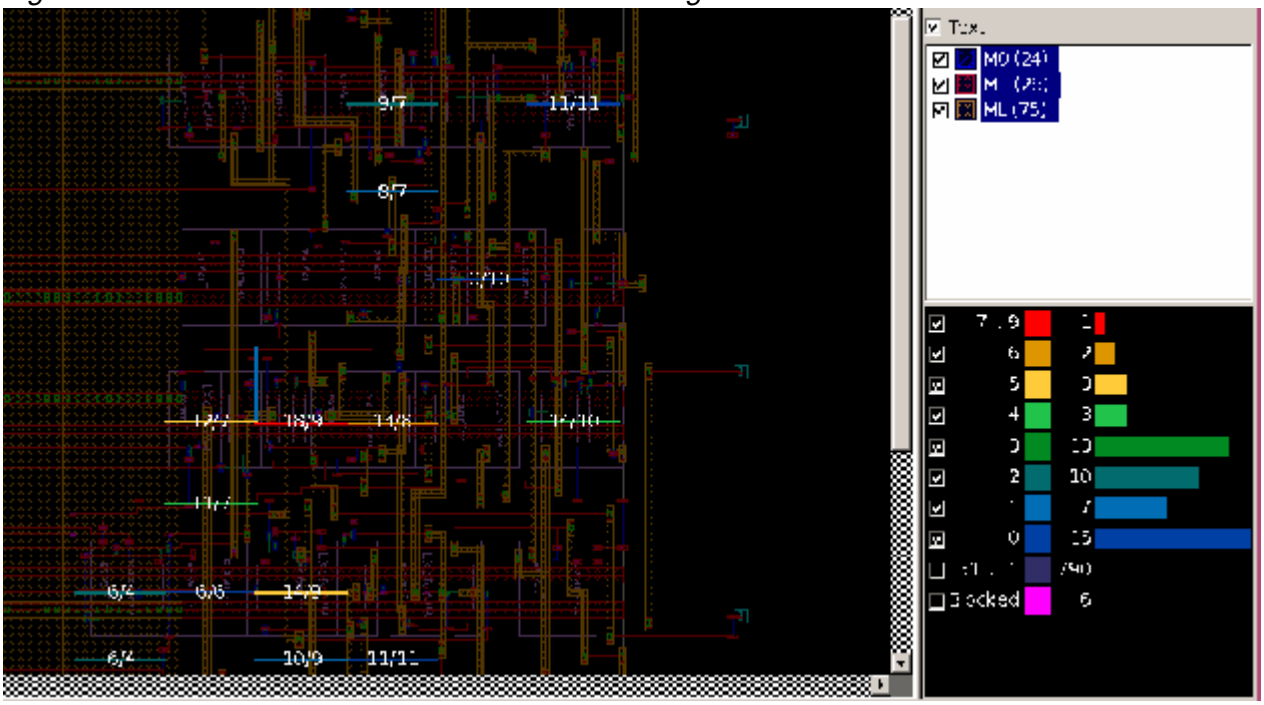
The congestion map shows the borders between global routing cells highlighted with different colors that represent different levels of overflow. The overflow values shown in the congestion map are determined by combining the overflow and underflow of all selected layers; they do not represent the maximum overflow values.

By default, all metal layers are selected in the congestion map. To display the congestion map for a subset of layers, select or deselect the layers on the Map Mode panel. For example, if the global routing report shows that the maximum overflow occurs on metal layer 2, you can deselect all layers, except for metal 2, to display only the metal 2 congestion.

The Map Mode panel also displays a histogram showing the number of global routing cells in different ranges (bins) of overflow values for the selected layers. You can select which bins to display in the congestion map by selecting or deselecting them on the Map Mode panel.

If the design shows congested areas, zoom into the congested area to see the overflow number on the global routing cell. For example, in Figure 3-2, the red highlight on the edge of the global routing cell shows 18/9. This means there are 9 wire tracks available, but 18 tracks are needed.

Figure 3-2 Global Route Overflow on Global Routing Cell

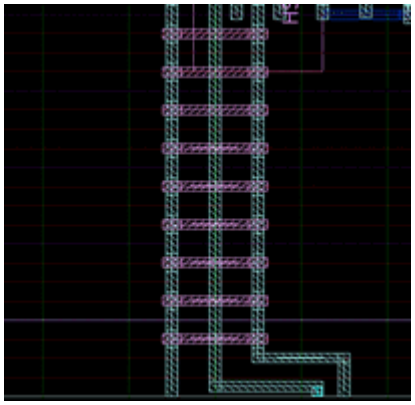


You can also generate congestion maps based on the track assignment and detail routing in your design. To generate a congestion map based on track assignment, choose Route > Track Assign Congestion Map in the GUI. To generate a congestion map based on detail routing, choose Route > Detail Route Congestion Map in the GUI. When you generate these congestion maps, the tool does not run track assignment or detail routing; it uses the existing information in the design. If the track assignment or detail route information does not already exist in the design database, the tool issues an error message and does not generate a congestion map.

Shielding Nets

The router shields routed nets by generating shielding wires that are based on the shielding widths and spacing defined in the shielding rules. In addition to shielding nets on the same layer, you also have the option to shield one layer above or one layer below or the layer above and the layer below. Shielding above or below the layer is called *coaxial shielding*. Coaxial shielding provides even better signal insulation than same-layer shielding, but it uses more routing resources. Figure 3-3 shows an example of coaxial shielding.

Figure 3-3 Coaxial Shielding



You can perform shielding either before or after signal routing. Shielding before signal routing, which is referred to as preroute shielding, provides better shielding coverage but can result in congestion issues during signal routing. Preroute shielding is typically used to shield critical clock nets. Shielding after signal routing, which is referred to as postroute shielding, has a very minimal impact on routability, but provides less protection to the shielded nets.

Before you perform shielding, you must

1. Define the shielding rules.

   To define shielding rules, use the `-shield_spacings` and `-shield_widths` options of the `define_routing_rule` command. For example, to specify a shielding rule that uses spacing of 0.1 microns and width of 0.1 microns for metal1 through metal5 and spacing of 0.3 microns and width of 0.3 microns for metal6, enter the following command:

   ```
   icc_shell> define_routing_rule shield_rule \
      -shield_widths {M1 0.1 M2 0.1 M3 0.1 M4 0.1 M5 0.1 M6 0.3} \
      -shield_spacings {M1 0.1 M2 0.1 M3 0.1 M4 0.1 M5 0.1 M6 0.3}
   ```

   For more information about the `define_routing_rule` command, see “Defining Nondefault Routing Rules” on page 2-7.

2. Assign shielding rules to the nets to be shielded.

   To avoid congestion issues and achieve the best balance of DRC convergence and timing closure, apply shielding rules only to high-frequency or critical clock nets and apply double-spacing rules to the lower-frequency clock nets.

   Use the `set_clock_tree_options` command to assign shielding rules to clock nets. For signal nets, use the `set_net_routing_rule` command to assign the shielding rules.

   Note:

   The `set_clock_tree_options` command assigns shielding rules to clock nets only before clock tree synthesis. After clock tree synthesis, you must use the `set_net_routing_rule` command to assign shielding rules to clock nets.

   For more information about these commands, see "Applying Nondefault Routing Rules" on page 2-8.

To shield nets with defined rules, use the `create_auto_shield` command (or choose Route > Create Shield in the GUI).

By default, the `create_auto_shield` command performs same-layer shielding on all nets with predefined shielding rules. The shielding wires are tied to ground, standard cell power and ground pins, and standard cell rails.

To explicitly specify the nets on which to perform shielding, use the `-nets` option. To perform coaxial shielding, use the `-coaxial_above` and `-coaxial_below` options. To tie the shielding wires to a named power or ground net, use the `-with_ground` option. To prevent connections to the standard cell power and ground pins, use the `-ignore_shielding_net_pins` option. To prevent connections to the standard cell rails, use the `-ignore_shielding_net_rails` option.

For example, to perform coaxial shielding below the shielded net segment layer on the clock nets and tie the shielding wires to VSS, enter

```
icc_shell> create_auto_shield -nets $clock_nets \
   -coaxial_below true -with_ground VSS
```

Performing ECO Routing

Whenever you modify the nets in your design, you need to run engineering change order (ECO) routing to reconnect the routing.

To run ECO routing, use the `route_eco` command (or choose Route > ECO Route in the GUI).

By default, the ECO router connects open nets and then fixes design rule violations in the design. All the open nets are run sequentially through global routing, track assignment, and detail routing. The ECO router can reroute existing wires to fix DRC violations. You can modify the default behavior by using the `route_eco` command options.

Table 3-7 describes the `route_eco` command options. For more information, see the man page.

Table 3-7 route_eco Command Options

Command option	Description
`-no_global` (“Global route” check box in Run section in the GUI)	Skips global routing.
`-no_track` (“Track assignment” check box in Run section in the GUI)	Skips track assignment.
`-no_detail` (“Detail route” check box in Run section in the GUI)	Skips detail routing.
`-auto` (Auto check box in the Run section in the GUI)	Enables automatic region-based ECO routing. You must use this option with the `-region_based` option.
`-region_based` *string* (“Region based nets filename” box in the GUI)	Specifies the name of the file that contains the list of nets on which to perform region-based ECO routing. You must use this option with the `-auto` option.
`-search_repair_loop` *int* (“Maximum Search & Repair cycles” box in the GUI)	Specifies the maximum number of search and repair loops. You must specify a value between 0 and 500. The default is 20.

Table 3-7 route_eco Command Options (Continued)

Command option	Description
`-reroute any_nets | modified_nets_only | modified_nets_first_then_others` (“Reroute nets” radio buttons in the GUI)	Controls which nets can be rerouted to fix DRC violations. The default is `any_nets`.
`-utilize_dangling_wires` (“Utilize dangling wires” check box in the GUI)	Enables the reuse of existing dangling routes to fix open nets.
`-freeze_routing_on_layer` *`list_of_layers`* (“Freeze routing on layers” list in the GUI)	Prevents routing changes on the specified layers.
`-freeze_vias_on_frozen_metal` (“Freeze vias on frozen metal” check box in the GUI)	Freezes vias on frozen metal layers.
`-num_cpus` *`int`* (“Number of CPUs” box in the GUI)	Specifies the number of CPUs to use for distributed routing. You must specify a value between 1 and 63. For information about setting up for distributed routing, see “Enabling Distributed Routing” on page 2-18. The default is 1.

Reporting Cell Placement and Routing Statistics

You can report on cell placement and routing statistics to help you determine whether to perform further optimizations to improve timing. For example, if the statistics show that an area has high congestion, an additional optimization might not have room to add or size up standard cells.

To view the place and route summary report, run the `report_design_physical -verbose` command.

Verifying the Routed Design

After you have completed routing with the classic router, you can check the design for routing design rule violations.

Note:

The classic router uses center lines to determine connectivity. If your design contains wires with zero wire extension, the `verify_route` command might report open and spacing violations due to these zero extension wires. To fix this problem, use the `convert_wire_ends` command to convert all wires with zero extension into wires with half-width extensions. If the `convert_wire_ends` command cannot convert a wire, it issues a warning message and removes the wire from the design cell. Because the `convert_wire_ends` command modifies the wire end information and, in some cases, removes wires, write commands such as `write_def` generate output files that are different from the original input files. For more information about the `convert_wire_ends` commands, see the man page.

The IC Compiler tool provides the following methods for verifying the routed design:

- Use the IC Validator or Hercules tool to check the routing design rules defined in the foundry runset.

  To use this method, run the `signoff_drc` command or choose Verification > Signoff DRC in the GUI.

  Note:

  An IC Validator or Hercules license is required to run the `signoff_drc` command.

- Use the Hercules tool to check the routing design rules defined in the Milkyway technology file.

  To use this method, run the `verify_drc` command or choose Verification > DRC in the GUI.

  Note:

  A Hercules license is required to run the `verify_drc` command.

- Use the IC Compiler DRC engine to check the routing design rules defined in the Milkyway technology file.

  To use this method, run the `verify_route` command or choose Route > Verify Route in the GUI.

- Import the DRC results from the Calibre tool.

The following sections describe these methods.

Using the signoff_drc Command

Signoff design rule checking runs the IC Validator or Hercules tool within the IC Compiler tool to check the routing design rules defined in the foundry runset. To perform signoff design rule checking, run the `signoff_drc` command (or choose Verification > Signoff DRC in the GUI).

Note:
An IC Validator or Hercules license is required to run the `signoff_drc` command.

To use the `signoff_drc` command to perform design rule checking,

- Set up the validation tool environment.
- Set up the physical signoff options.
- Run the `signoff_drc` command (or choose Verification > Signoff DRC in the GUI).

The following sections describe these tasks.

Setting Up the Validation Tool Environment

You can use either the IC Validator or Hercules tool to check the routing design rules with the `signoff_drc` command.

Setting Up the IC Validator Environment

To use the IC Validator tool when running the `signoff_drc` command, you must have an IC Validator license, and you must specify the location of the IC Validator executable by setting the `ICV_HOME_DIR` environment variable. You can set this variable in your .cshrc file. To specify the location of the IC Validator executable, use commands similar to those shown in the following example:

```
setenv ICV_HOME_DIR /root_dir/icv
set path = ($path $ICV_HOME_DIR/bin/AMD.64)
```

You must ensure that the version of the IC Validator executable that you specify is compatible with the IC Compiler version that you are using.

For more information about the IC Validator tool, see the IC Validator documentation, which is available on SolvNet.

Setting Up the Hercules Environment

To use the Hercules tool when running the `signoff_drc` command, you must have a Hercules license, and you must specify the location of the Hercules executable by setting the `HERCULES_HOME_DIR` environment variable. You can set this variable in your .cshrc file. For example,

```
setenv HERCULES_HOME_DIR /root_dir/hercules
set path = ($path $HERCULES_HOME_DIR/bin/AMD.64)
```

You must ensure that the version of the Hercules executable that you specify is compatible with the IC Compiler version that you are using. For more information about the Hercules tool, see the Hercules documentation, which is available on SolvNet.

Setting Up the Physical Signoff Options

To set up the physical signoff options, use the `set_physical_signoff_options` command (or choose Verification > Set Physical Signoff Options in the GUI). The settings that you specify are saved in the Milkyway design library.

Table 3-8 shows the physical signoff options that apply to the `signoff_drc` command.

Table 3-8 Physical Signoff Options for signoff_drc

Command option	Description
`-exec_cmd icv | hercules` (“Executable name” box in the GUI)	Specifies the validation tool to use for design rule checking (optional). If you do not specify this option, the `signoff_drc` command uses the value stored in the Milkyway design library. If there is no value stored in the Milkyway design library, an error occurs.
`-drc_runset` *`file_name`* (DRC box in the GUI “Foundry runset” area)	Specifies the foundry runset to use for design rule checking (required).
`-mapfile` *`file_name`* (“Layer mapping file” box in the GUI)	Specifies the layer mapping file (optional). For more information, see the next section, “Defining the Layer Mapping Between Milkyway and the Runset File.”

To report the current settings for the physical signoff options, use the `report_physical_signoff_options` command.

Defining the Layer Mapping Between Milkyway and the Runset File

In general, the Milkyway technology file and the runset file used by the IC Validator or Hercules tool use the same layer numbers. If they do not, you must supply a layer-mapping file to map the Milkyway layers to the layers used in the runset file.

The `signoff_drc` command accepts a layer-mapping file in IC Validator, Hercules, or Milkyway format.

- The syntax of the mapping information in an IC Validator or Hercules mapping file is

  `Milkyway_layer_list > runset_layer_list`

  The rules for specifying the layer lists are

  - Each layer in a layer list must be separated by a space. Or, if specified in a range, the range must consist of a starting layer and an ending layer joined by a dash (-).
  - Generally, parentheses should surround the layer list. However, you can omit parentheses if you specify only one layer.

  You can include comments in a layer mapping file by preceding the comment with a pound sign (#). All text that follows a pound sign on a given line is a comment.

- The syntax of the mapping information in a Milkyway mapping file is

  `MilkywayObjType MilkywayLayer``[:`*`DataType`*`]` *`RunsetLayer`*`[:`*`RunsetDataType`*`]`

  The *MilkywayObjType* argument is a string of one to three characters that identifies the Milkyway objects to be translated. The required first character is the code for the type of Milkyway object to be translated: A for all, T for text, or D for data. The optional second character specifies the net type, such as S for signal, P for power, or G for ground. An optional third character specifies the pin code. The Milkyway layer is a name or an integer. The runset layers and data types are integers.

  You can add a comment to the file by inserting a semicolon. Text is ignored from the semicolon to the end of the line.

  You can save the Milkyway layer mapping file in the Milkyway design library by using the `set_stream_layer_map_file -format out` command. When the Milkyway design library contains a layer mapping file, it defines the default mapping. The layer mapping file specified by the `set_physical_signoff_options -mapfile` option overrides the layer mapping file saved in the Milkyway design library.

  For more information about the Milkyway mapping file, see the “Layer Mapping File: Milkyway to GDSII or OASIS” section in Chapter 3 of the *Library Preparation for IC Compiler User Guide*.

Performing Distributed Design Rule Checking

By default, the signoff_drc command uses a single process to perform design rule checking. To reduce the turnaround time used for design rule checking, you can use distributed processing. To enable distributed processing, you must define the distributed processing configuration by using the set_host_options command.

If you have defined more than one distributed processing configuration with the set_host_options command, the signoff_drc command selects the IC Validator processing method in the following order of priority:

1. Job submission through a user-defined distributed processing script

   To enable job submission using your own script with the set_host_options command, use the -submit_command option to specify the location of your job submission script. For example, to specify a configuration named custom4 that enables a maximum of four processes using your job submission script, enter the following command:

   ```
   icc_shell> set_host_options -name custom4 -num_processes 4 \
      -submit_command /usr/local/bin/my_submit_command
   ```

2. Job submission through the Load Sharing Facility (LSF) or the Sun Grid Engine (SGE)

   To enable LSF or SGE job submission with the set_host_options command, use the -pool option to specify the mode and the -num_processes option to specify the maximum number of processes.

   For example, to specify a configuration named lsf4 that enables a maximum of four processes using LSF, enter the following command:

   ```
   icc_shell> set_host_options -name lsf4 -pool lsf -num_processes 4
   ```

   To specify a configuration named grd4 that enables a maximum of four processes using SGE, enter the following command:

   ```
   icc_shell> set_host_options -name grd4 -pool grd -num_processes 4
   ```

3. Distributed processing on the specified hosts

   To enable distributed processing with the set_host_options command, specify the hosts to use and use the -num_processes option to specify the maximum number of processes on each host. For example, to specify a configuration named dp4 that enables a maximum of four processes, with a maximum of two processes each on machineA and machineB, enter the following command:

   ```
   icc_shell> set_host_options -name dp4 -num_processes 2 \
      {machineA machineB}
   ```

4. Multithreading

   To enable multithreading on the current machine with the set_host_options command, use the -max_cores option to specify the number of threads. For example, to specify a

configuration named mt4 that enables a maximum of four threads on the current machine, enter the following command:

```
icc_shell> set_host_options -name mt4 -max_cores 4
```

To ensure that you are using the intended distributed processing configuration, remove the current configurations by using the `remove_host_options` command before defining the distributed processing configuration for the `signoff_drc` command. To report the current distributed processing configurations, use the `report_host_options` command.

For more information about the `set_host_options` command, see Chapter 2, “Working With the IC Compiler Tool,” in the *IC Compiler Implementation User Guide*.

Running the signoff_drc Command

By default, the `signoff_drc` command uses the FRAM view to check all cell types (standard cells, macro cells, and I/O pad cells) on all routing layers of the entire chip for all rules specified in the foundry runset. To use the CEL view instead of the FRAM view, use the `-read_cel_view` option or select the “CEL view” option in the GUI. Using the CEL view can expose problems that are masked by the FRAM view abstraction.

Note:

The `signoff_drc` command uses the on-disk design information, not the design information in memory. You must save the current state of the design before running the `signoff_drc` command.

You can specify additional options for the Hercules or IC Validator command line by using the `-user_defined_options` option when you run the `signoff_drc` command. The string that you specify in this option is added to the command line used to invoke design rule checking in the Hercules or IC Validator tool. The IC Compiler tool does not perform any checking on the specified string.

By default, the IC Validator tool checks for both top-level and child-level errors and reports a maximum of 1000 DRC errors per rule. To ignore child-level errors, use the `-ignore_child_cell_errors` option. You can override the maximum error count by using the `-max_errors_per_rule` option when you run the `signoff_drc` command.

You can restrict the design rule checking to specific areas of the chip, to specific design rules, and to specific layers.

- To restrict the checking to specific areas of the chip,
  - If you are using the command line, use the `-bounding_box` option to specify the coordinates of each area to check. To exclude checking in specific areas, use the `-excluded_bounding_boxes` option. For either option, you can specify multiple areas by specifying the coordinates for each area.
  - If you are using the GUI, specify the coordinates for each area to check in the “Included bounding boxes” list and specify the coordinates for each area to exclude

from checking in the “Excluded bounding boxes” list. To specify an area, either draw the bounding box or enter the x- and y-coordinates of the box. For each area, you can also enable or disable snapping. If snapping is enabled, you can specify that the bounding box should snap to the minimum grid (the default), placement site, routing track, middle routing track, or user grid.

- To restrict the checking to specific rules,
  - To check only the specified rules, specify the rules in the `-select_rule` option (or in the Pattern box in the “To use” section of the Rules area in the GUI).
  - To exclude checks for specific rule, specify the rules in the `-unselect_rule` option (or in the Pattern box in the Excluded section of the Rules area in the GUI).

  In either case, you specify the rules by specifying a matching pattern for the rule names. The rule names are specified in the COMMENT section in the runset file. You can use these options together to customize the set of rules checked by the `signoff_drc` command.

  For example, to restrict the `signoff_drc` command to route validation, select the metal layer rules and exclude the metal density rules.
- To restrict the checking to specific layers,
  - Specify the layers in the `-select_layers` option (or in the list associated with the “Selected layers” radio button in the “Check DRC violations on” area in the GUI).
- To restrict the checking to specific cell types,
  - Specify the cell types to exclude in the `-excluded_cell_types` option (or select the associated check box in the “Excluded cell types” area in the GUI).

After you complete design rule checking on the routing layers, you can perform a quick design rule check on the Milkyway database by rerunning the `signoff_drc` command with the `-check_all_layers` option. If you are using the GUI, set this option by selecting the “All runset layers (routing and device layers)” radio button in the “Check DRC violations on” area. When you use this option, the design information comes from the CEL view and the validation tool checks all layers, including the nonrouting layers.

The `signoff_drc` command creates a Milkyway error view that you can use to report or display the DRC violations. By default, the error view generated by the `signoff_drc` command is saved in a file called *design*_sdrc.err. You can control this name by using the `-error_view` option (or the “Error cell name” box in the GUI).

To make it easier to view specific violations in the error browser, you can enable error categorization for the generated error view. When you enable this feature, the violations in the error view are marked with their associated net type and route type. You can use these markings to filter the violations displayed in the error browser. To enable this feature, use the `-categorize_errors true` option when you run the `signoff_drc` command. Note that

using this feature slightly increases the runtime for the `signoff_drc` command. For information about using the error view to debug DRC violations, see "Analyzing DRC Violations" on page 3-47.

In addition to the error view, the `signoff_drc` command generates the following files, which you can use for debugging, and places them in a directory called signoff_drc_run under the current working directory:

- *design*.errsum or *design*.LAYOUT_ERRORS

  These files contain an error summary report.

- sdrc.ev

  This file contains the runset file, including any options added by the `signoff_drc` command.

- sdrc.log

  This file contains any errors that occurred during the `signoff_drc` run.

- ./run_details/*design*.sum

  This file contains the detailed log file from the validation tool.

You can control the directory location by using the `-run_dir` option (or the "Run directory" box in the GUI). You can specify either a relative path, in which case the directory is created under the current working directory, or an absolute path.

Using the verify_drc Command

If you do not have a foundry runset, you can use the `verify_drc` command to perform design rule checking with a runset based on the design rules defined in the Milkyway technology file.

Note:

The `verify_drc` command does not check design rules for technologies at the 45-nm process node and below. To check design rules for these technologies, you must use the `signoff_drc` command.

For information about the routing design rules and the technology file attributes that define them, see the *IC Compiler Technology File and Routing Rules Reference Manual*.

The `verify_drc` command generates a Hercules DRC runset based on the design rules defined in the Milkyway technology file, and then runs the Hercules tool to check for DRC violations. A Hercules license is required to run the `verify_drc` command. For more information about the Hercules tool, see the Hercules documentation, which is available on SolvNet.

To use the `verify_drc` command to perform design rule checking,

- Set up the Hercules environment.

  For information about setting up the Hercules environment, see "Setting Up the Validation Tool Environment" on page 3-37.

- Run the `verify_drc` command.

By default, the `verify_drc` command checks and reports the DRC violations for the entire design. To perform these checks only in a specific region, specify the coordinates of the region by using the `-selected_area` option (or by specifying the coordinates in the "Selected area" box in the "DRC Area" tab in the GUI). To perform these checks for the entire design, except for a specific region, specify the coordinates of the excluded region by using the `-excluded_area` option (or by specifying the coordinates in the "Whole chip except area" box in the "DRC Area" tab in the GUI).

Table 3-9 lists the design rules checked by the `verify_drc` command.

Table 3-9 Design Rules Checked by verify_drc

Rule	Checked by default	Command option to enable or disable (Check box in the "DRC Rules" tab in GUI)
Minimum and maximum width	Yes	`-ignore_width` ("Minimum and maximum width rule")
Metal spacing (includes minimum spacing, same-net spacing, fat metal spacing, the fat metal parallel length rule, and the fat metal extension range rule)	Yes	`-ignore_spacing` ("All spacing related rules")
Minimum area	Yes	`-ignore_area` ("Minimum area")
Metal density	Yes	`-ignore_density` ("Metal density rules")
Minimum enclosed area (default and fat metal)	Yes	`-ignore_enclosed_area` ("Basic minimum enclosed area rule and fat table minimum enclosed area rule")
Minimum number of consecutive short edges	Yes	`-ignore_min_edge_length` ("Minimum number of consecutive short edges rule")

Table 3-9 Design Rules Checked by verify_drc (Continued)

Rule	Checked by default	Command option to enable or disable (Check box in the "DRC Rules" tab in GUI)
Blockages on metal layers	No	`-check_blockage` ("Blockages on metal layers")
Via size rule	No	`-check_via_size` ("Via size rule")
Via spacing (includes minimum spacing, the fat contact rule, the fat contact spacing rule, and the fat contact extension range rule)	Yes	`-ignore_via_spacing` ("Via spacing rules")
Maximum number of via stack layers	Yes	`-ignore_stack_level` ("Maximum number of stack layer rule")
Stack via rule	Yes	`-ignore_stackable` ("Stack via rule")
Maximum number of adjacent vias and enclosed via rules	Yes	`-ignore_adjacent_via` ("Maximum number of adjacent via and enclosed via rules")
Minimum number of fat vias	Yes	`-ignore_min_via_number` ("Minimum number of vias defined in the fat contact table (including extension range)")
Via array size and spacing between via arrays	No	`-check_via_farm` ("Via array size and spacing between two via arrays")
Via enclosure rule	No	`-check_enclosure` ("Via Enclosure rule")
End-of-line rule	No	`-check_end_of_line` ("End of line rule")
Fat poly contact rule	No	`-check_fat_poly_contact` ("Fat poly contact rule")

The Hercules runset file generated by the `verify_drc` command is saved in adrc/adrc.ev. You can change the directory to which the runset file is saved by using the `-dir` option (or the “Hercules runset directory” box in the GUI).

The `verify_drc` command creates a Milkyway error view that you can use to report or display the DRC violations. By default, the `verify_drc` command saves the generated error view in a file called *design*_adrc.err. You can control this name by using the `-error_cell` option (or the “Error cell name” box in the GUI). For information about using the error view to debug DRC violations, see “Analyzing DRC Violations” on page 3-47.

Using the verify_route Command

The `verify_route` command checks for routing DRC violations by running an internal design rule checker that checks the design rules defined in the Milkyway technology file. The `verify_route` command also checks for open nets.

Note:

Unlike the `signoff_drc` and `verify_drc` commands, which check the design rules for all objects, the `verify_route` command checks only objects with a signal routing type. For more information about routing types, see “Setting Routing Types” on page 2-13.

For information about the routing design rules and the technology file attributes that define them, see the *IC Compiler Technology File and Routing Rules Reference Manual*.

By default, the `verify_route` command checks and reports the DRC violations and open nets for the entire design. To perform these checks only in a specific region, specify the coordinates of the region by using the `-bounding_box` option (or by specifying the coordinates in the “DRC Area” area in the GUI). To prevent checking of the DRC violations, use the `-no_drc` option when you run `verify_route`. To prevent checking for open nets, use the `-no_opens` option when you run `verify_route`.

You can reduce the `verify_route` runtime by using distributed processing. For information about how to use distributed processing, see “Enabling Distributed Routing” on page 2-18.

You can also use the `verify_route` command to check for antenna violations. For more information about checking for and fixing antenna violations, see “Preventing Antenna Problems” on page 4-2.

The `verify_route` command can create a Milkyway error view that you can use to report or display the DRC violations. To output a Milkyway error view, use the `-output` *`file_name`* option when you run the `verify_route` command. For information about using the error view to debug DRC violations, see “Analyzing DRC Violations” on page 3-47.

Using the Calibre Interface

You can perform design rule checking with the Calibre tool, convert the Calibre DRC error file to a Milkyway error view, and then report or view the errors in the IC Compiler tool. To convert the Calibre DRC error file to a Milkyway error view, use the `read_drc_error_file` command (or choose Verification > Read Third-party DRC Error File in the GUI).

For example,

```
icc_shell> read_drc_error_file Calibre_error_file
```

By default, the command saves the generated Milkyway error view in a file called *cell_name*.err, where *cell_name* is derived from the Calibre error report. You can specify a name for the Milkyway error view by using the `-error_cell` option.

Note:

The `read_drc_error_file` command supports only flat Calibre DRC error files; it does not support Calibre hierarchy DRC error files.

You can load the Milkyway error view created by the `read_drc_error_file` command into the error browser to report or display the DRC violations. For more information about using the error browser, see “Examining Routing and Verification Errors” section in Appendix A of the *IC Compiler Implementation User Guide* and the “Examining Routing and Verification Errors” topic in IC Compiler Help. To view IC Compiler Help, choose Help > IC Compiler Online Help in the GUI.

Analyzing DRC Violations

After you have generated an error view, either by running one of the Synopsys design rule checking commands or by converting the Calibre results to an error view, you can analyze the DRC violations by

- Using the DRC query commands
- Using the error browser

The following sections describe these capabilities.

Using the DRC Query Commands

You can get information about DRC violations by using the `get_drc_errors` command to create a collection of DRC violations and then using the `get_attribute` command to query the attributes of the errors. Some attributes that provide information about DRC errors are `type`, `bbox`, `nets`, and `shapes`. Note that the availability of attribute values depends on the error type and the verification method used. For a list of all attributes associated with DRC errors, use the `list_attributes -application -class drc_error` command.

By default, the `get_drc_errors` command creates a collection that contains all DRC violations detected during the most recent run of the `verify_route` command. You can modify the query by

- Using the `-type` option to restrict the errors returned by the command to a specific error type. To determine the error type values, use the `list_drc_error_types` or `report_drc_error_type` commands.
- Using the `-bbox` option to restrict the query to a specific region of the design.
- Using the `-error_view` option to query a specific error view.

For example, to create a collection that contains all shorted nets detected by the most recent `verify_route` run, enter the following command:

```
icc_shell> get_attribute [get_drc_errors -type {Short}] nets
```

To use the DRC query commands on a nondefault error view, such as from a third-party DRC error file, you must first open the error view, either by loading it into the error browser or by using the `open_mw_cel -not_as_current` command. For example,

```
icc_shell> read_drc_error_file -error_cell Route_Opt.err error.file
icc_shell> set cellId [open_mw_cel -not_as_current Route_Opt.err]
icc_shell> report_drc_error_type -error_view $cellId
icc_shell> get_drc_errors -error_view Route_Opt.err
```

For more information about these commands, see the man pages.

Using the Error Browser

The error browser lets you examine routing design rule errors in the GUI. You can filter the error list, highlight errors in the layout view, display information about an error on the Query panel, select errors interactively with the Error Selection tool, mark errors as fixed, and save the errors in a file.

For more information about using the error browser, see

- The Error Browser usage help

  To view the Error Browser usage help, choose Help > Error Browser in the error browser window.

- The “Examining Routing and Verification Errors” topic in IC Compiler Help

  To view IC Compiler Help, choose Help > IC Compiler Online Help in the GUI.

- The “Examining Routing and Verification Errors” section in Appendix A of the *IC Compiler Implementation User Guide*

4

Using the Classic Router for Design for Manufacturing and Chip Finishing

This chapter describes how to use the classic router for design for manufacturing (DFM) and chip finishing tasks. It contains the following sections:

For information about additional chip finishing tasks, see the *IC Compiler Implementation User Guide*.

Preventing Antenna Problems

In chip manufacturing, gate oxide can be easily damaged by electrostatic discharge. The static charge that is collected on wires during the multilevel metallization process can damage the device or lead to a total chip failure. The phenomena of an electrostatic charge being discharged into the device is referred to as either antenna or charge-collecting antenna problems.

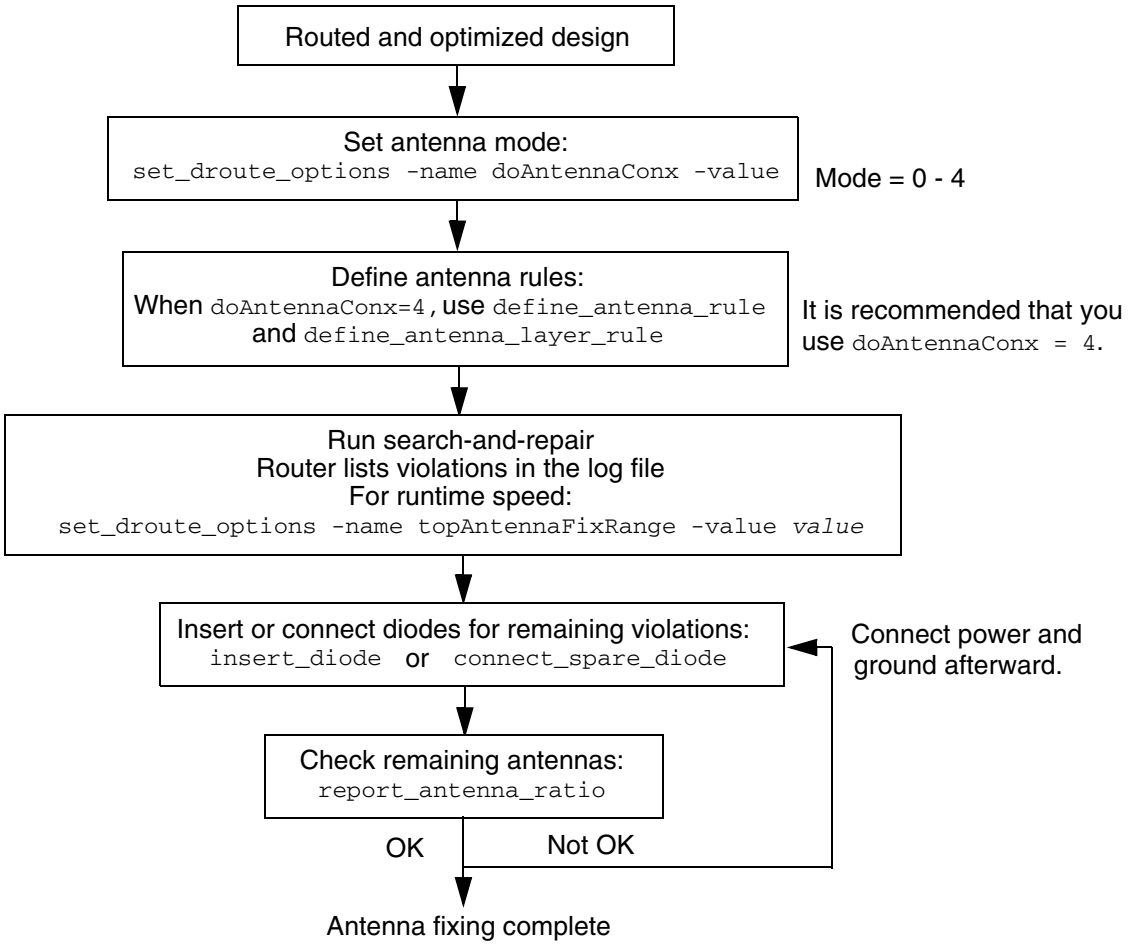
To prevent antenna problems, the IC Compiler tool verifies that for each input pin the metal antenna area divided by the gate area is less than the maximum antenna ratio given by the foundry:

(antenna-area)/(gate-area) < (max-antenna-ratio)

The antenna rules are categorized into basic and advanced rules. For process nodes above 0.18 microns, use the basic antenna rules.

Figure 4-1 shows how to use the IC Compiler tool to perform antenna checking and fixing.

Figure 4-1 IC Compiler Antenna Methodology



Setting the Antenna Mode

The IC Compiler tool supports single-layer, accumulative ratio, accumulative area, and advanced antenna modes.

Using the basic antenna rules, you can select one of the first three antenna modes and decide the area, which can be either surface (polygon) or sidewall, to use for antenna area calculation. Set the `doAntennaConx` detail route option to select an antenna mode and the `useSideWallForAntenna` (0 for surface area) detail route option to choose an area:

- Single-layer mode (Set `doAntennaConx` to 1)
- Accumulative ratio mode (Set `doAntennaConx` to 2)
- Accumulative area mode (Set `doAntennaConx` to 3)
- Advanced mode (Set `doAntennaConx` to 4)

This section discusses only the first three modes using the surface area. For the advanced antenna mode, use the `-mode` option to decide a mode and follow the advanced antenna rules. For more information, see "Setting the Antenna Mode for the Advanced Antenna Rules" on page 4-9.

Figure 4-2 shows a layout example with a lateral view, which is used to explain antenna modes in Table 4-1: single-layer mode, accumulative ratio mode, and accumulative area mode.

Figure 4-2 Layout Example

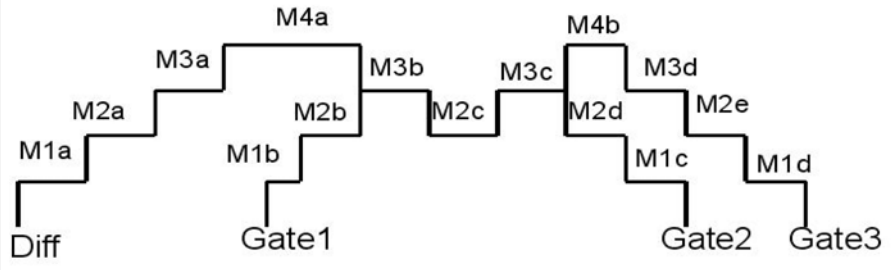


Table 4-1 Antenna Modes and Ratios

Antenna mode	Antenna ratio

Single-layer mode (ignores all lower-layer segments; allows best routability):
 	
`set_droute_options \`
`-name doAntennaConx \`
`-value 1`
	antenna_ratio = connected metal area of the layer / total gate area
	
	M1 ratios
	• Gate1: M1b / Gate1
	• Gate2: M1c / Gate2
	• Gate3: M1d / Gate3
	
	M2 ratios
	• Gate1: M2b / Gate1
	• Gate2: M2d / Gate2
	• Gate3: M2e / Gate3
	
	M3 ratios
	• Gate1, 2: (M3b + M3c) / (Gate1 + Gate2)
	• Gate3: M3d / Gate3
	
	M4 ratios
	• Gate1, 2, 3: (M4a + M4b) / (Gate1 + Gate2 + Gate3)

Table 4-1 Antenna Modes and Ratios (Continued)

Antenna mode	Antenna ratio

Accumulative ratio mode (includes lower-layer segments to the input pins):
 	
`set_droute_options \`
`-name doAntennaConx \`
`-value 2`
	antenna_ratio = accumulation of ratios for the layer and layers below
	
	M1 ratios
	• Gate1: M1b / Gate1
	• Gate2: M1c / Gate2
	• Gate3: M1d / Gate3
	
	M2 ratios
	• Gate1: M1b / Gate1 + M2b / Gate1
	• Gate2: M1c / Gate2 + M2d / Gate2
	• Gate3: M1d / Gate3 + M2e / Gate3
	
	M3 ratios
	• Gate1: M1b / Gate1 + M2b / Gate1 + (M3b + M3c) / (Gate1 + Gate2)
	• Gate2: M1c / Gate2 + M2d / Gate2 + (M3b + M3c) / (Gate1 + Gate2)
	• Gate3: M1d / Gate3 + M2e / Gate3 + M3d / Gate3
	
	M4 ratios
	• Gate1: M1b / Gate1 + M2b / Gate1 + (M3b + M3c) / (Gate1 + Gate2) + (M4a + M4b) / (Gate1 + Gate2 + Gate3)
	• Gate2: M1c / Gate2 + M2d / Gate2 + (M3b + M3c) / (Gate1 + Gate2) + (M4a + M4b) / (Gate1 + Gate2 + Gate3)
	• Gate3: M1d / Gate3 + M2e / Gate3 + M3d / Gate3 + (M4a + M4b) / (Gate1 + Gate2 + Gate3)

Table 4-1 Antenna Modes and Ratios (Continued)

Antenna mode	Antenna ratio

Accumulative area mode (includes all lower-layer segments; allows least routability):
 	
`set_droute_options \`
`-name doAntennaConx \`
`-value 3`
	antenna_ratio = all connected metal areas / total gate area
	
	M1 ratios
	• Gate1: M1b / Gate1
	• Gate2: M1c / Gate2
	• Gate3: M1d / Gate3
	
	M2 ratios
	• Gate1: (M1b + M2b) / Gate1
	• Gate2: (M1c + M2d) / Gate2
	• Gate3: (M1d + M2e) / Gate3
	
	M3 ratios
	• Gate1, 2: (M1b + M1c + M2b + M2c + M2d + M3b + M3c) / (Gate1 + Gate2)
	• Gate3: (M1d + M2e + M3d) / Gate3
	
	M4 ratios
	• Gate1, 2, 3: all metal areas / (Gate1 + Gate2 + Gate3)

Defining Antenna Rules

The IC Compiler tool reads the antenna data that is prepared by the Milkyway Environment tool to fix antenna violations. If a sidewall is used for antenna calculation, you need to define the metal thickness by specifying the `unitMinThickness`, `unitNomThickness`, and `unitMaxThickness` attributes in each layer section of the technology file.

The following three factors influence the computation of antenna rules:

- The antenna mode to identify which metal piece to count
  - Single-layer mode
  - Accumulative ratio mode
  - Accumulative area mode
  - Advanced mode

- The antenna area calculation
  - Surface area = W x L
  - Sidewall area = (W + L) x 2 x thickness
- Protection from diodes and pins

Basic Antenna Rules

The basic antenna rules define the maximum antenna ratio (antenna area to gate area) for metal layers and cut layers.

To use the basic antenna rules, set the `doAntennaConx` detail route option to 1, 2, or 3, depending on which metal pieces you want to count. For example, to set the antenna mode to single-layer mode, enter

```
icc_shell> set_droute_options -name doAntennaConx -value 1
```

You must also specify whether you want to use surface area or sidewall area. To select the area, set the `useSideWallForAntenna` detail route option to 0 (the default) for surface area. Set the option to 1 for sidewall area. Note that this option is ignored when `doAntennaConx` is set to 4 for advanced antenna mode.

After setting the antenna mode and the area, use the following detail route options to define the maximum antenna ratio:

- `maxAntennaRatio`

  This option defines the maximum antenna ratio for metal layers. For example,

  ```
  icc_shell> set_droute_options -name maxAntennaRatio -value 300
  ```

- `maxCutAntennaRatio`

  This option defines the maximum antenna ratio for cut layers. For example,

  ```
  icc_shell> set_droute_options -name maxCutAntennaRatio -value 20
  ```

To get a report on the options that have been set with the `set_droute_options` command, use the `report_droute_options` command as shown in the following example:

```
icc_shell> report_droute_options
```

The part of the report showing the antenna settings looks like this:

```
Integer Option for droute doAntennaConx 1
#       range [0,4], default=0, stored in cell;
;;      0: no charge-collecting antenna checking;
;;      [1-4] check and avoid charge-collecting antenna;
;;        1: ignore all lower-layer segments, (best routability);
;;        2: include lower-layer segments to the input pins;
;;        3: include all lower-layer segments (worst routability).
;;        4: (advanced mode) use antenna rules defined in libraries
;;      for a combination of mode 1 - 3 and limited diode protection
...
Integer Option for droute maxAntennaRatio 300
...
```

Advanced Antenna Rules

The advanced antenna rules give you more control over defining antenna rules. You can define the rules per layer and specify the diode ratios.

Setting the Antenna Mode for the Advanced Antenna Rules

To use the advanced antenna rules, set the `doAntennaConx` detail route option to 4 and define the antenna rules by using the `define_antenna_rule` and `define_antenna_layer_rule` commands. You specify which metal pieces to count during antenna calculation by using the `-mode` option. Table 4-2 shows the description for each mode.

Table 4-2 Antenna Models Using the -mode Option

Antenna mode	Description
Mode 1: Single-layer	Uses surface area and ignores all lower-layer segments.
Mode 2: Accumulative ratio	Uses surface area which includes all lower-layer segments to the input pins.
Mode 3: Accumulative area	Uses surface area which includes all lower-layer segments.
Mode 4: Single-layer	Uses sidewall area and ignores all lower-layer segments.
Mode 5: Accumulative ratio	Uses sidewall area which includes all lower-layer segments to the input pins.
Mode 6: Accumulative area	Uses sidewall area which includes all lower-layer segments.

For example, to set the advanced antenna mode and to use the advanced antenna rules, enter the following commands:

```
icc_shell> set_droute_options -name doAntennexConx -value 4

icc_shell> define_antenna_rule mw_lib -mode mode \
   -diode_mode diode_mode \
   -metal_ratio metal_ratio \
   -cut_ratio cut_ratio \
   [-protected_metal_scale metal_scale] \
   [-protected_cut_scale cut_scale]

icc_shell> define_antenna_layer_rule mw_lib -mode mode \
   -layer layer_name \
   -ratio ratio \
   [-pratio pratio] \
   [-nratio nratio] \
   -diode_ratio {v0 v1 v2 v3 [v4]} \
   [-scale_factor scale_factor]
```

Setting the Diode Mode

Antenna ratio calculation with diode protection is based on the vector `{v0 v1 v2 v3 [v4]}{s0 s1 s2 s3 s4 s5}` that you specify by using the `-diode_ratio` option of the `define_antenna_layer_rule` command. The vector defines the maximum allowable antenna ratio (max-antenna-ratio) of the antenna area to the gate area if the antenna is protected by diodes. The antenna ratio of each metal layer must be less than the allowable max-antenna-ratio (antenna-area / gate-area < max-antenna-ratio).

The default value of the vector `{v0 v1 v2 v3 [v4]}` is `{0 0 1 0 0}` and can be written as `{v0 v1 v2 v3} = {0 0 1 0}` without the upper limit diode protection `v4`. Both vectors, `{0 0 1 0 0}` and `{0 0 1 0}`, indicate that the upper limit of the diode protection is 0.

The value of diode protection, `dp`, is recorded on each output pin of a cell. If the value of the diode protection of an output pin is `dp`, the antenna ratio is calculated as follows:

- For diode modes 0 to 4, use vector `{v0 v1 v2 v3}` to calculate the allowable max-antenna-ratio.
- For diode modes 5 to 8, use vector `{v0 v1 v2 v3}` to calculate the antenna ratio.
- For diode modes 9 and 10, use vector {v0 v1 v2 v3 v4} to calculate the allowable max-antenna-ratio.
- For diode modes 11 and 12, use both vectors {v0 v1 v2 v3 v4} and {s0 s1 s2 s3 s4 s5} to calculate the allowable max-antenna ratio.

Note that vector {s0 s1 s2 s3 s4 s5} is used for diode modes 11 and 12 only.

Table 4-3 shows how to calculate the antenna ratio for each diode mode.

Table 4-3 Antenna Ratio Calculation Based on Diode Mode

Diode mode	Allowable ratio calculation
Diode modes 2 to 4	allowable max-antenna-ratio • When dp > 0, - if v4 <> 0, the allowable max-antenna-ratio = min (((dp + v1) * v2 + v3), v4) - If v4 == 0, the allowable max-antenna-ratio = (dp + v1) * v2 + v3 • When dp <= v0, the allowable max-antenna-ratio = layerMaxRatio
Diode mode 5	antenna_ratio = metal_area / (gate_area + equi_gate_area) • If max_diode_protection <= v0, equi_gate_area = 0 • Else if (v4 <> 0), equi_gate_area = min (((max_diode_protection + v1) * v2 + v3), v4) • Else equi_gate_area = (max_diode_protection + v1) * v2 + v3
Diode mode 6	antenna_ratio = metal_area / (gate_area + equi_gate_area) • If total_diode_protection <= v0, equi_gate_area = 0 • Else if (v4 <> 0), equi_gate_area = min (((total_diode_protection + v1) * v2 + v3), v4) • Else equi_gate_area = (total_diode_protection + v1) * v2 + v3
Diode mode 7	antenna_ratio = (metal_area - equi_metal_area) / gate_area • if max_diode_protection <= v0, equi_metal_area = 0 • Else if (v4 <> 0), equi_metal_area = min (((max_diode_protection + v1) * v2 + v3), v4) • Else equi_metal_area = (max_diode_protection + v1) * v2 + v3 • if equi_metal_area > metal_area, equi_metal_area = metal_area

Table 4-3 Antenna Ratio Calculation Based on Diode Mode (Continued)

Diode mode	Allowable ratio calculation
Diode mode 8	antenna_ratio = (metal_area - equi_metal_area) / gate_area • If total_diode_protection <= v0, equi_metal_area = 0 • Else if (v4 <> 0), equi_metal_area = min (((total_diode_protection + v1) * v2 + v3), v4) • Else equi_metal_area = (total_diode_protection + v1) * v2 + v3 • If equi_metal_area > metal_area, equi_metal_area = metal_area
Diode mode 9	antenna_ratio = scale * metal_area / gate_area • If max_diode_protection <= v0, scale = 1.0 • Else scale = max (1 / ((max_diode_protection + v1) * v2 + v3), v4)
Diode mode 10	antenna_ratio = scale * metal_area / gate_area • If total_diode_protection <= v0, scale = 1.0 • Else scale = max (1 / ((total_diode_protection + v1) * v2 + v3), v4)
Diode mode 11	antenna_ratio = scale * metal_area / gate_area • If max_diode_protection < v0, scale = s0 • If max_diode_protection < v1, scale = s1 • If max_diode_protection < v2, scale = s2 • If max_diode_protection < v3, scale = s3 • If max_diode_protection < v4, scale = s4 • If max_diode_protection >= v4, scale = s5

Table 4-3 Antenna Ratio Calculation Based on Diode Mode (Continued)

Diode mode	Allowable ratio calculation
Diode mode 12	antenna_ratio = scale * metal_area / gate_area • If total_diode_protection < v0, scale = s0 • If total_diode_protection < v1, scale = s1 • If total_diode_protection < v2, scale = s2 • If total_diode_protection < v3, scale = s3 • If total_diode_protection < v4, scale = s4 • If total_diode_protection >= v4, scale = s5

Example for Diode Modes 2, 3 and 4

The vector {v0 v1 v2 v3} is {0.7 0.0 200 2000}, three diodes are connected to a single net, and the value layerMaxRatio is 400.

```
Diode A: diode protection=0.5, allowable ratio is 400 if protected
Diode B: diode protection=1.0, allowable ratio is 2200 if protected
Diode C: diode protection=1.5, allowable ratio is 2200 if protected

The antenna_ratio calculation of a route net:
Diode mode 2: allowable ratio = 2300 (maximum value)
Diode mode 3: allowable ratio = 400+2200+2300 = 4900
Diode mode 4: allowable ratio = ((0.5+1.0+1.5)*200)+2000 = 2600
```

Example for Diode Modes 5 and 6

The vector {v0 v1 v2 v3} is {0.0 0 1 0}, three diodes are connected to a single net, and the value of layerMaxRatio is 400.

```
Gate area of input pin =0.6
Diode A: diode protection = 0.5
Diode B: diode protection = 1.0
Diode C: diode protection = 1.5
Total diode protection = 0.5+1.0+1.5 = 3.0

The antenna_ratio calculation of a route net:
Diode mode 5: (Maximum diode protection value dp = 1.5)
antenna_ratio = metal_area/(gate_area + equi_gate_area)
                      = metal_area/(0.6+((1.5+0)*1+0))
                      = metal_area/2.1
Diode mode 6: (Total diode protection value dp = 3.0)
antenna_ratio = metal_area/(gate_area + equi_gate_area)
                      = metal_area/(0.6+((3.0+0)*1))
                      = metal_area/3.6
```

Example for Diode Modes 7 and 8

The vector {v0 v1 v2 v3} is {0.7 0.0 150 800}, three diodes are connected to a single net, and the value of layerMaxRatio is 400.

```
Diode A: diode protection = 0.5
Diode B: diode protection = 1.0
Diode C: diode protection = 1.5
Total diode protection = 0.5+1.0+1.5 = 3.0

The antenna_ratio calculation of a route net:
Diode mode 7: (Use maximum dp to compute equi_metal_area)
antenna_ratio = (metal_area - equi_metal_area)/gate_area
                     = (metal_area - ((1.5+0)*150+800))/gate_area
                     = (metal_area - 1025)/gate_area
Diode mode 8: (Use total dp to compute equi_metal_area)
antenna_ratio = (metal_area - equi_metal_area)/gate_area
                     = (metal_area - ((3.0+0)*150+800))/gate_area
                     = (metal_area - 1250)/gate_area
```

The following shows an example of the advanced antenna rules:

```
set lib [current_mw_lib]
remove_antenna_rules $lib
define_antenna_rule $lib -mode 1 -diode_mode 4 -metal_ratio 300 \
   -cut_ratio 20
define_antenna_layer_rule $lib -mode 1 -layer "M1" -ratio 300 \
   -diode_ratio {0.09 0 123 16880}
define_antenna_layer_rule $lib -mode 1 -layer "M2" -ratio 300 \
   -diode_ratio {0.09 0 123 16880}
define_antenna_layer_rule $lib -mode 1 -layer "M3" -ratio 300 \
   -diode_ratio {0.09 0 123 16880}
define_antenna_layer_rule $lib -mode 1 -layer "M4" -ratio 300 \
   -diode_ratio {0.09 0 123 16880}
define_antenna_layer_rule $lib -mode 1 -layer "M5" -ratio 400 \
   -diode_ratio {0.09 0 123 20000}
define_antenna_layer_rule $lib -mode 1 -layer "VIA1" -ratio 20 \
   -diode_ratio {0.09 0 110 500}
define_antenna_layer_rule $lib -mode 1 -layer "VIA2" -ratio 20 \
   -diode_ratio {0.09 0 110 500}
define_antenna_layer_rule $lib -mode 1 -layer "VIA3" -ratio 20 \
   -diode_ratio {0.09 0 110 500}
define_antenna_layer_rule $lib -mode 1 -layer "VIA4" -ratio 20 \
   -diode_ratio {0.09 0 110 500}
```

Reporting Antenna Rules

You can get a report of antenna rules by using the `report_antenna_rules` command. For example,

```
icc_shell> report_antenna_rules -output ./antenna.rule design
```

Removing Antenna Rules

You can remove the antenna rules you set by using the `remove_antenna_rules` command. For example,

```
icc_shell> remove_antenna_rules design
```

Checking Antenna Rules

After the antenna modes and rules are defined, you can proceed with antenna rules checking. The design rule checker of the detail router checks antenna rules automatically. Every routing command checks and reports the number of antenna violations as follows:

```
@@@@ Total nets not meeting constraints = 116
```

where the number represents the number of nets with antenna violations.

To get a detailed report of antenna violations, you can use `verify_route -antenna` or the `report_antenna_ratio` command. Alternatively, you can choose Route > Verify Route or Route > Report Antenna Ratio in the GUI to get an antenna violation report.

Note:

You can set the `topAntennaFixRange` and `maxAntennaPinCount` detail route options to speed up the runtime of antenna checking and fixing.

Specifying Antenna Properties

You can set external antenna properties for all I/O ports or for a specified I/O port by using the `define_io_gate_size`, `define_io_diode_protection`, and `define_io_antenna_area` commands. To remove the definitions set by these commands, use the `remove_io_antenna_properties` command.

You can report the antenna properties set by these commands by using the `report_io_antenna_properties` command.

Fixing Antenna Violations

You can fix antenna violations either before or after running the `route_opt` command. If both antenna fixing and routing optimization are needed, running the `route_opt` command first can shorten the overall turnaround time. However, running the `route_opt` command after antenna fixing can generate a good layout initially. Note that you need to remove as many DRC violations as possible before the antenna fixing step. When DRC violations have been eliminated, you can fix the antenna violations by running search and repair and inserting diodes. Before performing these tasks, ensure that the antenna mode is set and that the antenna rules are defined as described in the previous sections.

Running Search and Repair

After antenna checking and fixing, run search and repair to fix antenna violations. For example,

```
icc_shell> route_search_repair -loop 20 -rerun_drc
```

Search and repair performs two types of metal-jog operations for antenna fixing. Both approaches decrease the antenna ratio by splitting a large metal polygon into several upper-level polygons.

- Break the antenna with a higher-level metal segment.

  Use this technique to fix most antenna violations. For antenna violations happening at metal-N, inserting a small segment of metal-(N+1) close to the gate reduces the ratio between the remaining metal-N, making the ratio much lower. This approach is not suitable for fixing top-metal layer antennas when the output pin can provide only limited protection because there is no way for the router to break antenna violations at the topmost metal layer.

- Move down to a lower-level metal.

  Use this technique to fix only the topmost layer antenna violations when output pins provide only limited protection. For antenna violations happening at metal-N, replace part of the metal-N with metal-(N-1 or lower) to reduce the ratio. However, splitting the metal layer into many pieces might have a negative impact on RC and timing delay.

Fore more information about search and repair, see “Search and Repair” on page 3-26.

Inserting Diodes

One way to protect gates from antenna effects is to provide a discharge path for the accumulated charge to leave the net. However, the discharge path should not allow current to flow during normal chip operation. Discharging can be accomplished by inserting a reverse-biased diode on the net close to the gate that is being protected.

Based on the results of the antenna checking, the tool inserts a diode cell or multiple diodes when the antenna ratio requires the protection of more than one diode for each violation.

The diode port of the diode cell must be defined as such in the library. In the Milkyway environment, you can use the `write_def` command to define diode pins. For details, see the *Library Data Preparation for IC Compiler User Guide*. In addition, the diode protection value must be defined in the physical library. For details, see the description of the `antenna_diffusion_area` attribute in the *Library Compiler Physical Libraries User Guide* and the *Library Compiler Physical Libraries Reference Manual*.

To minimize the impact on the existing placement and routing, you can choose to freeze the existing cell placement and the routing so that the diode cells can be placed in existing empty spaces only. You can also choose to complete the routing of the diode cells during the insertion process when the cell placement is already frozen. Whenever possible, the tool ties the diode cells to the existing wires without ripping up and rerouting the entire net.

The tool considers voltage areas when inserting diode cells and also observes the logic hierarchy assignments for diode cells.

To fix antenna violations, use the `insert_diode` command or choose Finishing > Insert Diode in the GUI. For example,

```
icc_shell> insert_diode -signal_route_options advanced
```

By default, the `insert_diode` command triggers the internal antenna checker to identify the antenna violations and then inserts one or multiple diode cells for each violation. For diode modes 2, 5, and 7, the `insert_diode` command inserts only one diode cell to fix an antenna violation. For diode modes 3, 4, 6, and 8, the `insert_diode` command inserts multiple diodes as needed to fix an antenna violation.

The command syntax is

```
insert_diode
  [-nets collection_of_nets]
  [-antenna_check_engine internal | hercules]
  [-internal_check_option all | top_layer_only]
  [-no_auto_cell_selection]
  [-diode_cells collection_of_diode_cells]
  [-prefix prefix_name]
  [-use_hierarchical_diode_instance_name]
  [-same_row]
  [[-dont_freeze_existing_placement] |
   [-routing skip | when_all_violations_are_gone | always]]
  [-signal_route_options ignore_lower_layers | include_lower_layers|
     include_all_lower_layers | advanced]
  [-max_ratio max_ratio_number]
```

By default, all nets are checked for antenna violations. To restrict checking to a list of nets, use the `-nets` option. By default, the tool uses its own internal antenna checker. To use the external Hercules check instead, specify `-antenna_check_engine hercules`. To report only top-level antenna violations rather than all violations, specify `-internal_check_option top_layer_only`.

By default, any diode cells can be used for fixing. To specify which diode cells to use, specify the `-no_auto_cell_selection` option together with the `-diode_cells` option. To attempt diode insertion in the same row as the violation, use the `-same_row` option; by default, there is no preferred row for insertion.

By default, diode insertion does not disturb existing standard cells. To allow existing cells to be moved to make room for inserted diodes, use the `-dont_freeze_existing_placement` option. The `-routing` option lets you specify whether to perform routing: `always`, `when_all_violations_are_gone`, or `skip`.

Use the `-signal_route_options` to specify the rules used for antenna checking: `ignore_lower_layers`, `include_lower_layers`, `include_all_lower_layers`, or `advanced`. The `-max_ratio` option specifies the maximum allowable ratio of wiring area to gate area to be used for antenna checking, overriding the default antenna checking rules.

For more information about the command options, see the man page for the `insert_diode` command.

To remove diodes, use the `remove_diode` command or choose Finishing > Remove Diodes in the GUI. The `remove_diode` command first disconnects all the diode ports on all the nets specified; then it removes the diode cells that are fully disconnected. For example,

```
icc_shell> remove_diode -nets net_name
icc_shell> remove_diode -nets [get_nets -hierarchical net_name]
```

Connecting Spare Diodes

Instead of inserting new diodes to fix antenna violations, you can insert spare diodes at the same time as standard cell placement and then connect the spare diodes as needed when antenna violations are found. The unused spare diodes are left unconnected. This technique lets you fix antenna violations without disturbing the placement of existing standard cells. However, you must allocate some placement area to accommodate the spare diodes. Adding more spare diodes uses more area resources but makes antenna fixing easier because a nearby spare diode is more likely to be available where needed.

The diode port of the diode cell must be defined as such in the library. In the Milkyway environment, you can use the `write_def` command to define diode pins. For details, see the *Library Data Preparation for IC Compiler User Guide*. In addition, the diode protection value must be defined in the physical library. For details, see the description of the `antenna_diffusion_area` attribute in the *Library Compiler Physical Libraries User Guide* and the *Library Compiler Physical Libraries Reference Manual*.

To take advantage of spare diodes for antenna violation fixing, you need to add the spare diodes either before or after standard cell placement and before routing the areas where antenna violations might occur. For details, see the “ECO Flow” chapter in the *IC Compiler Implementation User Guide*.

To fix antenna violations by connecting spare diodes, use the `connect_spare_diode` command or choose Finishing > Connect Spare Diode in the GUI. For example,

```
icc_shell> connect_spare_diode
```

The `connect_spare_diode` command checks for antenna violations and fixes those violations by connecting the problem nets to nearby spare diodes.

The command syntax is

```
connect_spare_diode
  [-exclude_nets collection_of_nets]
  [-antenna_check_engine internal | hercules]
  [-internal_check_option all | top_layer_only]
  [-routing skip | route]
  [-distance distance_number]
  [-signal_route_options ignore_lower_layers |
     include_lower_layers| include_all_lower_layers | advanced]
  [-max_ratio max_ratio_number]
```

The `-antenna_check_engine` option specifies which antenna checking engine to use, either the IC Compiler internal antenna checker (`internal`, the default) or the external Hercules antenna checker (`hercules`). If the internal checker is being used, the `-internal_check_option` option specifies whether to report all antenna violations (`all`) or only the top-layer violations (`top_layer_only`).

The `-distance` option specifies the maximum distance from the problem net that is searched for a spare diode. If no spare diode can be found within this distance, the

command reports an error. You specify the distance as an integer multiple of the global routing cell size. The default is 10, which means that spare diode cells should be made available within 10 global routing cell lengths of anywhere an antenna violation might occur. For example, an array of spare diode cells with a spacing of 14 global routing cell lengths might be sufficient.

The `-exclude_nets` option specifies a collection of nets that should not be connected to diodes. If you do not set this option, all nets are connected to diodes.

The `connect_spare_diode` command finds each antenna violation, identifies one or more nearby spare diodes required for fixing, and sets the net ID of the diode pins to the name of the problem net that must be connected. The number of diodes required to fix a particular antenna violation depends on the antenna checking method. By default, the command also invokes the detail router in ECO mode to perform the required routing between the problem net and the diode. However, if you use the `-routing skip` option, the routing step is skipped. In that case, you can perform the necessary routing separately at a later time. Detail routing should be complete before you attempt to connect the spare diodes.

By default, the `connect_spare_diode` command uses the antenna-checking mode specified by the `doAntennaConx` detail route option, as explained in "Defining Antenna Rules" on page 4-7. You can override the `set_droute_options` setting by using the `-signal_route_options` option with the `connect_spare_diode` command. The four possible override settings are `ignore_lower_layers`, `include_lower_layers`, `include_all_lower_layers`, and `advanced`. These settings correspond to the `doAntennaConx` antenna mode settings of 1, 2, 3, and 4, respectively. The `doAntennaConx` antenna mode must be set to 1, 2, 3, or 4 (not 0) or the `-signal_route_options` option must be used with the `connect_spare_diode` command, even if the external Hercules antenna checker is being used.

By default, for antenna mode settings of 1, 2, or 3, the `connect_spare_diode` command checks for an antenna-to-gate area ratio that exceeds the `maxAntennaRatio` option of the `set_droute_options` command, as explained in "Defining Antenna Rules" on page 4-7. You can override the `set_droute_options` setting by using the `-max_ratio` option with the `connect_spare_diode` command.

You can also perform the functions of the `connect_spare_diode` command in the GUI by choosing Finishing > Connect Spare Diode.

Setting Detail Route Options to Control Antenna Fixing

You can set the detail route options listed in Table 4-4 to fix antenna violations by using the `set_droute_options` command. The option settings are stored with the cell.

Table 4-4 Detail Route Options

Option	Range	Default	Description
`useSideWallForAntenna`	[0, 1]	0	0: Uses the wire area to compute antenna ratio. 1: Uses the wire sidewall area to compute antenna ratio. You need to specify the thickness in the technology file. This option is ignored when `doAntennaConx` is set to 4.
`maxCutAntennaRatio`	[0, 1000000]	0	0: Does not compute cut area. N: The ratio of the total area of the charge-collecting antenna cuts to the total gate size should not exceed the value N. This option is ignored when `doAntennaConx` is set to 4.
`maxAntennaRatio`	[0, 1000000]	1000000	N: The ratio of the total area of the charge-collecting antenna to the total gate size should not exceed the value N. This option is ignored when `doAntennaConx` is set to 4.
`maxAntennaPinCount`	[-1, 1000000]	-1	-1: Checks antenna on all nets. N: Skips checking antenna on nets with more than N pins.
`minAntennaRatioScale`	[0.000, 1.000]	0.000	N: Shows the minimal antenna ratio scale for diode modes 11 and 12.
`maxAntennaRatioScale`	[0.000, 1.000]	1.000	N: Shows the maximal antenna ratio scale for diode modes 11 and 12.

Table 4-4 Detail Route Options (Continued)

Option	Range	Default	Description
`topAntennaFixRange`	[-1, 10]	-1	-1: Fixes all top-layer antenna violations by pushing wires down. N: Fixes only top-layer antenna violations if the ratio is less than [safeRatio x (1 + 0.1 * N)] to avoid pushing too many wires to lower layers. This option is valid only when `doAntennaConx` is set to 4. (Note: When `topAntennaFixRange` is set, the log file also shows the number of nets that are not fixed by the router.)
`dontMergeGateForAntenna`	[0, 1]	0	When 2 or more input gates are connected, 0: The router adds their areas together to compute antenna ratio. 1: The router does not add the area of connected input gates.
`outputPinDischarge`	[0, 1]	1	0: Assumes that no output pin can discharge static electricity. 1: Assumes that output pins discharge static electricity. This option is ignored when `doAntennaConx` is set to 4.
`defaultTopPinExtAntennaArea`	[0.000, 1000000.000]	0.000	N: Sets the default `extAntennaArea` value at all metal layers for top-cell pins. This option replaces `breakAntennaToTopPin`.
`defaultTopPinExtGateSize`	[0.000, 1000000.000]	0.000	N: Sets the default `extGateSize` value for top-cell pins.

Table 4-4 Detail Route Options (Continued)

Option	Range	Default	Description
`breakAntennaToTopPin`	[0, 2]	0	0: Treats top-level pins as floating. 1: Treats top-level pins as they are connected to a huge antenna and breaks the antenna (Requires a jumper at top metal layer). 2: Connects all input pins to top-level pins only through the highest routing layer of the net.
`checkAntennaOnPG`	[0, 1]	0	0: Does not check antenna on power and ground nets. 1: Checks antenna on power and ground nets.

Reducing Critical Areas

A critical area is a region of the design where, if the center of a random particle defect falls there, the defect causes circuit failure, thereby reducing yield. A conductive defect causes a short fault, and a nonconductive defect causes an open fault.

The following sections describe how to

- Report critical areas
- Display critical area maps
- Reduce critical area short faults by performing wire spreading
- Reduce critical area open faults by performing wire widening

Reporting Critical Areas

After routing is complete, you can report layout-critical areas that are susceptible to random particle defects, which cause shorts and opens during the fabrication process.

The results from the critical area analysis report are output to an output_heatmap text file. You can see the critical area results graphically by displaying critical area heat maps.

To report critical areas, use the `report_critical_area` command (or choose Finishing > Report Critical Area Map in the GUI).

Table 4-5 describes the `report_critical_area` options. For more information, see the man page.

Table 4-5 report_critical_area Command Options

Command option	Description
`-fault_type short | open` (“Fault type” radio buttons in the GUI)	Specifies the type of fault to check for. The default is `short`.
`-particle_distr_func_file` *file_name* (“Particle distribution function file” box in the GUI)	The name of the input file that contains the particle distribution function to be used during the calculation. By default, the tool uses an internal particle probability function.
`-suppress_zeros_in_report` (“Suppress zeros in report” check box in the GUI)	Prevents reporting of zero results. By default, all results are reported.
`-multiparticle_report_format true | false` (“Multi particle report format” check box in the GUI)	Outputs a report containing the critical area analysis result of each individual particle size. By default, the tool outputs normalized results.
`-tsmc_encr_particle_distr_file` (“TSMC encrypted format” check box in the GUI)	Specifies that the particle distribution function file is in TSMC encrypted format.
`-layer_alias_DSD_format {`*X Y Z T R*`}` (“Metal scheme directives” boxes in the GUI)	Specifies the layer alias used for defect size distribution (DSD) of each layer.
`-input_layers` *layers* (“Input layer” list in the GUI)	Specifies the metal layers for which the critical area is calculated. By default, the critical area is calculated for all metal layers.

Often the particle probability function is considered sensitive data. When security for a sensitive particle distribution file from a foundry is of concern, use the `process_particle_probability_file` command, which provides a way to encrypt and decrypt the particle probability function. Critical area analysis can work with such an encrypted particle probability function.

A secret key is used to encrypt the particle probability function. The encrypted file cannot be decrypted without the key. To encrypt and decrypt a particle probability function file, use the `process_particle_probability_file` command.

The command syntax is

```
process_particle_probability_file
   -key string
   -input_file file_name
   -output_file file_name
```

Critical area analysis takes the encrypted file as its input and processes it without the key. The output is the heat map based on the encrypted particle probability function; it is in text format. The text format of the particle probability function is still accepted as input to critical area analysis. If an encrypted file is given as input, the file is internally decrypted and used.

Displaying Critical Area Maps

Critical area maps provide an indication of places where a chip might fail due to particle defects, which can cause shorts and opens.

To display a critical area for the current design,

1. Specify the type of defect to display by choosing one of the following:
   - Finishing > Open Critical Area Map
   - Finishing > Short Critical Area Map

   The Map Mode panel appears.
2. Select a metal layer for which critical area shorts or opens are to be displayed.
3. Modify the critical heat ranges by changing or removing the maximum or minimum threshold values.
4. Adjust other display parameters. You can make text visible in the map.
5. Click Apply.

   To update the critical area map data after you perform wire spreading, click Reload. This action opens the (Re)Calculate Open Critical Area Map Data dialog box, from which you can run the `report_critical_area -fault_type short` command.

   To update the critical area map data after you perform wire widening, click Reload. This action opens the (Re)Calculate Open Critical Area Map Data dialog box, from which you can run the `report_critical_area -fault_type open` command.

For more information about using critical area maps, see the “Examining Critical Area Maps” topic in IC Compiler Help.

Performing Wire Spreading

After you have performed detail routing, you can perform wire spreading to increase the average spacing between wires, which reduces the critical area short faults and therefore improves yield.

To perform wire spreading, use the `route_spreadwires` command (or choose Finishing > Route Spread Wires in the GUI).

The `route_spreadwires` command can make the following changes to the layout:

- Move wires on the same layer to widen the spaces between wires
- Move wires to the upper or lower metal layers as needed to resolve DRC violations caused by widening spaces

When spreading wires, a piece of the original wire is broken and pushed away, and jog wires are created at both ends to maintain a connection. By default, the minimum ratio of the jog length to the layer pitch is two. You can modify this minimum ratio by using the `-min_jog_length` option. After spreading, the `route_spreadwires` command performs search and repair to fix any DRC violations caused as a result of spreading. By default, the classic router performs 10 search-and-repair loops. You can control the number of loops by using the `-search_repair_loop` option. You can reduce the turnaround time for running the search-and-repair loops by using distributed routing. To use distributed routing, specify the number of CPUs by using the `-num_cpus` option. For information about setting up for distributed routing, see “Enabling Distributed Routing” on page 2-18.

Because any change to the routing pattern can affect the timing, the tool provides timing-driven wire spreading to minimize the timing impact. You enable timing-driven wire spreading by using the `-timing_driven` and `-setup_slack_threshold` options. When you specify these options, the command finds the timing-critical nets and prevents spreading from being performed on them. Timing-critical nets are those nets that have a slack less than the specified threshold value.

In most cases, timing is improved, due to the wider net spacing produced by wire spreading. However, having too many timing-violated nets in the design can have a negative impact on the final critical area improvement. Often, when too many nets have violations, the cause is that only a limited number of segments can be moved during spreading. To allow more nets to be spread, set the setup slack threshold option to a negative value. For example, if this option is set to –0.5, the router spreads all nets that have slack greater than –0.5. The default is 0.0.

For example, to perform timing-driven wire spreading followed by six search-and-repair loops, enter the following command:

```
icc_shell> route_spreadwires -search_repair_loop 6 \
   -timing_driven -setup_slack_threshold 0.15
```

Performing Wire Widening

After you have performed detail routing and wire spreading, you can perform wire widening to increase the average width of the wires, which reduces the critical area open faults and therefore improves yield.

To perform wire widening, use the `route_widen_wire` command (or choose Finishing > Route Widen Wires in the GUI).

The `route_widen_wire` command widens all wires in the design to 1.5 times their original width and then performs search and repair to fix any DRC violations caused as a result of widening. By default, the classic router performs 10 search-and-repair loops. You can control the number of loops by using the `-search_repair_loop` option.

Note:

The widened wires do not trigger fat wire spacing rules.

When you widen the wires, it changes the timing of your design. Wire widening has a timing-driven mode that allows you to perform wire widening with minimal impact on the design timing.

Use the `-timing_driven`, `-setup_slack_threshold`, and `-hold_slack_threshold` options to consider timing during wire widening. In that case, the command finds the timing-critical nets and prevents widening from being performed on them. Timing-critical nets are those nets that have a slack less than the specified threshold value.

For example, to perform wire widening followed by 20 search-and-repair loops, enter the following command:

```
icc_shell> route_widen_wire -search_repair_loop 20
```

Inserting Redundant Vias

After routing and postrouting optimization, you can replace single-cut vias with multiple-cut via arrays (a process sometimes called via doubling) or with a different via with the same layers. The via is replaced only if the via array or the new via does not introduce DRC violations.

To insert redundant vias, use the `insert_redundant_vias` command (or choose Finishing > Insert Redundant Vias).

The command syntax is

```
insert_redundant_vias
   -from_via list_of_from_vias
   -to_via list_of_to_vias
   [-to_via_x_size list_of_x-sizes]
   [-to_via_y_size list_of_y-sizes]
   [-via_array_no_swap]
   [-optimize_level level_of_optimization]
   [-num_cpus number_of_cpus]
   [-auto_mode preview | insert]
```

There are two redundant via insertion methods: manual and automatic.

The manual method uses the `-from_via`, `-to_via`, `-to_via_x_size`, and `-to_via_y_size` options to customize the insertion process by specifying which kinds of redundant vias are inserted and the allowed array dimensions. For example, the following command replaces all VIA23 single-cut vias with 1x3 VIA23 arrays and all VIA34 single-cut vias with 5x1 VIA34 arrays:

```
icc_shell> insert_redundant_vias \
   -from_via {VIA23 VIA34} -to_via {VIA23 VIA34} \
   -to_via_x_size {1 5} -to_via_y_size {3 1}
```

The automatic method uses the `-auto_mode` option. If you use the `-auto_mode insert` option, the command inserts the default vias defined in the technology file as redundant vias for all layers. For example,

```
icc_shell> insert_redundant_vias -auto_mode insert
```

If you use the `-auto_mode preview` option, the command lists all of the vias on all the layers without referencing the technology file. You can then cut and paste part or all of the listed vias into a new manual-method command that specifies the `-to_via` and `-from_via` options.

The Insert Redundant Vias dialog box in the GUI (Finishing > Insert Redundant Vias) is very helpful in setting the options for the `insert_redundant_vias` command.

Inserting Filler Cells

Filler cells fill gaps in the design to ensure that all power nets are connected and the spacing requirements are met.

- Before routing, you can
  - Insert standard cell fillers
  - Insert end cap cells
- After routing, you can
  - Insert well fillers
  - Insert pad fillers

The following sections describe how to insert these cells.

Inserting Standard Cell Fillers

You can fill empty spaces in the standard cell rows with instances of reference filler cells to make sure all power nets are connected. One method of improving the stability of the power supply is to add decoupling capacitors as filler cells.

One method of improving the stability of the power supply is to add decoupling capacitors as filler cells. However, these filler cells often contain metal internally, and they can cause a short or a spacing rule violation with existing metal routes in the design. During insertion, filler cells with metal are retained only when they do not cause DRC violations.

To insert standard cell fillers,

1. (Optional) Define the standard cell filler rules by using the `set_left_right_filler_rule` command to define the left and right filler rules. These rules specify the filler cell to insert immediately to the left and right, respectively, of specific standard cells.
2. Use the `insert_stdcell_filler` command (or choose Finishing > Insert Standard Cell Filler in the GUI) to insert the standard cell fillers.

The following sections provide more information about these steps.

Defining the Filler Rules

The left and right filler rules specify the filler cell to insert to the immediate left and immediate right, respectively, of specific standard cells. You use the `set_left_right_filler_rule` command to define these rules.

If there is only a single site between two standard cells, the rule used to fill that site depends on whether the references for the standard cells are the same or different. If the references for the two standard cells are the same, the tool uses the rules for the cell on the left and the rules for the cell on the right are ignored. If the references are different, the tool uses the rule that was defined first.

To report the right and left filler rules defined for your design, run the `report_left_right_filler_rule` command. The report shows the rules and the order in which they were defined.

By default, the left and right filler cells have a north (N) orientation or flipped-south (FS) orientation when the rows are flipped. You can choose to have the left and right filler cells follow the orientation of the standard cell by using the `set_left_right_filler_rule -follow_stdcell_orientation` option.

Inserting Filler Cells

To insert standard cell fillers, use the `insert_stdcell_filler` command or choose Finishing > Insert Standard Cell Filler in the GUI. Table 4-6 shows the available options.

Table 4-6 Options for Inserting the Filler Cells

Option	Description
`-cell_without_metal`	Specifies the list of filler cells to use.
`-cell_with_metal`	Specifies the master filler cells to use.
`-bounding_box`	Specifies the rectangular region to insert filler cells.
`-dont_respect_hard_placement_blockage`	Specifies whether to respect hard placement blockage.
`-dont_respect_soft_placement_blockage`	Specifies whether to respect soft placement blockage.
`-between_std_cells_only`	Inserts filler cells between two standard cells only.
`-randomize`	Specifies whether to randomize the selection of filler cells.

Table 4-6 Options for Inserting the Filler Cells (Continued)

Option	Description
`-respect_overlap`	Specifies whether to respect standard-cell overlap check points.
`-cell_without_metal_prefix`	Inserts a prefix to the instance names of the filler cells without metal.
`-cell_with_metal_prefix`	Inserts a prefix to the instance names of the filler cells with metal.
`-avoid_layers`	Prevents insertion of filler cells under the specified metal layers.
`-connect_to_power`	Specifies the power connection.
`-connect_to_ground`	Specifies the ground connection.
`-voltage_area`	Specifies the voltage areas to insert filler cells.
`-vt_filler`	Specifies the threshold voltage cells to be used as voltage threshold fillers.
`-check_only`	Runs the command in checking mode, rather than insertion mode.
`-restore_filler_snapshot`	Restores the filler cell placement of the snapshot previously created.
`-vt_filler_prefix`	Specifies the prefix for the collection of threshold voltage fillers to be inserted.
`-respect_keepout`	Specifies whether to respect keepout margins.

The following example fills empty spaces between standard cells with filler cells FILL_4X, FILL_2X, and FILL_1X, in that order of preference, in the bounding box with corners at (10,10) and (5000,5000). By specifying the filler cells from largest to smallest, the total number of filler cells is minimized by adding the larger filler cells first. Cells with metal are used where they do not cause DRC violations and cells without metal are used otherwise.

```
icc_shell> insert_stdcell_filler \
   -cell_without_metal {FILL_4X FILL_2X FILL_1X} \
   -cell_with_metal {FILL_4XM FILL_2XM FILL_1XM} \
   -bounding_box {{10.0 10.0} {5000.0 5000.0}}  \
   -between_std_cells_only
```

For more information about the `insert_stdcell_filler` command, see the man page.

Removing Filler Cells

To remove standard cell fillers, use the `remove_stdcell_filler -stdcell` command (or choose Finishing > Remove Fillers and select “Standard Cell” in the “Filler type” area in the GUI). By default, the filler cells are removed for the whole chip. You can optionally specify a bounding box from which to remove filler cells.

When filler cells are added after signal routing, you can remove all the filler cells that have routing design rule violations. To remove standard cell fillers with violations, use the `remove_filler_with_violation` command (or choose Finishing > Remove Fillers With Violation in the GUI). You can restrict the removal to specific instances by using the `-name` option.

Reporting Filler Cells

To report the type of filler cells and their locations, use the `report_filler_placement` command.

The command syntax is

```
report_filler_placement -lib_cell lib_cell_list [-abut]
```

Use the `-lib_cells` option to specify the type of filler cells that you want to report in your design. To report only the adjacent filler cells that form a consecutive pair in a cell row, use the `-abut` option. For example,

```
icc_shell> report_filler_placement -lib_cell FILL1BWP -abut
```

The example reports only the consecutive filler cells named FILL1BWP and their locations in the design.

Inserting End Caps

After placing standard cells and before routing, you can add end cap cells at both ends of a cell row. Typically, an end cap cell is a nonlogic cell that serves a certain purpose for the row such as providing a decoupling capacitor for the power rail. Because the IC Compiler tool accepts any standard cell as an end cap, you should specify a suitable end cap cell.

To insert end caps, use the `add_end_cap` command or choose Finishing > Insert End Cap in the GUI. You must specify the cell to use for the end caps by using the `-lib_cell` option. For example,

```
icc_shell> add_end_cap -lib_cell MY_END_CAP
```

By default, the command places the specified library cells in their default orientation at both ends of the horizontal cell rows without considering padding, blockages, or keepouts. To add end caps to only one end, specify which end by using the `-mode` option. To add end caps only at the left end, specify the `-mode bottom_left` option. To add end caps only at the right end, specify the `-mode upper_right` option.

To specify the cells to add as vertical end caps, use the `-vertical_cells` option, which inserts cells in the specified order and avoids unfilled space at the end of a cell row. When you add both horizontal and vertical end cap cells, you can fill the corners where the horizontal and vertical end caps meet by specifying the `-fill_corner` option, changing it from its default of off. The `-fill_corner` option takes effect only when you use it with the -`vertical_cells` option.

By default, if a voltage area has no guard bands, the `add_end_cap` command ignores the voltage area boundaries during end cap insertion. To insert horizontal end caps at both sides of a vertical voltage area boundary, use the `-at_va_boundary` option.

By default, the `add_end_cap` command ignores the plan group boundaries during end cap insertion. To insert horizontal end caps at both sides of a vertical plan group boundary, use the `-at_plan_group_boundary` option.

To flip the orientation of the end cap cells, use the `-mirror` option. The `-mirror` option applies to horizontal end caps only. To prevent the command from placing end caps inside padding areas, blockages, or keepouts, use the `-respect_padding`, `-respect_blockage`, and `-respect_keepout` options respectively.

When you specify the `-next_to_fixed` option, the command treats a fixed cell abutting the boundary as a macro and creates a horizontal end cap next to the fixed cell. Other fixed cells are ignored by this option. This option cannot be used with the `-skip_fixed` option.

During end cap insertion, the `add_end_cap` command ignores both hard and soft blockages by default. If you specify the `-respect_blockage` option, the command respects both hard and soft blockages. To ignore only soft blockages, use the -ignore_soft_blockage option. The `-ignore_soft_blockage` option must be used with the `-respect_blockage` option. When you specify both options, the command respects hard blockages but ignores soft

blockages. If you specify only the `-ignore_soft_blockage` option, the command issues an error.

You can remove end cap cells from your design by using the `remove_stdcell_filler -end_cap` command.

Inserting Well Fillers

After routing is complete, you can fill small gaps that violate the spacing rule for the well layer with well filler cells. You can fill gaps between cells in the same row or between rows.

To insert well fillers, use the `insert_well_filler` command (or choose Finishing > Insert Well Filler in the GUI).

The command syntax is

```
insert_well_filler
   -layer layer_name_or_number
   [-ignore_PRboundary]
   [-fill_gaps_smaller_than gap_size]
   [-higher_edge min | max]
   [-lower_edge min | max]
   [-gap_type {tt | bb | tb | bt}]
   [-respect_blockages]
   [-row_overlap {row_overlap_value}]
   [-min_gap  {min_gap_distance}]
   [-max_gap  {max_gap_distance}]
   [-enclosure_only  width]
```

For example,

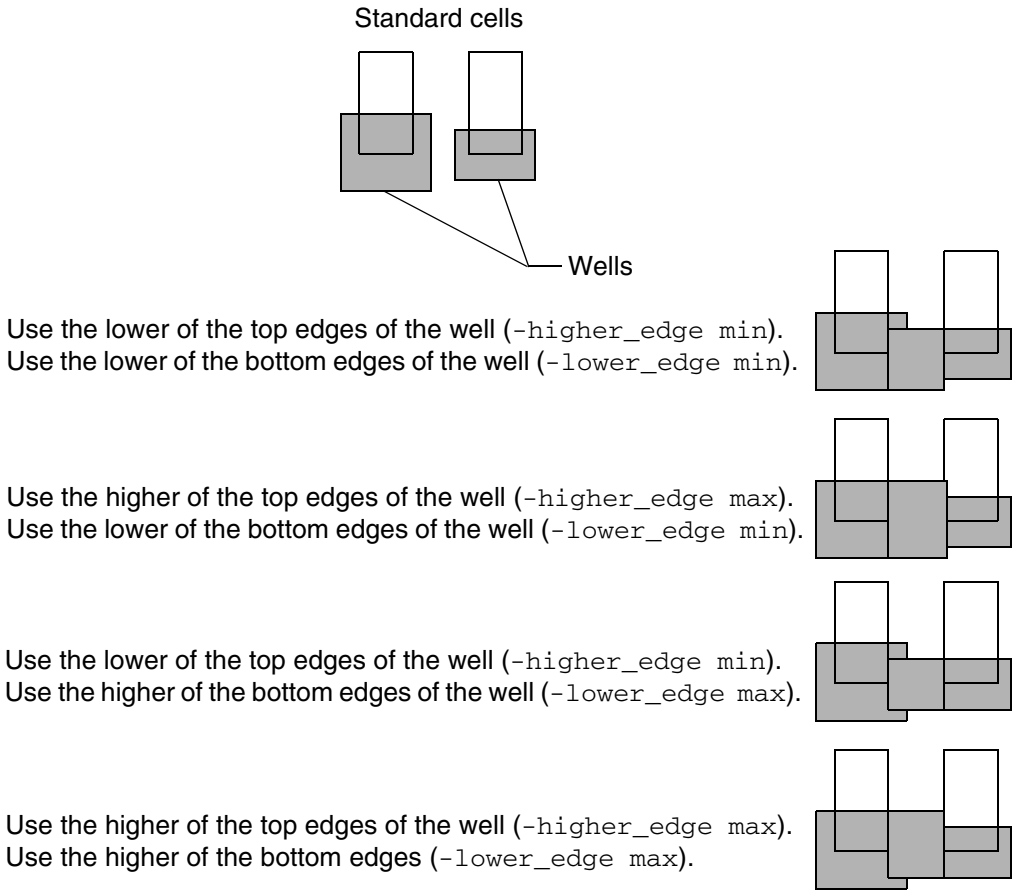
```
icc_shell> insert_well_filler -layer 10 \
   -fill_gaps_smaller_than 3.0 \
   -higher_edge "max" -lower_edge "min"
```

This example adds well filler on layer 10 for gaps smaller than 3.0 microns. If the wells from the two standard cells do not line up, the command creates the largest fill box by using the larger top edge and the smaller bottom edge.

By default, only the gaps between standard cells within a row are filled. To fill gaps between rows, use the `-gap_type` option and set it to `tt`, `bb`, `tb`, or `bt`.

The `-higher_edge` and `-lower_edge` options specify how to align the well filler with the wells in the two standard cells when the wells in the cells do not line up. Figure 4-3 shows the alignments and fill boxes that occur as a result of the various settings.

Figure 4-3 Filler Alignment



The `insert_well_filler` command also has options to specify the following:

- Whether to ignore the PR boundary layer (layer 207) that extends outside a cell
- Whether to respect placement blockages when the `-gap_type` option is used
- The amount of row overlap between the row and the gap when the `-gap_type` option is used
- The minimum and maximum gap sizes that are filled when the `-gap_type` option is used
- Whether to insert a well enclosure instead of well fill and, if so, the enclosure width

To remove well fillers, use the `remove_well_filler` command (or choose Finishing > Remove Well Fillers in the GUI).

Inserting Pad Fillers

After routing is complete, you can fill gaps in the pad ring with instances of pad filler cells. These are dummy pad cells that you can use to control the pad spacing and complement the n-well taps in pads. You should complete the routing process before you add pad filler cells.

To insert pad fillers, use the `insert_pad_filler` command (or choose Finishing > Insert Pad Filler in the GUI). This is the command syntax:

```
insert_pad_filler
   -cell lib_cells
   [-overlap_cell overlap_lib_cells]
   [-voltage_area voltage_area_list]
   [-bounding_box rectangle]
   [-prefix prefix]
   [-no_left]
   [-no_right]
   [-no_bottom]
   [-no_top]
```

For example,

```
icc_shell> insert_pad_filler -cell "PFILL_2X PFILL_1X" \
    -overlap_cell "PFILL_1X" -voltage_area {V1}
```

This example inserts pad filler cells PFILL_2X and PFILL_1X on the pad ring, with preference for PFILL_2X because it is listed first, inside voltage area V1 only. The PFILL_1X cell is allowed to overlap other pad filler cells to fill gaps that are too small for any pad filler cell.

The command also has options to restrict pad filler insertion to a specified rectangular area, to specify pad filler instance naming conventions, and to exclude pad filler insertion from left, right, top, or bottom boundaries.

To remove pad fillers, use the `remove_stdcell_filler -pad` command (or choose Finishing > Remove Fillers in the GUI and select “Pad” in the “Filler type” area). By default, pad fillers are removed for the whole chip. You can optionally specify a bounding box from which to remove pad fillers.

Inserting Metal Fill

After routing, you can fill the empty space in the design with metal wires to meet the metal density rules required by most fabrication processes. Before inserting metal fill, the design should be close to meeting timing and have only a very few or no DRC violations.

When you define minimum and maximum metal density rules in the technology file, the tool tries to create fill within the specified ranges.

The IC Compiler tool supports both real and emulation metal fill extraction.

- To perform emulation metal fill extraction, you must specify the emulation TLUPlus files by using the `-max_emulation_tluplus` and `-min_emulation_tluplus` options of the `set_tlu_plus_files` command.
- To perform real metal fill extraction, you must specify the `-real_metalfill_extraction` option of the `set_extraction_options` command. The tool can treat the metal fill polygons as either floating or grounded. You can either specify how to treat the metal fill polygons by using the `floating` or `grounded` keyword or let the tool determine the treatment, based on the fill wire track property, by using the `auto` keyword. When you use the `none` keyword, which is the default, fill is not considered during extraction.

  In addition to specifying the `-real_metalfill_extraction` option, you must specify TLUPlus files without any emulation information, using the `set_tlu_plus_files` command.

  Note:

  The tool uses nonemulation TLUPlus files when `set_extraction_options -real_metalfill_extraction` is set to `auto`, `floating`, or `grounded`. Therefore, you should select real metal fill for extraction only after the metal fill objects have been added.

You can insert metal fill by using the internal IC Compiler metal fill capability or by invoking the external IC Validator or Hercules metal fill capability. To use the internal IC Compiler metal fill capability, run the `insert_metal_filler` command or choose Finishing > Insert Metal Filler in the GUI. To invoke the external IC Validator or Hercules metal fill capability, use the `signoff_metal_fill` command or choose Finishing > Signoff Metal Fill in the GUI. Using the IC Validator or Hercules metal fill capability ensures DRC conformance and produces better quality of results but requires an IC Validator or Hercules license.

IC Compiler Metal Fill

To perform metal density filling in the IC Compiler tool, use the `insert_metal_filler` command (or choose Finishing > Insert Metal Filler in the GUI). This is the command syntax:

```
insert_metal_filler
   [-out self | cellview_name]
   [-purge]
   [-bounding_box {{llx lly} {urx ury}}]
   [-dont_overwrite]
   [-timing_driven]
   [-insert_as_instance instance_name]
   [-tie_to_net none | ground | net_name]
   [-create_floating_vias]
   [-floating_via_ftr_spacing]
   [-routing_space route_space]
   [-from_metal from_metal_number]
   [-to_metal to_metal_number]
   [-width {layer width}]
   [-space {layer space_between_fills}]
   [-min_length {layer min_length}]
   [-max_length {layer max_length}]
   [-space_to_route {layer keep_from_metal_space}]
   [-space_to_pg {layer keep_from_pg_nets_space}]
   [-stagger {layer}]
   [-x_offset {layer x_distance}]
   [-y_offset {layer y_distance}]
   [-dont_snap_fill_to_track]
   [-fill_poly]
   [-distance_to_boundary poly_to_cell_distance]
```

You can use the `-width`, `-space`, `-min_length`, `-max_length`, `-space_to_route`, `-space_to_pg`, `-x_offset`, and `-y_offset` options to specify the fill characteristics of one or more layers. You specify each characteristic using one or more pairs of values. Each pair consists of a layer name such as poly, M1, or M2, followed by the value. For example,

```
icc_shell> insert_metal_filler \
   -width {M1 0.1} \
   -space {poly 0.1} \
   -min_length {M1 2 M2 2 M3 2} \
   -max_length {M1 10 M2 10 M3 10}
```

This example fills tracks using width 0.1 for metal 1 and spacing 0.1 for poly, with lengths ranging from 2 to 10 for metal 1, metal 2, and metal 3. The default values are used for the remaining metal fill characteristics not specified explicitly in the command.

These are the fill characteristics you can set:

- `-width`: the width of fill tracks
- `-space`: the spacing between fill tracks on the same layer

- `-min_length`: the minimum length of fill tracks
- `-max_length`: the maximum length of fill tracks
- `-space_to_route`: the space to keep away from existing metal
- `-space_to_pg`: the space to keep away from power and ground nets
- -`x_offset`: the x-offset for staggered fill tracks
- `-y_offset`: the y-offset for staggered fill tracks

Fill tracks are staggered if you use the `-x_offset`, `-y_offset`, or `-stagger` option. In that case, if you do not want the fill snapped to wire tracks, use the `-dont_snap_fill_to_track` option.

Use the `-purge` option to remove, rather than insert, metal fill. You can remove fill-track objects from a specified range of layers using `-from_metal` and `-to_metal`. Otherwise, the `-purge` option removes fill-track objects from all layers.

The `insert_metal_filler` command also has options to specify the following:

- Whether to modify the current cell (the default) or to write the output to a new cell
- A bounding box in which to insert metal fill
- The net to which the metal fill must be connected
- The space between normal routing wires and the fill metal, expressed as a multiplier of the minimum spacing for the layer (default 1.0)
- The range of layer numbers to fill
- Whether to fill poly layers as well as metal layers (by default, only metal layers are filled)
- Whether to create floating vias, and if so, the required spacing
- Whether to check for timing effects during metal fill
- The spacing of poly to the cell boundary

Signoff Metal Fill

Without leaving the IC Compiler tool, you can invoke the IC Validator or Hercules tool to create, view, and change metal fill by using the `signoff_metal_fill` command or by choosing Finishing > Signoff Metal Fill in the GUI. Using the external IC Validator or Hercules metal fill capability ensures DRC conformance and produces the best quality of results. However, it requires a separate IC Validator or Hercules license.

To use the `signoff_metal_fill` command to insert metal fill,

- Set up the validation tool environment.
- Set up the physical signoff options.
- Set up distributed processing.
- Run the `signoff_metal_fill` command.

The following sections describe these tasks.

Setting Up the Validation Tool Environment

You can use either the IC Validator or Hercules tool to insert metal fill with the `signoff_metal_fill` command. For designs that are at the 32-nm process node or below, you should use the IC Validator tool for metal fill.

Setting Up the IC Validator Environment

To use the IC Validator tool when running the `signoff_metal_fill` command, you must have an IC Validator license, and you must specify the location of the IC Validator executable by setting the `ICV_HOME_DIR` environment variable. You can set this variable in your .cshrc file. To specify the location of the IC Validator executable, use commands similar to those shown in the following example:

```
setenv ICV_HOME_DIR /root_dir/icv
set path = ($ICV_HOME_DIR/bin/AMD.64 $path)
```

You must ensure that the version of the IC Validator executable that you specify is compatible with the IC Compiler version that you are using. For more information about the IC Validator tool, see the IC Validator documentation, which is available on SolvNet.

The IC Validator tool can generate a compressed hierarchical FILL view file, instead of the default hierarchical FILL view file. The compressed hierarchical FILL view reduces the size of the FILL view file without increasing runtime or memory requirements. To use the compressed hierarchical FILL view file instead of the default hierarchical FILL view file, set

the ICV_ENABLE_GDSREF environment variable to 1 before invoking the IC Compiler tool. You can also set this variable in your .cshrc file.

```
setenv ICV_ENABLE_GDSREF 1
```

Setting Up the Hercules Environment

To use the Hercules tool when running the signoff_metal_fill command, you must have a Hercules license, and you must specify the location of the Hercules executable by setting the HERCULES_HOME_DIR environment variable. You can set this variable in your .cshrc file. For example,

```
setenv HERCULES_HOME_DIR /root_dir/hercules
set path = ($path $HERCULES_HOME_DIR/bin/AMD.64)
```

You must ensure that the version of the Hercules executable that you specify is compatible with the IC Compiler version that you are using. For more information about the Hercules tool, see the Hercules documentation, which is available on SolvNet.

Setting Up the Physical Signoff Options

To prepare for signoff metal fill, use the set_physical_signoff_options command to specify the name of the executable, either icv or hercules, and the runset file for metal fill. For example,

```
icc_shell> set_physical_signoff_options -exec_cmd icv \
   -fill_runset my_fill_runset_file
```

You can report the option settings by using the report_physical_signoff_options command.

Setting Up Distributed Processing

By default, the signoff_metal_fill command uses a single process to perform metal fill insertion. To reduce the turnaround time for metal fill insertion, you can use distributed processing. To enable distributed processing, you must define the distributed processing configuration by using the set_host_options command.

If you have defined more than one distributed processing configuration with the set_host_options command, the signoff_metal_fill command selects the IC Validator processing method in the following order of priority:

1. Job submission through a user-defined distributed processing script

   To enable job submission using your own script with the set_host_options command, use the -submit_command option to specify the location of your job submission script. For example, to specify a configuration named custom4 that enables a maximum of four processes using your job submission script, enter the following command:

   ```
   icc_shell> set_host_options -name custom4 -num_processes 4 \
      -submit_command /usr/local/bin/my_submit_command
   ```

2. Job submission through the Load Sharing Facility (LSF) or the Sun Grid Engine (SGE)

   To enable LSF or SGE job submission with the set_host_options command, use the -pool option to specify the mode and the -num_processes option to specify the maximum number of processes.

   For example, to specify a configuration named lsf4 that enables a maximum of four processes using LSF, enter the following command:

   ```
   icc_shell> set_host_options -name lsf4 -pool lsf -num_processes 4
   ```

   To specify a configuration named grd4 that enables a maximum of four processes using SGE, enter the following command:

   ```
   icc_shell> set_host_options -name grd4 -pool grd -num_processes 4
   ```

3. Distributed processing on the specified hosts

   To enable distributed processing with the set_host_options command, specify the hosts to use and use the -num_processes option to specify the maximum number of processes on each host. For example, to specify a configuration named dp4 that enables a maximum of four processes, with a maximum of two processes each on machineA and machineB, enter the following command:

   ```
   icc_shell> set_host_options -name dp4 -num_processes 2 \
      {machineA machineB}
   ```

4. Multithreading

   To enable multithreading on the current machine with the set_host_options command, use the -max_cores option to specify the number of threads. For example, to specify a configuration named mt4 that enables a maximum of four threads on the current machine, enter the following command:

   ```
   icc_shell> set_host_options -name mt4 -max_cores 4
   ```

To ensure that you are using the intended distributed processing configuration, remove the current configurations by using the remove_host_options command before defining the

distributed processing configuration for the `signoff_metal_fill` command. To report the current distributed processing configurations, use the `report_host_options` command.

For more information about the `set_host_options` command, see Chapter 2, “Working With the IC Compiler Tool,” in the *IC Compiler Implementation User Guide*.

Running the signoff_metal_fill Command

Before you run the `signoff_metal_fill` command, the design must be fully routed and have only a very few or no DRC violations, and you must save the most recent revision of the design in a CEL view. The IC Validator or Hercules tool reads and operates on the design data in CEL, FRAM, and FILL views stored on disk, not on the current design in IC Compiler memory.

You can use the `signoff_metal_fill` command to perform the following tasks, which are described in this section:

- Standard Metal Fill Insertion
- Timing-Driven Metal Fill Insertion
- Metal Fill Removal
- Post-ECO Metal Fill Cleanup

The following sections describe these tasks.

Note:

When you run the `signoff_metal_fill` command, you can specify additional options for the Hercules or IC Validator command line by using the `-user_defined_options` option. The string that you specify in this option is added to the command line used to invoke metal fill insertion in the Hercules or IC Validator tool. The IC Compiler tool does not perform any checking on the specified string.

Standard Metal Fill Insertion

Standard metal fill insertion is the default mode for the `signoff_metal_fill` command. If you run the `signoff_metal_fill` command without any options, it performs the following tasks:

- Removes existing metal fill from the entire design.

  You can skip this step and perform incremental metal fill insertion by using the `-append` option (or by selecting “Keep existing metal fills in output view” in the Fill Options tab in the GUI). When you perform incremental metal fill insertion, you must use the default FILL view.

- Inserts metal fill in the empty regions for the whole design using the metal fill mode specified in the runset file, which is either hierarchical or flat.

  You can force the use of the flat metal fill mode by using the `-mode flat` option.

- Stores the inserted metal fill information in the default FILL view.

  You can specify a different name for the FILL view by using the `-output_view` option (or by entering the name in the “Output FILL view name” text box in the Fill Options tab in the GUI).

You can restrict the metal fill insertion to specific layers or specific regions of the design.

- Specifying the layers for metal fill insertion

  By default, the `signoff_metal_fill` command inserts metal fill on all the metal routing layers. To perform metal fill insertion on a specific set of metal and via layers, specify the layers in the `-select_layers` option (or select them in the list in the “Insert/Remove metal fills in selected layers” area in the GUI).

  By default, when you use this option, the `signoff_metal_fill` command removes all existing metal fill from the design and then inserts metal fill only on the specified layers. To keep the existing metal fill on the unspecified layers and to redo metal fill insertion only on the specified layers, use the `-eco` option. If you use the `-eco` option, you must use the default FILL view.

  For example, to remove all metal fill on layers M1 and M3 and refill those two layers without affecting the metal fill on other layers, enter the following command:

  ```
  icc_shell> signoff_metal_fill -eco -select_layers {M1 M3}
  ```

- Specifying the regions for metal fill insertion

  By default, the `signoff_metal_fill` command inserts metal fill for the whole chip. You can restrict metal fill insertion to one or more regions of the design by specifying regions

in which to insert metal fill or by specifying regions in which to prevent metal fill insertion or both.

To restrict metal fill insertion to specific regions of the design, use the `-bounding_boxes` option to specify the coordinates of each region in which to insert metal fill. You can specify multiple areas by specifying the coordinates for each area. If you are using the GUI, identify the regions in which to insert metal fill by selecting “Selected areas” and specifying the coordinates of each bounding box in the “Bounding Boxes” list. To specify the coordinates for a region, either draw the bounding box or enter the x- and y-coordinates of the box. For each region, you can also enable or disable snapping. If snapping is enabled, you can specify that the bounding box should snap to the minimum grid (the default), placement site, routing track, middle routing track, or user grid.

Note:

The bounding box coordinates passed to the Hercules or IC Validator tool in the `METAL_FILL_SELECT_WINDOW` parameter are enlarged by 1 µm to avoid DRC violations on the boundary of the specified regions during metal fill insertion. The actual metal fill insertion occurs within the regions specified by the `-bounding_boxes` option.

In addition, when you use the `-bounding_boxes` option, the `signoff_metal_fill` command always uses flat metal fill mode.

To prevent metal fill insertion in specific regions, use the `-excluded_bounding_boxes` option or select “Excluded areas” and specify the coordinates of each bounding box in the “Bounding Boxes” list in the GUI.

By default, when you use these options, the `signoff_metal_fill` command removes all existing metal fill from the design and then inserts metal fill only in the specified regions. To redo metal fill insertion only on the specified regions and keep the other existing metal fill, use the `-eco` option. If you use the `-eco` option, you must use the default FILL view.

For example, to remove all metal fill from the design and then fill all empty regions outside the bounding box with corners at (100,150) and (300,200), enter the following command:

```
icc_shell> signoff_metal_fill \
   -excluded_bounding_boxes {{100 150} {300 200}}
```

Timing-Driven Metal Fill Insertion

Timing-driven metal fill insertion inserts metal fill in the specified regions of the design, except around timing-critical nets. You can either explicitly specify the timing-critical nets or you can specify slack thresholds that enable the tool to determine the timing-critical nets automatically.

Note:

Timing-driven metal fill insertion is supported only in the IC Validator tool.

To explicitly specify the critical nets, use the `-timing_preserve_nets` option. To specify a setup slack threshold, use the `-timing_preserve_setup_slack_threshold` option. To specify a hold slack threshold, use the `-timing_preserve_hold_slack_threshold` option. If you are using the GUI, you can set these values in the Timing Preserve tab.

By default, the minimum spacing between a critical net and metal fill is twice the minimum spacing for the layer on which the net occurs. In addition, no metal fill is inserted within the minimum spacing around the vertical extension of the net on the layers above and below the net. You can specify a different same-layer minimum spacing requirement by using the `-space_to_critical_nets` option. You can allow metal fill insertion within the vertical extension of the net on the adjacent layers by using the `-fill_over_critical_nets` option.

If you specify a large minimum spacing requirement, timing-driven metal fill insertion might cause density errors. To prevent the introduction of density errors during timing-driven metal fill insertion, use the `-fix_density_errors true` option. Note that this option requires that the metal density rules are defined in the technology file. For information about defining metal density rules in the technology file, see the *IC Compiler Technology File and Routing Rules Reference Manual*.

Note:

You can specify the regions in which to perform timing-driven metal fill insertion, as described in "Standard Metal Fill Insertion" on page 4-44; however, you cannot specify the layers. When performing timing-driven metal fill insertion, the `signoff_metal_fill` command inserts metal fill on all routing layers.

During timing-driven metal fill insertion, the `signoff_metal_fill` command

- Performs timing analysis, including multicorner-multimode analysis, to minimize timing impact.
- Identifies timing-critical nets based on the options you specify.
- Invokes the IC Validator tool to perform metal fill insertion.

  When you perform timing-driven metal fill insertion, you must always use the default FILL view.

  By default, when you run timing-driven metal fill insertion, the `signoff_metal_fill` command removes all existing metal fill from the design and then inserts metal fill in the specified regions, except in the areas around the critical nets. To do timing-driven metal fill insertion on the specified regions and keep the other existing metal fill, use the `-eco` option. You cannot use `-append` option when performing timing-driven metal fill insertion.

Metal Fill Removal

To remove all metal fill from the design, use the `-purge` option (or select “Remove fills” in the GUI). When you use this option, you must use the default FILL view. You can specify the layers on which to remove the metal fill by using the `-select_layers` option. You can specify the regions from which to remove the metal fill by using the `-bounding_boxes` option. No other options are supported with the `-purge` option. For more information about specifying the layers and regions for metal fill removal, see “Standard Metal Fill Insertion” on page 4-44.

Post-ECO Metal Fill Cleanup

After you perform metal fill insertion and ECO routing, use the `signoff_metal_fill` command to remove any metal fill that overlaps a net. You can use the following methods to remove the metal fill:

- Remove the metal fill based on foundry-specific rules

  The `signoff_metal_fill` command supports metal fill removal based on the fill spacing rules for several foundry and node combinations. To determine the supported foundry and node combinations, use the `signoff_metal_fill -remove_overlap_by_rules list` command.

  To remove the metal fill that overlaps any net based on foundry-specific rules,

  1. Prepare the fill removal runset include file.

     The fill removal runset include file specifies foundry’s fill spacing rule values. For details about how to prepare this file, see SolvNet article 035828.

  2. Specify the fill removal runset include file by using the `-fill_removal_runset_include_file` option with the `set_physical_signoff_options` command.

     ```
     icc_shell> set_physical_signoff_options \
        -fill_removal_runset_include_file file_name
     ```

  3. Perform fill removal by using the `-remove_overlap_by_rules` option with the `signoff_metal_fill` command to specify one of the keywords returned by the `signoff_metal_fill -remove_overlap_by_rules list` command.

     ```
     icc_shell> signoff_metal_fill \
        -remove_overlap_by_rules foundry_node
     ```

- Remove the metal fill based on minimum spacing

  The `signoff_metal_fill` command supports metal fill removal based on either the minimum spacing rules defined in the technology library or user-specified minimum spacing values. When removing metal fill based on the minimum spacing rules defined in the technology library, the tool removes the metal fill around each net such that there

is no metal fill within twice the minimum spacing for the layer on which the net occurs. You can either remove the metal fill on all nets or only on specified nets.

To remove the metal fill that overlaps any net based on the minimum spacing rules, use the `-remove_overlap_by_rules min_spacing` option. To use user-defined spacing values instead of the minimum spacing rules, you must also use the `-space_to_critical_nets` option to specify the spacing requirements for each layer.

For example, to use the minimum spacing rules to remove the metal fill that overlaps the existing nets, enter the following command:

```
icc_shell> signoff_metal_fill -remove_overlap_by_rules min_spacing
```

To remove the metal fill that overlaps specific nets based on the minimum spacing rules, use the `-remove_overlap_with_nets` *nets* option. To use user-defined spacing values instead of the minimum spacing rules, you must also use the `-space_to_critical_nets` option to specify the spacing requirements for each layer.

For example, to remove the metal fill that overlaps the n1 net with a minimum spacing of 0.2 microns on the M1, M2, and M3 metal layers, enter the following command:

```
icc_shell> signoff_metal_fill -remove_overlap_with_nets n1 \
   -space_to_critical_nets {M1 0.2 M2 0.2 M3 0.2}
```

Note:

You can specify the layers on which to remove the metal fill, as described in "Standard Metal Fill Insertion" on page 4-44; however, you cannot specify the regions. When removing overlapping metal fill, the `signoff_metal_fill` command removes the metal fill around the specified nets.

For example, to use the minimum spacing rules to remove the metal fill on layers M1, M2, and M3 that overlaps existing nets, enter the following command:

```
icc_shell> signoff_metal_fill -remove_overlap_by_rules min_spacing \
   -select_layers {M1 M2 M3} \
```

Performing Notch and Gap Filling

After routing is complete, you can fill notches and gaps that are smaller than the minimum distance limit between objects of the same net on the same layer. The generated notch-and-gap-filling information is stored in the FILL view cell and can be used when you translate your design data to GDSII format.

To perform notch and gap filling, use the `insert_ng_filler` command (or choose Finishing > Insert Notch Gap Filler in the GUI). This is the command syntax:

```
insert_ng_filler
   [-include_existing_notch_gap_fill_cell]
   [-skip_filling_child_cell]
   [-dont_apply_fat_wire_rule]
   [-dont_fill_corner_to_corner]
   [-output outname]
   [-notch_layer_data_type notch_datatype]
   [-gap_layer_data_type gap_datatype]
   [-layers layer_list]
```

For example,

```
icc_shell> insert_ng_filler -include_existing_notch_gap_fill_cell \
   -dont_apply_fat_wire_rule -output test.err
```

The command has options to specify the following:

- Whether to use existing notch and gap cells existing in the netlist and having the .FILL extension
- Whether to skip reading the child cell's port data and applying fill on those ports
- Whether to ignore the fat wire rule when filling notches and gaps
- Whether to ignore corner-to-corner notch errors
- Whether to save the notch and gap errors into a file
- The notch layer data type and gap layer data type numbers
- The list of layers for which to fill notches and gaps

You can also perform notch and gap filling during the search-and-repair process. Doing so enables the router to correct DRC violations as it fills notches and gaps, including violations of 90-nm rules. Set the relevant routing option by using the `set_route_options -same_net_notch check_and fix` command (or by choosing Route > Routing Setup > Set Route Options in the GUI and selecting the corresponding option under the Miscellaneous tab). This setting instructs the router to fix same-net notch violations during the search-and-repair process. For more information, see "Search and Repair" on page 3-26.

Lithography Compliance Checking

Lithography compliance checking (LCC) is a method of increasing yields by analyzing the chip layout in detail for conditions that can lead to lithography defects and that cannot be fixed by ordinary optical proximity correction methods. PrimeYield LCC is a Synopsys product designed to find these conditions and generate reports on LCC hotspots, which are locations of potential defects. It simulates the full resolution-enhancement technology tape-out flow using the same production-baseline technology and manufacturing models as those used by foundries and integrated device manufacturers. It reports the location and type of each occurrence of lithographic sensitivity to potential defects such as line-end narrowing and line-end bridging. It ranks these problems by severity and presents them for review.

In the IC Compiler tool, you can detect potential defect locations with the `detect_lcc_hotspot` command, which invokes PrimeYield LCC in the background to find the hotspots. You can view the reported hotspots in the IC Compiler GUI and fix them with the `fix_lcc_hotspot` command. The fixing command repairs the reported conditions by moving, filling, and widening routes and by adding, deleting, and moving vias; or in cases where these local fixing methods cannot be used effectively, it rips up and reroutes the problem net.

PrimeYield LCC licensing is required to use these commands. The PrimeYield LCC and Hercules version used by the IC Compiler tool must be the same as the version used to generate the LCC data files. For information about setting up and using the PrimeYield LCC tool, see the *PrimeYield LCC User Guide* provided with the PrimeYield LCC product.

Before running LCC detection, perform via optimization and critical area reduction and ensure that the design does not have any DRC violations.

Detecting LCC Hotspots

The `detect_lcc_hotspot` command invokes PrimeYield to detect potential lithography-related defects. This is the command syntax:

```
detect_lcc_hotspot
   -lcc_file_path lcc_path_name
   -layers list_of_layers
   [-dp_hosts list_of_dp_hosts]
```

For example,

```
icc_shell> detect_lcc_hotspot \
   -lcc_file_path /LCC/full-chip \
   -layers {M2 M3 M4} \
   -dp_hosts {lin1 lin2 lin3 lin4 lin5 lin6 lin7 lin8}
```

This command performs full-chip LCC detection and produces two fixing guidance files in the working directory, out00 and fout00. These two files contains hotspot information and guidance for local fixing and reroute fixing. They are required to run the `fix_lcc_hotspot` command.

The hotspot detection process also produces an error view named *top_cell_name*_lcc.err. You can open this error view in the GUI using Verification > Error Browser and view the hotspots in the current design. Each hotspot is highlighted as a DRC error.

In the `detect_lcc_hotspot` command, you must specify the full path name of the directory containing the PrimeYield LCC runset files necessary for full-chip LCC detection. For information about preparing runset files, see the *PrimeYield LCC User Guide* provided with the PrimeYield LCC product, or see the article “Using PrimeYield LCC,” which is available as SolvNet article 023079. You must also specify the layers to be checked.

The `detect_lcc_hotspot` command requires distributed processing because of the long runtime for full-chip LCC detection. You can specify the machines to be used with the `-dp_hosts` option, the .dprc file under the working directory, or the .dprc file specified under the *lcc_path_name* directory. Running hotspot detection on a single CPU is not supported.

The distributed processing engine used by the `detect_lcc_hotspot` command is different from that used by the `fix_lcc_hotspot` command and many other routing commands. You do not need to run the `set_distributed_route` command before the `detect_lcc_hotspot` command. If you run the `set_distributed_route` command, run the `remove_distributed_route` command before the `detect_lcc_hotspot` command to release the reserved network resources. Otherwise, the communication port between machines might become occupied and block distributed processing functionality.

Fixing LCC Hotspots

The `fix_lcc_hotspot` command attempts to fix hotspots previously detected with the `detect_lcc_hotspot` command. To obtain maximum fixing quality and to ensure iteration convergence, the design must not have DRC violations. The two fixing guidance files generated by the `detect_lcc_hotspot` command, out00 and fout00, must be present the working directory.

This is the syntax of the LCC hotspot-fixing command:

```
fix_lcc_hotspot
   -lcc_file_path lcc_path_name
   [-types list_of_types]
   [-level level]
   [-num_loops number_of_loops]
   [-num_cpus number_of_cpus]
```

For example,

```
icc_shell> set_distributed_route \
   -jp_machines {linux1 linux2 linux3 linux4 linux5}

icc_shell> fix_lcc_hotspot \
   -lcc_file_path /root/LCC/Local-LCC \
   -types {line space lineend slotend} \
   -level 1 -num_cpus 10
```

Distributed processing is recommended because runtimes can be long. First run the `set_distributed_route` command to set up the machines to be used. Then, in the `fix_lcc_hotspot` command, set the `-num_cpus` option to a value greater than 1. For more information about distributed routing, see the “Enabling Distributed Routing” on page 2-18.

In the `fix_lcc_hotspot` command, you must specify the full path name of the same LCC directory used by the `detect_lcc_hotspot` command. This is the directory containing the PrimeYield LCC runset files necessary for full-chip LCC detection. You can optionally specify the types of hotspots to fix, the hotspot severity levels to fix, the maximum number of fixing loops to attempt, and the number of CPUs to use.

The `-types` option specifies the types of hotspots to fix, which can be any combination of `line`, `lineend`, `space`, `slotend`, and `voc` (via overlay check). The default behavior is to fix all types of hotspots.

The `-levels` option specifies the severity levels to fix. If you use this option, the command attempts to fix only the hotspots having a severity number equal to or less than the specified number. Lower numbers represent the more severe hotspots.

There are two severity level systems, LCC-t and LCC-g. The IC Compiler tool gets the severity level system, either LCC-t or LCC-t, from the LCC runsets. You need to know which level system is being used so you can enter an appropriate level number. The LCC-g system

has the possible level values 1, 2, 3, 4, and 5. If you specify the `-levels 3` option in this system, the command fixes only the hotspots having a severity level of 1, 2, or 3. The default maximum severity value for this system is 2.

The `fix_lcc_hotspot` command attempts to fix LCC hotspots using multiple iterations of fixing and rechecking. In each iteration, it first tries to fix hotspots by local-fixing and rerouting. New DRC violations or new LCC hotspots might occur as a result of fixing, so it performs internal DRC checking and launches LCC checking jobs to check the changed patterns. This process is repeated until there are no more LCC hotspots or DRC violations, or until the number of iterations reaches the maximum value specified by the `-num_loops` option. Most designs that can converge do so in fewer than 10 iterations.

Index

D

E

F

G

H

I

L

M

N

O

P

R

S

T

V

W

Y